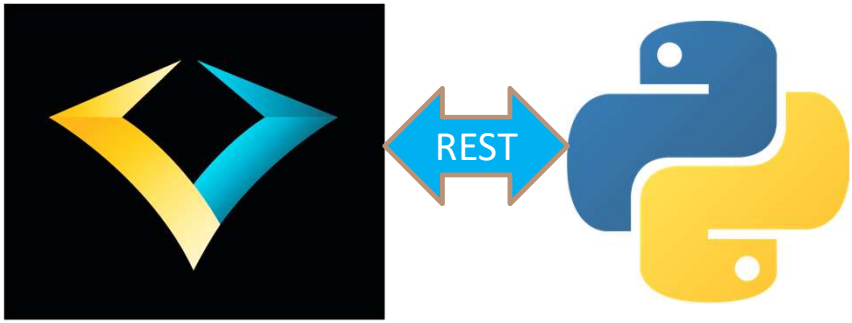
Fusion Remote Control API
Using REST
(REpresentational State Transfer)

© Andor Technology 2019

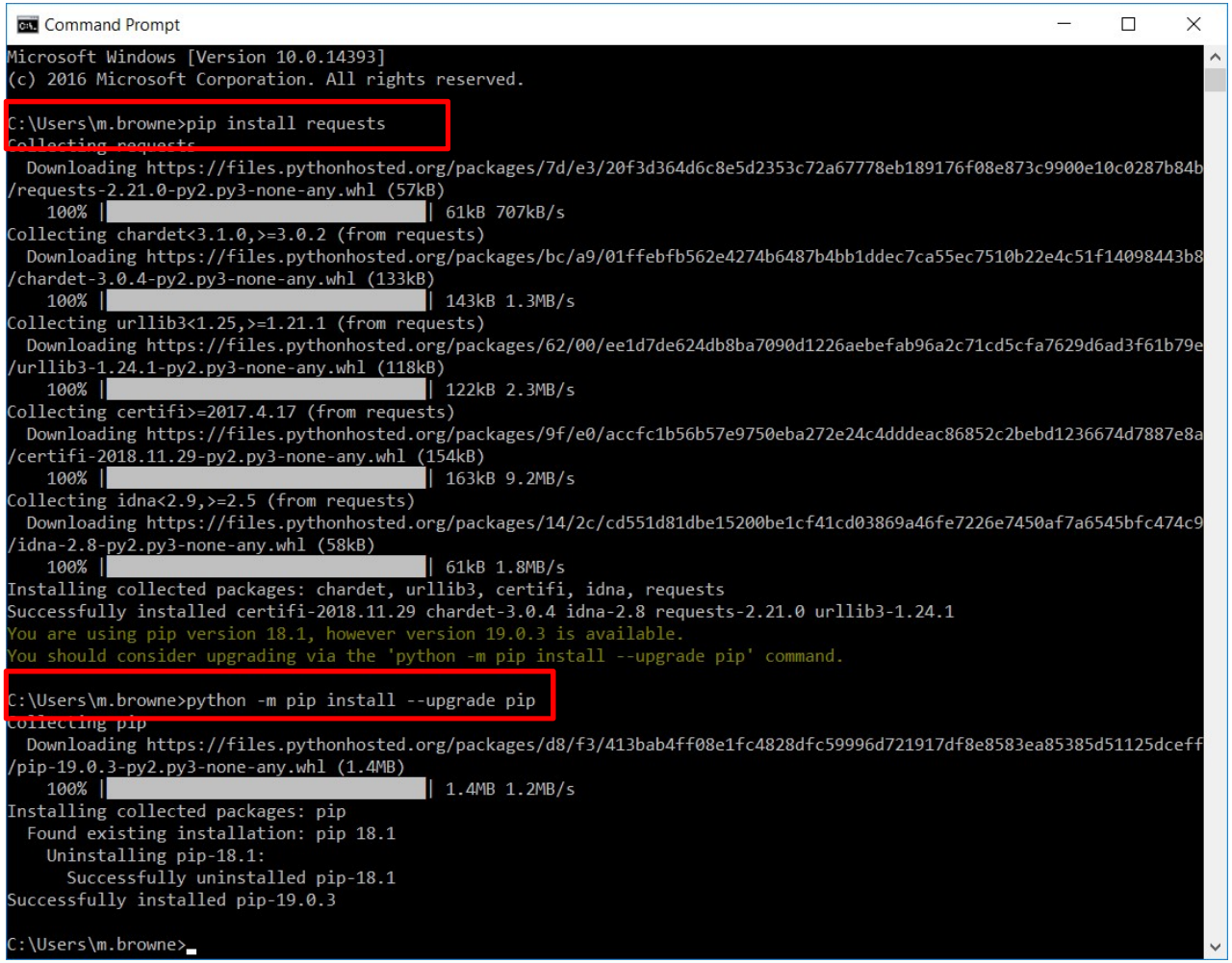


© Oxford Instruments 2015

First get Python - https://www.python.org/ftp/python/3.7.2/python-3.7.2-amd64.exe

OXFORD INSTRUMENTS
The Business of Science®

1. Download installer from Link above
2. Run installer in Express mode
3. Start Windows Command Prompt
4. Run pip install requests as shown
5. Note update in yellow below
6. Run update of pip



© Oxford Instruments 2015

If running Windows 7 then the ACL needs to be updated

ACL – refers to the Access Control List which provides permissions to access files or other resources.

