Python Workshop – November 2023

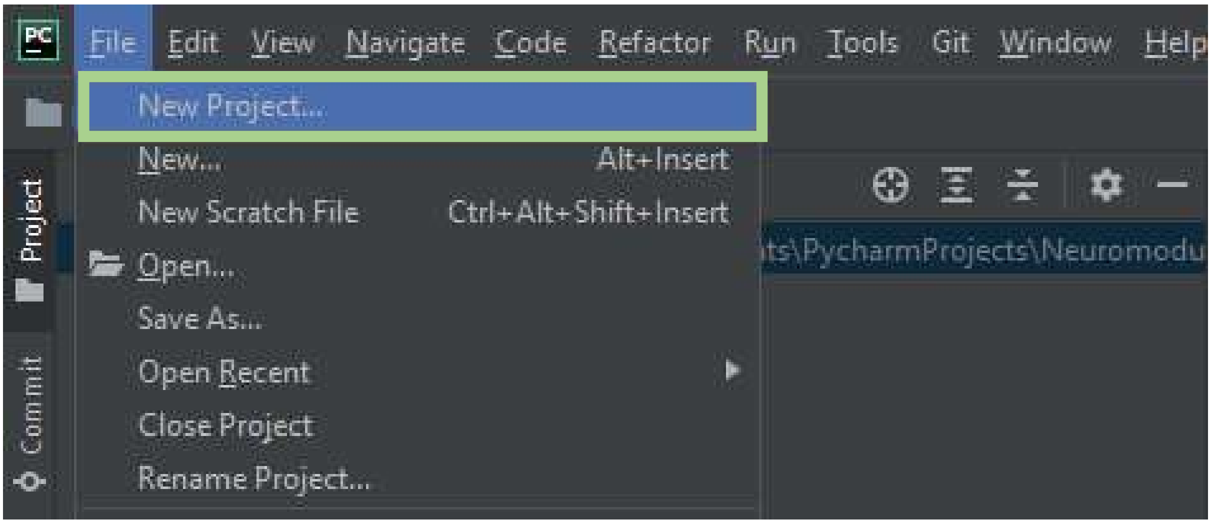
Introduction

Welcome to the coding section of the Neuromodulation Workshop! In two separated sessions we hope to bring you into the programming world (if you are not already familiar). We aim to step through some simple coding exercises which will lead into some traditional image processing techniques. The goal is to try and use Python (a programming language) to identify and count the number of regions of interest (ROIs) within an image obtained on a microscope so that you do not have to manually count them yourselves.

