Pychron Documentation

Release Dev

Jake Ross

Sep 08, 2023

CONTENTS


New
Mexico
Geochronological
Research
Laboratory


pychronLABS


NEW MEXICO TECH

CHAPTER

ONE

PYCHRON INTRODUCTION

1.1 Abstract

Pychron is a fully featured open source project for automated data collection and processing. The majority of Pychron's codebase is written using Python, an increasing popular programming language within the scientific and open source communities. Pychron is optimized for use with the next generation multi-collector mass spectrometers from Thermo Scientific. However, Pychron was designed with flexibility and extensibility in mind and application to other hardware and isotopic systems is readily achievable.

The applications that comprise Pychron use Qt for a modern graphical user interface. GUI's are provided for all major hardware components such as the mass spectrometer and laser systems, in addition to interfaces for configuring and executing sequences of automated analyses.

Pychron can save data to a variety of SQL databases such as MySQL and PostgreSQL. Export facilities are available to translate analysis data and metadata to numerous formats including, XML, CSV, HDF5 and the MassSpec MySQL schema.

Analyses are processed manually using a streamlined workflow. Pychron also provides configurable batch processing procedures to automate the data reduction process. Pychron features interactive plots including, time series, ideograms, spectra, and inverse isochrons. Summary PDF tables following the Renne et al., 2009 format and appendix style data tables in PDF, CSV or Excel 97-2004 format are generated from easy-to-use dialogs.

Built-in unittests are provided with Pychron to verify the accuracy of the Ar-Ar and other statistical calculations. Unittests are executed autonomously and continuously using the Continuous Integration service, Travis CI. Documentation is located at ReadTheDocs.org (https://pychron.rtfd.org). The entire Pychron source code is licensed under the liberal Apache 2.0 open source license and publicly hosted at GitHub.com (https://github.com/nmgrl/pychron). A beta version of Pychron is currently available that includes both data acquisition and processing capabilities, identified by the DOI 10.5281/zenodo.9884 (Ross, 2014).

1.2 Introduction

Software has become an essential part of the Ar-Ar geochronology technique, both for data collection and data processing. The technique has greatly benefited from inexpensive and accessible computing hardware facilitating fully automated data acquisition. Storage of raw isotopic measurements and metadata in central relational database management systems (RDBMS) such as MySQL have dramatically increased the quality and throughput of analytical data from Ar-Ar facilities. Although software and interfacing with hardware has long been recognized as a critical piece of a laboratory's infrastructure currently, there exists only one widely used and fully integrated package for Ar-Ar analysis, Mass Spec by Alan Deino of Berkeley Geochronology Center (BGC). Mass Spec has been of great value to the community for 20+ years, however we do not consider it a viable and sustainable product for the future in its current form. Mass Spec does not make use of the many advancements in software development and engineering that have occurred in the recent past. Mass spec has no adequate version control scheme and relies completely on version

numbers for maintaining a history of changes. No API or extensive documentation is available for Mass Spec, making upgrades, bug fixes and generic modifications time consuming and inefficient. Mass Spec uses a custom esoteric scripting language for data collection that greatly limits is flexibility and extensibility. For all of these reasons and more we determined that a new platform was necessary for sustaining the high quality and highly desired results produced by Ar-Ar geochronology.

To address many of our concerns with Mass Spec and the general software ecosystem in Ar-Ar geochronology we chose to develop an open-source and extensible software product named Pychron. Pychron is freely available and makes use of many recent advancements in software development. It features robust version control via Git and is publicly hosted at GitHub. Pychron leverages the tremendous efforts of the open-source and scientific communities and uses widely used and accepted packages such as Numpy, Scipy (Jones et al., 2001) and SQLAlchemy (Bayer, 2006) to ensure sustainability and efficient design.

The name Pychron is a portmanteau of Python (the main language used for implementation) and Chronology (its main domain space). We choose Python because of its increasing popularity among the scientific community as well as the web development community, an area we hope Pychron will benefit from in the near future. Python allows for rapid prototyping of new features, a critical property for evolving experimental scientific laboratories. As articulated in the Zen of Python (PEP 20), Python follows the paradigm that code is more often read than written, making it an ideal language for both novice and advanced programmers, the full range of which is found in the scientific community. Python's ``batteries included'' concept makes it easy to implement new features, robustly, and when included components are not adequate an extensive body of open source and proprietary packages are readily available. Python, being an interpreted language, is often criticized for being too slow. However if computation speed becomes a limiting factor, algorithms can be implemented using lower level compiled languages, such as Fortran or C, and accessed directly from python modules. Python tools designed for optimization and speed such as Cython or its predecessor Pyrex provide additional means to mitigate any speed limitations.

Pychron was specifically designed to operate with Thermo Scientific new generation multicollector mass spectrometers, notably the ArgusVI. Instead of directly communicating with the ArgusVI's electronics we wrote a lightweight remote method invocation server in C-sharp, called `RemoteControlServer.cs`. This relatively simple UDP/TCP server runs within Thermo's Qtegra environment and facilitates remote control of the mass spectrometer either locally or over a shared network. The `RemoteControlServer.cs` model, over the past 1-2 years has proven itself as a easy and rapid way to control the new generation of mass spectrometers, allowing users to quickly move from instrument setup and installation to the end goal of making isotopic measurements. It is installed at numerous noble gas laboratories around the world and is currently the de facto method for interfacing third party software with Qtegra.

Running Pychron on any system is achievable by a variety of mechanisms. A simple entry point script is provided for developers to launch from the command-line or their preferred IDE (Integrated Development Environment). For end users, a self-contained application is constructed using a built-in python script called `app_maker.py`. The resulting application bundle allows the user to launch from the dock or start menu. (Currently `app_maker.py` is only setup for Mac OSX, but variants for Windows and Linux are possible)

CHAPTER

TWO

PYTHON PRIMER

Think of python as a program. what it does is takes human readable source code (.py) and parses, compiles and executes in real time.

Python 2.7 Documentation

Best place to start. Python Tutorial

You can run python in 2 ways. (there is actually many ways to run from the command line but only 2 are used frequently)

Open terminal

1. with no arguments. opens the python interpreter

```
>>> python
```

2. with a path to python script (absolute or relative path). executes the script >>> python hello_world.py

Using Python Interpreter Zen of Python

```
>>> import this
```

you can use the command line as a powerful calculator see http://docs.python.org/tutorial/introduction.html

```
>>> x=1
>>> y=2
>>> x+y
3
>>> x=50
```

Use the up and down arrows to cycle thru previous commands

```
>>> x+y
53
>>> y=x+y
>>> x+y
103
>>> i=0
>>> i+=1    #same as i = i+1 works for all math operators i -=1, i*=2 etc...
```

strings are defined using ', ", or ''' blocks. ' and " are equivalent. convenient when needing to nest quotes.

```
>>> s='foo'
>>> b='bar'
```

(continues on next page)

(continued from previous page)

```
>>> print s,b
foo bar
>>> s
'foo'
>>> b
'bar'
>>> "foo" == 'foo' == '''foo'''
True
```

to make a list of items use a list or tuple see http://docs.python.org/tutorial/datastructures.html

```
>>> l1=[1,2,3,4]
>>> l2=['foo','bar']
>>> t1=(1,2,3)
>>> t2=('foo','bar')
```

lists are mutable, tuples are immutable

```
>>> l1[0]=10
>>> l1
[10,2,3,4]
>>> t1[0]=10
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

get the length of the sequence use builtin len

```
>>> len(l1)
4
```

to generate a list of numbers use builtin range

```
>>> range(10)
[0,1,2,3,4,5,6,7,8,9]
```

```
>>> range(0,10,2)
[0,2,4,6,8]
```

get item from list

```
>>> l1[0]
1
```

get the last item

```
>>> l1[-1]
9
```

get a sublist

```
>>> l1[0:2]    #list[startindex:stopindex:step]
               #each parameter is optional but at least needs to be set
```

(continues on next page)

(continued from previous page)

```
                #startindex defaults to 0
                #stopindex defaults to the last index
                #step defaults to 1
                #l1[0:2] same as l1[0:2:1] and l1[:2] (preferred)
```

same slicing operations work on strings. just think of them as a list of characters

```
>>> s= 'hello world'
>>> s[:5]
'hello'
>>> s[6:]
'world'
>>> s[-5:]
'world'
```

you can split and join strings easily

```
>>> s.split(' ') #str.list(character to  split on) returns a list
['hello', 'world']
>>> ', '.join(s.split(' ')) #join_str.join(list of strings to join)
hello, world
>>> '\n'.join(['this is a good','way to write multi','line text'])
this is a good
way to write multi
line text
```

Dictionaries are key:value containers. There are two syntaxes for creating a dictionary

```
>>> d=dict(name='Jake', office=316, building='MSEC')
>>> d2 = {'name':'Jake','office':316, 'building':'MSEC'} #convenient when the keys are␣
↪variables as well
>>> key1='person'
>>> key2='id'
>>> val1='John'
>>> val2=10394303
>>> d3 = {key1:val1, key2:val2}
```

to get a value from the dictionary you specifiy a key. To get the definition of a word you find the word (key) in are dictionary and read the associated entry

```
>>> d['name']
Jake
```

entries can be modified

```
>>> d['name']='Jake Ross'
>>> d['name']
Jake Ross
```

String formating is awesome in python. Lets say you want to display some text with your results

```
>>> 'the result of {} plus {} is {}'.format(x,y,x+y)
'the result of 50 plus 53 is 103'
```

(continues on next page)

(continued from previous page)

```
>>> 'the result of {1} plus {0} is {2}'.format(x,y,x+y)
'the result of 53 plus 50 is 103'
```

you can use pass in a key:pairs

```
>>> "{name}'s office is {building} {office}".format(name='Jake',building='MSEC',
↪office=316)
```

or better

```
>>> "{name}'s office is {building} {office}".format(**d2)
"Jake's office is MSEC 316"
```

CHAPTER

THREE

PYCHRON USER GUIDE

3.1 Pychron Setup

3.1.1 Install Directions

General Instructions

1. Setup github account
2. Download and install git
3. Download and install anaconda3
4. Download and run installer
5. Launch and setup Pychron

Installer Scripts

- Mac Installer Script
- PC Installer Script

3.1.2 Quick Start (ish)

Note: The following information mostly accurate but this section is out of date (7/10/19)

Step 0. Downloads

A. (Optional) Download and install the excellent python IDE, PyCharm. A free community edition is available at https://www.jetbrains.com/pycharm/

B. Install Git. https://git-scm.com/downloads

C. Download the Anaconda Python 3.6 Distribution from https://store.continuum.io/cshop/anaconda/. This download includes the Python standard library and numerous open source packages for science, engineering and application development. The excellent package manager `conda` is also included to install any additional dependencies

D. Create an account at https://github.com. This free account will give you access to Pychron's source files, support files and even the source for this documentation.

Note: For steps A-C you must open the downloaded package and run the installer.

For Update Plugin

1. Create hidden directory `.pychron.<APPLLICATION_ID>` in your Home folder. Replace `<APPLICATION_ID>` with an integer. e.g. 0
2. Clone the pychron source. Replace `<ORGANIZATION>` with the appropriate fork e.g. NMGRL

```
cd ~/.pychron.0
git clone https://github.com/<ORGANIZATION>/pychron.git updates
```

Launcher Script

Create the file `pychron_launcher.sh` in a convenient place

```
#!/bin/bash

export APPLICATION_ID=0

ROOT=~/.pychron.$APPLICATION_ID/updates

echo Using $ROOT as "ROOT" directory

ENTRY_POINT=$ROOT/launchers/launcher.py

export PYCHRON_APPNAME=pyexperiment
export PYTHONPATH=$ROOT

export GITHUB_USER=
export GITHUB_PASSWORD=
export GITHUB_ORGANIZATION=
export MassSpecDBVersion=16
export CONDA_DISTRO=~/anaconda3
export CONDA_ENV=pychron
export QT_API=pyqt

$CONDA_DISTRO/envs/$CONDA_ENV/bin/pythonw $ENTRY_POINT
```

Setup Environment

Install command line developer tools

```
xcode-select --install
```

Setup the conda environment

```
conda create -n pychron3 python=3.5 python.app
source activate pychron3
conda install pymysql gitpython sqlalchemy reportlab lxml pyyaml yaml
conda install envisage pyqt=4 statsmodels
conda install xlrd xlwrt xlsxwriter
conda install requests certifi
conda install swig cython
pip install chaco uncertainties peakutils qimage2ndarray
```

Manual

1. Create a directory called `Pychron` in your Home folder. ie `/Users/<username>/Pychron` where `<username>` is replaced with your user name, for the remainder of this documentation we will assume the username is `argonlab`. This location will serve as the root directory for all Pychron configuration and data files.
2. Create a directory called `Programming` in your Home folder. `/Users/argonlab/Programming` This location will hold source files
3. Open a terminal window. `/Applications/Utilities/Terminal.app`
4. Execute the following commands. These commands will move you to the `Programming` directory, then download the pychron source files into a directory called `pychron`

   ```
   cd ~/Programming
   git clone https://github.com/<organization>/pychron.git
   ```

   Note: Replace `<organization>` with the name of your github organization. For example U. of Manitoba has its own pychron fork located at https://github.com/UManPychron/pychron.git

5. Check to make sure you have the source files. You should see a number of files and subdirectories after executing the following commands

   ```
   cd ~/Programming/pychron
   ls
   ```

6. checkout the version of Pychron you want to use. By default you should be on the "develop" branch. This is the

   leading edge of pychron development will have buggy features. The current public release of pychron is v16.7. To

   use the current release.

   ```
   cd ~/Programming/pychron
   git checkout release/v16.7
   ```

1. Before you can launch Pychron you must install some dependencies.

```
cd ~/Programming/pychron
cd app_utils/requirements
conda install --yes --file ./conda_requirements.txt
pip install -r ./pip_requirements.txt
```

2. Download the Pychron support files.

```
cd ~/Programming
git clone https://github.com/<organization>/support_pychron.git
```

Note: Again replace <organization> with the name of your github organization.

3. Move the directories in ~/Programming/support_pychron to ~/Pychron

Auto (Beta)

Warning: This feature is experimental and should be used with caution.

use the installer script, install.sh or install_development.sh

3.1.3 Spectrometer

Argus

1. Set the values in ~/Pychron/setupfiles/spectrometer/config.cfg

```
[General]
name = jan

[SourceParameters]
ionrepeller = 10.0
electronenergy = 40.20

[SourceOptics]
ysymmetry = 50.0
zsymmetry = 06.0
zfocus = 70.0
extractionlens = 01.0

[Deflections]
h2 = 0.0
h1 = 0.0
ax = 454.0
l1 = 0.0
l2 = 0.0
cdd = 0.0
```

(continues on next page)

(continued from previous page)

```
[CDDParameters]
operatingvoltage = 618.0

[Protection]
use_detector_protection=True
detectors=CDD
detector_protection_threshold=0.1
use_beam_blank=False
beam_blank_threshold=0.1
```

2. Set the `~/Pychron/setupfiles/spectrometer/mftable.csv`. Each value in the table is the peak center in DAC space for a given isotope on a given detector.

   Note: The DAC value is for a Deflection voltage=0

   To populate the table,

   1. Set the deflection to zero,
   2. Peak center Ar40 on H2
   3. Record the DAC value in the approriate cell
   4. Repeat 1-3 for Ar40 on H1,AX,..., etc
   5. Repeat 1-4 for Ar39,Ar36

3. Setup the deflection correction files

   1. Create directory `~/Programming/Pychron/setupfiles/spectrometer/deflections`
   2. For each detector create a text file using the detector name as the name of the file. e.g AX

      Note: Use all capital letters and leave off the .txt extension

   3. The detector files are simple csv files where each row represents a `deflection,dac` pair. The dac value is the voltage required to center a reference isotope (Ar40) on this detector for a given deflection. For example the AX file might look like

      ```
      0,6.003672
      135,5.99933
      ```

      Note: you can add as many calibration points as you like, but in practice the deflection voltages do not change over time so you can just define two points, 1) deflection=0 and 2) deflection=<normal operating voltage>

3.1.4 Extraction Line

1. Edit `Pychron/setupfiles/extractionline/valves.xml`. This file defines the valves used in the extraction line. The name and location of the valves file is configurable via Preferences/Extraction Line. Both .xml and .yaml file formats are supported 2. Edit `Pychron/setupfiles/canvas2D/canvas.xml`. This file defines the positions of various extraction line elements, namely valves 3. Edit `Pychron/setupfiles/canvas2D/canvas_config.xml`. Defines some global aspects of the main canvas

3.1.5 Example Valves.yaml

```yaml
- name: A
  address: Ftkh
  state_device:
    name: foo
    address: bar
    inverted: True
  description: Furnace Turbo
- name: H
  address: Blep
  interlock:
    - I
  description: Outer pipette
- name: I
  address: Blop
  interlock: H
  description: Inner pipette
- name: Air
  kind: pipette
  inner: G
  outer: F
- name: Cocktail
  kind: pipette
  inner: I
  outer: H
```

3.1.6 Example Valves.xml

```xml
<?xml version='1.0' encoding='ASCII'?>
<root>
    #Furnace
    <valve>A
        <address>FTkh</address>
        <state_device>foo
            <address>bar</address>
        </state_device>
        <description>Furnace Turbo</description>
    </valve>

  #Ref (Ar)
  <valve>H
```

(continues on next page)

(continued from previous page)

```
      <address>Pipet Ref. Out Set</address>
      <description>Ar Outer</description>
      <interlock>I</interlock>
  </valve>
  <valve>I
      <track>True</track>
      <address>Pipet Ref. In Set</address>
      <description>Ar Inner</description>
      <interlock>H</interlock>
  </valve>

  <switch>J<address>Getter manual 1 Degas Set</address><description>Getter manual 1 Degas
↪</description></switch>


  <manual_valve>T<description>CO2 Inlet</description></manual_valve>
  <manual_valve>U<description>Excimer Inlet</description></manual_valve>
  <manual_valve>ADiode</manual_valve>
  <manual_valve>RDiode</manual_valve>

  <pipette>Air
   <inner>G</inner>
   <outer>F</outer>
  </pipette>
  <pipette>Ar
   <inner>I</inner>
   <outer>H</outer>
  </pipette>
</root>
```

3.1.7 Plugins

Plugins are enabled/disabled in the `setupfiles/initialization.xml` file.

List of Plugins

- **General**
  - **Experiment** - Execute sets of automated runs.
  - **MassSpec** - Mass Spec plugin.
  - **PyScript** - Edit PyScripts; pychron's internal scripting language.
  - **ArArConstants** - List of Ar/Ar geochronology constants.
  - **Database** - SQL database interface.
  - **Loading** - Laser tray loading plugin.
  - **Pipeline** - Pychron's pipeline based processing workflow
  - **Entry** - Enter/Edit irradiation data.
  - **Workspace** - Git-enabled workspace repository.

  - **DVC** - Pychron's custom Data Version Control system.
  - **GitLab** - Private git repository hosting.
  - **GitHub** - Public git repository hosting at GitHub.com.
  - **MediaServer** - Image server/client.
  - **LabBook** - Git-enabled labbook repository.
  - **Video** - Video server/client.
  - **DashboardServer** - Publish various laboratory values.
  - **DashboardClient** - Listen to the Dashboard server.
  - **LabspyClient** - Labspy client. push updates to the labspy database.
  - **Update** - Update plugin.
  - **Image** - Use to take snapshots with a connected camera and save to file or database.
  - **IGSN** - International Geo Sample Number.
  - **Geochron** - Upload analyses to Geochron.org
- **Hardware**
  - **ExtractionLine** - Control extraction line components.
  - **ClientExtractionLine** - Remotely control extraction line components.
  - **ArgusSpectrometer** - Thermo ArgusVI plugin.
  - **HelixSpectrometer** - Thermo Helix plugin.
  - **NGXSpectrometer** - Isotopx NGX plugin.
  - **NMGRLFurnace** - NMGRL's resistance furnace plugin.
  - **ChromiumCO2** - Photon Machines Fusions CO2 control via "Chromium"
  - **FusionsCO2** - Photon Machines Fusions CO2.
  - **FusionsDiode** - Photon Machines Fusions Diode.
  - **FusionsUV** - NMGRL's custom Fusions UV.
  - **ExternalPipette** - Interface with the APIS pipette system.
  - **CanvasDesigner** - Visual editor for the Extraction Line Schematic.
- **Social** * **Email** - Allows pychron to send emails

Example Data Processing Initialization File

```
<root>
  <globals>
  </globals>
  <plugins>
    <general>
      <plugin enabled="true">Database</plugin>
      <plugin enabled="true">Processing</plugin>
      <plugin enabled="true">ArArConstants</plugin>
      <plugin enabled="true">Entry</plugin>
```

(continues on next page)

(continued from previous page)

```
      <plugin enabled="true">SystemMonitor</plugin>
    </general>
  </plugins>
</root>
```

Example Experiment Initialization File

```
<root>
  <globals>
  </globals>
  <plugins>
    <general>
      <plugin enabled="true">Database</plugin>
      <plugin enabled="true">Experiment</plugin>
      <plugin enabled="true">Processing</plugin>
      <plugin enabled="true">PyScript</plugin>
      <plugin enabled="true">ArArConstants</plugin>
      <plugin enabled="true">Entry</plugin>
      <plugin enabled="true">DashboardServer</plugin>
    </general>
    <hardware>
        <plugin enabled="false">Spectrometer
            <device enabled="true">spectrometer_microcontroller
              <klass>ArgusController</klass>
            </device>
        </plugin>
        <plugin enabled="true">ExtractionLine
            <processor enabled="false">/tmp/hardware-extractionline</processor>
            <manager enabled="false">gauge_manager
            <device enabled="true">bone_micro_ion_controller
                <klass>MicroIonController</klass>
            </device>
            <device enabled="false">microbone_micro_ion_controller
                <klass>MicroIonController</klass>
                <required>false</required>
            </device>
            </manager>
            <manager enabled="true">valve_manager
                <device enabled="true">valve_controller</device>
            </manager>
            <device enabled="true">air_transducer
                <klass>Transducer</klass>
            </device>
        </plugin>
    </hardware>
    <data>
    </data>
  </plugins>
</root>
```

Example Laser Initialization File

```
<root>
  <globals>
  </globals>
  <plugins>
    <general>
      <plugin enabled="true">Database</plugin>
      <plugin enabled="false">Experiment</plugin>
      <plugin enabled="true">Processing</plugin>
      <plugin enabled="false">PyScript</plugin>
      <plugin enabled="false">ArArConstants</plugin>
      <plugin enabled="false">Entry</plugin>
      <plugin enabled="false">SystemMonitor</plugin>
      <plugin enabled="true">DashboardServer</plugin>
    </general>
    <hardware>
        <plugin enabled="false">Spectrometer
            <device enabled="true">spectrometer_microcontroller
              <klass>ArgusController</klass>
            </device>
        </plugin>
        <plugin enabled="true">ExtractionLine
            <processor enabled="false">/tmp/hardware-extractionline</processor>
            <manager enabled="false">gauge_manager
            <device enabled="true">bone_micro_ion_controller
                <klass>MicroIonController</klass>
            </device>
            <device enabled="false">microbone_micro_ion_controller
                <klass>MicroIonController</klass>
                <required>false</required>
            </device>
            </manager>
            <manager enabled="true">valve_manager
                <device enabled="true">valve_controller</device>
            </manager>
            <device enabled="true">air_transducer
                <klass>Transducer</klass>
            </device>
        </plugin>
    </hardware>
    <data>
    </data>
    <social>
      <plugin enabled="true">Email</plugin>
      <plugin enabled="false">Twitter</plugin>
    </social>
  </plugins>
</root>
```

3.1.8 Extraction Line Canvas

This section describes how to construct and modify an extraction line canvas.

Two files are required to define an extraction line canvas.

1. `setupfiles/canvas2D/canvas.xml` Defines the canvas elements
2. `setupfiles/canvas2D/canvas_config.xml` Defines globals parameters e.i origin, bgcolor
3. Optional. `setupfiles/canvas2D/alt_canvas.xml` Defines globals for the secondary canvas (Canvas displayed as a pane within a window). Useful for defining a smaller font when canvas displayed with Experiment or Spectrometer tasks.

Lets construct an example canvas.xml file to see how it works.

The file starts with the standard <root> xml tag. All other elements will be children of `root`. Various graphical elements can be easily added to the canvas. Lets start with a `stage` element. `Stages` are generic areas that connect multiple valves.