In order to provide permissions to the current user then the following commands must be run from an **administrator command prompt.** (right-click Command Prompt in the start menu, pick "Run as administrator").

Run the below two commands to allow port 15120, which is the default Fusion port.
You'll need to update the "DOMAIN\username" part based on whatever user account will be running Fusion.

```
netsh http add urlacl url=http://127.0.0.1:15120/ user=DOMAIN\username
netsh http add urlacl url=http://localhost:15120/ user=DOMAIN\username
```

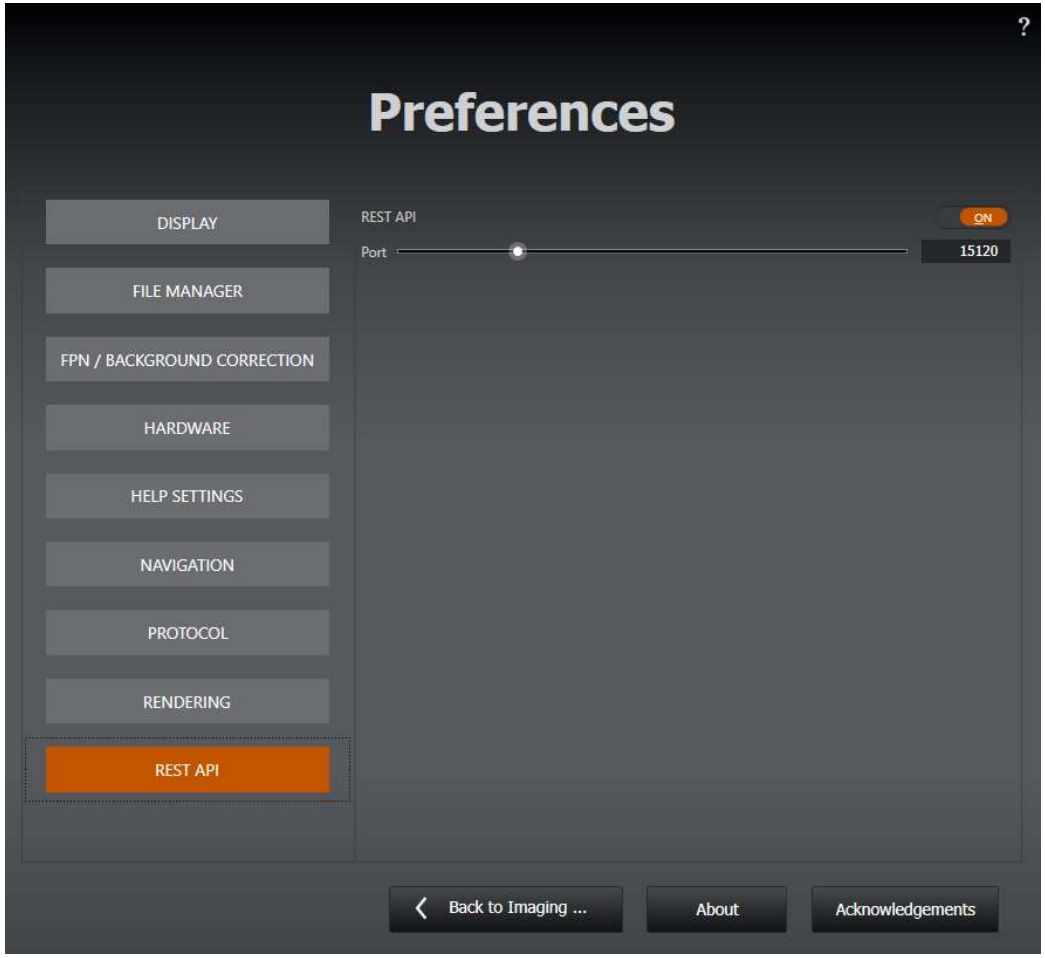
If using just a local user account on the PC, the user account will be "PCNAME\username".
If you're not sure, you can check the exact account name from a normal (non-admin) command prompt by:

```
echo %USERDOMAIN%\%USERNAME%
```