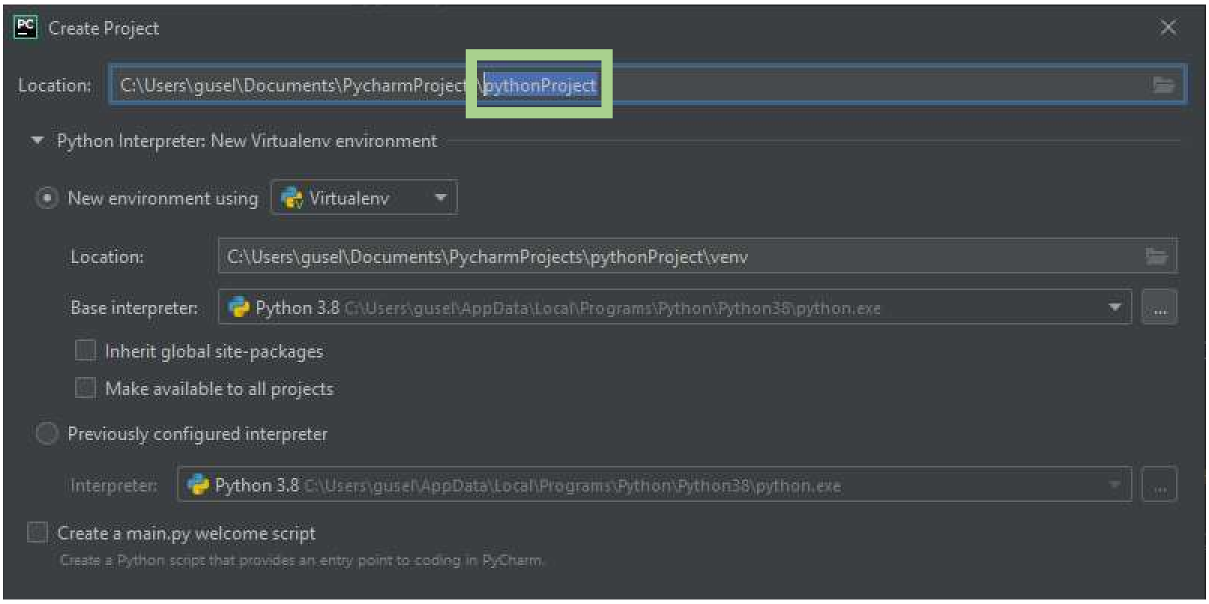
Module 1: PyCharm

PyCharm is an integrated development environment (IDE) for programmers, which can be downloaded and installed from https://www.jetbrains.com/help/pycharm/installation-guide.html . There are many other python software packages that can be used to run python scripts, but for this journey we will stick to PyCharm. Start by opening PyCharm Community or Professional App. Then

1. Create a new project: File -> New Project…



2. Rename Python Project by typing the name where the highlighted font is below.



Create Project

Location: C:\Users\gusel\Documents\PycharmProjects\pythonProject

▾ Python Interpreter: New Virtualenv environment

● New environment using Virtualenv

Location: C:\Users\gusel\Documents\PycharmProjects\pythonProject\venv

Base interpreter: Python 3.8 C:\Users\gusel\AppData\Local\Programs\Python\Python38\python.exe