```
<root>
    <stage>Bone
       <translation>10,5</translation>
       <dimension>5,2</dimension>
       <color>255,0,0</color>
    </stage>
</root>
```

The above snippet defines a `stage` named **"Bone"** that is 5 units wide, 2 units tall and positioned 10 units to the right and 5 units above the center of the canvas. The color of the stage is defined using the color tag and is red in this case- 255,0,0 (R,G,B). By default the name of the stage is rendered by can be disabled using `display_name="false"`

```
...
<stage display_name="false">Bone
...
```

Before we can see what our canvas looks like, we need to make a `canvas_config.xml` file. This file contains X important elements.

1. origin. Shift the center of the canvas X,Y
2. xview. Left bounds, Right bounds
3. yview. Bottom bounds, Top bounds
4. color tags. default colors for canvas elements

```
<root>
    <origin>0,2.5</origin>
    <xview>-25,25</xview>
    <yview>-22,22</yview>
    <color tag="bgcolor">lightblue</color>
    <color tag="getter">green</color>
    <!-- optional-->
    <font>arial 12</font>
</root>
```

The above snippet shifts the canvas up 2.5 units and sets the background color to `lightblue`. The default color for `"getter"` elements is "green". The left bounds of the canvas is -25 and the right 25 (width=50 units). The upper and

lower bounds are -22 and 22 respectively (height=44 units). Optionally the font used for labels can be set using a `font` tag.

(Note. Defining the canvas is independent of defining hardware connections. Elements defined here need not have physical representations)

3.1.9 Startup Tests

Startup tests are enabled using `setupfiles/startup_tests.yaml`

```yaml - plugin: DatabasePlugin
    **tests:**
        - test_pychron
        - test_pychron_version
- plugin: MassSpecPlugin tests:
    - test_database
- plugin: LabBookPlugin tests:
- plugin: ArArConstantsPlugin tests:
- plugin: ArgusSpectrometerPlugin tests:
    - test_communication
    - test_intensity
- plugin: ExtractionLinePlugin tests:
    - test_valve_communication
    - test_gauge_communication
- plugin: DVC tests:
    - test_database
    - test_dvc_fetch_meta
- plugin: GitHub tests:
    - test_api
- plugin: NMGRLFurnacePlugin tests:
    - test_furnace_api
    - test_furnace_cam
```

3.1.10 Tray Maps Format

Example tray map file. Tray maps are located in `Pychron/setupfiles/tray_maps/37_hole.txt`

```
circle,3.74              # shape, dimension (mm)
19                       # holes to draw. Not currently used
2,21,36,16,19            # calibration holes. N,E,S,W,center
                         # X,Y
-4.8006, 14.4018
0, 14.4018
4.8006, 14.4018
-9.6012, 9.6012
-4.8006, 9.6012
0, 9.6012
4.8006, 9.6012
9.6012, 9.6012
-14.4018, 4.8006
-9.6012, 4.8006
-4.8006, 4.8006
0, 4.8006
4.8006, 4.8006
9.6012, 4.8006
14.4018, 4.8006
-14.4018, 0
-9.6012, 0
-4.8006, 0
0, 0
4.8006, 0
9.6012, 0
14.4018, 0
-14.4018, -4.8006
-9.6012, -4.8006
-4.8006, -4.8006
0, -4.8006
4.8006, -4.8006
9.6012, -4.8006
14.4018, -4.8006
-9.6012, -9.6012
-4.8006, -9.6012
0, -9.6012
4.8006, -9.6012
9.6012, -9.6012
-4.8006, -14.4018
0, -14.4018
4.8006, -14.4018
```

3.1.11 Ratio Change Detection

Ratio Change Detection are enabled `Preferences/Experiment` and configured using `setupfiles/ratio_change_detection.yaml`

```yaml - ratio: Ar40/Ar36

    nanalyses: 5 threshold: 1 percent_threshold: 1 nominal_ratio: 295 nsigma: 3 analysis_type: air failure_count: 2 consecutive_failure: True

- ratio: Ar40/Ar39 nanalyses: 5 threshold: 1 percent_threshold: 1 nominal_ratio: 10 nsigma: 3 analysis_type: cocktail failure_count: 2 consecutive_failure: True
```

3.1.12 DVC Configuration

Meta Repository

Cocktail Irradiation

Example cocktail.json file

```
{
  "chronology": "2016-06-01 17:00:00",
  "j": 4e-4,
  "j_err": 4e-9
}
```

Repositories

DVC Database

Ar-Ar Mapping

Example *arar_mapping.yaml*

```
{
    Ar40: 'Ar40',
    Ar39: 'Ar39',
    Ar38: 'Ar38',
    Ar37: 'Ar37',
    Ar36: 'Ar36L1'
}
```

3.2 Preferences

3.2.1 General

1. Confirm Quit. Ask user to confirm quitting the application
2. Random Tip. Display a random tip at start up
3. Default PI. Name of the default Principal Investigator

Root

4. Pychron Directory

Login

5. Use Login
6. Multi User

Organization

7. Name. Name of GitHub organization for submitting bugs

Laboratory Repo

8. Name. Name of GitHub repository for Laboratory notes

3.2.2 Browser

References Padding (hrs)
: Number of hours used when finding Reference analyses (airs, blanks, etc) associated with a set of analyses. e.g if References Padding = 10 then pychron will get references between oldest_analysis_time - 10 and youngest_analysis_time+10

Max. Analysis Sets
: Maximum number of analysis sets to maintain.

Analysis Colors

Use Analysis Colors
Enable color coding the analyses by analysis type in the Browser Analysis Table.

Unknown
Color for unknown and monitor analyses

Blank
Color for all blank analysis types, e.g. blank_unknown, blank_air, etc

Air
Color for air analyses

Browser Loading

Auto Load
Load the browser search boxes, e.g. project, principal_investigator etc, ever time the browser is opened. Disable this option for speed and/or testing.

Load Prior Selection
Load the prior selection, e.g. select the previously selected project, principal_investigator etc, when the browser is opened.

3.2.3 Hardware

Pychron Proxy Server

1. Enabled

- Available Protocols *

A. ValveProtocol

B. FusionsCO2Protocol

C. FusionsDiodeProtocol

D. FusionsUVProtocol

E. FurnaceProtocol

3.2.4 Entry

Irradiations

1. Irradiation Prefix
2. Monitor Name. Sample name for neutron flux monitors
3. J Multiplier. J per hour, used to estimate J for a given irradiation chronology

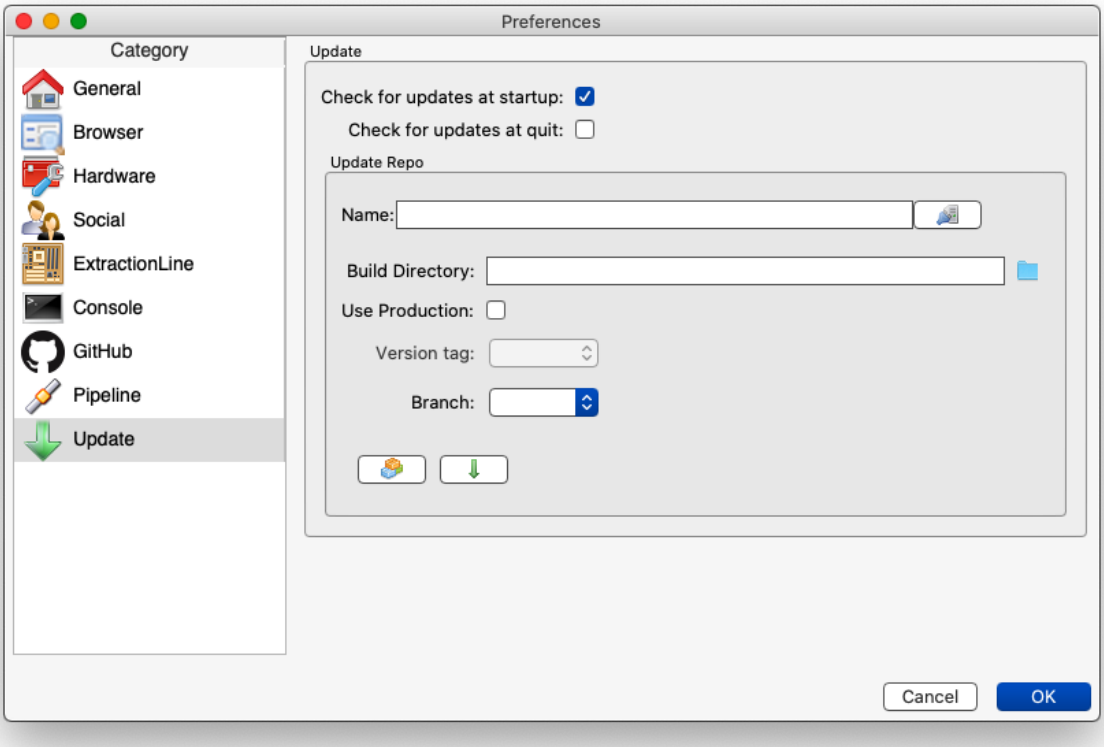
3.2.5 Loading

1. Output Directory

3.2.6 Pipeline

3.2.7 Update

1. Enable Update Plugin using the `initialization.xml` file or `Menu/Help/Edit Initialization`
2. Restart Pychron
3. Open Pychron's `Preferences` and goto the `Update` section
4. In the Update Repo section set the "Name" of Pychron fork you will be using. Each organization typically has its own fork. For example `NMBGMR/Pychron`.



5. After setting a valid name, test connection to the fork and load available tags and branches using the "Test Connection" button to the right of the "Name" entry field.
6. Set the build directory. This is the location on your computer where the code actually "lives". If your version of Pychron was installed using the installer script your should already have a build directory on your computer, typically `/Users/<username>/.pychron.0/pychron`
7. Set the Branch
8. Checkout the Branch
9. Update the Branch

3.2.8 DVC

Organization

1. Name. Name of the Organization/Group to save data
2. Default team.

Meta

3. Name. Name of the MetaData repository

Pychron DB

4. Kind. Type of Database. SQLite, MySQL, etc
5. Database Name. Name of the database
6. Favorites
7. Host. Host e.g 192.168.0.1

8. User. Username for accessing database 10 Password. Password for accessing database

11. Add. Add a new connection
12. Delete. Delete selected connection
13. Test Connection. Test the selected connection

3.2.9 Mass Spec

1. Use MassSpec

Connection

2. Database
3. Host
4. Name
5. Password

Isotope

6. Reference Isotope

Detector

7. Set By Isotope
8. Reference Detector

3.2.10 GitHub

Credentials

1. Username
2. Password
3. Token
4. Test Connection
6. Default Remote

3.2.11 GitLab

3.2.12 Social

Email

1. Host
2. User
3. Password
4. Port
5. Test Connection

3.3 Operation

3.3.1 Pychron and the Spectrometer

Peak Hop Definition

This section describes how to define a peak hopping sequence.

A peak hop sequence, `HOPS` , is a list of peak hops (i.e magnet moves).

```
HOPS=[('Ar40:H1:10,      Ar39:AX,      Ar36:CDD',       5, 1),
      ('Ar37:CDD',                                     5, 1)]
```

A `peak hop` is a list of `iso:detector` pairs plus two configuration values,

1. the number of counts at the current position
2. the settling time (s) after magnet positioning and before measurement starts.

The first `iso:detector` pair of each hop is used for positioning.

To specify a non-nominal deflection value for a detector use `iso:detector:deflection`

```
Ar40:H1:30
```

If no value is specified and the deflection value had been changed by a previous cycle then the deflection is set to the value stored in the spectrometer configuration file.

The following sequence,

```
('Ar40:H1,      Ar39:AX,      Ar36:CDD',      10, 1)
('Ar40:L2,      Ar39:CDD',                    20, 5)
```

translates to

1. position Ar40 on detector H1, wait 1s and record 10 H1,AX,and CDD measurements.
2. After 10 measurements position Ar40 on detector L2, wait 5s, then record 20 measurements.

Source Parameters

There are three ways in which pychron can interact with the spectrometers sources settings.

1. **Readonly** - Pychron only reads the sources parameters from the spectrometer. For each analysis these values are saved to the pychron database in the `meas_SpectrometerParametersTable`.

   !Pychron currently does not save the source values to the secondary database.!

2. **Read/Write** - Pychron sets the source parameters in the **measurement** script. By using a pyscript the user has fine-grained control of when and what values are set. See below for detailed description of setting source parameters in a pyscript.
3. **Config Write** - On startup Pychron sets the source parameters using values retrieved from a configuration file located at `setupfiles/spectrometer/config.py`

Setting Source parameters

Set source parameters individually by name

```
set_extraction_lens(103.4)
```

Set parameters using a configuration file located at `setupfiles/spectrometer/config.py`

```
set_source_parameters()
set_source_optics()
```

Values from the configuration file can be overwritten by specifying a parameter name and value

```
set_source_parameter(IonRepeller=134)
```

Detector Parameters

Interacting with the detector parameters is the same as with the source parameters. The deflection values are saved in the pychron database in the `meas_SpectrometerDeflectionsTable`

Reference

Pyscript Commands

```
set_ysymmetry(100)
set_zsymmetry(100)
set_zfocus(100)
set_extraction_lens(100)
set_deflection('H2',0)

# IonRepeller, ElectronVolts
set_source_parameters()

# ExtractionLens, YSymmetry, ZSymmetry, ZFocus
set_source_optics()
```

Config file Example

```
[SourceParameters]
ionrepeller = 0
electronvolts = 0

[SourceOptics]
ysymmetry = 0
zsymmetry = 0
zfocus = 0
extractionlens = 0

[Deflections]
h2 = 0
h1 = 0
ax = 0
l1 = 0
l2 = 0
cdd = 0

[CDDParameters]
operatingvoltage = 618.0
```

3.3.2 Pychron and the Database

3.3.3 Pychron and Valves

3.3.4 Pychron and a Laser

3.3.5 Experiments

This section describes how to write an experiment with Pychron. A Pychron `experiment` in Mass Spec parlance is a `Multiple Runs Sequence`.

Position Rules

The following is a list of rules for how a position entry is interpreted

1. 4 or p4 (Goto position 4)
2. 3,4,5 (Goto positions 3,4,5. Treat as one analysis)
3. 7-12 (Goto positions 7,8,9,10,11,12. Treat as individual analyses)
4. 7:12 (Same as #3)
5. 10:16:2 (Goto positions 10,12,14,16. Treat as individual analyses)
6. D1 (Drill position 1)
7. T1-2 (Goto named position T1-2 i.e transect 1, point 2)
8. L3 (Trace path L3)
9. 1-6;9;11;15-20 (Combination of rules 2. and 3. Treat all positions as individual analyses)

Here are a few examples of how Rule #9 is processed

```
user_input= 1-6;9

#resulting positions
positions= 1,2,3,4,5,6,9


user_input= 1-3;9;11-13

#resulting positions
positions= 1,2,3,4,5,6,9,11,12,13
```

The starting position, i.e 1 in the above case, can be greater than the end position i.e 6. If the start > end, positions will decrease from start to end

```
user_input= 9;6-1

#resulting positions
positions= 9,6,5,4,3,2,1
```

If auto-increment position is enabled pychron starts incrementing from the last value and follows the same pattern as the rule.

```
user_input= 1-6;9

#resulting auto-incremented positions
positions= 10-15;18


user_input= 1-6

#resulting auto-incremented positions
positions= 7-12
```

3.3.6 Scripting

General

```
#=============================================================
def main():
    info('this is an info message')
    acquire('pipette')
    info('aquired resourece - pipette')
    sleep(15)
    release('pipette')
    begin_interval(120)
    #do some stuff ...
    complete_interval() #wait until 120 seconds have passed
#=============================================================
```

functions

info(*msg*)

add info `msg` to the log

acquire(*resource*)

reserve the `resource` for use. blocks other scripts from using `resource` until *release()* is called.

release(*resource*)

release `resource` so other scripts can use it.

sleep(*seconds*)

sleep for `seconds`. if `seconds>5` a timer will appear. decimal seconds are allowed e.g `sleep(0.5)`

gosub(*path_to_script*)

execute a pyscript located at `path_to_script`. `path_to_script` is relative to the current script. e.g `gosub(commonscripts/fuse.py)`. `commonscripts` must be a directory in the same directory this script is saved in.

begin_interval(*timeout*)

start an interval. if `timeout>5` a timer will appear.

complete_interval()

wait unit `timeout` has elapsed

Measurement Scripts

Note: Please see *Examples* for more a comprehensive set of examples

Functions

class pychron.pyscripts.measurement_pyscript.**MeasurementPyScript**(**args: Any*, ***kwargs: Any*)

MeasurementPyScripts are used to collect isotopic data

activate_detectors(**dets*, ***kw*)

set the active detectors

Parameters
dets – list

coincidence()

Do a coincidence scan. Peak center all active detectors simulatenously. calculate required deflection corrections to bring all detectors into coincidence

property eqtime

Property. Equilibration time. Get value from `AutomatedRun`.

Returns
float, int

equilibrate(*eqtime=20*, *inlet=None*, *outlet=None*, *do_post_equilibration=True*, *close_inlet=True*, *delay=3*)

equilibrate the extraction line with the mass spectrometer

inlet or outlet can be a single valve name or a list of valve names.

```
'A', ('A','B'), ['A','B'], 'A,B'
```

Parameters

- **eqtime** – int, equilibration duration in seconds
- **inlet** – str, tuple or list, inlet valve
- **outlet** – str, tuple or list, ion pump valve
- **do_post_equilibration** – bool
- **close_inlet** – bool
- **delay** – int, delay in seconds between close of outlet and open of inlet

generate_ic_mftable(*detectors*, *refiso='Ar40'*, *peak_center_config=''*, *n=1*, *calc_time=False*)

Generate an IC MFTable. Use this when doing a Detector Intercalibration. peak centers the `refiso` on a list of `detectors`. MFTable saved as ic_mftable

cancel script if generating mftable fails

Parameters

- **detectors** (*list*) – list of detectors to peak center
- **refiso** (*str*) – isotope to peak center

peak_center(*detector=''*, *isotope=''*, *integration_time=1.04*, *save=True*, *calc_time=False*, *directions='Increase'*, *config_name='default'*)

Calculate the peak center for `isotope` on `detector`.

Parameters

- **detector** – str
- **isotope** – str
- **integration_time** – float
- **save** – bool

position_magnet(*pos*, *detector='AX'*, *use_dac=False*, *for_collection=True*)

Parameters

- **pos** (*str*) – location to set magnetic field
- **detector** – detector to position `pos`
- **use_dac** (*bool*) – is the `pos` a DAC voltage

examples:

```
position_magnet(4.54312, use_dac=True) # detector is not relevant
position_magnet(39.962, detector='AX')
position_magnet('Ar40', detector='AX') #Ar40 will be converted to 39.962 use␣
↪mole weight dict
```

post_equilibration(*block=False*)

Run the post equilibration script.

reset(*arun*)

Reset the script with a new automated run

Parameters

arun (`AutomatedRun`) – A new `AutomatedRun`

set_baseline_fits(**fits*)

set baseline fits for detectors

Parameters

fits –

set_fits(**fits*)

set time vs intensity regression fits for isotopes

Parameters

fits – str, list, or tuple

set_integration_time(*v*)

Set the integration time

Parameters

v (*float*) – integration time in seconds

set_time_zero(*offset=0*)

set the time_zero value. add offset to time_zero

```
T_o= ion pump closes
offset seconds after T_o. define time_zero

T_eq= inlet closes
```

sink_data(*n=100*, *delay=1*, *calc_time=False*)

@param n: number of measurements @param period: delay between measurements @param calc_time: @return:

property time_zero_offset

Property. Substract `time_zero_offset` from time value for all data points

Returns
float, int

property truncated

Property. True if run was truncated otherwise False

Returns
bool

property use_cdd_warming

Property. Use CDD Warming. Get value from `AutomatedRunSpec`

Returns
bool

whiff(*ncounts=0*, *conditionals=None*)

Do a whiff measurement.

Whiff's are quick measurements with conditionals. use them to take action at the beginning of a measurement. For example do a whiff to determine if intensity to great.

Parameters

- **ncounts** – int
- **conditionals** – list of dicts

Variables

truncated

eqtime

use_cdd_warming

Multicollect

```
#!Measurement

#counts
MULTICOLLECT_COUNTS= 4

#baselines
BASELINE_COUNTS= 2
```

(continues on next page)

(continued from previous page)

```
BASELINE_DETECTOR= 'H1'
BASELINE_MASS= 39.5
BASELINE_BEFORE= False
BASELINE_AFTER= True

#peak center
PEAK_CENTER_BEFORE= False
PEAK_CENTER_AFTER= False
PEAK_CENTER_DETECTOR= 'H1'
PEAK_CENTER_ISOTOPE= 'Ar40'

#equilibration
EQ_TIME= 2
INLET= 'R'
OUTLET= 'S'
DELAY= 3.0
TIME_ZERO_OFFSET=5

ACTIVE_DETECTORS=('H1','AX')
FITS=[
      ((0,5),('linear', 'linear')),
      ((5,None),('linear', 'parabolic'))
      ]
USE_FIT_BLOCKS=True

ACTIONS= [(False,('age','<',10.6,20,10,'',False)),
         ]

TRUNCATIONS = [(False, ('age','<',10.6,20,10,)),
               ]

TERMINATIONS= [(False, ('age','<',10.6,20,10))
               ]

def main():
    #this is a comment
    '''
        this is a multiline
        comment aka docstring
    '''
    #display information with info(msg)
    info('unknown measurement script')

    #set the spectrometer parameters
    #provide a value
    set_source_parameters(YSymmetry=10)

    #or leave blank and values are loaded from a config file (setupfiles/spectrometer/
↪config.cfg)
    set_source_optics()

    #set the cdd operating voltage
```

(continues on next page)

(continued from previous page)

```
    set_cdd_operating_voltage(100)

    if PEAK_CENTER_BEFORE:
        peak_center(detector=PEAK_CENTER_DETECTOR,isotope=PEAK_CENTER_ISOTOPE)

    #open a plot panel for this detectors
    activate_detectors(*ACTIVE_DETECTORS)

    if BASELINE_BEFORE:
        baselines(ncounts=BASELINE_COUNTS,mass=BASELINE_MASS, detector=BASELINE_DETECTOR)
    #set default regression
    regress(*FITS)

    #position mass spectrometer
    position_magnet('Ar40', detector='H1')

    #gas is staged behind inlet

    #post equilibration script triggered after eqtime elapsed
    #equilibrate is non blocking
    #so use either a sniff of sleep as a placeholder until eq finished
    equilibrate(eqtime=EQ_TIME, inlet=INLET, outlet=OUTLET)

    for use,args in ACTIONS:
        if use:
            add_action(*args)

    for use,args in TRUNCATIONS:
        if use:
            add_truncation(*args)

    for use, args in TERMINATIONS:
        if use:
            add_termination(*args)

    #equilibrate returns immediately after the inlet opens
    set_time_zero(offset=TIME_ZERO_OFFSET)

    sniff(EQ_TIME)

    #multicollect on active detectors
    multicollect(ncounts=MULTICOLLECT_COUNTS, integration_time=1)

    clear_conditionals()

    if BASELINE_AFTER:
        baselines(ncounts=BASELINE_COUNTS,mass=BASELINE_MASS, detector=BASELINE_DETECTOR)
    if PEAK_CENTER_AFTER:
        peak_center(detector=PEAK_CENTER_DETECTOR,isotope=PEAK_CENTER_ISOTOPE)


    #WARM CDD
```

(continues on next page)

(continued from previous page)

```
    warm_cdd()

    info('finished measure script')

def warm_cdd():
    '''
        1. blank beam
        2. move to desired position
        3. unblank beam
    '''
    if not is_last_run():
        set_deflection('CDD',2000)

        position_magnet(28.04, detector='H1')
        #or
        #position_magnet(5.00, dac=True)

        #return to config.cfg deflection value
        set_deflection('CDD')


↪#========================EOF==============================================================
```

Peak Hop

For a detailed description of a Peak Hop sequence see hops. Instead of defining the peak hop sequence directly within the measurement script it is cleaner to load the sequence from a separate file using `load_hops`

```
#!Measurement

#==============================================================================
# parameter definitions
#==============================================================================
#multicollect
MULTICOLLECT_COUNTS       = 1003
MULTICOLLECT_ISOTOPE      = 'Ar40'
MULTICOLLECT_DETECTOR     = 'H1'

#baselines
BASELINE_COUNTS           = 10
BASELINE_DETECTOR         = 'H1'
BASELINE_MASS             = 39.5
BASELINE_BEFORE           = False
BASELINE_AFTER            = True

#peak center
PEAK_CENTER_BEFORE        = False
PEAK_CENTER_AFTER         = False
PEAK_CENTER_DETECTOR      = 'H1'
PEAK_CENTER_ISOTOPE       = 'Ar40'
```

(continues on next page)

(continued from previous page)

```
#equilibration
EQ_TIME                 = 1.0
EQ_INLET                = 'S'
EQ_OUTLET               = 'O'
EQ_DELAY                = 3.0

#PEAK HOP
USE_PEAK_HOP            = True
NCYCLES                 = 3
BASELINE_NCYCLES        = 3


"""
    HOPS definition

    HOPS is a list of peak hops.
    a peak hop is a list of iso:detector pairs plus the number of counts to measure
    for this hop. The first iso:detector pair is used for positioning.

    added rev 1665
        specify a deflection for the iso:det pair
        Ar40:H1:30
        if no value is specified and the deflection value had been changed by a previous␣
↪cycle
        then set the deflection to the config. value


    ('Ar40:H1,      Ar39:AX,      Ar36:CDD',      10)
    ('Ar40:L2,      Ar39:CDD',                    20),
    means position Ar40 on detector H1 and
    record 10 H1,AX,and CDD measurements. After 10 measurements
    position Ar40 on detector L2, record 20 measurements.

    repeat this sequence NCYCLES times

"""

HOPS=[('Ar40:H1:10,      Ar39:AX,      Ar36:CDD',       5, 1),
      #('Ar40:L2,      Ar39:CDD',                    5, 1),
      #('Ar38:CDD',                                  5, 1),
      ('Ar37:CDD',                                   5, 1),
      ]

#Detectors
ACTIVE_DETECTORS        = ('H2','H1','AX','L1','L2','CDD')
FITS                    = ('average','parabolic','parabolic','parabolic','parabolic',
↪'linear')
#=============================================================================
#
#=============================================================================
```

(continues on next page)

(continued from previous page)

```
def main():
    #this is a comment
    '''
        this is a multiline
        comment aka docstring
    '''
    #display information with info(msg)
    info('unknown measurement script')

    #set the spectrometer parameters
    #provide a value
    #set_source_parameters(YSymmetry=10)

    #or leave blank and values are loaded from a config file (setupfiles/spectrometer/
↪config.cfg)
    #set_source_optics()

    #set the cdd operating voltage
    #set_cdd_operating_voltage(100)

    if PEAK_CENTER_BEFORE:
        peak_center(detector=PEAK_CENTER_DETECTOR,isotope=PEAK_CENTER_ISOTOPE)

    activate_detectors(*ACTIVE_DETECTORS)

    if BASELINE_BEFORE:
        baselines(ncounts=BASELINE_COUNTS,mass=BASELINE_MASS, detector=BASELINE_DETECTOR)
    #set default regression
    regress(*FITS)

    #position mass spectrometer even though this is a peak hop so an accurate sniff/eq␣
↪is measured
    position_magnet(MULTICOLLECT_ISOTOPE, detector=MULTICOLLECT_DETECTOR)

    #gas is staged behind inlet

    #post equilibration script triggered after eqtime elapsed
    #equilibrate is non blocking
    #so use either a sniff of sleep as a placeholder until eq finished
    equilibrate(eqtime=EQ_TIME, inlet=EQ_INLET, outlet=EQ_OUTLET)

    #equilibrate returns immediately after the inlet opens
    set_time_zero()

    sniff(EQ_TIME)

    if USE_PEAK_HOP:
        hops=load_hops('hops/hop.txt')
        info(hops)

        peak_hop(ncycles=NCYCLES, hops=HOPS)
    else:
```

(continues on next page)

(continued from previous page)

```
        #multicollect on active detectors
        multicollect(ncounts=MULTICOLLECT_COUNTS, integration_time=1)

    if BASELINE_AFTER:
        if USE_PEAK_HOP:
            peak_hop(ncycles=BASELINE_NCYCLES, hops=BASELINE_HOPS, baseline=True)
        else:
            baselines(ncounts=BASELINE_COUNTS,mass=BASELINE_MASS,
                      detector=BASELINE_DETECTOR)
    if PEAK_CENTER_AFTER:
        peak_center(detector=PEAK_CENTER_DETECTOR,isotope=PEAK_CENTER_ISOTOPE)

    info('finished measure script')



↪#========================EOF==============================================
```

Annotated Multicollect