© Oxford Instruments 2015

Switch ON Fusion REST API

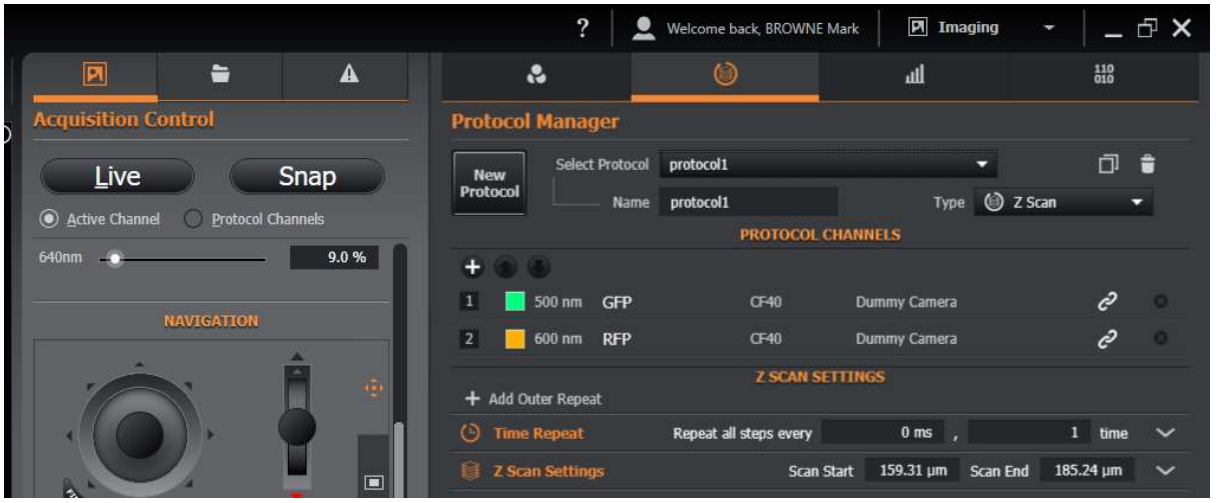
1. Start Fusion_REST – version 2.1.0.34 or later
2. Switch to Preferences View
3. Select REST API
4. Enable REST_API slider to “ON”
5. Default Port 15120 should be left at Default
6. Click “Back to Imaging”…