□ Inherit global site-packages

□ Make available to all projects

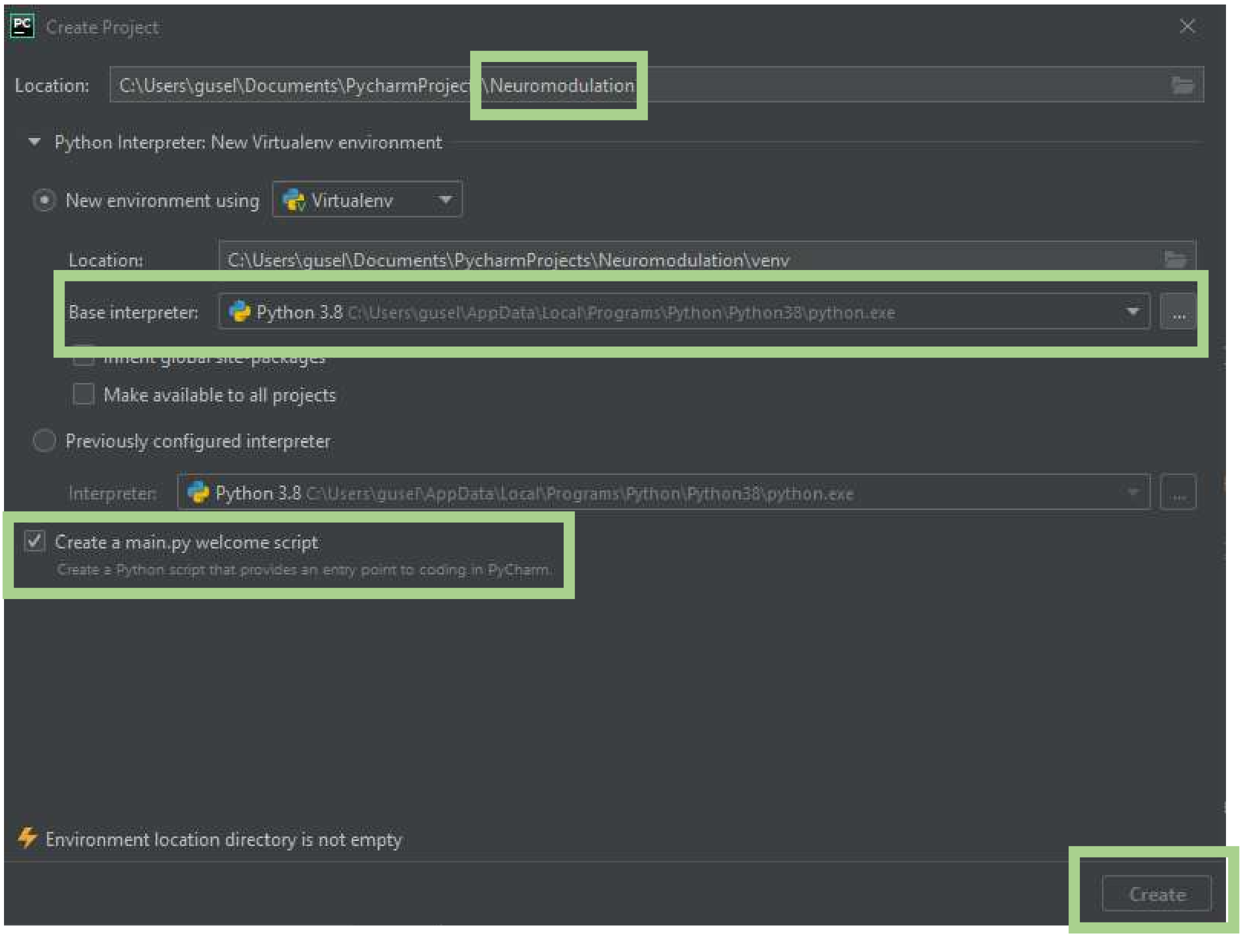
○ Previously configured interpreter

Interpreter: Python 3.8 C:\Users\gusel\AppData\Local\Programs\Python\Python38\python.exe

□ Create a main.py welcome script

Create a Python script that provides an entry point to coding in PyCharm.

3. Tick the ‘create a main.py welcome script’ and select create in the bottom right-hand corner of the dialogue box. Make sure you also select Python 3.x. as your base interpreter.



Create Project

Location: C:\Users\gusel\Documents\PycharmProjects\Neuromodulation

Python Interpreter: New Virtualenv environment

● New environment using Virtualenv

Location: C:\Users\gusel\Documents\PycharmProjects\Neuromodulation\venv

Base interpreter: Python 3.8 C:\Users\gusel\AppData\Local\Programs\Python\Python38\python.exe

□ Inherit global site-packages

□ Make available to all projects

○ Previously configured interpreter

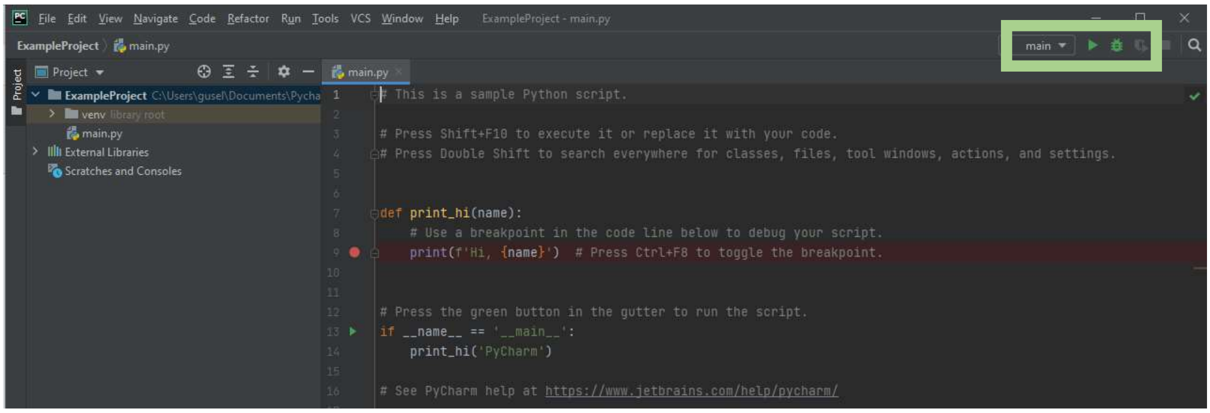
Interpreter: Python 3.8 C:\Users\gusel\AppData\Local\Programs\Python\Python38\python.exe

☑ Create a main.py welcome script

Create a Python script that provides an entry point to coding in PyCharm.

Environment location directory is not empty

A main.py script should now appear in front of you with some example code as follows. Try running the script by clicking the play button.