```
#!Measurement

def main():
    '''
    display a message in the Experiment Executor and add to log
    '''
    info('example measurement script')

    '''
    define which detectors to collect with
    '''
    activate_detectors('H1','AX','L1','L2','CDD')

    '''
    set the fit types
    if a single value is supplied it's applied to all active detectors
    '''
    regress('parabolic')

    #or a list of fits the same len as active detectors is required
    regress('parabolic','parabolic','linear','linear','parabolic')

    '''
    position the magnet

    position_magnet(4.54312, dac=True) # detector is not relevant
    position_magnet(39.962, detector='AX')
    position_magnet('Ar40', detector='AX') #Ar40 will be converted to 39.962 use mole␣
↪weight dict
    '''
```

(continues on next page)

(continued from previous page)

```
#position isotope Ar40 on detector H1
position_magnet('Ar40', detector='H1')

'''
sniff and split

    1. isolate sniffer volumne
    2. equilibrate
    3. sniff gas
    4. test condition

gas is staged behind inlet
isolate sniffer volume
'''
close('S')
sleep(1)

'''
equilibrate with mass spec
set outlet to make a static measurement

set do_post_equilibration to False so that the gas in the microbone
is not pumped away
'''
equilibrate(eqtime=20, inlet='R', do_post_equilibration=False)
set_time_zero()

#display pressure wave
sniff(20)

#define sniff/split threshold
sniff_threshold=100

#test condition
if get_intensity('H1')>sniff_threshold:
    extraction_gosub('splits:jan_split')
    '''
    extraction_gosub is same as
    gosub('splits:jan_split', klass='ExtractionLinePyScript')
    '''

'''
gas has been split down and staged behind the inlet
post equilibration script triggered after eqtime elapsed
equilibrate is non-blocking so use a sniff or sleep as a placeholder
e.g sniff(<equilibration_time>) or sleep(<equilibration_time>)
'''
equilibrate(eqtime=5,inlet='R', outlet='V')
set_time_zero()

#sniff the gas during equilibration
```

(continues on next page)

(continued from previous page)

```
    sniff(5)
    sleep(1)

    '''
    Set conditionals

    order added defines conditional precedence.
    conditionals after the first true conditional are NOT evaluated

    terminate if age < 10000 ma after 5 counts, check every 2 counts
    terminate means do not finish measurement script and immediately execute
    the post measurement script
    '''
    add_termination('age','<',10000, start_count=5, frequency=2)

    '''
    truncate means finish the measurement block immediately and continue to next
    command in the script
    '''
    add_truncation('age','>',10.6, start_count=20, frequency=10)

    '''
    use add_action to specify an action to take for a given conditional

    action can be a code snippet 'sleep(10)', 'gosub("example_gosub")' or
    a callable such as a function or lambda

    the resume keyword (default=False) continues measurement after executing
    the action
    '''
    add_action('age','>',10.6, start_count=20, frequency=10,
               action='sleep(10)')
    add_action('age','<',10000, start_count=5, frequency=2,
                  action=func)
    add_action('age','<',10000, start_count=5, frequency=2,
                  action='sleep(7)',
                  resume=True
                  )
    add_action('age','<',10000, start_count=5, frequency=2,
                  action='gosub("snippet")')

    #measure active detectors for ncounts
    multicollect(ncounts=50, integration_time=1)

    '''
    clear the conditionals when measuring baseline
    also have oppurtunity to add new conditionals
    '''
    clear_conditionals()

    #multicollect baselines for ncounts
    baselines(ncounts=5,mass=39.5)
```

(continues on next page)

(continued from previous page)

```
    info('finished measure script')

def func():
    info('action performed')

#============================EOF=============================================
```

Writing Extraction Line Scripts

Gas handling is controlled by three separate scripts

1. Extraction
2. Post Equilibration (optional)
3. Post Measurement (optional)

Each type of scripts resides in a dedicated directory. e.g scripts/extraction/unknown_diode

The extraction script is responsible for staging the gas for the mass spectrometer. The script should end just prior to opening the inlet. (The inlet and ion pump are controlled by the measurement script).

The post equilibration script is an optional script that is triggered by an `equilibrate` command in the measurement script.

The post measurement script is an optional script that runs after the measurement script completes. Commonly this script only opens the mass spectrometer ion pump because the post equilibration script should have already pumped out analysis chambers.

Exposed names

- `extract_value` = the extraction device setpoint in `extract_units`
- `extract_units` = units for setting the extraction device e.g watts or percent
- `ramp_rate` = rate to increase the setpoint from 0 to `extract_value` in `extract_units` per second
- `duration` = period to maintain `extract_value` in seconds
- `cleanup` = delay for gettering
- `position` = list of hole numbers or `X,Y[,Z]` coordinates
- `disable_between_analyses` = for multiple position analyses disable extraction device while moving

Obama Unknown CO2

```
'''
this script will not work well for a multiple position analysis
because move_to_position([1,2,3]) with move to holes 1,2,3 before
firing the laser
'''
MOVE_DURATION=15
def main():
```

(continues on next page)

(continued from previous page)

```
    info('Obama unknown laser analysis')

    gosub('obama:PrepareForCO2Analysis')

    if analysis_type=='blank':
        info('is blank. not heating')

        #wait move_duration seconds to have equal blanks
        sleep(MOVE_DURATION)

    else:
        info('move to position {}'.format(position))
        begin_interval(MOVE_DURATION)

        '''
        position can be a single value or a list
        all moves will take place before the laser is enabled and firing
        see Single or Multiple Position script below if you plan on doing
        multiple position analyses
        '''
        move_to_position(position)
        complete_interval()
        if ramp_rate>0:
            '''
            style 1.
            '''
#             begin_interval(duration)
#             info('ramping to {} at {} {}/s'.format(extract_value, ramp_rate, extract_
↪units)
#             ramp(setpoint=extract_value, rate=ramp_rate)
#             complete_interval()
            '''
            style 2.
            '''
            elapsed=ramp(setpoint=extract_value, rate=ramp_rate)
            sleep(min(0, duration-elapsed))

        else:
            info('set extract to {}'.format(extract_value))
            extract(extract_value)
            sleep(duration)

    if not analysis_type=='blank':
        end_extraction()
    sleep(cleanup)
```

Single or Multiple Position

```
'''
    this script should work for single of multiple position analyses
'''

def main():
    info('Obama unknown laser analysis')

    gosub('obama:PrepareForCO2Analysis')

    if analysis_type=='blank':
        info('is blank. not heating')
    else:

        acquire('ObamaCO2Flag')
        '''
        this is the most generic what to move and fire the laser
        position is always a list even if only one hole is specified
        '''
        for pi in position:
            info('move to position {}'.format(pi))
            move_to_position(pi)
            do_extraction()
            sleep(duration)
            if disable_between_positions:
                end_extract()

    end_extract()
    sleep(cleanup)

    '''
        release the ObamaCO2Flag at the end of post equilibration
        using:
            release('ObamaCO2Flag')
    '''


def do_extraction():
    if ramp_rate>0:
        '''
        style 1.
        '''
 #       begin_interval(duration)
 #       info('ramping to {} at {} {}/s'.format(extract_value, ramp_rate, extract_units)
 #       ramp(setpoint=extract_value, rate=ramp_rate)
#        complete_interval()
        '''
        style 2.
        '''
        elapsed=ramp(setpoint=extract_value, rate=ramp_rate)
        sleep(min(0, duration-elapsed))
```

(continues on next page)

(continued from previous page)

```
    else:
        info('set extract to {}'.format(extract_value))
        extract(extract_value)
        sleep(duration)
```

PrepareForCO2Analysis

```
'''
mass spec equivalent
Message "Preparing for CO2 Analysis"
Close "Mass Spec Inlet"
Open "MS Ion Pump"
Close "Bone to Minibone"
Open "Bone to Turbo"
Open "Bone to Getter GP-50"
Close "Bone to Diode Laser"
Close "CO2 Laser to Jan"
Open "CO2 Laser to Obama"
Open "Bone to CO2 Laser"
'''
def main():
    info('Preparing for CO2 Analysis')
    close(description='Obama Inlet')
    open(description='Obama Ion Pump')
    close(description='Bone to Minibone')
    open(description='Bone to Turbo')
    open(description='Bone to Getter GP-50')
    close(description='Bone to Diode Laser')
    close(description='CO2 Laser to Jan')
    open(description='CO2 Laser to Obama')
    open(description='Bone to CO2 Laser')
```

Writing Bakeout Scripts

Here's how to write a bakeoutscript

1. Launch BakeoutManager (Bo on the dock)
2. Open the script editor (`Edit scripts` button). An empty script is opened as a default
3. Use `Open` to open an existing script or just start writing one from scratch
4. Write your script. see below
5. You may check the syntax of your script at any point by hitting `Test`. A dialog will pop up saying if there was a problem and what it was or if the script passed with no errors. The scripts syntax is also automatically checked before saving.
6. Use `Save` and `Save As` in the normal manner. script files should end with **.py** . if the file ending is omitted a **.py** is appended automatically
7. Close the script editor

Here is an example script

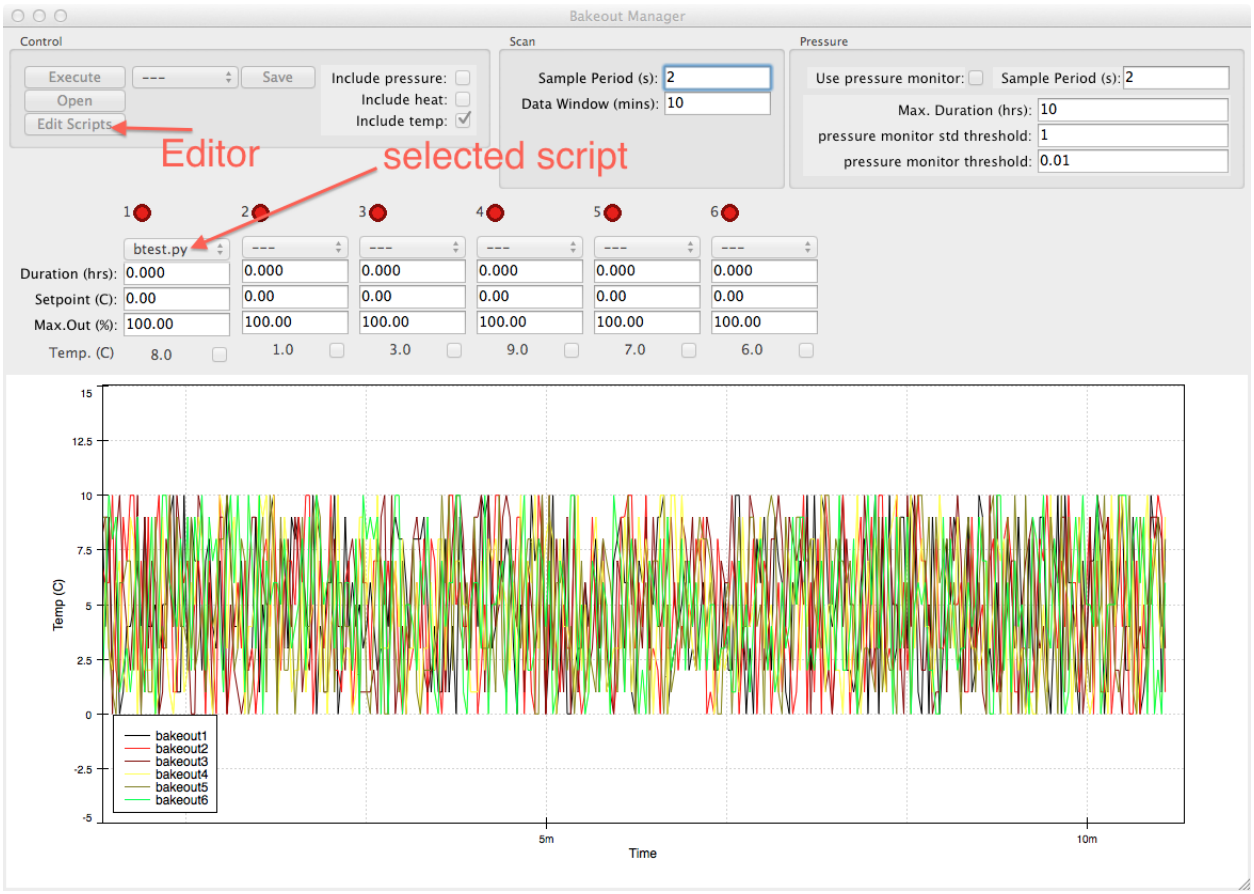
```
#================================================================================
#this is a comment in the script. use it at your leisure
#the following line def main(): is required and the entry point for the script
def main():
    #this is the body of the main function
    #it should be one tab in
    ramp(150,100,start=45)  # comments can go on the same line as functions
    setpoint(150,4)
    ramp(0,-150)
#end of script
#================================================================================
```

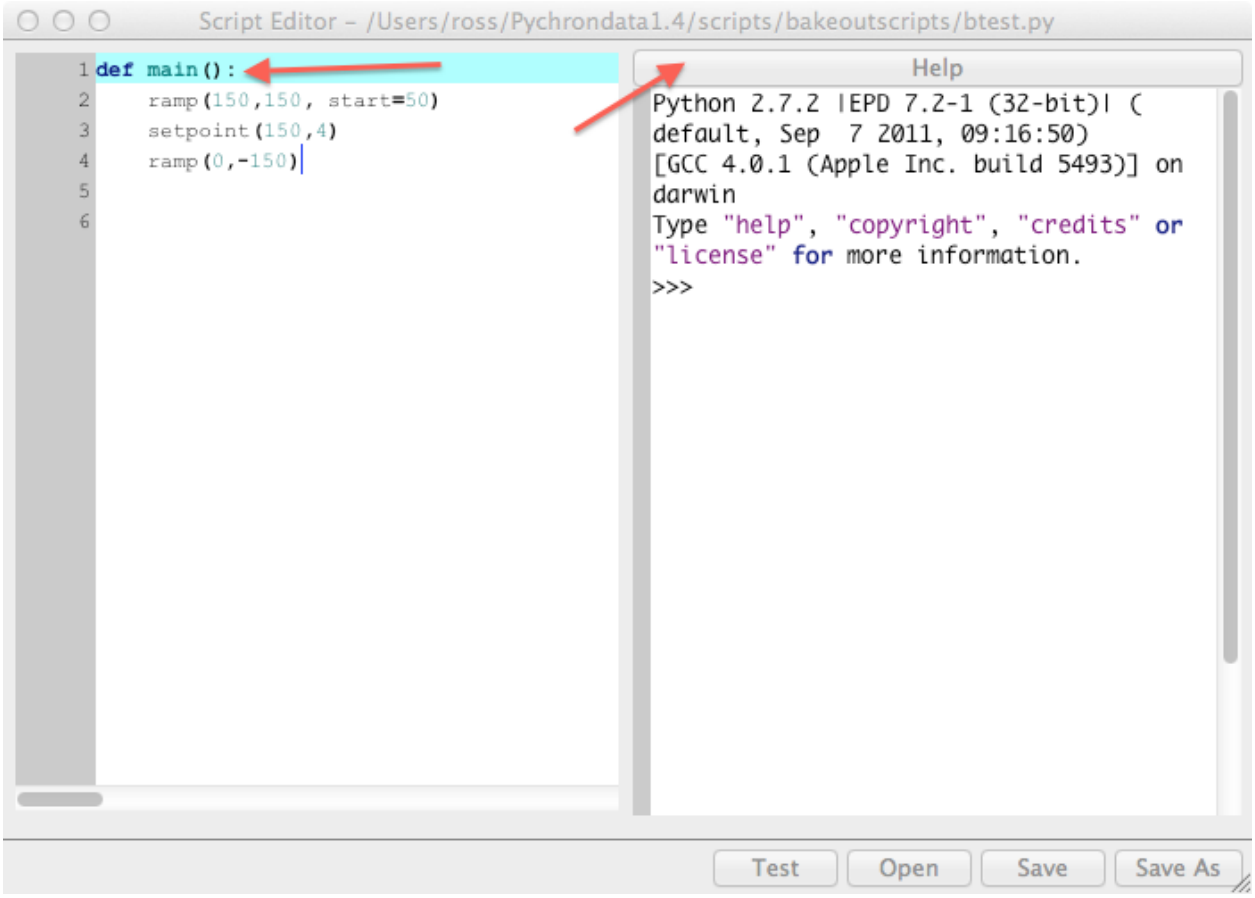
Lets break it down line by line.

- A comment can be added anywhere using the # character
- `def main():` defines a required function called "main" that will be executed when the script is run. **The main function is necessary** and is added to an empty script by default.
- The function `ramp(150,100,start=45)` sets the controller to 45 C, then increases the temperature setting to 150 C at a rate of 100 C/hr
- The function `setpoint(150,4)` sets the controller to a temperature of 150 C and holds it for 4 hours
- `ramp(0,-100)` decreases the temperature setting from the current temperature to 0 C at a rate of -100 C/hr.
- Notice that the `start` parameter of the *ramp()* function is optional. If omitted the controller's current temperature is used as the starting point

To activate a script for a given controller use the drop down menu located above the `Duration` box

To execute the bakeout hit `Execute`. `Stop` to Stop.


Bakeout Manager
Control
Execute
Open
Edit Scripts
Save
Include pressure:
Include heat:
Include temp:
Scan
Sample Period (s): 2
Data Window (mins): 10
Pressure
Use pressure monitor:
Sample Period (s): 2
Max. Duration (hrs): 10
pressure monitor std threshold: 1
pressure monitor threshold: 0.01
Editor
selected script
1
2
3
4
5
6
btest.py
Duration (hrs): 0.000
Setpoint (C): 0.00
Max.Out (%): 100.00
Temp. (C) 8.0
1.0
3.0
9.0
7.0
6.0
Temp (C)
Time
5m
10m
bakeout1
bakeout2
bakeout3
bakeout4
bakeout5
bakeout6



bakeout functions

ramp(*setpt*, *rate*[, *start=None*, *period=60*])

ramp controller's setpoint from `start` to `setpt` at a rate of `rate` C\hr. if `start=None` then the controller's current `temperature` is used. `period` defines seconds between setpoint updates. e.g. `period=30` sets the controllers setpoint every 30 seconds

setpoint(*temperature*, *duration*)

set controller's setpoint to `temperature` for `duration` hours

Examples

Helix

Helix Examples

Helix Extraction

Helix Measurement

Helix Post Equilibration

Helix Post Measurement

Argus

Argus Examples

Argus Extraction

Argus Measurement

Argus Post Equilibration

Argus Post Measurement

3.3.7 Conditionals

Conditionals are the pychron mechanism used to take action if a given condition evaluates to True, e.g. age>2.0. There are three levels of Conditionals 1) System 2) Queue 3) Run. System conditionals are applied to every run of every experiment. A typical system conditional is to cancel the experiment if the CDD is not on/enabled. Queue Conditionals can be specified per experiment queue. Queue conditionals are applied to all runs in the experiment. Run conditionals are specified on a per run basis. There is also multiple types of conditionals. System and Queue conditionals fall into five categories 1) Actions, 2) Pre Run Terminations 3) Truncations 4) Terminations and 5) Post Run Terminations. Run conditionals have all the same categories except for Pre and Post Terminations.

Truncations conditionals truncate the current run (curtail in MassSpec parlance) and do an abbreviated baseline measurement. Terminations cancel the experiment immediately. Actions allow you to specify a action to take such as run a blank, etc. Custom actions can be programmed using pyscripts.

Levels

Name	Description
System	Executed for all runs in all experiment queues
Queue	Executed for all runs in this experiment queue
Run	Executed for only this run

Conditional Types

Name	Description
Action	
Truncation	
Termination	
Cancelation	
PreRunTermination	
PostRunTermination	

Functions/Modifiers

Note: User supplies the values contained in brackets (i.e. [ ])

Name	Description	Example
min([value])		min(Ar40)
max([value])		max(Ar40)
average([value])		average(Ar40)
slope([value])		slope(Ar40)
[name].current	get the last measured intensity for an isotope	Ar40.current
[name].cur	same as current	Ar40.cur
device.[name]	get a device value	device.pneumatics
[controller].[gauge].pressure	get a pressure from an controller	bone.ig.pressure
[name].deflection	get detector's deflection	H1.deflection
[isotope]/[isotope]	get a baseline corrected ratio	Ar40/Ar36
between([value], [v1],[v2])	check if a value is between v1,v2	between(max(Ar40), 10, 100)
not	invert logic	not Ar40<100

Writing Conditionals

There is a primitive GUI for editing AutomatedRun Conditionals. You can probably get away with using the GUI 95% of the time, however, there are a few cases that are not currently implemened in the GUI, but nevertheless maybe used by editing the specific conditionals file. All conditional files use standard yaml syntax. Here is an example system_conditionals.yaml

```
actions: []
pre_run_terminations:
 - check: CDD.inactive
 - check: CDD.deflection==2000
 - check: bone.ig.pressure > 1e-8
 - check: microbone.ig.pressure > 1e-8
 - check: device.pneumatics < 10
post_run_terminations:
 - check: Ar40 < $MIN_INTENSITY
   analysis_types:
    - air
    - cocktail
terminations: []
truncations: []
```

Notice the special syntax used Post Run Termination (Ar40 < $MIN_INTENSITY). This is the interpolation/templating syntax. The interpolation syntax ($VARIABLE_NAME) is used to define the conditional's sentinel values at runtime. For instance the above post run termination conditional terminates the experiment if an Air's Ar40 intensity is below a certain threshold. The reason for interpolating the minimum intensity is that this value is dependent on the extraction script, e.i. either a full air shot or a sniff air. To specify an interpolatable value simply assign the variable in the extraction scripts metadata docstring.

```
'''
sensitivity_multiplier: 0.5
```

(continues on next page)

(continued from previous page)

```
modifier: 1
MIN_INTENSITY: 10
'''
def main():
    info('Jan Air Script')
```

In this case, the Post Run termination conditional would trip if the Ar40 intensity was less the 10 fA.

3.4 Pipeline

The Pipeline (aka Pipeline task) is your go to location for reducing your and producing figures and tables.

3.4.1 Recall

Definitions

I = T-zero intercept
I-Bs = I - Baseline
I-Bs-Bk = I - Baseline - Blank
S*IC = (I-Bs-Bk)*ICFactor
S*IC*DC = (I-Bs-Bk)*ICFactor*DecayFactor

IFC = Interference Corrected Values

Ar40 = Ar40_measured - Ar40_K
Ar39 = Ar39_K_decay_corrected
Ar38 = Ar38
Ar37 = Ar37_measured*DecayFactor
Ar36 = Ar36_Atm

3.4.2 Data

CSV

required columns: runid, age, age_err optional columns: group, aliquot

```
runid, age, age_error
SampleA, 10, 0.24
SampleB, 11, 0.32
SampleC, 10, 0.40
```

3.4.3 Fit

Define Equilibration

Isotope Evolutions

Blanks

IC Factors

Flux

3.4.4 Pipeline

Ideogram

Plotting ideograms of attributes other than Age

Available attributes

Name	Description
Age	
Ar40*/Ar39K	F value
Ar40/Ar36	Blank,Baseline,DetectorCorrected

Isochron

Spectrum

Series

3.4.5 Tagging

There are two types if tagging in Pychron

1. *Analysis Tagging*
2. *Data Reduction Tagging*

Analysis Tagging

Analysis tagging is used to tag individual analyses and specify how pychron should handle the analysis when processing and plotting. There are two generic tags, **OK** and **Invalid**. **OK** is the default tag for an analysis. It indicates that the analysis is acceptable a should be included in all processes. **Invalid** indicates that the analysis is *bad* and should not be hidden from the browser and plotting routines

---

Note: The **Browser** includes a checkbox to include **Invalid** analyses in the results. By default this it is set to False

---

In addition to the two generic tags, users have to option to create there own tags. This user defined tags can have any name and also come with four options for handling the plotting behavior. These options are

1. *omit_ideo*

2. *omit_spec*
3. *omit_iso*
4. *omit_series*

These options indicate whether the analysis should be excluded from the plot. Excluded in this context does not mean *not visible*, rather the analysis is given a special marker indicating it is not used in the calculations. For example omitted analyses in an ideogram are displayed as red open squares and are not used in the calculation of the mean age.

To add an analysis tag to one or more analyses

1. **Select a set of analyses using one of the following steps**
    a. select analyses from browser
    b. select analyses from Unknowns
    c. graphically select analyses on figure
2. Go to *Data/Tag*
3. Select existing tag or make a new one
4. (optional) fine-tune analysis select
5. Click *OK* to apply of *Cancel* to cancel

Data Reduction Tagging

Data reduction tagging is used to preserve the state of a set of analyses. This feature is slightly analogous to the branching data model used in software development, namely Git. For example, say you would like to reduce a set of data with all parabolic fits and then compare it to all linear fits. Data reduction tagging allows the user to save each data reduction session, one for parabolic fits and one for linear, and easily toggle between the two sets.

To add a data reduction tag

1. Perform the desired data reduction
2. Selected a set of analyses in the browser or Unknowns pane
3. Go to *Data/Data Reduction Tag*
4. Give the tag a name
5. Click *OK* to apply of *Cancel* to cancel

To toggle between data reduction tags

1. Go to *Data/Set Data Reduction Tag*
2. Select the desired tag from the table. You can filter by tag name and/or user
3. Click *OK* to apply of *Cancel* to cancel

Note: If you select a set of analyses prior to step 1. pychron will only display the data reduction tags which contain the selected set of analyses

---

3.4.6 Sharing Changes

Problem. You've made changes on Computer A and want to see those changes on Computer B.

Solution. Share you changes.

Sharing changes is the process of uploading (`push`) your local changes (Data reduction done on Computer A) to an accessible location (e.i. "The Cloud", GitHub). Once the changes are shared other instances of pychron will automatically download (`pull`) the changes to the current computer.

How to share changes

1. Automatically. If you've made changes and close the Pipeline Window pychron will ask

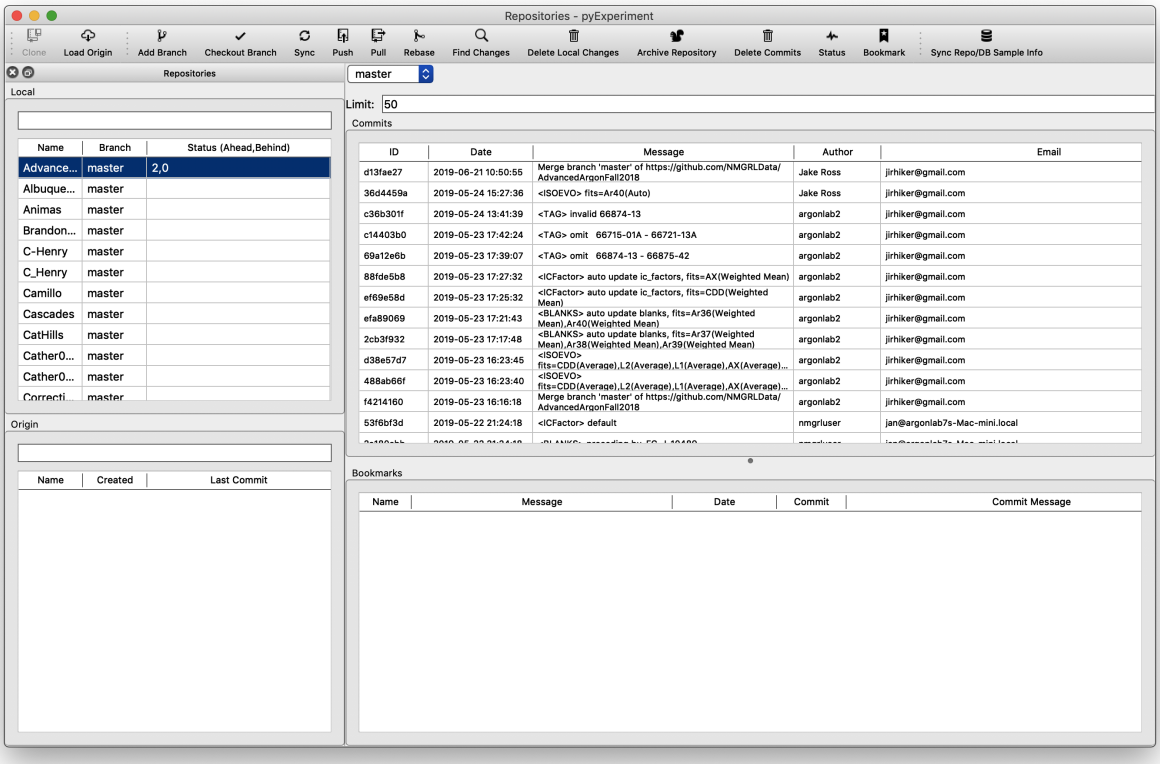
   "You have changes to analyses. Would you like to share them?"

   There are cases where pychron asks simply

   "Would you like to share your changes?"

   Answering **Yes** to either of these will push (upload) your current changes to `github .com/ <your_organizations_data>/<repo>`

2. Manually. Use `MenuBar/View/Repositories` to open the window used to interact with pychron's data repositories.



Note: Many tasks can be accomplished from the "Repositories" window. For now we will only cover sharing your changes.

1. On the LHS, in the "Local" group, select a repository from the list.
2. Click "Find Changes" on the toolbar

Note: pychron is examining your local version of the repository and comparing it to the version stored on github.com. The results are presented as <Number of changes ahead>, <Number of changes behind>. For example if the repository's status is 2,0 then your local version has 2 changes that don't exist in the github.com version. You can "Push" to share your 2 changes with github. If the status is 0,3 then your local version is missing 3 changes. You can "Pull" to update your version to the github version.

---

3. Push changes. If your status was X,0 then there is little chance of an issue when pushing. If your status is 0,Y pushing will do nothing because you have no changes to share. If your status is X,Y the process can get tricky. If you are comfortable with git proceed with sync, rebase, or via command line. Otherwise contact Pychron Labs.

3.4.7 Script

Access the pipelines state using *state*

e.g.