© Oxford Instruments 2015

Set up Fusion Protocols for REST control

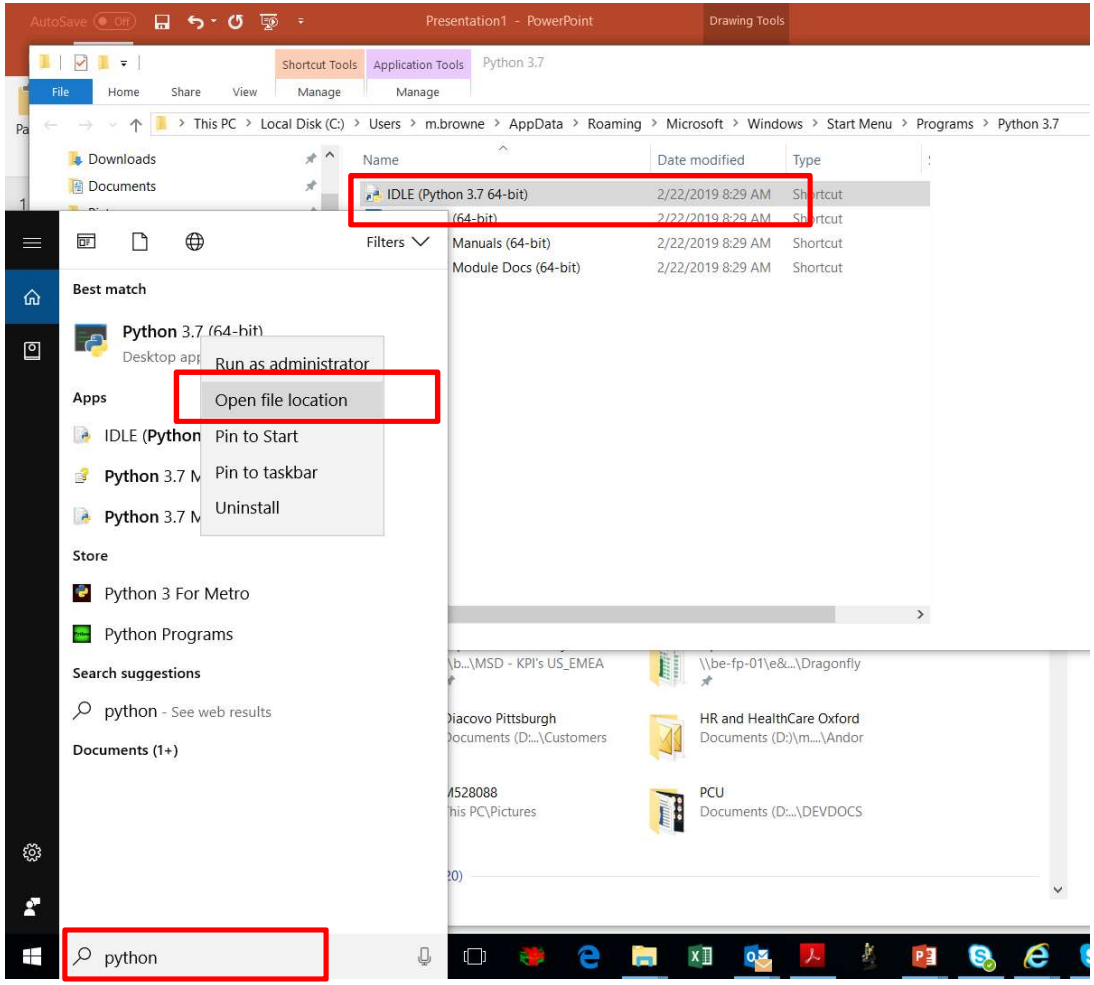
1. Ensure correct Channels exist
2. Build Protocols
3. Check protocols match script names
4. Here we have “protocol1” and “protocol2”



© Oxford Instruments 2015

Locate Python shortcuts

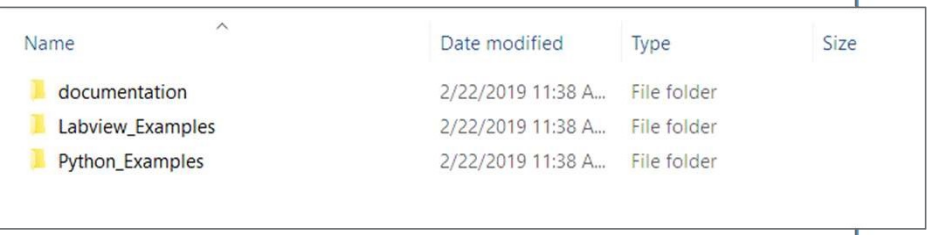
1. Type “Python” into Windows search box
2. Windows results opens – Python on top
3. Right mouse Open file location - click
4. Windows File Explorer shows folder
5. Double Click on “IDLE (Python 3.7 64 bit)
6. Right Mouse “Pin to Taskbar” if required