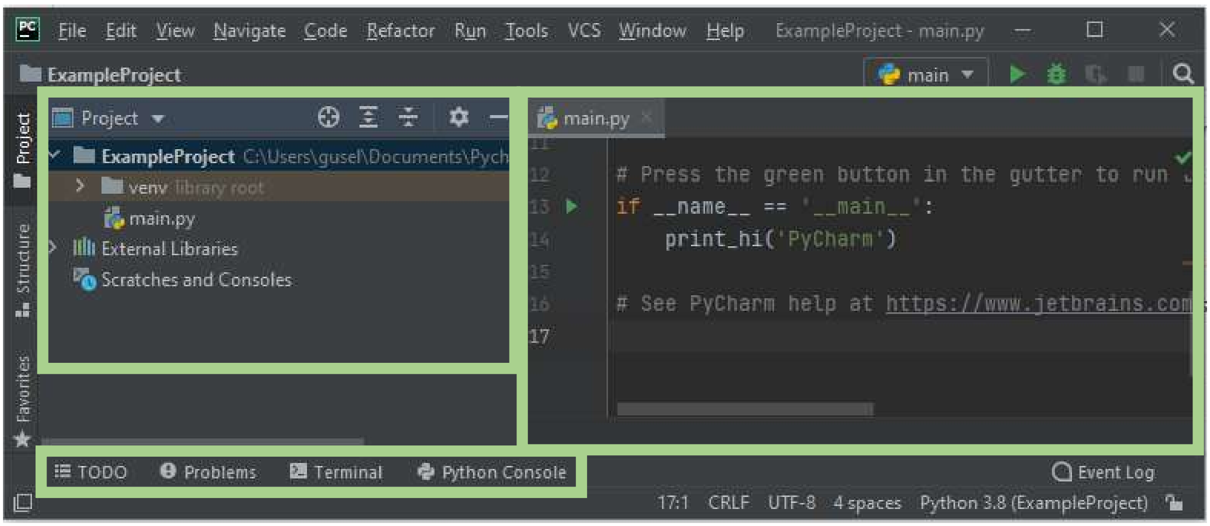


```
# This is a sample Python script.

# Press Shift+F10 to execute it or replace it with your code.
# Press Double Shift to search everywhere for classes, files, tool windows, actions, and settings.


def print_hi(name):
    # Use a breakpoint in the code line below to debug your script.
    print(f'Hi, {name}')  # Press Ctrl+F8 to toggle the breakpoint.


# Press the green button in the gutter to run the script.
if __name__ == '__main__':
    print_hi('PyCharm')

# See PyCharm help at https://www.jetbrains.com/help/pycharm/
```

Notice that there are different Tabs (see figure below):

- Script/editor tab: where you can edit your code.
- Project tab lists the current project/working directory and files.
- Problems: when you run your code problems/warnings will appear here.
- Terminal/console: you can run commands or code directly from the computer terminal or python console (we will not be doing this)

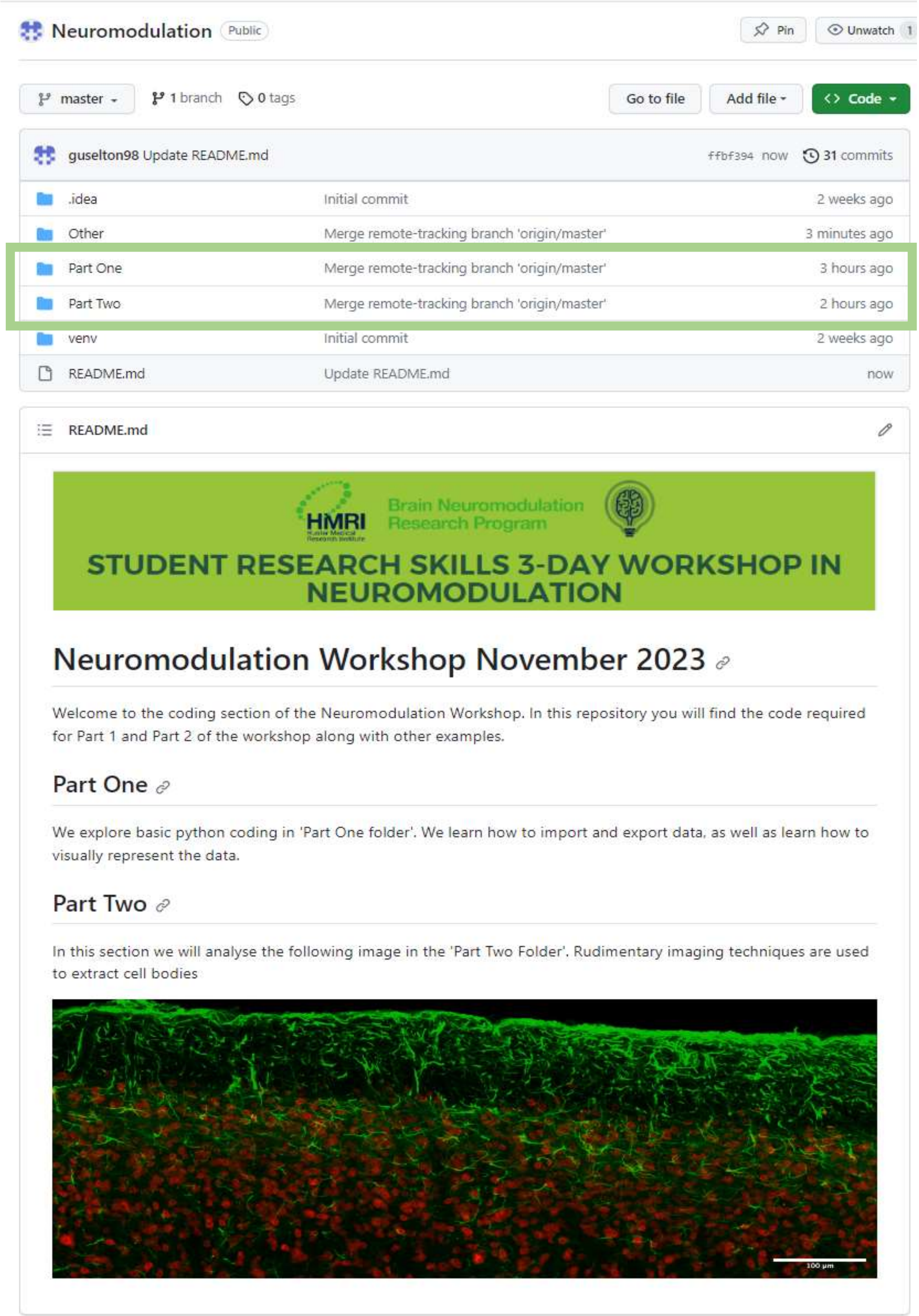
- TODO tab: you can create items for you to complete which will be listed here.