```
unknown_analyses = state.unknowns
```

for convenience the unknowns are exposed directly via *unknowns*

```
#the follow are equivalent
ans = state.unknowns
ans = unknowns
```

Available attributes for analyses

```
# the first analysis in the list
analysis = unknowns[0]

# a dictionary of `Isotope` objects
isotopes = analysis.isotopes

age = analysis.age

# age as a `ufloat` object
uage = analysis.uage

from uncertainties import nominal_value, std_dev
age = nominal_value(uage)
age_err = std_dev(uage)

# or
age = analysis.age
age_w_j_err = analysis.age_err

# see ArArAge for comprehensive view of the Analysis object
```

Analysis Attributes

Attribute	Type	Description
kca	ufloat	K/Ca
cak	ufloat	Ca/K
kcl	ufloat	K/Cl
clk	ufloat	Cl/K
radiogenic_yield	ufloat	%40Ar*
rad40	ufloat	40Ar*
total40	ufloat	
k39	ufloat	
uF	ufloat	
F	float	
F_err	float	
F_err_wo_irrad	float	
uage	ufloat	
uage_w_j_err	ufloat	
uage_w_position_err	ufloat	
age	float	
age_err	float	
age_err_wo_j	float	
age_err_wo_irrad	float	
age_err_wo_j_irrad	float	
ar39decayfactor	float	
ar37decayfactor	float	
isotopes	dict	Dictionary of isotopes
isochron3940	ufloat	
isochron3640	ufloat	
lambda_k	ufloat	
irradiation_label		
decay_days		
moles_k39		

Analysis Methods

Method	Arguments	Return	Description
get_interference_corrected_value	str	ufloat	
get_corrected_ratio	str, str	ufloat	
get_value	str	ufloat	

```
get_interference_corrected_value('Ar40')
get_corrected_ratio('Ar38', 'Ar36')  # returns Ar38/Ar36 ratio
get_value('age')  # helper function for getting commonly accessed values
```

Isotope Attributes

Attribute	Type	Description
name	str	Name of the isotope e.g. Ar40
detector	str	Name of detector used to measure isotope e.g. H1
n	int	Total number of data points collected. e.g. len(xs)
fn	int	Total number of data points collected. e.g. len(xs)
offset_xs	array	
xs	array	
ys	array	
fit	str	
value	float	
error	float	
uvalue	ufloat	
baseline	Isotope	
blank	Isotope	

Isotope Methods

Method	Return	Description
get_baseline_corrected_value	ufloat	
get_ic_decay_corrected_value	ufloat	
get_decay_corrected_value	ufloat	
get_interference_corrected_value	ufloat	
get_disc_corrected_value	ufloat	
get_ic_corrected_value	ufloat	
get_non_detector_corrected_value	ufloat	

Plotting Examples

Plot with error bars

```
import matplotlib.pyplot as plt
from uncertainties import nominal_value, std_dev

def main():

    ns = [a.get_value('Ar40/Ar38') for a in unknowns]
    ds = [a.get_value('Ar40/Ar36') for a in unknowns]

    xs = [nominal_value(di) for di in ds]
    ys = [nominal_value(ni) for ni in ns]

    xerrs = [std_dev(di) for di in ds]
    yerrs = [std_dev(ni) for ni in ns]

    plt.ylabel('Ar40/Ar36')
    plt.xlabel('Ar40/Ar38')
```

(continues on next page)

(continued from previous page)

```
    plt.errorbar(xs, ys, xerr=xerrs, yerr=yerrs, fmt='bo')

    plt.show()
```

Simple scatter plot

```
import matplotlib.pyplot as plt
from uncertainties import nominal_value, std_dev

def main():

    ns = [a.get_value('Ar40/Ar38') for a in unknowns]
    ds = [a.get_value('Ar40/Ar36') for a in unknowns]

    xs = [nominal_value(di) for di in ds]
    ys = [nominal_value(ni) for ni in ns]

    plt.ylabel('Ar40/Ar36')
    plt.xlabel('Ar40/Ar38')

    plt.plot(xs, ys, 'bo')

    plt.show()
```

3.5 Entry

3.5.1 Import Analyses From File

It is possible to import analyses from a Excel, csv, or yaml file into the pychron database.

Excel

To import analyses from an Excel file follow the following file format.

Note: Column order is not important, however, column names must match the following table. In addition the column names should be on the second row.

Example Excel template

`analysis_import.xls`

Necessary Columns

name	type	example	optional	note
identifier	str	12345		

continues on next page

Table 1 – continued from previous page

name	type	example	optional	note
project	str	Foo		
sample	str	Bar		
material	str	Feldspar		
aliquot	int	1		
step	int or str	2 or B		
analysis_type	str	unknown		
mass_spectrometer	str	jan		
analysis_time	date	6/7/12		
irradiation	str	NM-256	Yes	
comment	str	This is a comment	Yes	
Ar40	str or float	10.123 or Sheet2		Use sheet name to link to sheet with raw data
Ar40err	str or float			
Ar39	str or float			
Ar39err	str or float			
Ar38	str or float			
Ar38err	str or float			
Ar37	str or float			
Ar37err	str or float			
Ar36	str or float			
Ar36err	str or float			
Ar40bs	str or float			
Ar40bserr	str or float			
Ar39bs	str or float			
Ar39bserr	str or float			
Ar38bs	str or float			
Ar38bserr	str or float			
Ar37bs	str or float			
Ar37bserr	str or float			
Ar36bs	str or float			
Ar36bserr	str or float			
Ar40bk	float			
Ar40bkerr	float			
Ar39bk	float			
Ar39bkerr	float			
Ar38bk	float			
Ar38bkerr	float			
Ar37bk	float			
Ar37bkerr	float			
Ar36bk	float			
Ar36bkerr	float			
Ar40det	str			
Ar39det	str			
Ar38det	str			
Ar37det	str			
Ar36det	str			

3.5.2 Import Samples From File

It is possible to import analyses from a Excel

Excel

To import samples from an Excel file follow the following file format.

Note: Column order is not important, however, column names must match the following table. In addition the column names should be on the second row.

Example Excel template

`sample_import.xls`

Necessary Columns

name	type	example	optional	note
sample	str	foo-001		
project	str	Bar		
material	str	Feldspar		

3.6 Dashboard

maxdepth
1

config

3.7 Data Mapper

3.7.1 USGS Menlo (Volcano Science Center)

Importing analyses and metadata into pychron

1. Import Irradiation
    - Menubar> Entry> Import Irradiation...
    - Select USGSVSC MAP
    - Click the Folder icon and select the appropriate irradiation file
    - Click "Import" to import the selected irradiation file

    example Irradiation File

```
IRR330
09/23/2014, 1 hour TRIGA, Cd shielded, use average 04/2011 values
steps       1
1.0 3494329980      1
36/37       2.810000E-4     6.210000E-6
```

(continues on next page)

(continued from previous page)

```
40/39       1.003000E-3     3.790000E-4
39/37       7.100000E-4     4.960000E-5
38/37       3.290000E-5     7.500000E-6
38/39       1.314000E-2     1.200000E-5
```

2. Import Analyses

   Analyses can be imported either individually (file) or as a group (directory)

   - Menubar> Entry> Import Analyses
   - Select USGSVSC MAP from the first drop down
   - Select a Repository Identifier. You can also add a repository using the (+) button
   - Select either an entire directory or an individual analysis file to import.
   - Click "Import" to import analyses

3.8 Post Analysis Modifications

Unfortunately it is not uncommon of an analysis to be collected and saved with in correct information. For example the analysis by have been with a typo in the sample name or an incorrect sample name entirely. Another example is the analysis is run with the wrong "identifier".

3.8.1 Fix Sample MetaData

A sample's metadata is stored in the database. When an analysis of a sample is collected however, the sample information is saved in the analysis json file. So simply changing the sample metadata will only affect future analyses. In order to sync collected analyses with the database you need to use "Sync Repo/DB Sample Info" toolbar action in the "Repositories" window. Select the repository that contains the analyses you want to update and click "Sync Repo/DB Sample Info"

3.8.2 Fix Aliquot/Increment

Use Bulk Edit pipeline

3.9 ArAr Age Calculations

3.9.1 F-valve Calculation

```
# user supplied values
# ic/decay corrected intensities
a40=  #Ar40
a39=  #Ar39
a38=  #Ar38
a37=  #Ar37
a36=  #Ar36
k4039= #(40/39)K production ratio
```

(continues on next page)

(continued from previous page)

```
ca3937= #(39/37)Ca
k3739= #(37/39)K
k3839= #(38/39)K
ca3637= #(36/37)Ca
ca3837= #(38/37)Ca
cl3638= #(36/38)Cl
lambda_cl36= # Cl36 decay constant
decay_time= # time since irradiation
atm3836 = # trapped Ar38/Ar36
trapped_4036= # trapped Ar40/Ar36

# ca-correction
# ca37 = a37 - k37
# ca39 = ca3937 * ca37
# k39 = a39 - ca39
# k37 = k3739 * k39

# k39 = a39-ca3937*(a37-(k3739*k39))
#    = a39-ca3937*a37+k3739*k39*ca3937
# k39*(1-k3739*ca3937) = a39-ca3937*a37

k39 = (a39-ca3937*a37)/(1-k3739*ca3937)
k37 = k3739 * k39
k38 = k3839 * k39

ca37 = a37 - k37
ca39 = ca3937 * ca37

ca36 = ca3637 * ca37
ca38 = ca3837 * ca37

# cl-correction
# calculate atm36, cl36, cl38

# starting with the following equations
# atm36 = a36 - ca36 - cl36

# m = cl3638*lambda_cl36*decay_time
# cl36 = cl38 * m

# cl38 = a38 - k38 - ca38 - ar38atm
# ar38atm = atm3836 * atm36

# rearranging to solve for atm36
# cl38 = a38  - k38 - c38 - atm3836 * atm36

# cl36 = m * (a38  - k38 - ca38 - atm3836 * atm36)
#     = m (a38  - k38 - ca38) - m * atm3836 * atm36
# atm36 = a36 - ca36 - m (a38  - k38 - ca38) + m * atm3836 * atm36
# atm36 - m * atm3836 * atm36 =  a36 - ca36 - m (a38  - k38 - ca38)
# atm36 * (1 - m*atm3836) = a36 - ca36 - m (a38  - k38 - ca38)
# atm36 = (a36 - ca36 - m (a38  - k38 - c38))/(1 - m*atm3836)
```

(continues on next page)

(continued from previous page)

```
m = cl3638 * lambda_Cl36 * decay_time
atm36 = (a36 - ca36 - m*(a38 - k38 - ca38)) / (1 - m * atm3836)

atm40 = atm36 * trapped_4036

k40 = k39 * k4039

rad40 = a40 - atm40 - k40
f = rad40 / k39
```

3.9.2 Age Calculation

```
lambda_k =  # total 40K decay constant
f = # F-value e.g Ar40*/Ar39K
j = # J-value e.g. neutron flux parameter
age = lambda_k ** -1 * ln(1 + j * f))
```

3.9.3 Apply Fixed (37/39)K

```
"""
     x=ca37/k39
     y=ca37/ca39
     T=s39dec_cor

     T=ca39+k39
     T=ca37/y+ca37/x

     ca37=(T*x*y)/(x+y)
 """

 k3739 = # (37/39)K
 ca39 =  # (39/37)Ca

 x = k3739
 y = 1 / ca3937

 ca37 = (a39 * x * y) / (x + y)

 ca39 = ca3937 * ca37
 k39 = a39 - ca39
 k37 = x * k39
```

3.9.4 Decay Factors

```
"""
    McDougall and Harrison
    p.75 equation 3.22

    the book suggests using ti==analysis_time-end of irradiation segment_i

    mass spec uses ti==analysis_time-start of irradiation segment_i

    using start seems more appropriate
"""

    dc37 = # Ar37 decay constant
    dc39 = # Ar39 decay constant

    a = sum([pi * ti for pi, ti, _, _, _ in segments])

    b = sum([pi * ((1 - math.exp(-dc37 * ti)) / (dc37 * math.exp(dc37 * dti)))
         for pi, ti, dti, _, _ in segments])

    c = sum([pi * ((1 - math.exp(-dc39 * ti)) / (dc39 * math.exp(dc39 * dti)))
         for pi, ti, dti, _, _ in segments])


    df37 = a / b
    df39 = a / c
```

3.9.5 Abundance Sensitivity

```
s40 = # m/e=40 intensity


# correct for abundance sensitivity
# assumes symmetric and equal abundant sens for all peaks
n40 = s40 - abundance_sensitivity * (s39 + s39)
n39 = s39 - abundance_sensitivity * (s40 + s38)
n38 = s38 - abundance_sensitivity * (s39 + s37)
n37 = s37 - abundance_sensitivity * (s38 + s36)
n36 = s36 - abundance_sensitivity * (s37 + s37)
```

3.9.6 Integrated Age Weighting Factors

Variance

```
W_i = 1/Ar39K_i_sigma**2
```

Volume

```
W_i = (39ArK_i/Total_39ArK*39ArK_i_sigma)^2
```

3.10 Glossary

DVC
Data Version Control. A system for tracking all tracks to data

Editor
A tab within the editor area.

Editor Area
A region within a window that contains one or more Editors.

Pyscript
Pychron's mini scripting language used for extraction line and measurement automation.

Python
The high level scripting programming language using to write Pychron.

Pane
A movable component at adds extra functionality to a task.

Task
A set of panes and editors used to perform a specific operation.

Window
A top-level container for a task. Tasks and windows in most cases can be thought of as synonymous, the difference is that a single window can be switched to display a different task.

3.10.1 Git Glossary

Adapted from https://help.github.com/articles/github-glossary/

Branch
A branch is a parallel version of a repository. It is contained within the repository, but does not affect the primary or master branch allowing you to work freely without disrupting the "live" version. When you've made the changes you want to make, you can merge your branch back into the master branch to publish your changes.

Clone
A clone is a copy of a repository that lives on your computer instead of on a website's server somewhere, or the act of making that copy. With your clone you can edit the files in your preferred editor and use Git to keep track of your changes without having to be online. It is, however, connected to the remote version so that changes can be synced between the two. You can push your local changes to the remote to keep them synced when you're online.

Commit
A commit, or "revision", is an individual change to a file (or set of files). It's like when you save a file, except with Git, every time you save it creates a unique ID (a.k.a. the "SHA" or "hash") that allows you to keep record of what changes were made when and by who. Commits usually contain a commit message which is a brief description of what changes were made.

Merge
Merging takes the changes from one branch (in the same repository or from a fork), and applies them into another. This often happens as a Pull Request (which can be thought of as a request to merge), or via the command line. A merge can be done automatically via a Pull Request via the GitHub.com web interface if there are no conflicting changes, or can always be done via the command line.

Pull
Pull refers to when you are fetching in changes and merging them. For instance, if someone has edited the remote file you're both working on, you'll want to pull in those changes to your local copy so that it's up to date.

Push

Pushing refers to sending your committed changes to a remote repository such as GitHub.com. For instance, if you change something locally, you'd want to then push those changes so that others may access them.

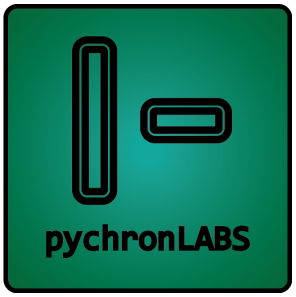
Repository

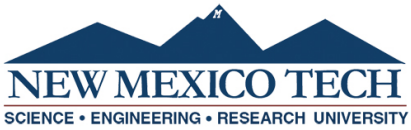
A repository is the most basic element of Git. They're easiest to imagine as a project's folder. A repository contains all of the project files (including documentation), and stores each file's revision history. Repositories can have multiple collaborators and can be either public or private.

CHAPTER

FOUR

PYCHRON DEVELOP GUIDE





4.1 Launching Pychron

4.1.1 Envisage

All pychron applications are based on Enthought's envisage application framework. The envisage framework is used to make pychron extensible and pluggable. For more information see http://docs.enthought.com/envisage/

4.1.2 Entry Point

The entry point for all pychron applications are scripts located in the `launchers` package. For example, to launch pyexperiment

```
cd pychron
python launchers/pyexperiment.py
```

This script calls imports and calls `entry_point` from the `helpers` module.

The `entry_point` function performs multiples tasks.

1. sets the GUI backend to Qt `ETSConfig.toolkit = "qt4"`
2. updates the pythonpath
3. sets global variables and pychron filesystem paths
4. constructs missing directories in Pychrondata...
5. sets up logging
6. calls the `launch` function

The `launch` function takes one argument, a PychronApplication class. The `launch` function is located at `pychron.envisage.pychron_run`.

`launch` does the following

1. checks that necessary dependencies are installed
2. constructs the application object using `app_factory`
3. launches the application by calling the application's `run` method

During Step. 2 the necessary plugins are constructed and added to the application. The plugins to use are specified in the `initialization.xml` file. Parsing of the `initialization.xml` file is handled by an `InitializationParser` object.

4.1.3 Initialization

If **Hardware** plugins are included, such as **FusionsCO2**, the **HardwarePlugin** is added to the application plugins. When the **HardwarePlugin** starts, it uses an `Initalizer` object to bootstrap the necessary managers and devices.

4.2 Hardware

4.2.1 Make a new Device

You have two options for making a new device, CoreDevice and AbstractDevice. Use CoreDevice when you are directly interfacing with a physical device. Use AbstractDevice if you interface with the device via a secondary device. For example when you want to access a physical device with an analog output. Lets say you want to read the analog output from a power meter. First create a PowerMeter(AbstractDevice) class than use or create a ADC(CoreDevice) class. ADC(CoreDevice) class communicates with the Analog-to-Digital Converter device and the PowerMeter class converts the raw signal from ADC(CoreDevice) into a power value i.e. Watts

Using the AbstractDevice allows you to interface with the physical power meter using a variety of ADC's that typically will have different communication protocols. Switching ADCs is a simple as switch the a single value in the PowerMeter(AbstractDevice)'s configuration file.

4.3 Database

```
update analysestable
set aliquot_pychron = CAST(analysestable.aliquot as SIGNED INTEGER)
where `IrradPosition` in #(LIST OF LABNUMBERS)
```

```
delimiter $$
CREATE TRIGGER update_aliquot_pychron
BEFORE INSERT ON analysestable
FOR EACH ROW BEGIN
SET NEW.aliquot_pychron = CAST(NEW.aliquot as SIGNED INTEGER);
END$$
delimiter ;
```

4.3.1 Install

Installing a Mass Spec-compatible MySQL database

Author
Rich Esser

Date
9/27/07

Updated
Jake Ross

Date
4/4/13

1. Download the MySQL package installer. As of mid September 2007, it is located at:

   http://dev.mysql.com/get/Downloads/MySQL-5.0/mysql-5.0.45-osx10.4-i686.dmg/from/ http://mirror.services.wisc.edu/mysql/

   But as time goes by and new versions come out, the above link may no longer work. Going to the general MySQL download site, you will be able to find a recent version for the computer in question (you must know if you're installing MySQL on a PowerPC (PPC) Mac or Intel Mac and download that particular version): http://dev.mysql.com/downloads/. Open up the disk image that the installer arrives in. Inside you will find and install the " mysql-5.X.XX-osxXXX.pkg" (the actual MySQL database application), the "MySQLStartupitem.pkg" (a small application that will automatically launch the MySQL program when the computer is restarted) and the "MySQL.prefPane" (a System Preferences pane to easily turn MySQL on/off). Double-click and install in the order above (you'll need to enter your admin password for most of these installers). Once all is installed, open the 'MySQL' preference pane (in System Preferences) and start the MySQL Server and click the checkbox to automatically start MySQL on reboot. At this point, you should probably restart the computer.

2. To verify the success of the installation, issue the command

   ```
   /usr/local/mysql/bin/mysql --version
   ```

   in the Terminal. The result should be something like

   ```
   /usr/local/mysql/bin/mysql  Ver 14.12 Distrib 5.0.45, for apple-darwin8.5.1 (i686)␣
   ↪using readline 5.0
   ```

If you don't get this similar return back, open the MySQL preference pane and make sure that MySQL is running ("running" in green text). If the MySQL server is running, you can now give the server a root password. The root password for MySQL is independent of the MacOS admin or root passwords. In the Terminal, issue the following command:

```
sudo /usr/local/mysql/bin/mysqladmin -u root password "Argon"
```

where "Argon" (include the quotes in the Terminal command) is the password to access the database. However, you can make the password whatever you like. Remember, because you issue the command with the sudo, you must enter your Mac admin password to get the command to work.

3. With the root password set, you can now log into the MySQL database with client software (e.g. Sequel Pro). Using Sequel Pro or another MySQL editor, log into the new database (host=localhost, user=root, password=Argon). In the "Choose database…" popup list on the left side of the resulting window, select the 'mysql' database. Go to the bottom of the table list and click on 'user'. In the upper-right window, switch from the "Structure" tab to the "Content" tab. Highlight the row that has 'localhost' and 'root" (there is probably only 1, maybe 2 rows in the table). Duplicate the highlighted row by clicking the 'Duplicate' icon at the bottom of the window. In the resulting row, change the 'root' to 'massspec' (double-click the 'root' name to edit). Enter to take the change. Click on the "Flush Privileges" icon at the top of the Sequel Pro window.

4. The Mass Spec program (specifically, RealBasic) uses an older style of password/database hashing (look it up, if you're curious), so you must tell the newly installed database server to use this older style hashing. If you skip this step, the Mass Spec program will not be able to use the database. In the Terminal, log into the MySQL database:

```
sudo /usr/local/mysql/bin/mysql -u root -p
```

Once again, the first password is your MacOS admin password, followed by the MySQL root password (Argon). You should now see the mysql prompt (mysql>). Type

```
use mysql;
```

with the semi-colon closing the command (all commands typed into the MySQL command line editor must be closed/ended by the semi-colon). Hit return.