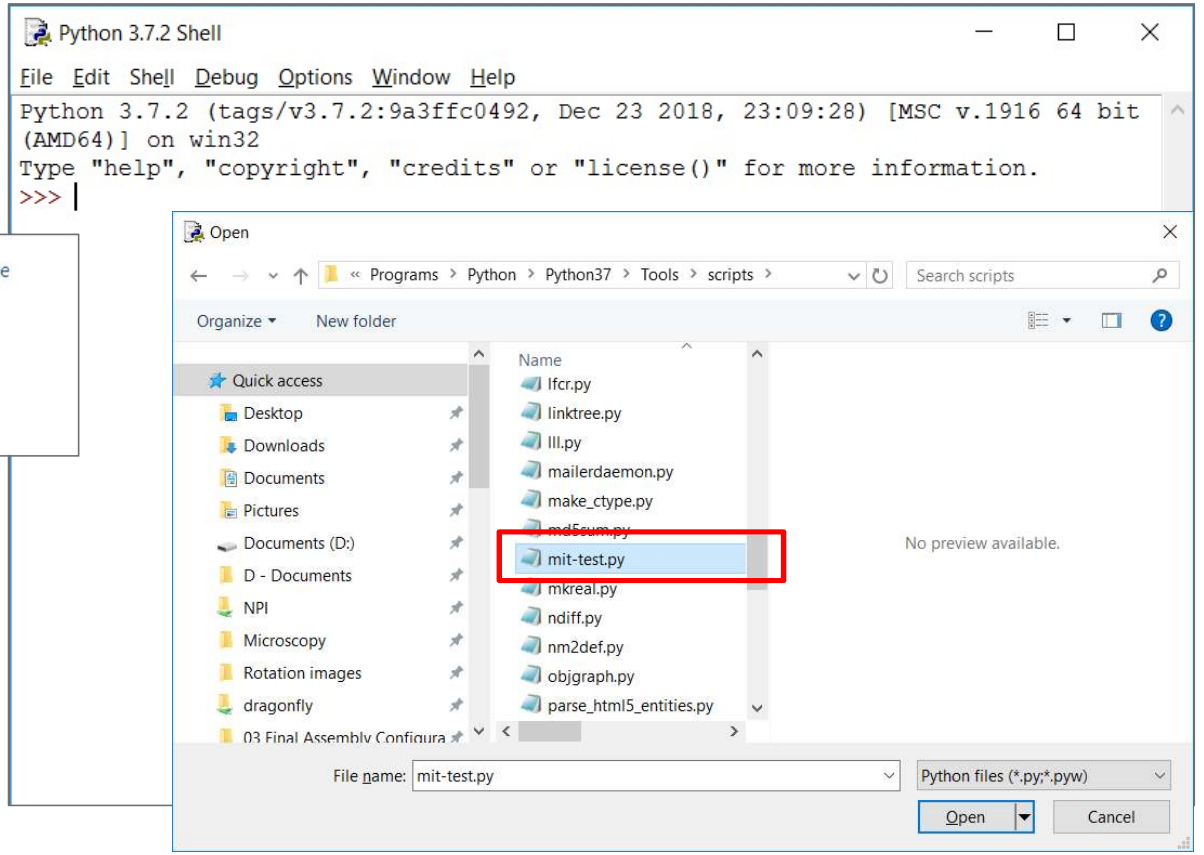


© Oxford Instruments 2015

Load Fusion Python Scripts

1. IDLE editor will Open as shown
2. Locate Fusion REST Docs and Examples
3. File Open script file “mit-test.py”

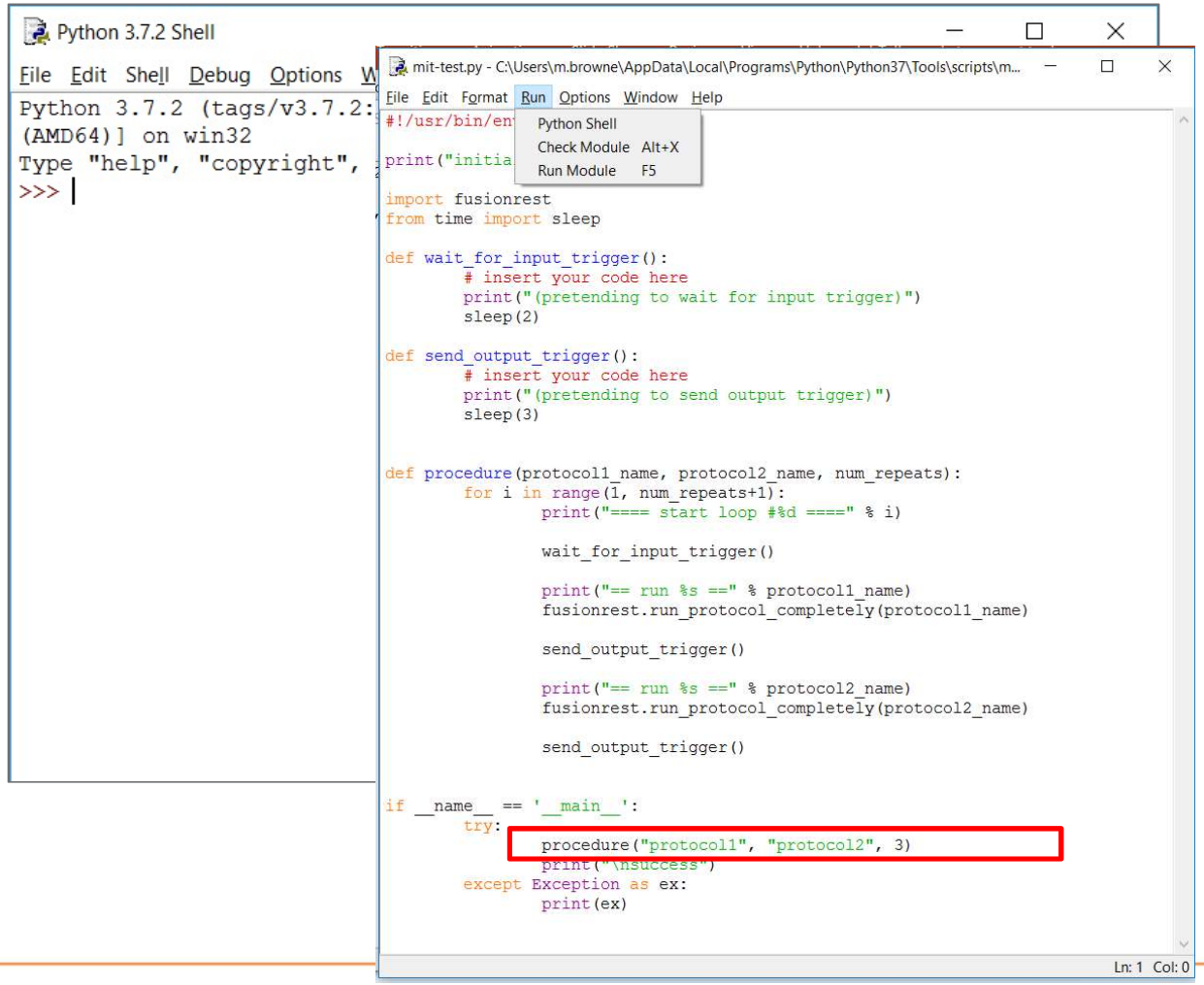




© Oxford Instruments 2015

Edit Fusion Python Scripts

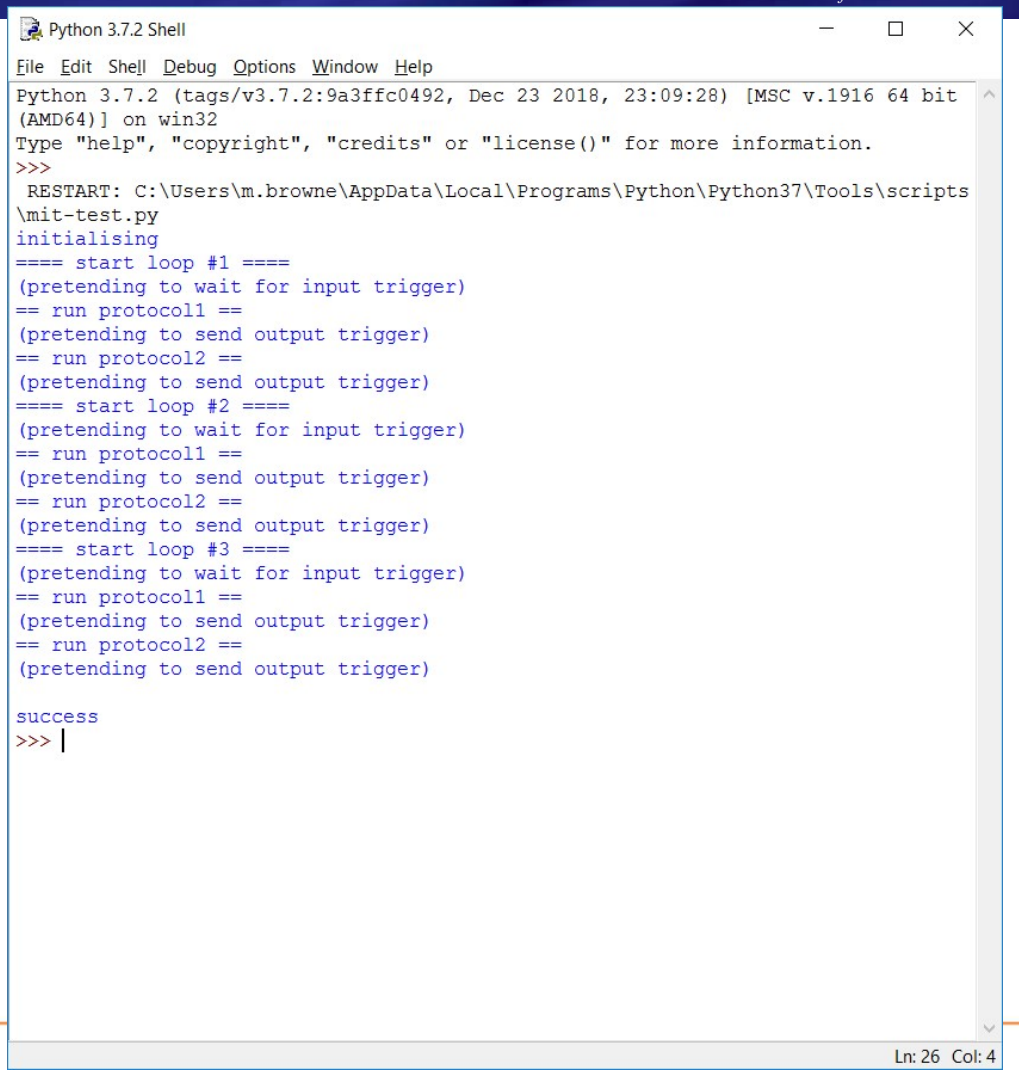
1. mit-test.py opens in its own window
2. Edit protocol names as required
3. Change dummy processes to real triggers etc



© Oxford Instruments 2015

Run Fusion Python Scripts

1. Select “Run” from the menu
2. Run Module mit-test.py
3. Observe Fusion executes scripts, IDLE reports output



© Oxford Instruments 2015

End of Preliminary setup

1. Now you can create your own protocol sequences
2. Control external hardware through Python or other

3. Note that fusionrest.py includes only the definitions