File Edit View Navigate Code Refactor Run Tools VCS Window Help ExampleProject - main.py
ExampleProject
main
Project
Structure
Favorites
Project
ExampleProject C:\Users\gusel\Documents\Pych
venv library root
main.py
External Libraries
Scratches and Consoles
main.py
TODO
Problems
Terminal
Python Console
Event Log
17:1 CRLF UTF-8 4 spaces Python 3.8 (ExampleProject)

```
# Press the green button in the gutter to run
if __name__ == '__main__':
    print_hi('PyCharm')

# See PyCharm help at https://www.jetbrains.com
```

Module 2: GitHub Repository

In the coding sessions we will be following the examples outlined in the GitHub repository: https://github.com/guselton98/Neuromodulation .


Neuromodulation
Public
Pin
Unwatch 1
master
1 branch
0 tags
Go to file
Add file
Code
README.md

guselton98 Update README.md ffbf394 now 31 commits

.idea	Initial commit	2 weeks ago
Other	Merge remote-tracking branch 'origin/master'	3 minutes ago
Part One	Merge remote-tracking branch 'origin/master'	3 hours ago
Part Two	Merge remote-tracking branch 'origin/master'	2 hours ago
venv	Initial commit	2 weeks ago
README.md	Update README.md	now

HMRI Hunter Medical Research Institute
Brain Neuromodulation Research Program
STUDENT RESEARCH SKILLS 3-DAY WORKSHOP IN NEUROMODULATION

Neuromodulation Workshop November 2023

Welcome to the coding section of the Neuromodulation Workshop. In this repository you will find the code required for Part 1 and Part 2 of the workshop along with other examples.

Part One

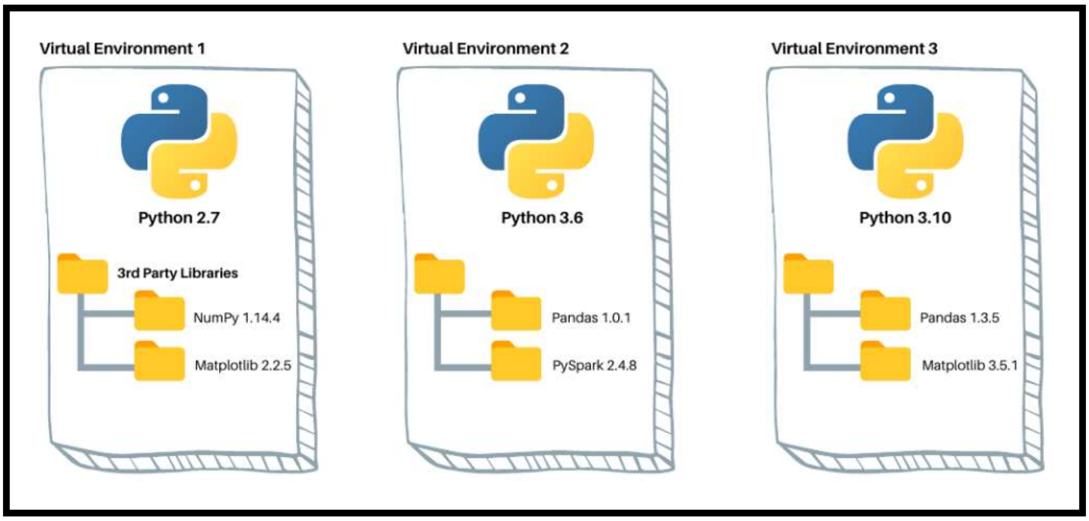
We explore basic python coding in 'Part One folder'. We learn how to import and export data, as well as learn how to visually represent the data.