```
Database changed
```

Next command:

```
UPDATE user SET Password = OLD_PASSWORD('Argon') WHERE Host = 'localhost' AND
User = 'root';
```

```
"Query OK, 1 rows affected...".
```

Follow it by:

```
UPDATE user SET Password = OLD_PASSWORD('Argon') WHERE Host = 'localhost' AND
User = 'massspec';
```

```
Another "Query OK, 1 rows affected...".
```

And once again, flush the privileges, either with this command (flush privileges;) or with Sequel Pro.

That should do it. You're now ready to import a 'massspecdata' database using the normal procedures. Remember to first create the database (using CocoaMySQL) that you're going to import in to. Then import:

```
sudo /usr/local/mysql/bin/mysql -u root -p massspecdata < the_backed_up_database_
↪location_and_name
```

The above are bare-bones instructions for getting MySQL up and running for a local-only system. In other words, no one from outside the computer you just installed MySQL on can access this database server. If this is desired, there are several additional steps needed.

4.3.2 Replication

Setup a MySQL replication server

Author
Alan Deino

Date
2/2011

Source
Mass Spec Manual Version 7.849

Modified
Jake Ross

Date
4/4/2013

A significant improvement over the single MySQL server approach is to set up a replication MySQL server on a separate computer, which will automatically mirror the database on the main ('master') server. Any changes that occur in the master, like storage of new analytical data, are immediately updated in the replication server, or 'slave.' You can do the periodic (e.g., nightly) database dumps on the slave, so that the master doesn't get bogged down and potentially drastically slow the mass spectrometer data collection operation during the period of the backup. Also, the intention is in the future to have Mass Spec detect failure of the master server, and automatically continue operation with the slave. This is not yet implemented as of 7.782.

To set up a replication server, you could do the following:

1. Stop all data collection and reduction operations.
2. Create a dump of the MassSpecData database on the master MySQL server.
3. Create a new MySQL server on the replication computer. Note that the best approach is to also update the master server so that the two MySQL versions are the same.
4. Set up the slave MySQL account in the same way as the master (see the User table in the MySQL database; remember to Flush Priviledges after changing the User account).
5. Load the data backup from step #2 into the slave MySQL in the manner indicated in the previous section.
6. You now have two identical MySQL instances. One must be designated the master and the other the slave through the use of configuration files ('my.cnf), read by the instances on startup.
7. Master my.cnf: Edit or create the my.cnf file in the MySQL data folder, located usually at /usr/local/mysql/data. You can also find or create it in /etc. Add to this file:

```
[mysqld]
log-bin
binlog-do-db=massspecdata
server-id=1
```

8. Stop the master server (typically using the MySQL preference pane in the System Preferences window).

9. Restart the master server.
10. From the terminal command line, on in Sequel Pro or YourSQL, issue the following SQL command:

```
SHOW MASTER STATUS
```

The response should be a file name, position, and the name of the database being logged.

11. Slave my.cnf: Edit or create the my.cnf file in the MySQL data folder, located usually at /usr/local/mysql/data. You can also find or create it in /etc. Add to this file (replace italicized fields with your appropriate data):

```
[mysqld]
server-id=2
master-host= put the IP address of the master here master-user=massspec
master-password=the Mass Spec database access password master-connect-retry=60
replicate-do-db=massspecdata
```

12. If you place your .cnf files in /etc, make sure the owner/group is mysql, with the following commands issued in a terminal window:

```
sudo chown mysql:mysql /etc/my.cnf
sudo chmod 770 /etc/my.cnf
```

13. Stop/start the slave MySQL instance.
14. Issue the following SQL command:

```
STOP SLAVE
```

15. Issue the following SQL command:

```
CHANGE MASTER TO master_host='MH',master_user='MU',master_password='MP', master_log_
↪file='MLF',master_log_pos=MLP
```

where MH= IP address of master, MU= master username e.g massspec, MP= the Mass Spec database access password, MLF= the file name obtained in step #10 and MLP= the postion obtained in step #10

16. Issue the following SQL command:

```
START SLAVE
```

The replication server should now be operational. Test it by making a small change in the master database; this change should be reflected in the slave database.

4.4 DVC

4.4.1 Analysis

example. See also https://github.com/NMGRLData/Lusk01121/blob/master/666/81-03L.json

```
{
    "acquisition_software": "pychron py3/18.2",
    "aliquot": 3,
    "analysis_type": "unknown",
```

(continues on next page)

(continued from previous page)


```
    "analyst_name": "",
    "arar_mapping": {
        "Ar36": "Ar36",
        "Ar37": "Ar37",
        "Ar38": "Ar38",
        "Ar39": "Ar39",
        "Ar40": "Ar40"
    },
    "collection_version": "1.0:2.0",
    "comment": "C:16 Muscovite, fine, 4.98 mg",
    "commit": "f3c45de575b8ae63b7fee625a0156943968ccbb4",
    "conditionals": [
        {
            "analysis_types": [],
            "frequency": 10,
            "hash_id": -3426856737248020020,
            "level": 10,
            "location": "setupfiles/spectrometer/default_conditionals.yaml",
            "ntrips": 1,
            "start_count": 5,
            "teststr": "L2(CDD).deflection==3250"
        }
    ],
    "data_reduction_software": "pychron py3/18.2",
    "detectors": {
        "AX": {
            "deflection": 0.0,
            "gain": 1
        },
        "H1": {
            "deflection": 0.0,
            "gain": 1
        },
        "H2": {
            "deflection": 0.0,
            "gain": 1
        },
        "L1": {
            "deflection": 0.0,
            "gain": 1
        },
        "L2(CDD)": {
            "deflection": 0.0,
            "gain": 1
        }
    },
    "environmental": {
        "lab_humiditys": [
            {
                "device": "EnvironmentalMonitor",
                "name": "Lab Hum.",
                "pub_date": "2019-03-15T07:53:04",
```


(continues on next page)

(continued from previous page)


```
                "title": "iServer Hum.",
                "value": 18.3
            }
        ],
        "lab_pneumatics": [
            {
                "device": "AirPressure",
                "name": "Pressure",
                "pub_date": "2019-03-15T12:43:35",
                "title": "Building",
                "value": 101.1
            },
            {
                "device": "AirPressure2",
                "name": "Pressure2",
                "pub_date": "2019-03-15T07:53:04",
                "title": "Lab",
                "value": 87.5
            },
            {
                "device": "MicroBoneGaugeController",
                "name": "JanDecabinPressure",
                "pub_date": "2019-03-15T07:53:04",
                "title": "Jan Decabin",
                "value": 2.0
            }
        ],
        "lab_temperatures": []
    },
    "experiment_queue_name": "LoadLF0001",
    "experiment_type": "Ar/Ar",
    "extraction": "felix_furnace.py",
    "grainsize": "",
    "identifier": "66681",
    "increment": 11,
    "instrument_name": "",
    "intensity_scalar": 0.0,
    "irradiation": "NM-300",
    "irradiation_level": "C",
    "irradiation_position": 16,
    "isotopes": {
        "Ar36": {
            "detector": "L2(CDD)",
            "name": "Ar36",
            "serial_id": "00000"
        },
        "Ar37": {
            "detector": "L1",
            "name": "Ar37",
            "serial_id": "00000"
        },
        "Ar38": {
```


(continues on next page)

(continued from previous page)


```
            "detector": "AX",
            "name": "Ar38",
            "serial_id": "00000"
        },
        "Ar39": {
            "detector": "H1",
            "name": "Ar39",
            "serial_id": "00000"
        },
        "Ar40": {
            "detector": "H2",
            "name": "Ar40",
            "serial_id": "00000"
        }
    },
    "laboratory": "",
    "latitude": "",
    "lithology": "",
    "lithology_class": "",
    "lithology_group": "",
    "lithology_type": "",
    "longitude": "",
    "mass_spectrometer": "felix",
    "material": "",
    "measurement": "felix_analysis340_60_CDD_center.py",
    "post_equilibration": "felix_pump_extraction_line.py",
    "post_measurement": "felix_pump_ms.py",
    "principal_investigator": "",
    "project": "",
    "queue_conditionals_name": "",
    "repository_identifier": "Lusk01121",
    "rlocation": null,
    "sample": "LS-248",
    "source": {
        "emission": 770.169,
        "trap": 248.607
    },
    "spec_sha": "857b14d91aad9a1d167c9a6f53bb8f032c5a9e87",
    "timestamp": "2019-03-15T14:49:55.061740",
    "tripped_conditional": null,
    "unit": null,
    "username": "",
    "uuid": "a251fda6-5bd9-4c5d-bb5b-c77bcd8dbaec",
    "whiff_result": null
}
```

4.4.2 Blank

Default Blank file example for an analysis collected by Pychron


```
{
  "Ar36": {
      "error": 0.0002047115883032583,
      "error_type": "",
      "fit": "previous",
      "references": [
          {
              "exclude": false,
              "record_id": "bu-NF-F-236"
          }
      ],
      "value": 0.004194779935647032
  },
  "Ar37": {
      "error": 0.014948890705747078,
      "error_type": "",
      "fit": "previous",
      "references": [
          {
              "exclude": false,
              "record_id": "bu-NF-F-236"
          }
      ],
      "value": 0.007742686081190914
  },
  "Ar38": {
      "error": 0.005029408341337333,
      "error_type": "",
      "fit": "previous",
      "references": [
          {
              "exclude": false,
              "record_id": "bu-NF-F-236"
          }
      ],
      "value": -0.0030836048108839054
  },
  "Ar39": {
      "error": 0.011975702475760708,
      "error_type": "",
      "fit": "previous",
      "references": [
          {
              "exclude": false,
              "record_id": "bu-NF-F-236"
          }
      ],
      "value": -0.008069258524676111
  },
```


(continues on next page)

(continued from previous page)


```
  "Ar40": {
      "error": 0.024465880424101836,
      "error_type": "",
      "fit": "previous",
      "references": [
          {
              "exclude": false,
              "record_id": "bu-NF-F-236"
          }
      ],
      "value": 2.8342343480026635
  }
}
```


Example Blank file for Analysis run thru “Blanks” pipeline


```
{
  "Ar36": {
      "error": 0.002530676861539285,
      "error_type": "",
      "fit": "Parabolic_SD",
      "references": [
          {
              "exclude": "ok",
              "record_id": "bu-FC-J-9999",
              "uuid": "203a75ef-33cf-4b5c-a8d7-4a68beb11cf9"
          },
          {
              "exclude": "ok",
              "record_id": "bu-FC-J-10000",
              "uuid": "610dbdbe-5663-41d5-a871-c318de82e871"
          },
          {
              "exclude": "ok",
              "record_id": "bu-FC-J-10001",
              "uuid": "156651dc-84f9-452e-8436-517922e14973"
          },
          {
              "exclude": "ok",
              "record_id": "bu-FC-J-10002",
              "uuid": "3f6c528a-833e-4094-b550-5526fd658d04"
          },
          {
              "exclude": "ok",
              "record_id": "bu-FC-J-10003",
              "uuid": "adee3e32-8b11-4a2b-bf9e-3afaa7f87cb4"
          },
          {
              "exclude": "ok",
              "record_id": "bu-FC-J-10004",
              "uuid": "aadadc1b-b363-4824-ac84-8b78988cd925"
          },
```


(continues on next page)

(continued from previous page)


```
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10005",
            "uuid": "ceae7843-b8bf-4979-80c4-38d47230f03d"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10006",
            "uuid": "b7802c50-bc12-494e-9b3b-f611dca52f39"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10007",
            "uuid": "9ed44d3e-2f30-4aa4-a57e-e36015154bcb"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10008",
            "uuid": "7cdb4a76-aae1-4299-8832-913b1fb90fbc"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10009",
            "uuid": "c5ae7ec6-504a-469a-aa88-f261638b0db6"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10010",
            "uuid": "39c8bca3-d1fd-49a3-8826-02e659b34a1b"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10011",
            "uuid": "0f91075c-852d-4193-96ec-9b229e5ae823"
        },
        {
            "exclude": "omit",
            "record_id": "bu-FC-J-10012",
            "uuid": "fa273be2-8805-4ac4-a2a2-5e53cf242eda"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10013",
            "uuid": "cf6926a6-28f7-4fa3-840e-b54ae4d47b0e"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10014",
            "uuid": "236fb911-03b4-423e-8a64-7c545ad74d07"
        },
        {
            "exclude": "omit",
```


(continues on next page)

(continued from previous page)


```
            "record_id": "bu-FC-J-10015",
            "uuid": "88b57793-559c-4036-8bab-1674c55dfb12"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10016",
            "uuid": "4db2f0a4-a125-4ea7-abf1-f36aeb3412bd"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10017",
            "uuid": "898e97c3-0e12-4b8c-aabb-053c540b1638"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10018",
            "uuid": "fd0b06c2-d005-417f-8140-acfe493d7efe"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10019",
            "uuid": "32bdfc37-6b20-4d6f-9dea-325f785f6825"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10020",
            "uuid": "c84ef7fe-5f08-4a78-ab63-66c81e72cd0d"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10021",
            "uuid": "41eb9da1-17f9-4d44-82b6-39bb31bcfb43"
        },
        {
            "exclude": "omit",
            "record_id": "bu-FC-J-10022",
            "uuid": "9e9a11d1-a714-45ec-9998-79aaf51dca3b"
        },
        {
            "exclude": "omit",
            "record_id": "bu-FC-J-10023",
            "uuid": "feb18f7d-5d11-43dc-93f5-24668ac65d73"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10024",
            "uuid": "a49aa507-ffc2-4e11-842e-8b16a4a3ee3f"
        },
        {
            "exclude": "omit",
            "record_id": "bu-FC-J-10025",
            "uuid": "591afc5e-2287-40a0-8010-0e1f096cb325"
```


(continues on next page)

(continued from previous page)


```
            }
        ],
        "reviewed": true,
        "value": 0.026285578599641108
    },
    "Ar37": {
        "error": 0.00786620718599018,
        "error_type": "",
        "fit": "Average_SD",
        "references": [
            {
                "exclude": "ok",
                "record_id": "bu-FC-J-9999",
                "uuid": "203a75ef-33cf-4b5c-a8d7-4a68beb11cf9"
            },
            {
                "exclude": "ok",
                "record_id": "bu-FC-J-10000",
                "uuid": "610dbdbe-5663-41d5-a871-c318de82e871"
            },
            {
                "exclude": "ok",
                "record_id": "bu-FC-J-10001",
                "uuid": "156651dc-84f9-452e-8436-517922e14973"
            },
            {
                "exclude": "ok",
                "record_id": "bu-FC-J-10002",
                "uuid": "3f6c528a-833e-4094-b550-5526fd658d04"
            },
            {
                "exclude": "ok",
                "record_id": "bu-FC-J-10003",
                "uuid": "adee3e32-8b11-4a2b-bf9e-3afaa7f87cb4"
            },
            {
                "exclude": "ok",
                "record_id": "bu-FC-J-10004",
                "uuid": "aadadc1b-b363-4824-ac84-8b78988cd925"
            },
            {
                "exclude": "ok",
                "record_id": "bu-FC-J-10005",
                "uuid": "ceae7843-b8bf-4979-80c4-38d47230f03d"
            },
            {
                "exclude": "ok",
                "record_id": "bu-FC-J-10006",
                "uuid": "b7802c50-bc12-494e-9b3b-f611dca52f39"
            },
            {
                "exclude": "ok",
```


(continues on next page)

(continued from previous page)


```
            "record_id": "bu-FC-J-10007",
            "uuid": "9ed44d3e-2f30-4aa4-a57e-e36015154bcb"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10008",
            "uuid": "7cdb4a76-aae1-4299-8832-913b1fb90fbc"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10009",
            "uuid": "c5ae7ec6-504a-469a-aa88-f261638b0db6"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10010",
            "uuid": "39c8bca3-d1fd-49a3-8826-02e659b34a1b"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10011",
            "uuid": "0f91075c-852d-4193-96ec-9b229e5ae823"
        },
        {
            "exclude": "omit",
            "record_id": "bu-FC-J-10012",
            "uuid": "fa273be2-8805-4ac4-a2a2-5e53cf242eda"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10013",
            "uuid": "cf6926a6-28f7-4fa3-840e-b54ae4d47b0e"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10014",
            "uuid": "236fb911-03b4-423e-8a64-7c545ad74d07"
        },
        {
            "exclude": "omit",
            "record_id": "bu-FC-J-10015",
            "uuid": "88b57793-559c-4036-8bab-1674c55dfb12"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10016",
            "uuid": "4db2f0a4-a125-4ea7-abf1-f36aeb3412bd"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10017",
            "uuid": "898e97c3-0e12-4b8c-aabb-053c540b1638"
```


(continues on next page)

(continued from previous page)


```
            },
            {
                "exclude": "ok",
                "record_id": "bu-FC-J-10018",
                "uuid": "fd0b06c2-d005-417f-8140-acfe493d7efe"
            },
            {
                "exclude": "ok",
                "record_id": "bu-FC-J-10019",
                "uuid": "32bdfc37-6b20-4d6f-9dea-325f785f6825"
            },
            {
                "exclude": "ok",
                "record_id": "bu-FC-J-10020",
                "uuid": "c84ef7fe-5f08-4a78-ab63-66c81e72cd0d"
            },
            {
                "exclude": "ok",
                "record_id": "bu-FC-J-10021",
                "uuid": "41eb9da1-17f9-4d44-82b6-39bb31bcfb43"
            },
            {
                "exclude": "omit",
                "record_id": "bu-FC-J-10022",
                "uuid": "9e9a11d1-a714-45ec-9998-79aaf51dca3b"
            },
            {
                "exclude": "omit",
                "record_id": "bu-FC-J-10023",
                "uuid": "feb18f7d-5d11-43dc-93f5-24668ac65d73"
            },
            {
                "exclude": "ok",
                "record_id": "bu-FC-J-10024",
                "uuid": "a49aa507-ffc2-4e11-842e-8b16a4a3ee3f"
            },
            {
                "exclude": "omit",
                "record_id": "bu-FC-J-10025",
                "uuid": "591afc5e-2287-40a0-8010-0e1f096cb325"
            }
        ],
        "reviewed": true,
        "value": 0.03449851337958194
    },
    "Ar38": {
        "error": 0.004198981148223098,
        "error_type": "",
        "fit": "Average_SD",
        "references": [
            {
                "exclude": "ok",
```


(continues on next page)

(continued from previous page)


```
            "record_id": "bu-FC-J-9999",
            "uuid": "203a75ef-33cf-4b5c-a8d7-4a68beb11cf9"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10000",
            "uuid": "610dbdbe-5663-41d5-a871-c318de82e871"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10001",
            "uuid": "156651dc-84f9-452e-8436-517922e14973"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10002",
            "uuid": "3f6c528a-833e-4094-b550-5526fd658d04"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10003",
            "uuid": "adee3e32-8b11-4a2b-bf9e-3afaa7f87cb4"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10004",
            "uuid": "aadadc1b-b363-4824-ac84-8b78988cd925"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10005",
            "uuid": "ceae7843-b8bf-4979-80c4-38d47230f03d"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10006",
            "uuid": "b7802c50-bc12-494e-9b3b-f611dca52f39"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10007",
            "uuid": "9ed44d3e-2f30-4aa4-a57e-e36015154bcb"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10008",
            "uuid": "7cdb4a76-aae1-4299-8832-913b1fb90fbc"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10009",
            "uuid": "c5ae7ec6-504a-469a-aa88-f261638b0db6"
```


(continues on next page)

(continued from previous page)


```
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10010",
            "uuid": "39c8bca3-d1fd-49a3-8826-02e659b34a1b"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10011",
            "uuid": "0f91075c-852d-4193-96ec-9b229e5ae823"
        },
        {
            "exclude": "omit",
            "record_id": "bu-FC-J-10012",
            "uuid": "fa273be2-8805-4ac4-a2a2-5e53cf242eda"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10013",
            "uuid": "cf6926a6-28f7-4fa3-840e-b54ae4d47b0e"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10014",
            "uuid": "236fb911-03b4-423e-8a64-7c545ad74d07"
        },
        {
            "exclude": "omit",
            "record_id": "bu-FC-J-10015",
            "uuid": "88b57793-559c-4036-8bab-1674c55dfb12"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10016",
            "uuid": "4db2f0a4-a125-4ea7-abf1-f36aeb3412bd"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10017",
            "uuid": "898e97c3-0e12-4b8c-aabb-053c540b1638"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10018",
            "uuid": "fd0b06c2-d005-417f-8140-acfe493d7efe"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10019",
            "uuid": "32bdfc37-6b20-4d6f-9dea-325f785f6825"
        },
        {
```


(continues on next page)

(continued from previous page)


```
                "exclude": "ok",
                "record_id": "bu-FC-J-10020",
                "uuid": "c84ef7fe-5f08-4a78-ab63-66c81e72cd0d"
            },
            {
                "exclude": "ok",
                "record_id": "bu-FC-J-10021",
                "uuid": "41eb9da1-17f9-4d44-82b6-39bb31bcfb43"
            },
            {
                "exclude": "omit",
                "record_id": "bu-FC-J-10022",
                "uuid": "9e9a11d1-a714-45ec-9998-79aaf51dca3b"
            },
            {
                "exclude": "omit",
                "record_id": "bu-FC-J-10023",
                "uuid": "feb18f7d-5d11-43dc-93f5-24668ac65d73"
            },
            {
                "exclude": "ok",
                "record_id": "bu-FC-J-10024",
                "uuid": "a49aa507-ffc2-4e11-842e-8b16a4a3ee3f"
            },
            {
                "exclude": "omit",
                "record_id": "bu-FC-J-10025",
                "uuid": "591afc5e-2287-40a0-8010-0e1f096cb325"
            }
        ],
        "reviewed": true,
        "value": 0.01028580551619398
    },
    "Ar39": {
        "error": 0.012217213055969865,
        "error_type": "",
        "fit": "Average_SD",
        "references": [
            {
                "exclude": "ok",
                "record_id": "bu-FC-J-9999",
                "uuid": "203a75ef-33cf-4b5c-a8d7-4a68beb11cf9"
            },
            {
                "exclude": "ok",
                "record_id": "bu-FC-J-10000",
                "uuid": "610dbdbe-5663-41d5-a871-c318de82e871"
            },
            {
                "exclude": "ok",
                "record_id": "bu-FC-J-10001",
                "uuid": "156651dc-84f9-452e-8436-517922e14973"
```


(continues on next page)

(continued from previous page)


```
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10002",
            "uuid": "3f6c528a-833e-4094-b550-5526fd658d04"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10003",
            "uuid": "adee3e32-8b11-4a2b-bf9e-3afaa7f87cb4"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10004",
            "uuid": "aadadc1b-b363-4824-ac84-8b78988cd925"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10005",
            "uuid": "ceae7843-b8bf-4979-80c4-38d47230f03d"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10006",
            "uuid": "b7802c50-bc12-494e-9b3b-f611dca52f39"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10007",
            "uuid": "9ed44d3e-2f30-4aa4-a57e-e36015154bcb"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10008",
            "uuid": "7cdb4a76-aae1-4299-8832-913b1fb90fbc"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10009",
            "uuid": "c5ae7ec6-504a-469a-aa88-f261638b0db6"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10010",
            "uuid": "39c8bca3-d1fd-49a3-8826-02e659b34a1b"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10011",
            "uuid": "0f91075c-852d-4193-96ec-9b229e5ae823"
        },
        {
```


(continues on next page)

(continued from previous page)


```
        "exclude": "omit",
        "record_id": "bu-FC-J-10012",
        "uuid": "fa273be2-8805-4ac4-a2a2-5e53cf242eda"
    },
    {
        "exclude": "ok",
        "record_id": "bu-FC-J-10013",
        "uuid": "cf6926a6-28f7-4fa3-840e-b54ae4d47b0e"
    },
    {
        "exclude": "ok",
        "record_id": "bu-FC-J-10014",
        "uuid": "236fb911-03b4-423e-8a64-7c545ad74d07"
    },
    {
        "exclude": "omit",
        "record_id": "bu-FC-J-10015",
        "uuid": "88b57793-559c-4036-8bab-1674c55dfb12"
    },
    {
        "exclude": "ok",
        "record_id": "bu-FC-J-10016",
        "uuid": "4db2f0a4-a125-4ea7-abf1-f36aeb3412bd"
    },
    {
        "exclude": "ok",
        "record_id": "bu-FC-J-10017",
        "uuid": "898e97c3-0e12-4b8c-aabb-053c540b1638"
    },
    {
        "exclude": "ok",
        "record_id": "bu-FC-J-10018",
        "uuid": "fd0b06c2-d005-417f-8140-acfe493d7efe"
    },
    {
        "exclude": "ok",
        "record_id": "bu-FC-J-10019",
        "uuid": "32bdfc37-6b20-4d6f-9dea-325f785f6825"
    },
    {
        "exclude": "ok",
        "record_id": "bu-FC-J-10020",
        "uuid": "c84ef7fe-5f08-4a78-ab63-66c81e72cd0d"
    },
    {
        "exclude": "ok",
        "record_id": "bu-FC-J-10021",
        "uuid": "41eb9da1-17f9-4d44-82b6-39bb31bcfb43"
    },
    {
        "exclude": "omit",
        "record_id": "bu-FC-J-10022",
```


(continues on next page)

(continued from previous page)


```
                "uuid": "9e9a11d1-a714-45ec-9998-79aaf51dca3b"
            },
            {
                "exclude": "omit",
                "record_id": "bu-FC-J-10023",
                "uuid": "feb18f7d-5d11-43dc-93f5-24668ac65d73"
            },
            {
                "exclude": "ok",
                "record_id": "bu-FC-J-10024",
                "uuid": "a49aa507-ffc2-4e11-842e-8b16a4a3ee3f"
            },
            {
                "exclude": "omit",
                "record_id": "bu-FC-J-10025",
                "uuid": "591afc5e-2287-40a0-8010-0e1f096cb325"
            }
        ],
        "reviewed": true,
        "value": 0.01441814607864897
    },
    "Ar40": {
        "error": 0.7319125809914061,
        "error_type": "",
        "fit": "Parabolic_SD",
        "references": [
            {
                "exclude": "ok",
                "record_id": "bu-FC-J-9999",
                "uuid": "203a75ef-33cf-4b5c-a8d7-4a68beb11cf9"
            },
            {
                "exclude": "ok",
                "record_id": "bu-FC-J-10000",
                "uuid": "610dbdbe-5663-41d5-a871-c318de82e871"
            },
            {
                "exclude": "ok",
                "record_id": "bu-FC-J-10001",
                "uuid": "156651dc-84f9-452e-8436-517922e14973"
            },
            {
                "exclude": "ok",
                "record_id": "bu-FC-J-10002",
                "uuid": "3f6c528a-833e-4094-b550-5526fd658d04"
            },
            {
                "exclude": "ok",
                "record_id": "bu-FC-J-10003",
                "uuid": "adee3e32-8b11-4a2b-bf9e-3afaa7f87cb4"
            },
            {
```


(continues on next page)

(continued from previous page)


```
            "exclude": "ok",
            "record_id": "bu-FC-J-10004",
            "uuid": "aadadc1b-b363-4824-ac84-8b78988cd925"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10005",
            "uuid": "ceae7843-b8bf-4979-80c4-38d47230f03d"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10006",
            "uuid": "b7802c50-bc12-494e-9b3b-f611dca52f39"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10007",
            "uuid": "9ed44d3e-2f30-4aa4-a57e-e36015154bcb"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10008",
            "uuid": "7cdb4a76-aae1-4299-8832-913b1fb90fbc"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10009",
            "uuid": "c5ae7ec6-504a-469a-aa88-f261638b0db6"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10010",
            "uuid": "39c8bca3-d1fd-49a3-8826-02e659b34a1b"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10011",
            "uuid": "0f91075c-852d-4193-96ec-9b229e5ae823"
        },
        {
            "exclude": "omit",
            "record_id": "bu-FC-J-10012",
            "uuid": "fa273be2-8805-4ac4-a2a2-5e53cf242eda"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10013",
            "uuid": "cf6926a6-28f7-4fa3-840e-b54ae4d47b0e"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10014",
```


(continues on next page)

(continued from previous page)