4. This means they do not execute but define REST methods

5. For more help contact the Andor team
6. email: TBD
7. The Fusion team will be engaged as needed.

fusionrest.py - C:\Users\m.browne\AppData\Local\Programs\Python\Python37\Tools\scripts\fusionrest.py (3.7.2)

File Edit Format Run Options Window Help

```
import requests
import json
import time

host = "localhost"
port = 15120

class ApiError(Exception):
        """
        Indicates an error while calling the Fusion REST API.
        """
        def __init__(self, endpoint, code, reason):
                """
                Creates an new `ApiError` instance.
                """
                self._endpoint = endpoint
                self._code = code
                self._reason = reason

        def __repr__(self):
                return "<ApiError at {}: {} {}>".format(self._endpoint, self._code, self._reason)

        def __str__(self):
                return self.__repr__()

        def endpoint(self):
                """
                Gives the name of the API endpoint for which the error happened.
                """
                return self._endpoint

        def code(self):
                """
                Gives the HTTP response code for the error, as returned by the API.
                Also see `.reason()` for a more readable description of the problem.
                """
                return self._code

        def reason(self):
                """
                Gives the reason for the error, as returned by the API. (a string)
                """
                return self._reason

def __make_address(endpoint):
        return "http://{}:{}{}".format(host, port, endpoint)
```

Ln: 1 Col: 0

© Oxford Instruments 2015

© Oxford Instruments 2015