```
            "uuid": "236fb911-03b4-423e-8a64-7c545ad74d07"
        },
        {
            "exclude": "omit",
            "record_id": "bu-FC-J-10015",
            "uuid": "88b57793-559c-4036-8bab-1674c55dfb12"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10016",
            "uuid": "4db2f0a4-a125-4ea7-abf1-f36aeb3412bd"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10017",
            "uuid": "898e97c3-0e12-4b8c-aabb-053c540b1638"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10018",
            "uuid": "fd0b06c2-d005-417f-8140-acfe493d7efe"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10019",
            "uuid": "32bdfc37-6b20-4d6f-9dea-325f785f6825"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10020",
            "uuid": "c84ef7fe-5f08-4a78-ab63-66c81e72cd0d"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10021",
            "uuid": "41eb9da1-17f9-4d44-82b6-39bb31bcfb43"
        },
        {
            "exclude": "omit",
            "record_id": "bu-FC-J-10022",
            "uuid": "9e9a11d1-a714-45ec-9998-79aaf51dca3b"
        },
        {
            "exclude": "omit",
            "record_id": "bu-FC-J-10023",
            "uuid": "feb18f7d-5d11-43dc-93f5-24668ac65d73"
        },
        {
            "exclude": "ok",
            "record_id": "bu-FC-J-10024",
            "uuid": "a49aa507-ffc2-4e11-842e-8b16a4a3ee3f"
        },
```


(continues on next page)

(continued from previous page)


```
            {
                "exclude": "omit",
                "record_id": "bu-FC-J-10025",
                "uuid": "591afc5e-2287-40a0-8010-0e1f096cb325"
            }
        ],
        "reviewed": true,
        "value": 7.196981549647406
    },
    "Ar41": {
        "error": 0.0033257704726891275,
        "error_type": "",
        "fit": "previous",
        "references": [
            {
                "exclude": false,
                "record_id": "bu-FC-J-10010"
            }
        ],
        "value": 0.0298623430332032
    }
}
```


4.4.3 Extraction JSON

example. See also https://github.com/NMGRLData/Lusk01121/blob/master/666/extraction/81-03L.extr.json


```
{
    "beam_diameter": 1.25,
    "cleanup_duration": 240.0,
    "commit": "afadbe983d50e31a4821ee2e7a435b5847286752",
    "extract_device": "Fusions CO2",
    "extract_duration": 30.0,
    "extract_units": "watts",
    "extract_value": 0.2,
    "grain_polygons": [],
    "load_holder": "221-hole",
    "load_name": "C0014",
    "measured_response": "",
    "pattern": "",
    "positions": [
        {
            "is_degas": false,
            "position": "33",
            "x": null,
            "y": null,
            "z": null
        }
    ],
    "ramp_duration": 5.0,
    "ramp_rate": 0.0,
```


(continues on next page)

(continued from previous page)

```
  "requested_output": "",
  "setpoint_stream": "",
  "snapshots": [],
  "tray": "",
  "videos": [],
  "weight": 0.0
}
```

4.5 Database Analysis

4.5.1 Analysis Objects

IsotopeRecordView used for display in tables and fast creation and access.

IsotopeDatabaseManager.make_analyses(list_of_analyses) used to convert a IsotopeRecordView to a DBAnalysis. make_analyses retrieves the analysis from the db using a uuid. DBAnalysis is synced with the database record.

DBAnalysis is a subclass of ArArAge, Analysis. loading the isotopes from the db is the costliest process.

View of the analysis is handled by an AnalysisView. each DBAnalysis has an analysis_view object. the analysis is passed into analysis_view for creation.

AnalysisView is composed of multiple subview objects; MainView, HistoryView, ...

4.6 Automated Analysis

Automated analysis is handled by `ExperimentExecutor`, `ExperimentQueue`, and `AutomatedRun`.

4.6.1 ExperimentExecutor

`ExperimentExcecutor` is a top level object for coordinating the running of automated analyses.

4.6.2 ExperimentQueue

The `ExperimentQueue` contains a list of `AutomatedRunSpec`'s and some global metadata.

4.6.3 AutomatedRunSpec

An `AutomatedRunSpec` is a simple container object the holds all of the `AutomatedRun` information, such as labnumber, aliquot, pyscript names, etc.

4.6.4 AutomatedRun

The `AutomatedRun` contains the top level logic for executing an automated analysis. The `AutomatedRun` object is reused and the `ExperimentExecutor` has `measuring_run` and `extracting_run` objects. Two `AutomatedRun` objects are required to handle overlap. `AutomatedRun` is executed using `start()`.

Execution Sequence

1. Start button pressed, calls `ExperimentExecutor.execute`
2. pre execute check. `ExperimentExecutor._pre_execute_check`
3. New thread started. function= `ExperimentExecutor._execute`
4. Each queue in `ExperimentExecutor.experiment_queues` is run using `ExperimentExecutor._execute_queues`
5. pre run check
6. runspec retrieved from `new_runs_generator`
7. `AutomatedRun` updated with runspec data
8. if overlap new thread started else wait for run to complete return to step 5
9. `ExperimentExecutor._do_run` starts the `AutomatedRun`
    a) `AutoamtedRun._start`
    b) `AutoamtedRun._extraction`
    c) `AutoamtedRun._measurement`
    d) `AutoamtedRun._post_measurement`

4.6.5 PyScripts

Pyscript Sequence

1. Extraction
2. Measurement
3. Post Equilibration
4. Post Measurement

Each script is executed sequentially, except for measurement and post_equilibration. Post Equilibration and Measurement will be running concurrently after equilibration period has finished. If a script fails the run is stopped.

Extraction

Preforms the extraction of gas. Has access to valves and extraction devices. When finished gas should be staged for equilibration with the mass spectrometer

Measurement

Performs the measurement of the gas. Has access to valves and mass spectrometer.

Post Equilibration

Pumps out the extraction line following equilibration and isolation of the mass spectrometer from the extraction line. Has access to the valves.

Post Measurement

Runs after measurement is finished. Typically only pumps out the mass spectroemter. Has access to the valves.

4.6.6 AutomatedRunPersister

`AutoamtedRunPersister` is object used to save an analysis to the database. Uses `MassSpecDatabaseImporter` to save to MassSpec schema. `AutomatedRunPersister` uses `IsotopeAdapter` to save data to Pychron schema

Multiple database backends are handled by the `DataHub` and its *stores*. DataHub.main_store provides access to `IsotopeAdapter`.

MassSpecDatabaseImporter

4.6.7 DataHub

4.7 Experiment Construction

```
Class(name)

ExperimentEditorTask->Experimentor(manager)

Experimentor->ExperimentExecutor(executor)
            ->ExperimentFactory->QueueFactory(queue_factory)
                               ->AutomatedRunFactory(run_factory)->FactoryView(factory_
↪view)

ExperimentEditor->ExperimentQueue(queue)->List(executed_runs)
                                        ->List(automated_runs)


ExperimentFactoryPane -- QueueFactory, AutomatedRunFactory, FactoryView
```

4.7.1 New

4.7.2 Open

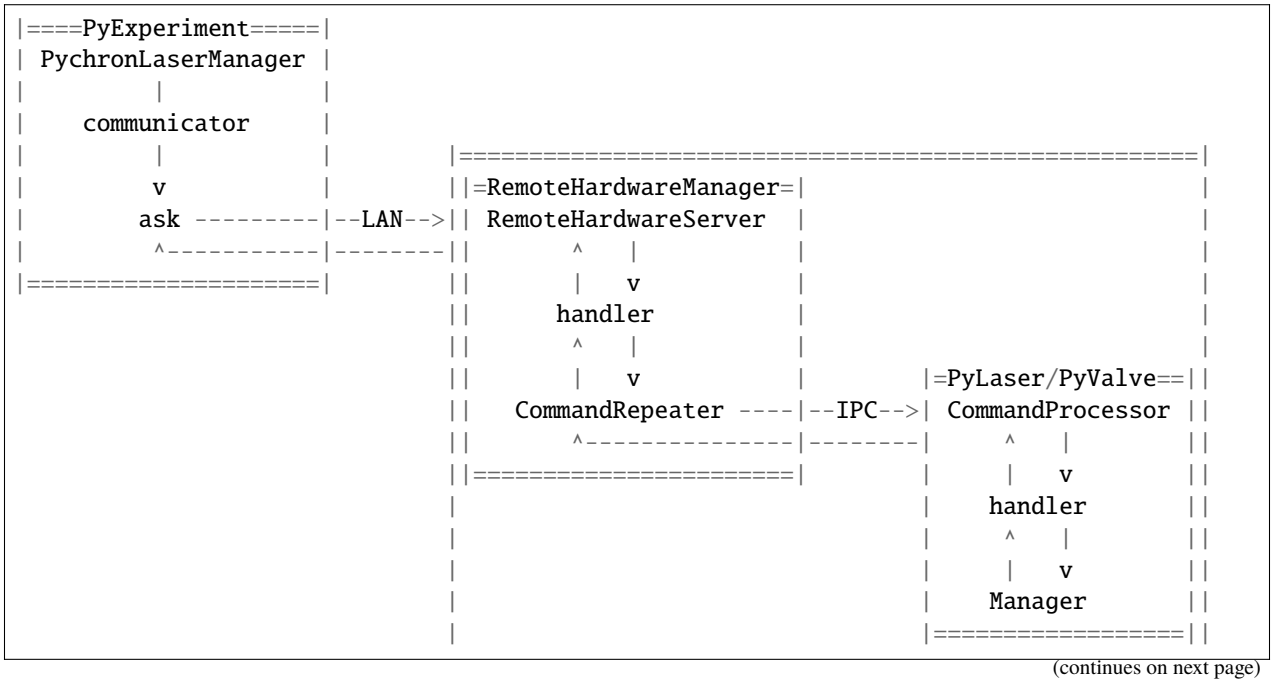
What Happens when Add Fired

```
_add_button_fired
         |
  add_consumable


  ConsumerMixin
         |
     _add_run
         |
AutomatedRunFactory
         |
      new_runs => runs, freq
                      |
             ExperimentQueue
               |
             add_runs
               |
             automated_runs.extend(runs)
```

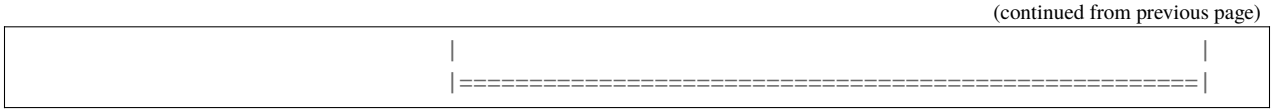
4.8 Communications

4.8.1 Pychron/Pychron Communication

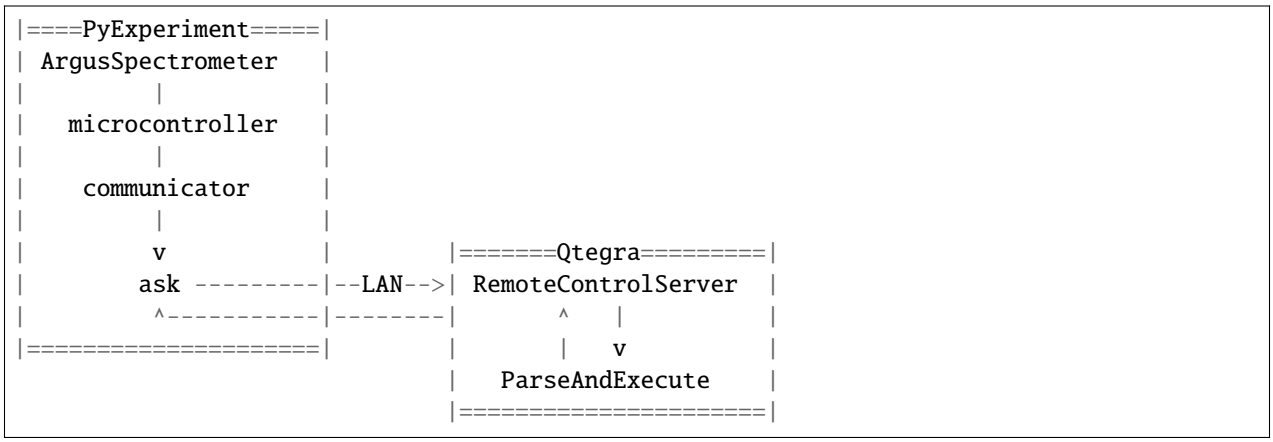


(continues on next page)

(continued from previous page)



4.8.2 Pychron/Qtegra Communications



4.8.3 Description

RemoteCommandServer is a top level object that doesnt do much. TCPServer is a subclass of SocketServer.ThreadingTCPServer TCPServer doesnt do much either. MessagingHandler is where the real work happens.

The TCPServer listens for incoming commands and creates a new MessageHandler object for each request ie command and calls its handle function

The handler object gets the data from the socket, processes it, and returns the result

To process requests the command is repeated using the CommandRepeater to the CommandProcessor living in Pychron

The CommandRepeater prepends a process_type string to the beginning of the request so the RemoteHardwareManager knows how to handle the request. The process_type and data are separated by a pipe character ex. 'System|Open V'

CommandProcessor is a simple socket server, (NOTE: isnt a subclass of a SocketServer, uses a listener thread to wait for requests from CommandRepeater)

When CommandProcessor receives a request its passed to RemoteHardwareManager.process_server_request

4.8.4 ——————————RemoteHardwareServer———————————

4.8.5 TCPServer –> MessagingHandler.handle –> CommandRepeater.get_response –>

4.8.6 ————————————————-Pychron———————————————

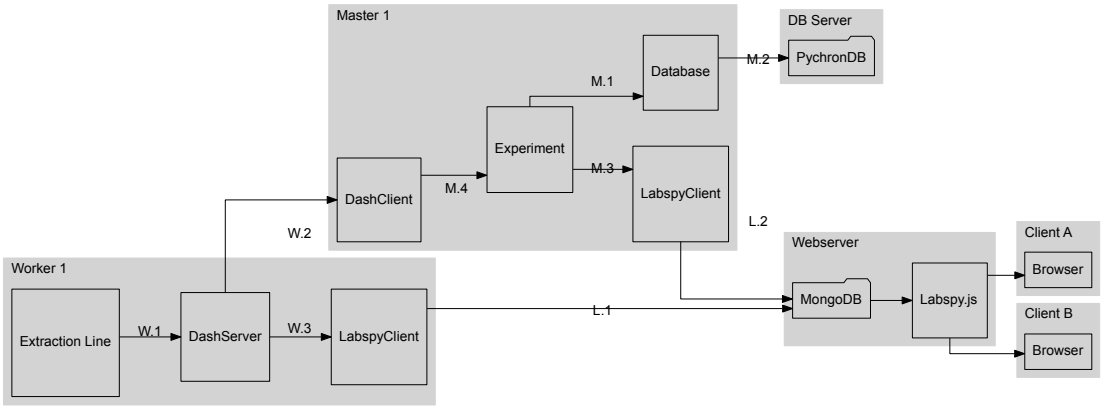
CommandProcessor._handler –> RemoteHardwareManager.process_server_request –>

4.8.7 [Request_type]Handler.handle –> result

return result up the call stack

4.9 Notification Network

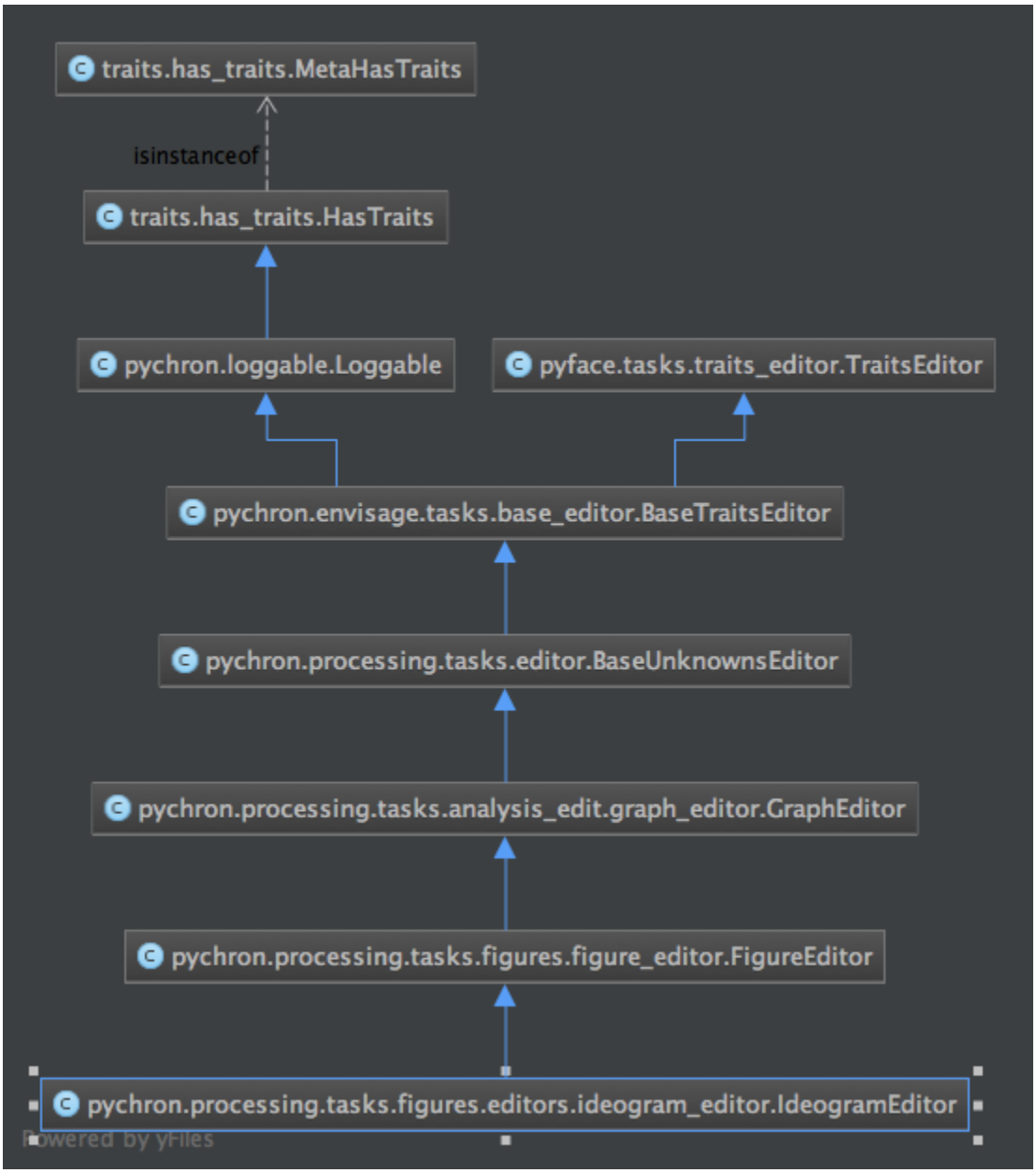
Dashboard + Labspy.js



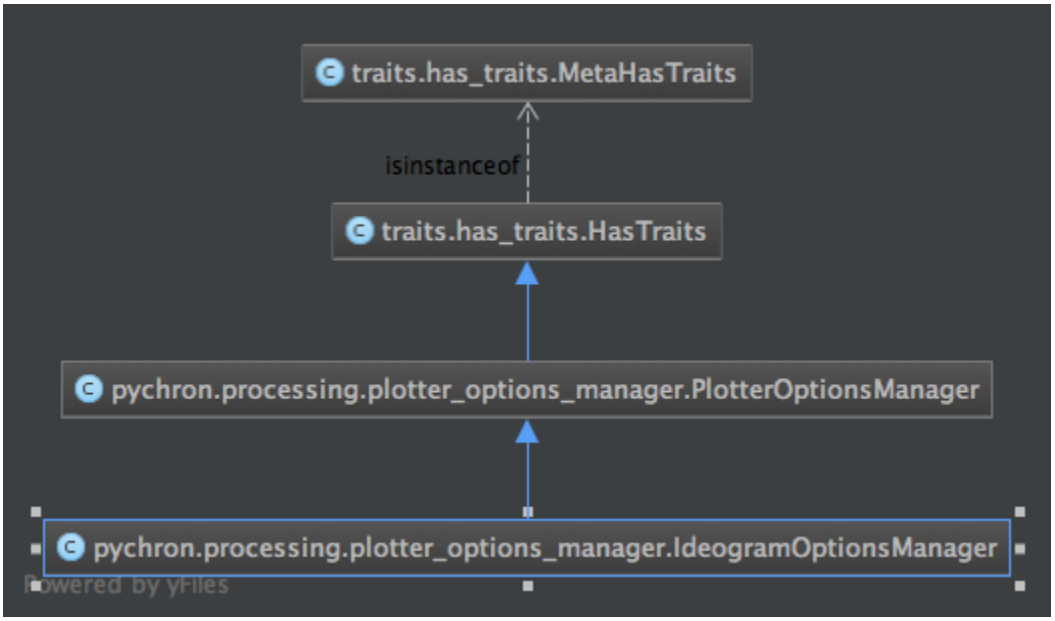
- **M1. Experiment - Database** – Experiment uses Database to save data
- **M2. Database - PychronDB** – Database saves data to a MySQL db
- **M3. Experiment - LabspyClient** – Experiment uses LabspyClient to update Labspy.js
- **M4. DashClient - Experiment** – Experiment tests DashClient error flag
- **W1. ExtractionLine - DashServer** – DashServer displays devices/values exposed by ExtractionLine
- **W2. DashServer - DashClient** – DashServer pushes notifications to DashClient
- **W3. DashServer - LabspyClient** – DashServer uses LabspyClient to update Labspy.js
- **L1. LabspyClient - MongoDB** – LabspyClient writes to MongoDB
- **L2. LabspyClient - MongoDB**

4.10 Figure

4.10.1 IdeogramEditor



4.10.2 Ideogram Plotter Options Manager



FigureEditor -> get_component [shape=circle] FigureEditor -> figure_container

4.11 Remote Hardware

Remote hardware is used to allow other software clients access to pychron hardware, such as valves and laser systems. A simple messaging system is used to pass information between pychron and a client. Currently the most active client is Mass Spec which uses the remote hardware protocol to do all of its hardware tasks.

The protocol is broken in two sections *Extraction Line Calls* and *Laser Calls*. Calls are simple ASCII text messages sent over the ethernet using either the UDP or TCP internet protocols

A response to a call is `OK`, a value, or an ErrorCode

4.11.1 Extraction Line Calls

`Open alias`
: Open the valve called `alias`. *InvalidValveErrorCode* return if `alias` not available

`Close alias`
: Close the valve called `alias`. *InvalidValveErrorCode* return if `alias` not available

`GetValveState alias`
: Get `alias` state. Returns `0` for closed `1` for open

`GetValveStates`
: Get all the valves states as a word. Returns a string <alias><state> e.g. `A1B0C1D1E0F0`

`GetValveLockStates`
: Get all the valves lock states as a word. Returns a string <alias><lock_state> e.g. `A1B0C1D1E0F0`

`StartMultRuns multruns_id`

CompleteMultRuns

StartRun runid

CompleteRun

PychronScript script

ScriptState

4.11.2 Laser Calls

Enable

Enable the laser. This is required before the laser's power can be set using *SetLaserPower*

Disable

Disable the laser

SetLaserPower power

Set the laser's power to `power`. `power` must be between 0-100.

ReadLaserPower

Read the lasers internal power meter. Returns an 8 bit value i.e 0-255

GetLaserStatus

Return OK if the laser can be enabled. If an interlock is enabled, such as insufficient coolant flow, an error will be returned

SetBeamDiameter

Set the beam diameter setting.

GetBeamDiameter

Get the beam diameter setting.

SetZoom zoom

Set zoom. `zoom` must be between 0-100.

GetZoom

Get zoom. returns value between 0-100.

GetPosition

Returns a comma separated list of positions X,Y,Z

GoToHole holenum

Go to hole `holename`. InvalidHoleErrorCode returned if hole is not in the current stage map.

GetJogProcedures

Return a list of available Jog procedures. Jog is a MassSpec term and a misnomer. Pychron internal refers to them as Patterns.

DoJog name

Launch the jog named `name`.

AbortJog

Abort the current jog

SetX xpos

Set the laser's stage controller X axis to `xpos`.

SetY ypos

Set the laser's stage controller X axis to `ypos`.

SetZ zpos

Set the laser's stage controller X axis to `zpos`.

SetXY xypos

Set the laser's stage controller X and Y axes to `xypos`. `xypos` should be a comma separated list of numbers. e.g `SetXY 10.1,-5.03`

GetXMoving

GetYMoving

GetDriveMoving

StopDrive

SetDriveHome

SetHomeX

SetHomeY

SetHomeZ

4.12 Extraction Line Error Codes

Name	Code	Description
Name	Code	Message
PyScriptErrorCode	001	invalid pyscript None does not exist
FuncCallErrorCode	002	func call problem: err= None args= None
InvalidCommandErrorCode	003	invalid command: None
InvalidArgumentsErrorCode	004	invalid arguments: None None
InvalidValveErrorCode	005	None is not a registered valve name
InvalidValveGroupErrorCode	006	Invalid valve group - None
ManagerUnavaliableError-Code	007	manager unavaliable: None
DeviceConnectionErrorCode	008	device None not connected
InvalidIPAddressErrorCode	009	None is not a registered ip address
NoResponseErrorCode	010	no response from device
PychronCommErrorCode	011	could not communicate with pychron through None. socket.error = None
SystemLockErrorCode	012	Access restricted to None (None). You are None
SecurityErrorCode	013	Not an approved ip address None
ValveSoftwareLockErrorCode	014	Valve None is software locked
ValveActuationErrorCode	015	Valve None failed to actuate None

PychronCommErrorCode

InvalidArgumentsErrorCode

ManagerUnavaliableErrorCode

InvalidValveGroupErrorCode

InvalidValveErrorCode

DeviceConnectionErrorCode

ValveActuationErrorCode

FuncCallErrorCode

InvalidCommandErrorCode

SecurityErrorCode

PyScriptErrorCode

InvalidIPAddressErrorCode

ValveSoftwareLockErrorCode

NoResponseErrorCode

SystemLockErrorCode

4.13 Laser Error Codes

Name	Code	Description
InvalidSampleHolderErrorCode	201	Invalid sample holder. {}
EnableErrorCode	202	Laser failed to enable {}
LogicBoardCommErrorCode	203	Failed communication with logic board
DisableErrorCode	204	Laser failed to disable {}

InvalidSampleHolderErrorCode

EnableErrorCode

LogicBoardCommErrorCode

DisableErrorCode

4.14 Update Plugin

The update plugin is used to manage the current pychron version. It uses Git as a backend for version control and distribution.

pychron stores its source code in a git repository located at [pychron_root]/.hidden/pychron

if the repo doesn't exist it is cloned. The source code repository name and default branch are stored in preferences. Default values of `NMGRL/pychron` and `master` are provided.

4.14.1 Update Process

- check for updates
  - compare the local commit to the remote commit
- if updates available
  - pull updates
  - build egg and resources
  - move egg and resources into the application bundle.

4.15 Style Guide

- Follow pep8.
- use 4 spaces
- max line length 120
- method/functions all lowercase

`python def foobar(): `

- classes PascalCase

`python class FooBar: `

- variables should be all lowercase

`python x = 10 xmax = 102 x_min =0 `

- global variables all UPPERCASE

`python DEBUG = True `

- use single quotes for strings

`python msg = 'Hello world' `

- use triple double quotes for doc strings

``` def foobar():
    “”” this is a docstring “””
```

- import individual items from numpy

` from numpy import array `

DO NOT USE

` import numpy as np `

- multiline list, dict, tuples. No orphaned opening or closing brackets

``` x = [1,2,3,
    4,5,6]
**d = {‘a’: 1,**
    ‘b’: 2}
```

t= (1,2,
3,4)

``` DO NOT WRITE

``` x=[

1,2,3 ]

**d = {'a': 1,**
'b': 2 }

```

4.15.1 Pycharm Template

Use this template for new files. Pycharm templates are located in Preferences>Editor>File and Code Templates. note for Windows, Preferences are referred to as Settings

```python # ===============================================================================
# Copyright ${YEAR} ${USER} # # Licensed under the Apache License, Version 2.0 (the "License"); # you may not use this file except in compliance with the License. # You may obtain a copy of the License at # # http://www.apache.org/licenses/LICENSE-2.0 # # Unless required by applicable law or agreed to in writing, software # distributed under the License is distributed on an "AS IS" BASIS, # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. # See the License for the specific language governing permissions and # limitations under the License.
# ===============================================================================

# ============= standard library imports ======================== # ============= local library imports ==========================

# ============= EOF ============================================= ```

## 4.16 Interpreted Ages

### 4.16.1 JSON format

```
{
"analyses": [
    {
        "age": float,
        "age_err": float,
        "age_err_wo_j": float,
        "baseline_corrected_intercepts": {},
        "blanks": {},
        "extract_value": float,
        "ic_corrected_values": {},
        "icfactors": {},
        "interference_corrected_values": {},
        "kca": float,
        "kca_err": float,
        "kcl": float,
        "kcl_err": float,
        "plateau_step": false,
        "radiogenic_yield": float,
```

(continues on next page)

(continued from previous page)

```
        "radiogenic_yield_err": float,
        "record_id": str,
        "tag": str,
        "uuid": str
    },

],
"name": str,
"preferred": {
    "age": float,
    "age_err": float,
    "ages": {
        "integrated_age": float,
        "integrated_age_err": float,
        "isochron_age": float,
        "isochron_age_err": float,
        "plateau_age": float,
        "plateau_age_err": float,
        "weighted_age": float,
        "weighted_age_err": float
    },
    "arar_constants": {
        "abundance_sensitivity": float,
        "atm4036": float,
        "atm4036_err": float,
        "atm4038": float,
        "atm4038_err": float,
        "fixed_k3739": float,
        "fixed_k3739_err": float,
        "lambda_Ar37": float,
        "lambda_Ar37_err": float,
        "lambda_Ar39": float,
        "lambda_Ar39_err": float,
        "lambda_Cl36": float,
        "lambda_Cl36_err": float,
        "lambda_k": float,
        "lambda_k_err": float
    },
    "display_age_units": str,
    "include_j_error_in_mean": bool,
    "include_j_error_in_plateau": bool,
    "include_j_position_error": bool,
    "monitor_age": float,
    "monitor_reference": str,
    "mswd": float,
    "nanalyses": int,
    "preferred_kinds": [
        {
            "attr": <"age", "kca", "kcl", "radiogenic_yield", "moles_k39", "signal_k39"
            "error": float,
            "error_kind": <"SD", "SEM", "SEM, but if MSWD>1 use SEM * sqrt(MSWD)">,
            "kind": <"Weighted Mean", >,
```

(continues on next page)

(continued from previous page)

```
            "value": float,
            "weighting": str
        },

    ]
},
"sample_metadata": {
    "grainsize": str,
    "irradiation": str,
    "irradiation_level": str,
    "irradiation_position": int,
    "latitude": float,
    "lithology": str,
    "lithology_class": str,
    "lithology_group": str,
    "lithology_type": str,
    "longitude": float,
    "material": str,
    "principal_investigator": str,
    "project": str,
    "rlocation": str,
    "sample": str
},
"uuid": str
}
```

4.16.2 Example


```
{
"analyses": [
    {
        "age": 27.719555312784266,
        "age_err": 0.48224260012401765,
        "age_err_wo_j": 0.48224260012401765,
        "baseline_corrected_intercepts": {
            "Ar36": {
                "error": 0.0005994962038630668,
                "value": 0.04723067451378344
            },
            "Ar37": {
                "error": 0.005503939925822024,
                "value": 0.11309640278377814
            },
            "Ar38": {
                "error": 0.002341430315982881,
                "value": 0.2361151444228655
            },
            "Ar39": {
                "error": 0.013954133127184293,
                "value": 15.974211414208787
```


(continues on next page)

(continued from previous page)


```
            },
            "Ar40": {
                "error": 0.01590441601008484,
                "value": 74.78410787929032
            },
            "Ar41": {
                "error": 0.001530540372273 1324,
                "value": 0.013232090318390084
            }
        },
        "blanks": {
            "Ar36": {
                "error": 0.002530676861539285,
                "value": 0.026285578599641108
            },
            "Ar37": {
                "error": 0.00786620718599018,
                "value": 0.03449851337958194
            },
            "Ar38": {
                "error": 0.004198981148223098,
                "value": 0.01028580551619398
            },
            "Ar39": {
                "error": 0.012217213055969865,
                "value": 0.01441814607864897
            },
            "Ar40": {
                "error": 0.7319125809914061,
                "value": 7.196981549647406
            },
            "Ar41": {
                "error": 0.0033257704726891275,
                "value": 0.0298623430332032
            }
        },
        "ic_corrected_values": {
            "Ar36": {
                "error": 0.002615646198459 1295,
                "value": 0.02106494810372595
            },
            "Ar37": {
                "error": 0.009600550515464271,
                "value": 0.07859788940419621
            },
            "Ar38": {
                "error": 0.004807674969019501,
                "value": 0.2258293389066715
            },
            "Ar39": {
                "error": 0.0189981835950715,
                "value": 15.963026765621615
```


(continues on next page)

(continued from previous page)


```
            },
            "Ar40": {
                "error": 0.7320853615953016,
                "value": 67.58712632964291
            },
            "Ar41": {
                "error": 0.0036610521804761047,
                "value": -0.016630252714813117
            }
        },
        "icfactors": {
            "Ar36": {
                "error": 0.0007633894469259384,
                "value": 1.0057222077222714
            },
            "Ar37": {
                "error": 1e-20,
                "value": 1.0
            },
            "Ar38": {
                "error": 1e-20,
                "value": 1.0
            },
            "Ar39": {
                "error": 0.00025692197839542535,
                "value": 1.0002026027177893
            },
            "Ar40": {
                "error": 1e-20,
                "value": 1.0
            },
            "Ar41": {
                "error": 1e-20,
                "value": 1.0
            }
        },
        "interference_corrected_values": {
            "Ar36": {
                "error": 0.0026156750775804503,
                "value": 0.021023296922508376
            },
            "Ar37": {
                "error": 0.017507863602034712,
                "value": 0.14333351953315343
            },
            "Ar38": {
                "error": 0.004807674969019972,
                "value": 0.225827734744206
            },
            "Ar39": {
                "error": 0.019002275618033367,
                "value": 15.966353214167436
```


(continues on next page)

(continued from previous page)


```
            },
            "Ar40": {
                "error": 0.7321010417064032,
                "value": 67.48517797176511
            },
            "Ar41": {
                "error": 0.0,
                "value": 0.0
            }
        },
        "kca": 57.86448398141012,
        "kca_err": 7.199353476239163,
        "kcl": 1758.301586538721,
        "kcl_err": 213.78707269155652,
        "plateau_step": false,
        "radiogenic_yield": 90.56231365030379,
        "radiogenic_yield_err": 1.1600258740023146,
        "record_id": "66714-01A",
        "tag": "ok",
        "uuid": "e4edbbed-1fae-413d-9d29-42bd6553f61a"
    },
    {
        "age": 27.19176900710652,
        "age_err": 0.011303308461930284,
        "age_err_wo_j": 0.011303308461930284,
        "baseline_corrected_intercepts": {
            "Ar36": {
                "error": 0.0005885739390490279,
                "value": 0.03852176691264056
            },
            "Ar37": {
                "error": 0.0057490619669011075,
                "value": 3.45429957981653
            },
            "Ar38": {
                "error": 0.0033829188468008317,
                "value": 10.134664901311618
            },
            "Ar39": {
                "error": 0.056042465844671034,
                "value": 892.6225273747481
            },
            "Ar40": {
                "error": 0.12363467388955417,
                "value": 3373.5458784531465
            },
            "Ar41": {
                "error": 0.0015080380730257462,
                "value": 0.12610673955911397
            }
        },
        "blanks": {
```


(continues on next page)

(continued from previous page)


```
            "Ar36": {
                "error": 0.002530676861621177,
                "value": 0.02616848527176168
            },
            "Ar37": {
                "error": 0.00786620718599018,
                "value": 0.03449851337958194
            },
            "Ar38": {
                "error": 0.004198981148223098,
                "value": 0.01028580551619398
            },
            "Ar39": {
                "error": 0.012217213055969865,
                "value": 0.01441814607864897
            },
            "Ar40": {
                "error": 0.7319126030362056,
                "value": 7.161701124116693
            },
            "Ar41": {
                "error": 0.0033257704726891275,
                "value": 0.0298623430332032
            }
        },
        "ic_corrected_values": {
            "Ar36": {
                "error": 0.002613104085564699,
                "value": 0.012423969684479709
            },
            "Ar37": {
                "error": 0.009743147796897693,
                "value": 3.419801066436948
            },
            "Ar38": {
                "error": 0.005392177909451174,
                "value": 10.124379095795424
            },
            "Ar39": {
                "error": 0.22919253475203238,
                "value": 892.7479146167714
            },
            "Ar40": {
                "error": 0.7422813422624946,
                "value": 3366.3841773290296
            },
            "Ar41": {
                "error": 0.003651702078032389,
                "value": 0.09624439652591077
            }
        },
        "icfactors": {
```


(continues on next page)

(continued from previous page)


```
            "Ar36": {
                "error": 0.0007633894469259384,
                "value": 1.0057222077222714
            },
            "Ar37": {
                "error": 1e-20,
                "value": 1.0
            },
            "Ar38": {
                "error": 1e-20,
                "value": 1.0
            },
            "Ar39": {
                "error": 0.000248593700629227,
                "value": 1.0001566257203542
            },
            "Ar40": {
                "error": 1e-20,
                "value": 1.0
            },
            "Ar41": {
                "error": 1e-20,
                "value": 1.0
            }
        },
        "interference_corrected_values": {
            "Ar36": {
                "error": 0.0026131461049083635,
                "value": 0.010708269470396504
            },
            "Ar37": {
                "error": 0.017770194313444784,
                "value": 6.237256451140361
            },
            "Ar38": {
                "error": 0.005392177909499971,
                "value": 10.124289479023856
            },
            "Ar39": {
                "error": 0.22924186600949456,
                "value": 892.9352934062675
            },
            "Ar40": {
                "error": 0.7891411431378178,
                "value": 3360.6826069310478
            },
            "Ar41": {
                "error": 0.0,
                "value": 0.0
            }
        },
        "kca": 74.75390813988191,
```


(continues on next page)

(continued from previous page)


```
        "kca_err": 0.39479729918549195,
        "kcl": -81096.19073918163,
        "kcl_err": 45898.849977837715,
        "plateau_step": false,
        "radiogenic_yield": 99.73566732777049,
        "radiogenic_yield_err": 0.024504026259052673,
        "record_id": "66714-01B",
        "tag": "ok",
        "uuid": "07db433f-1ef3-4f9f-a86b-4257bc7b66ec"
    },
    {
        "age": 28.077041834912727,
        "age_err": 1.0831169368560338,
        "age_err_wo_j": 1.0831169368560338,
        "baseline_corrected_intercepts": {
            "Ar36": {
                "error": 0.0004299589870458764,
                "value": 0.028314796367070936
            },
            "Ar37": {
                "error": 0.005602722653107384,
                "value": 0.0806406276715168
            },
            "Ar38": {
                "error": 0.0024130480783469245,
                "value": 0.12608958192546776
            },
            "Ar39": {
                "error": 0.013245702205643093,
                "value": 7.070455599926101
            },
            "Ar40": {
                "error": 0.013957558105976904,
                "value": 35.249057192632584
            },
            "Ar41": {
                "error": 0.0016832882633947878,
                "value": 0.012745520931239408
            }
        },
        "blanks": {
            "Ar36": {
                "error": 0.002530676861972484,
                "value": 0.025626797314095968
            },
            "Ar37": {
                "error": 0.00786620718599018,
                "value": 0.03449851337958194
            },
            "Ar38": {
                "error": 0.004198981148223098,
                "value": 0.01028580551619398
```


(continues on next page)

(continued from previous page)


```
            },
            "Ar39": {
                "error": 0.012217213055969865,
                "value": 0.01441814607864897
            },
            "Ar40": {
                "error": 0.7319126965606669,
                "value": 6.998865020529088
            },
            "Ar41": {
                "error": 0.0033257704726891275,
                "value": 0.0298623430332032
            }
        },
        "ic_corrected_values": {
            "Ar36": {
                "error": 0.002581631171217421,
                "value": 0.0027033803419133596
            },
            "Ar37": {
                "error": 0.009657521246187668,
                "value": 0.04614211429193487
            },
            "Ar38": {
                "error": 0.004842958157112939,
                "value": 0.11580377640927378
            },
            "Ar39": {
                "error": 0.01809056121293549,
                "value": 7.055615611059068
            },
            "Ar40": {
                "error": 0.7320457696175756,
                "value": 28.250192172103496
            },
            "Ar41": {
                "error": 0.003727493610282049,
                "value": -0.017116822101963792
            }
        },
        "icfactors": {
            "Ar36": {
                "error": 0.0007633894469259384,
                "value": 1.0057222077222714
            },
            "Ar37": {
                "error": 1e-20,
                "value": 1.0
            },
            "Ar38": {
                "error": 1e-20,
                "value": 1.0
```


(continues on next page)

(continued from previous page)


```
            },
            "Ar39": {
                "error": 0.0002284464313923612,
                "value": 0.999940215341664
            },
            "Ar40": {
                "error": 1e-20,
                "value": 1.0
            },
            "Ar41": {
                "error": 1e-20,
                "value": 1.0
            }
        },
        "interference_corrected_values": {
            "Ar36": {
                "error": 0.0025816598854381256,
                "value": 0.0026782432606453515
            },
            "Ar37": {
                "error": 0.017624693242188124,
                "value": 0.08420798467278917
            },
            "Ar38": {
                "error": 0.004842958157113373,
                "value": 0.11580306623350124
            },
            "Ar39": {
                "error": 0.018094462938368167,
                "value": 7.057073121479985
            },
            "Ar40": {
                "error": 0.7320488406202872,
                "value": 28.205131349099723
            },
            "Ar41": {
                "error": 0.0,
                "value": 0.0
            }
        },
        "kca": 43.33891696981006,
        "kca_err": 9.195528920548588,
        "kcl": 890.4075256113907,
        "kcl_err": 124.47542755103413,
        "plateau_step": false,
        "radiogenic_yield": 97.01001764226808,
        "radiogenic_yield_err": 2.729519464106105,
        "record_id": "66714-02A",
        "tag": "ok",
        "uuid": "3d0fc7d9-fdbe-4273-91d4-3d08fa3d3875"
    },
    {
```


(continues on next page)

(continued from previous page)


```
        "age": 27.20578856468821,
        "age_err": 0.008984828080871995,
        "age_err_wo_j": 0.008984828080871995,
        "baseline_corrected_intercepts": {
            "Ar36": {
                "error": 0.0005156569867258774,
                "value": 0.03268865013767039
            },
            "Ar37": {
                "error": 0.006285618771135262,
                "value": 4.692054070934782
            },
            "Ar38": {
                "error": 0.0035473824640944057,
                "value": 14.668141452121183
            },
            "Ar39": {
                "error": 0.0714188158731977,
                "value": 1296.8816804669736
            },
            "Ar40": {
                "error": 0.1495589243817402,
                "value": 4896.041232850392
            },
            "Ar41": {
                "error": 0.001549585156482157,
                "value": 0.1839271092016435
            }
        },
        "blanks": {
            "Ar36": {
                "error": 0.002530676862038943,
                "value": 0.025514325015847744
            },
            "Ar37": {
                "error": 0.00786620718599018,
                "value": 0.03449851337958194
            },
            "Ar38": {
                "error": 0.004198981148223098,
                "value": 0.01028580551619398
            },
            "Ar39": {
                "error": 0.012217213055969865,
                "value": 0.01441814607864897
            },
            "Ar40": {
                "error": 0.7319127140768318,
                "value": 6.965135517701946
            },
            "Ar41": {
                "error": 0.0033257704726891275,
```


(continues on next page)

(continued from previous page)


```
                "value": 0.0298623430332032
            }
        },
        "ic_corrected_values": {
            "Ar36": {
                "error": 0.0025974627554178,
                "value": 0.0072153781004368235
            },
            "Ar37": {
                "error": 0.010069072391683928,
                "value": 4.657555557555201
            },
            "Ar38": {
                "error": 0.005496850464556723,
                "value": 14.65785564660499
            },
            "Ar39": {
                "error": 0.3053349690822899,
                "value": 1296.7304212151641
            },
            "Ar40": {
                "error": 0.7470368751872541,
                "value": 4889.07609733269
            },
            "Ar41": {
                "error": 0.0036690548366303676,
                "value": 0.1540647661684403
            }
        },
        "icfactors": {
            "Ar36": {
                "error": 0.0007633894469259384,
                "value": 1.0057222077222714
            },
            "Ar37": {
                "error": 1e-20,
                "value": 1.0
            },
            "Ar38": {
                "error": 1e-20,
                "value": 1.0
            },
            "Ar39": {
                "error": 0.0002287167863376534,
                "value": 0.9998944833371105
            },
            "Ar40": {
                "error": 1e-20,
                "value": 1.0
            },
            "Ar41": {
                "error": 1e-20,
```


(continues on next page)

(continued from previous page)


```
                "value": 1.0
            }
        },
        "interference_corrected_values": {
            "Ar36": {
                "error": 0.002597519173467174,
                "value": 0.004883069506186372
            },
            "Ar37": {
                "error": 0.018378118561798284,
                "value": 8.50098975988744
            },
            "Ar38": {
                "error": 0.005496850464641618,
                "value": 14.657725507807312
            },
            "Ar39": {
                "error": 0.30540076957693696,
                "value": 1297.0033286855808
            },
            "Ar40": {
                "error": 0.8422989275748238,
                "value": 4880.794471732948
            },
            "Ar41": {
                "error": 0.0,
                "value": 0.0
            }
        },
        "kca": 79.79005818318248,
        "kca_err": 0.4144529334766862,
        "kcl": -49088.79633100066,
        "kcl_err": 16638.294686564474,
        "plateau_step": false,
        "radiogenic_yield": 99.8007955734752,
        "radiogenic_yield_err": 0.017746891705874197,
        "record_id": "66714-02B",
        "tag": "ok",
        "uuid": "3c110d6c-e8ea-4947-8561-ad8df56f48f6"
    },
    {
        "age": 27.05196522081795,
        "age_err": 0.10183095672511248,
        "age_err_wo_j": 0.10183095672511248,
        "baseline_corrected_intercepts": {
            "Ar36": {
                "error": 0.001704049992893757,
                "value": 0.4254132837797189
            },
            "Ar37": {
                "error": 0.005890772322284931,
                "value": 2.421729792372489
```


(continues on next page)

(continued from previous page)


```
            },
            "Ar38": {
                "error": 0.0021662585558443427,
                "value": 1.1360499350201467
            },
            "Ar39": {
                "error": 0.020065294937529957,
                "value": 83.83985323379515
            },
            "Ar40": {
                "error": 0.039711915054936875,
                "value": 440.7745775100887
            },
            "Ar41": {
                "error": 0.0016883051991301717,
                "value": 0.029795339207878906
            }
        },
        "blanks": {
            "Ar36": {
                "error": 0.0025306768621025056,
                "value": 0.025402549702945698
            },
            "Ar37": {
                "error": 0.00786620718599018,
                "value": 0.03449851337958194
            },
            "Ar38": {
                "error": 0.004198981148223098,
                "value": 0.01028580551619398
            },
            "Ar39": {
                "error": 0.012217213055969865,
                "value": 0.01441814607864897
            },
            "Ar40": {
                "error": 0.7319127307822177,
                "value": 6.931643673110177
            },
            "Ar41": {
                "error": 0.0033257704726891275,
                "value": 0.0298623430332032
            }
        },
        "ic_corrected_values": {
            "Ar36": {
                "error": 0.0030835352429197617,
                "value": 0.40229967858829874
            },
            "Ar37": {
                "error": 0.009827431711587812,
                "value": 2.387231278992907
```


(continues on next page)

(continued from previous page)


```
            },
            "Ar38": {
                "error": 0.00472484061253941,
                "value": 1.1257641295039527
            },
            "Ar39": {
                "error": 0.030420992914993072,
                "value": 83.81275660236166
            },
            "Ar40": {
                "error": 0.7329892780105407,
                "value": 433.84293383697855
            },
            "Ar41": {
                "error": 0.003729761880123292,
                "value": -6.700382532429452e-05
            }
        },
        "icfactors": {
            "Ar36": {
                "error": 0.0007633894469259384,
                "value": 1.0057222077222714
            },
            "Ar37": {
                "error": 1e-20,
                "value": 1.0
            },
            "Ar38": {
                "error": 1e-20,
                "value": 1.0
            },
            "Ar39": {
                "error": 0.00023062248286486071,
                "value": 0.9998487513325571
            },
            "Ar40": {
                "error": 1e-20,
                "value": 1.0
            },
            "Ar41": {
                "error": 1e-20,
                "value": 1.0
            }
        },
        "interference_corrected_values": {
            "Ar36": {
                "error": 0.0030835688313353674,
                "value": 0.4010702882022748
            },
            "Ar37": {
                "error": 0.01793936914076707,
                "value": 4.357742862409555
```


(continues on next page)

(continued from previous page)


```
            },
            "Ar38": {
                "error": 0.0047248406125404915,
                "value": 1.1257555095051646
            },
            "Ar39": {
                "error": 0.030427563432013357,
                "value": 83.82768763313119
            },
            "Ar40": {
                "error": 0.7334205869901604,
                "value": 433.3076772888897
            },
            "Ar41": {
                "error": 0.0,
                "value": 0.0
            }
        },
        "kca": 9.844837783377043,
        "kca_err": 0.041208242888349955,
        "kcl": 3879.757091962007,
        "kcl_err": 220.00581486326422,
        "plateau_step": false,
        "radiogenic_yield": 72.2759597316246,
        "radiogenic_yield_err": 0.21926992957115193,
        "record_id": "66714-03A",
        "tag": "ok",
        "uuid": "353ecedf-1138-491d-b00a-708627bb1d17"
    },
    {
        "age": 27.257140377955807,
        "age_err": 0.009870173988038862,
        "age_err_wo_j": 0.009870173988038862,
        "baseline_corrected_intercepts": {
            "Ar36": {
                "error": 0.0011740404627554493,
                "value": 0.17608560651941374
            },
            "Ar37": {
                "error": 0.0066242839210119125,
                "value": 8.655850950817861
            },
            "Ar38": {
                "error": 0.003368833749325161,
                "value": 13.104874179515685
            },
            "Ar39": {
                "error": 0.06709999536211372,
                "value": 1141.7321025755205
            },
            "Ar40": {
                "error": 0.14601043577843667,
```


(continues on next page)

(continued from previous page)


```
                "value": 4361.511631866489
            },
            "Ar41": {
                "error": 0.0015375250230911944,
                "value": 0.1472912533230021 4
            }
        },
        "blanks": {
            "Ar36": {
                "error": 0.002530676862163096,
                "value": 0.02529147137538983
            },
            "Ar37": {
                "error": 0.00786620718599018,
                "value": 0.03449851337958194
            },
            "Ar38": {
                "error": 0.004198981148223098,
                "value": 0.01028580551619398
            },
            "Ar39": {
                "error": 0.012217213055969865,
                "value": 0.01441814607864897
            },
            "Ar40": {
                "error": 0.7319127466664469,
                "value": 6.8983894867537785
            },
            "Ar41": {
                "error": 0.0033257704726891275,
                "value": 0.0298623430332032
            }
        },
        "ic_corrected_values": {
            "Ar36": {
                "error": 0.002808072473526074,
                "value": 0.15165701050861827
            },
            "Ar37": {
                "error": 0.010283888027351354,
                "value": 8.62135243743828
            },
            "Ar38": {
                "error": 0.005383352441901346,
                "value": 13.09458837399949
            },
            "Ar39": {
                "error": 0.2758636521557973,
                "value": 1141.4927881127328
            },
            "Ar40": {
                "error": 0.7463345872254825,
```


(continues on next page)

(continued from previous page)


```
                "value": 4354.613242379735
            },
            "Ar41": {
                "error": 0.0036639776791954452,
                "value": 0.11742891028979893
            }
        },
        "icfactors": {
            "Ar36": {
                "error": 0.0007633894469259384,
                "value": 1.0057222077222714
            },
            "Ar37": {
                "error": 1e-20,
                "value": 1.0
            },
            "Ar38": {
                "error": 1e-20,
                "value": 1.0
            },
            "Ar39": {
                "error": 0.00023412359079403406,
                "value": 0.9998030193280036
            },
            "Ar40": {
                "error": 1e-20,
                "value": 1.0
            },
            "Ar41": {
                "error": 1e-20,
                "value": 1.0
            }
        },
        "interference_corrected_values": {
            "Ar36": {
                "error": 0.0028081313819327625,
                "value": 0.14726825995280343
            },
            "Ar37": {
                "error": 0.018775005530363115,
                "value": 15.739758580779586
            },
            "Ar38": {
                "error": 0.0053833524419721265,
                "value": 13.094473362585436
            },
            "Ar39": {
                "error": 0.2759231389141926,
                "value": 1141.7272701964312
            },
            "Ar40": {
                "error": 0.8211803243999938,
```


(continues on next page)

(continued from previous page)


```
                "value": 4347.323085460973
            },
            "Ar41": {
                "error": 0.0,
                "value": 0.0
            }
        },
        "kca": 37.44364818670555,
        "kca_err": 0.09445756296981635,
        "kcl": 81362.83610704304,
        "kcl_err": 45829.61448767505,
        "plateau_step": false,
        "radiogenic_yield": 98.82289562480149,
        "radiogenic_yield_err": 0.020825217118115198,
        "record_id": "66714-03B",
        "tag": "ok",
        "uuid": "3a0f1610-a843-471d-982c-ac5ff9db3fd9"
    }
],
"name": "01",
"preferred": {
    "age": 27.21904871046781,
    "age_err": 0.014539368948486097,
    "ages": {
        "integrated_age": 27.224606105198074,
        "integrated_age_err": 0.04786378364736604,
        "isochron_age": 27.220009733430548,
        "isochron_age_err": 0.016358126816747635,
        "plateau_age": 27.19980252231228,
        "plateau_age_err": 0.007015880443098190 5,
        "weighted_age": 27.21904871046781,
        "weighted_age_err": 0.014539368948486097
    },
    "arar_constants": {
        "abundance_sensitivity": 1e-07,
        "atm4036": 298.56,
        "atm4036_err": 0.31,
        "atm4038": 1583.87,
        "atm4038_err": 3.01,
        "fixed_k3739": 0.00013,
        "fixed_k3739_err": 0.011,
        "lambda_Ar37": 0.0197500001,
        "lambda_Ar37_err": 0.01,
        "lambda_Ar39": 7.0680001e-06,
        "lambda_Ar39_err": 0.01,
        "lambda_Cl36": 6.308e-09,
        "lambda_Cl36_err": 0.0,
        "lambda_k": 5.464e-10,
        "lambda_k_err": 0.0
    },
    "display_age_units": "Ma",
    "include_j_error_in_mean": true,
```


(continues on next page)

(continued from previous page)


```
    "include_j_error_in_plateau": true,
    "include_j_position_error": false,
    "mswd": 5.4587149741362255,
    "nanalyses": 6,
    "preferred_kinds": [
        {
            "attr": "age",
            "error": 0.014539368948486097,
            "error_kind": "SEM, but if MSWD>1 use SEM * sqrt(MSWD)",
            "kind": "Weighted Mean",
            "value": 27.21904871046781,
            "weighting": ""
        },
        {
            "attr": "kca",
            "error": 5.786677620450036,
            "error_kind": "SEM, but if MSWD>1 use SEM * sqrt(MSWD)",
            "kind": "Weighted Mean",
            "value": 15.338441091495552,
            "weighting": ""
        },
        {
            "attr": "kcl",
            "error": 539.3999618451804,
            "error_kind": "SEM, but if MSWD>1 use SEM * sqrt(MSWD)",
            "kind": "Weighted Mean",
            "value": 1642.7849155072458,
            "weighting": ""
        },
        {
            "attr": "radiogenic_yield",
            "error": 0.6851802497583488,
            "error_kind": "SEM, but if MSWD>1 use SEM * sqrt(MSWD)",
            "kind": "Weighted Mean",
            "value": 99.3902152923649,
            "weighting": ""
        },
        {
            "attr": "moles_k39",
            "error": 3.1218504932975703e-15,
            "error_kind": "SEM, but if MSWD>1 use SEM * sqrt(MSWD)",
            "kind": "Weighted Mean",
            "value": 2.328687191170094e-15,
            "weighting": ""
        },
        {
            "attr": "signal_k39",
            "error": 39.02313116621963,
            "error_kind": "SEM, but if MSWD>1 use SEM * sqrt(MSWD)",
            "kind": "Weighted Mean",
            "value": 29.108589889626177,
            "weighting": ""
```


(continues on next page)

(continued from previous page)


```
        }
    ]
},
"sample_metadata": {
    "grainsize": "",
    "irradiation": "NM-300",
    "irradiation_level": "F",
    "irradiation_position": 9,
    "latitude": 72.0,
    "lithology": "tuff",
    "lithology_class": "igneous",
    "lithology_group": "felsic",
    "lithology_type": "volcanic",
    "longitude": 106.0,
    "material": "Sanidine",
    "principal_investigator": "Zimmerer, M",
    "project": "AdvancedArgonFall2018",
    "rlocation": "",
    "sample": "04L-37 JM w/ inclusion"
},
"uuid": "a8ea1025-01af-4d9a-8fff-b4235a14ce62"
}
```

CHAPTER

FIVE

API

5.1 Database Package

class pychron.database.core.database_adapter.**DatabaseAdapter**(**args: Any*, ***kwargs: Any*)

The DatabaseAdapter is a base class for interacting with a SQLAlchemy database. Two main subclasses are used by pychron, IsotopeAdapter and MassSpecDatabaseAdapter.

This class provides attributes for describing the database url, i.e host, user, password etc, and methods for connecting and opening database sessions.

It also provides some helper functions used extensively by the subclasses, e.g. `_add_item`, `_retrieve_items`

commit()

commit the session

connect(*test=True*, *force=False*, *warn=True*, *version_warn=True*, *attribute_warn=False*)

Connect to the database

Parameters

- **test** – Test the connection by running `test_func`
- **force** – Test connection even if connection parameters haven't changed
- **warn** – Warn if the connection test fails
- **version_warn** – Warn if database/pychron versions don't match

Returns
True if connected else False

Return type
bool

create_all(*metadata*)

Build a database schema with the current connection

Parameters
metadata – SQLAchemy MetaData object

flush()

flush the session

get_migrate_version(***kw*)

Query the AlembicVersionTable

reset_connection()

Trip the connection_parameters_changed flag. Next connect call with use the new values

class pychron.database.core.database_adapter.**SessionCTX**(*parent*, *use_parent_session=True*)

5.2 DVC

5.2.1 DVC

5.2.2 Git

class pychron.dvc.meta_repo.**MetaRepo**(**args: Any*, ***kwargs: Any*)

get_cocktail_irradiation()

example cocktail.json

{

"chronology": "2016-06-01 17:00:00", "j": 4e-4, "j_err": 4e-9

}

Returns

make_geometry_file(*name*, *holes*)

holes = [(x,y,r,id),] :param holes: :return:

class pychron.git_archive.repo_manager.**GitRepoManager**(**args: Any*, ***kwargs: Any*)

manage a local git repository

get_modification_date(*path*)

"Fri May 18 11:13:57 2018 -0600" :param path: :return:

init_repo(*path*)

path: absolute path to repo

return True if git repo exists

open_repo(*name*, *root=None*)

name: name of repo root: root directory to create new repo

pull(*branch='master'*, *remote='origin'*, *handled=True*, *use_progress=True*, *use_auto_pull=False*)

fetch and merge

if use_auto_pull is False ask user if they want to accept the available updates

reset()

delete index.lock

unpack_blob(*hexsha*, *p*)

p: str. should be absolute path

5.3 Experiment

5.3.1 Automated Run

class pychron.experiment.automated_run.automated_run.**AutomatedRun**(**args: Any*, ***kwargs: Any*)

The `AutomatedRun` object is used to execute automated analyses.

It mostly delegates responisbility to other objects. It provides an interface for `MeasurementPyscripts`. All measurement script commands have a corresponding function defined here. A commands corresponding function is defined as py_{function_name}

for example `position_magnet` calls `AutomatedRun.py_position_magnet`

data collection is handled by either `MultiCollector` or `PeakHopCollector`

persistence (saving to file and database) is handled by `AutomatedRunPersister`

An automated run is executed in four steps by the `ExperimentExecutor`.

1. start
2. extraction
3. measurement
    a. equilibration
    b. post_equilibration
4. post_measurement

equilibration and post_equilibration are executed concurrently with the measurement script this way equilibration gas can be measured.

four pyscripts (all optional) are used to program analysis execution

1. extraction
2. measurement
3. post_equilibration
4. post_measurement

four types of conditionals are available

1. termination_conditionals
2. truncation_conditionals
3. action_conditionals
4. cancelation_conditionals

cancel_run(*state='canceled'*, *do_post_equilibration=True*)

terminate the measurement script immediately

do post termination
post_eq and post_meas

don't save run

get_interpolation_value(*value*)

value is a string in the format of $VALUE. Search for VALUE first in the options file then in the extraction scripts metadata

Parameters
value –

Returns

persistence_spec

alias of PersistenceSpec

py_add_action(***kw*)

attr must be an attribute of arar_age

perform a specified action if teststr evaluates to true

py_add_cancelation(***kw*)

cancel experiment if teststr evaluates to true

py_add_termination(***kw*)

attr must be an attribute of arar_age

terminate run and continue experiment if teststr evaluates to true

py_add_truncation(***kw*)

attr must be an attribute of arar_age

truncate measurement and continue run if teststr evaluates to true default kw: attr='', comp='',start_count=50, frequency=5, abbreviated_count_ratio=1.0

py_define_hops(*hopstr*)

set the detector each isotope add additional isotopes and associated plots if necessary

py_sink_data(*n=100*, *delay=1*)

new measurement interface for just sinking the data from a ring buffer

script_info

alias of ScriptInfo

truncate_run(*style='normal'*)

truncate the measurement script

style:
normal- truncate current measure iteration and continue quick- truncate current measure iteration use truncated_counts for following

measure iterations

wait_for_overlap()

by default overlap_evt is set after equilibration finished

5.3.2 Collectors

class pychron.experiment.automated_run.data_collector.**DataCollector**(**args: Any*, ***kwargs: Any*)

Base class for `Collector` objects. Provides logic for iterative measurement.

class pychron.experiment.automated_run.multi_collector.**MultiCollector**(**args: Any*, ***kwargs: Any*)

Collector class for doing multi-collection, i.e. measuring multiple intensities simultaneously.

`MulticCollector` and `PeakHopCollector` conceptually very similar and potentially could be merged and simplified. MultiCollection is a simple case of PeakHopCollection in which only one peak hop is made at the beginning.

class pychron.experiment.automated_run.peak_hop_collector.**PeakHopCollector**(**args: Any*, ***kwargs: Any*)

Collector class for doing a peak hop measurement. Measure one or more intensities at given mass for ncounts then jump magnet to next new mass.

5.3.3 Persistence

5.3.4 Conditional

class pychron.experiment.conditional.conditional.**BaseConditional**(**args: Any*, ***kwargs: Any*)

check(*run*, *data*, *cnt*)

check conditional if cnt is greater than start count cnt-start count is greater than 0 and cnt-start count is divisable by frequency

returns True if check passes. e.i. Write checks to trip on success. To terminate if Ar36 intensity is less than x use Ar36<x

Parameters

- **run** – `AutomatedRun`
- **data** – 2-tuple. (keys, signals) where keys==detector names, signals== measured intensities
- **cnt** – int

Returns
True if check passes. e.i. Write checks to trip on success.

class pychron.experiment.conditional.conditional.**AutomatedRunConditional**(**args: Any*, ***kwargs: Any*)

class pychron.experiment.conditional.conditional.**TruncationConditional**(**args: Any*, ***kwargs: Any*)

stops the current measurement and continues to next step in pyscript. If more measure calls are made use abbreviated_count_ratio to reduce the number of counts. for example of abbreviated_count_ratio = 0.5 and the original baseline counts = 100, only 50 counts will be made for a truncated run.

class pychron.experiment.conditional.conditional.**TerminationConditional**(**args: Any*, ***kwargs: Any*)

Stop the current analysis immediately. Don't save to database. Continue to next run in experiment queue

class pychron.experiment.conditional.conditional.**CancelationConditional**(**args: Any*, ***kwargs: Any*)

Stop the current analysis immediately then stop the experiment.

class pychron.experiment.conditional.conditional.**ActionConditional**(**args: Any*, ***kwargs: Any*)

Executes a specified action. The action string is executed as pyscript snippet. actions therefore may be any valid measuremnt pyscript code. for example:

```
# call a gosub
gosub("someGoSub")

# open a valve
open("V")
```

perform(*script*)

perform the specified action.

use `MeasurementPyScript.execute_snippet` to perform desired action

Parameters
script – MeasurementPyScript

5.4 PyScripts

PyScripts are used to automate various processes, such as extraction of gas and measurement of isotopes.

5.4.1 PyScript

class pychron.pyscripts.pyscript.**PyScript**(**args: Any*, ***kwargs: Any*)

calculate_estimated_duration(*ctx=None*, *force=False*)

maintain a dictionary of previous calculated durations. key=hash(ctx), value=duration

5.4.2 ValvePyScript

class pychron.pyscripts.valve_pyscript.**ValvePyScript**(**args: Any*, ***kwargs: Any*)

5.4.3 Ramper

class pychron.pyscripts.extraction_line_pyscript.**Ramper**

ramp(*func*, *start*, *end*, *duration*, *rate=0*, *period=1*)

rate = units/s duration= s

use rate if specified

5.4.4 ExtractionPyScript

class pychron.pyscripts.extraction_line_pyscript.**ExtractionPyScript**(**args: Any*, ***kwargs: Any*)

The ExtractionPyScript is used to program the extraction and gettering of sample gas.

extract_pipette(*identifier=''*, *timeout=300*)

this is an atomic command. use the apis_controller config file to define the isolation procedures.

get_extraction_positions(*clear=True*)

Returns a list of x,y,z tuples each tuple represents where the extraction occurred

if clear is True (default) `self._extraction_positions` set to an empty list

Returns
list of x,y,z tuples

Return type
list of tuples

get_output_blob()

Get the extraction device's output blob

Returns
output blob: binary string representing time v percent output

Return type
str

get_response_blob()

Get the extraction device's response blob

Returns
response blob. binary string representing time v measured output

Return type
str

get_setpoint_blob()

Get the extraction device's setpoint blob

Returns
setpoint blob: binary string representing time v requested setpoint

Return type
str

load_pipette(*identifier*, *timeout=300*)

this is a non blocking command. it simply sends a command to apis to start one of its runscripts.

it is the ExtractionPyScripts responsiblity to handle the waiting. use the waitfor command to wait for signals from apis.

output_achieved()

Return a formated string with the extraction "heating" results:

```
Requested Output= 100.000
Achieved Output= 99.012
```

Returns
Formatted string with results

Return type
str

property position

if position is 0 return None

set_default_context(***kw*)

provide default values for all the properties exposed in the script

snapshot(*name=''*, *prefix=''*, *view_snapshot=False*, *pic_format='.jpg'*)

if name not specified use RID_Position e.g 12345-01A_3

waitfor(*func_or_tuple*, *start_message=''*, *end_message=''*, *check_period=1*, *timeout=0*, *func_kw=None*)

tuple format: (device_name, function_name, comparison, ...) addition tuple elements are passed to function_name

comparison

```
x<10
10<x<20
```

callable can of form `func() or func(ti) or func(ti, i)` where `ti` is the current relative time (relative to start of waitfor) and `i` is a counter

Parameters

- **func_or_tuple** (*callable, tuple*) – wait for function to return True
- **start_message** (*str*) – Message to display at start
- **end_message** (*str*) – Message to display at end
- **check_period** (*int, float*) – Delay between checks in seconds
- **timeout** (*int, float*) – Cancel waiting after `timeout` seconds

5.4.5 MeasurementPyScript

class pychron.pyscripts.measurement_pyscript.**MeasurementPyScript**(**args: Any*, ***kwargs: Any*)

MeasurementPyScripts are used to collect isotopic data

activate_detectors(**dets*, ***kw*)

set the active detectors

Parameters
dets – list

coincidence()

Do a coincidence scan. Peak center all active detectors simulatenously. calculate required deflection corrections to bring all detectors into coincidence

property eqtime

Property. Equilibration time. Get value from `AutomatedRun`.

Returns
float, int

equilibrate(*eqtime=20*, *inlet=None*, *outlet=None*, *do_post_equilibration=True*, *close_inlet=True*, *delay=3*)

equilibrate the extraction line with the mass spectrometer

inlet or outlet can be a single valve name or a list of valve names.

```
'A', ('A','B'), ['A','B'], 'A,B'
```

Parameters

- **eqtime** – int, equilibration duration in seconds
- **inlet** – str, tuple or list, inlet valve
- **outlet** – str, tuple or list, ion pump valve
- **do_post_equilibration** – bool
- **close_inlet** – bool
- **delay** – int, delay in seconds between close of outlet and open of inlet

generate_ic_mftable(*detectors*, *refiso='Ar40'*, *peak_center_config=''*, *n=1*, *calc_time=False*)

Generate an IC MFTable. Use this when doing a Detector Intercalibration. peak centers the `refiso` on a list of `detectors`. MFTable saved as ic_mftable

cancel script if generating mftable fails

Parameters

- **detectors** (*list*) – list of detectors to peak center
- **refiso** (*str*) – isotope to peak center

peak_center(*detector=''*, *isotope=''*, *integration_time=1.04*, *save=True*, *calc_time=False*, *directions='Increase'*, *config_name='default'*)

Calculate the peak center for `isotope` on `detector`.

Parameters

- **detector** – str
- **isotope** – str
- **integration_time** – float
- **save** – bool

position_magnet(*pos*, *detector='AX'*, *use_dac=False*, *for_collection=True*)

Parameters

- **pos** (*str*) – location to set magnetic field
- **detector** – detector to position `pos`
- **use_dac** (*bool*) – is the `pos` a DAC voltage

examples:

```
position_magnet(4.54312, use_dac=True) # detector is not relevant
position_magnet(39.962, detector='AX')
position_magnet('Ar40', detector='AX') #Ar40 will be converted to 39.962 use␣
↪mole weight dict
```

post_equilibration(*block=False*)

Run the post equilibration script.

reset(*arun*)

Reset the script with a new automated run

Parameters
arun (AutomatedRun) – A new AutomatedRun

set_baseline_fits(**fits*)

set baseline fits for detectors

Parameters
fits –

set_fits(**fits*)

set time vs intensity regression fits for isotopes

Parameters
fits – str, list, or tuple

set_integration_time(*v*)

Set the integration time

Parameters
v (*float*) – integration time in seconds

set_time_zero(*offset=0*)

set the time_zero value. add offset to time_zero

```
T_o= ion pump closes
offset seconds after T_o. define time_zero

T_eq= inlet closes
```

sink_data(*n=100*, *delay=1*, *calc_time=False*)

@param n: number of measurements @param period: delay between measurements @param calc_time: @return:

property time_zero_offset

Property. Substract time_zero_offset from time value for all data points

Returns
float, int

property truncated

Property. True if run was truncated otherwise False

Returns
bool

property use_cdd_warming

Property. Use CDD Warming. Get value from AutomatedRunSpec

Returns
bool

whiff(*ncounts=0*, *conditionals=None*)

Do a whiff measurement.

Whiff's are quick measurements with conditionals. use them to take action at the beginning of a measurement. For example do a whiff to determine if intensity to great.

Parameters

- **ncounts** – int
- **conditionals** – list of dicts

5.5 Spectrometer Package

class pychron.spectrometer.field_table.**FieldTable**(**args: Any*, ***kwargs: Any*)

map a voltage to a mass

load_table(*path=None*, *load_items=False*)

mftable format- first line is a header followed by Isotope, Dac_i, Dac_j,....

Dac_i is the magnet dac setting to center Isotope on detector i example:

```
iso, H2,     H1,      AX,     L1,     L2,     CDD
Ar40,5.78790,5.895593,6.00675,6.12358,6.24510,6.35683
Ar39,5.89692,5.788276,5.89692,5.89692,5.89692,5.89692
Ar36,5.56072,5.456202,5.56072,5.56072,5.56072,5.56072
```

update_field_table(*det*, *isotope*, *dac*, *message=''*, *save=True*, *report=False*, *update_others=True*)

dac needs to be in axial units

update_others. If false only

class pychron.spectrometer.fieldmixin.**FieldMixin**(**args: Any*, ***kwargs: Any*)

class pychron.spectrometer.base_magnet.**BaseMagnet**(**args: Any*, ***kwargs: Any*)

finish_loading()

initialize the mftable

read DAC from device :return:

map_dac_to_isotope(*dac=None*, *det=None*, *current=True*)

convert a dac voltage to isotope name for a given detector

Parameters

- **dac** – float, voltage
- **det** – str, detector name
- **current** – bool, get current hv

Returns
str, e.g Ar40

map_dac_to_mass(*dac*, *detname*)

convert a DAC value (voltage) to mass for a given detector use the mftable

Parameters

- **dac** – float, voltage (0-10V)
- **detname** – str, name of a detector, e.g H1

Returns
float, mass

`map_mass_to_dac`(*mass*, *detname*)

convert a mass value from amu to dac for a given detector

Parameters

- **mass** – float, amu
- **detname** – std, name of a detector, e.g. H1

Returns
float, dac voltage

`mass_change`(*m*)

set the self.mass attribute suppress mass change handler

Parameters
m – float

Returns

`class pychron.spectrometer.base_detector.BaseDetector`(**args: Any*, ***kwargs: Any*)

5.5.1 Thermo

The thermo package (pychron.spectrometer.thermo) contains abstractions for interfacing a Thermo Scientific mass spectrometer via RemoteControlService.cs

Spectrometers

`class pychron.spectrometer.thermo.spectrometer.base.ThermoSpectrometer`(**args: Any*, ***kwargs: Any*)

`get_deflection`(*name*, *current=False*)

get deflection by detector name

Parameters

- **name** – str, detector name
- **current** – bool, if True query qtegra

Returns
float

`get_detector_active`(*dname*)

return True if dname in the list of intensity keys e.g.

keys, signals = get_intensities return dname in keys

Parameters
dname –

Returns

get_intensity(*dkeys*)

dkeys: str or tuple of strs

load()

load detectors load setupfiles/spectrometer/config.cfg file load magnet load deflections coefficients

Returns

load_configurations()

load configurations from Qtegra :return:

load_current_detector_gains()

load the detector gains from the spectrometer

set_gains(*history=None*)

Parameters
history –

Returns
list

set_integration_time(*it*, *force=False*)

Parameters

- **it** – float, integration time
- **force** – set integration even if "it" is not different than self.integration_time

Returns
float, integration time

test_intensity()

test if intensity is changing. make 2 measurements if exactlly the same for all detectors make third measurement if same as 1,2 make fourth measurement if same all four measurements same then test fails :return:

class pychron.spectrometer.thermo.spectrometer.argus.**ArgusSpectrometer**(**args: Any*, ***kwargs: Any*)

Interface to a Thermo Scientific Argus Mass Spectrometer via Qtegra and RemoteControlServer.cs magnet control provided by ArgusMagnet source control provided by ArgusSource

direct access to RemoteControlServer.cs API via microcontroller e.g. microcontroller.ask('GetIntegrationTime')

magnet_klass

alias of *ArgusMagnet*

microcontroller_klass

alias of ArgusController

class pychron.spectrometer.thermo.spectrometer.helix.**HelixSpectrometer**(**args: Any*, ***kwargs: Any*)

magnet_klass

alias of *HelixMagnet*

microcontroller_klass

alias of HelixController

class pychron.spectrometer.thermo.magnet.base.**ThermoMagnet**(**args: Any*, ***kwargs: Any*)

Magnet interface to Qtegra.

uses MFTable object of mapping dac to mass

class pychron.spectrometer.thermo.magnet.argus.**ArgusMagnet**(**args: Any*, ***kwargs: Any*)

class pychron.spectrometer.thermo.magnet.helix.**HelixMagnet**(**args: Any*, ***kwargs: Any*)

5.5.2 Isotopx

The Isotopx package (pychron.spectrometer.isotopx) contains abstractions for interfacing a Isotopx mass spectrometer via IsotopxRCS

Spectrometers

class pychron.spectrometer.isotopx.spectrometer.base.**IsotopxSpectrometer**(**args: Any*, ***kwargs: Any*)

class pychron.spectrometer.isotopx.spectrometer.ngx.**NGXSpectrometer**(**args: Any*, ***kwargs: Any*)

finish_loading()

finish loading magnet send configuration if self.send_config_on_startup set in Preferences :return:

get_update_period(*it=None*, *is_scan=False*)

acquisition period is always set to 1s so update period always needs to be <1s

microcontroller_klass

alias of NGXController

set_integration_time(*it*, *force=False*)

Parameters

- **it** – float, integration time in seconds
- **force** – set integration even if "it" is not different than self.integration_time

Returns

float, integration time

class pychron.spectrometer.isotopx.magnet.base.**IsotopxMagnet**(**args: Any*, ***kwargs: Any*)

class pychron.spectrometer.isotopx.magnet.ngx.**NGXMagnet**(**args: Any*, ***kwargs: Any*)

set_mass(*v*, *delay=None*, *deflect=False*)

Parameters

- **v** – mass
- **delay** – settling time in ms
- **deflect** –

Returns

5.5.3 MAP

The map package (pychron.spectrometer.map) contains abstractions for interfacing with a Mass Analyzer Products (MAP) mass spectromter. Developed for New Mexico Geochronology Research Laboratory's MAP215-50

class pychron.spectrometer.map.magnet.**MapMagnet**(**args: Any*, ***kwargs: Any*)

Abstraction for the MAP Magent

set_dac(*v*, *verbose=False*)

set the magnet dac voltage.

set the range then send W[v]

```
dev.tell('W4.543')
```

Parameters

- **v** – v, dac voltage
- **verbose** –

Returns
bool, True if dac changed else False

set_range(*r*, *verbose=False*)

if float convert to integer and use as range.

send B[r-1]

```
# r = 6
dev.tell('B5')
```

Parameters

- **r** – float or int
- **verbose** –

5.6 Pipeline

5.6.1 Nodes

5.7 Remote Control Protocols

5.7.1 Service

class pychron.tx.protocols.service.**ServiceProtocol**(**args: Any*, ***kwargs: Any*)

5.7.2 Laser

class pychron.tx.protocols.laser.**LaserProtocol**(**args: Any*, ***kwargs: Any*)

5.7.3 Valve

class pychron.tx.protocols.base_valve.**BaseValveProtocol**(**args: Any*, ***kwargs: Any*)

class pychron.tx.protocols.valve.**ValveProtocol**(**args: Any*, ***kwargs: Any*)

Furnace

class pychron.tx.protocols.furnace.**FurnaceProtocol**(**args: Any*, ***kwargs: Any*)

CHAPTER

SIX

MASS SPEC - PYCHRON DIFFERENCES

As of 5/24/17

1. Mass Spec does not propogate baseline error
2. Mass Spec does not correct Ca/K for 37ArK
3. Mass Spec does not propogate IC error correctly?
4. Pychron estimates J error ~an order of magnitude greater than Mass Spec.

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

N

O

P

R

S

T

U

V

W