Part Two

In this section we will analyse the following image in the 'Part Two Folder'. Rudimentary imaging techniques are used to extract cell bodies

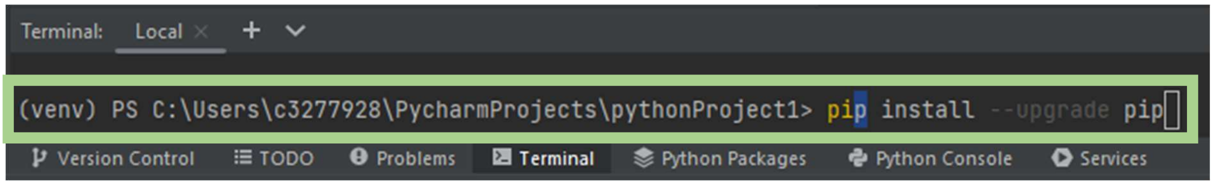
Module 3: Setting an environment: using pip install

The virtual environment that you have *magically* created contains the python compiler and some pre-installed python library packages.



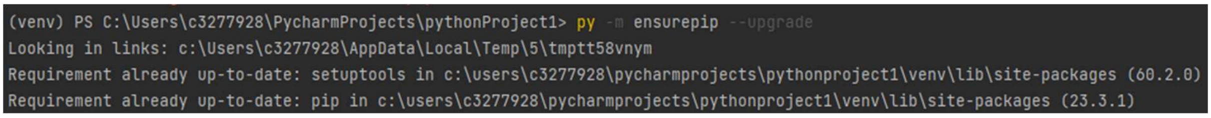
In this workshop we will also require a few more libraries which allows us to use tools already available. To install these packages, we use the *pip install*, and we need to make sure that the pip installer is upgraded:

1. Go to the terminal tab
2. Enter the command: pip install –upgrade pip



```
(venv) PS C:\Users\c3277928\PycharmProjects\pythonProject1> pip install --upgrade pip
```

3. Enter the command: py -m ensurepip –upgrade

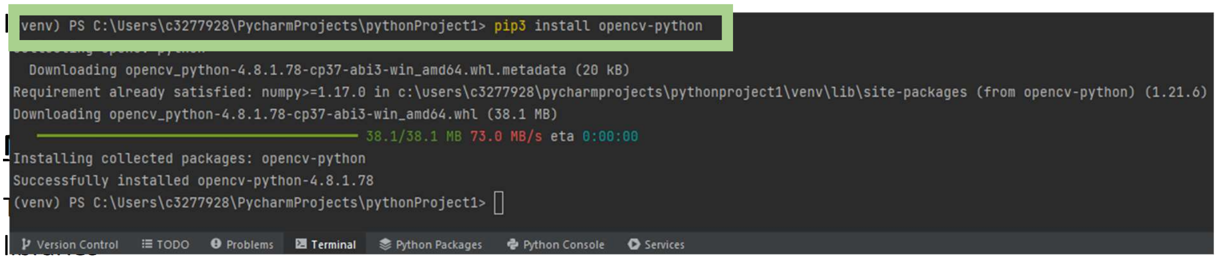


```
(venv) PS C:\Users\c3277928\PycharmProjects\pythonProject1> py -m ensurepip --upgrade
Looking in links: c:\Users\c3277928\AppData\Local\Temp\5\tmptt58vnym
Requirement already up-to-date: setuptools in c:\users\c3277928\pycharmprojects\pythonproject1\venv\lib\site-packages (60.2.0)
Requirement already up-to-date: pip in c:\users\c3277928\pycharmprojects\pythonproject1\venv\lib\site-packages (23.3.1)
```

You should see ‘requirement already up to update with version 23.3.1 installed.’

Now install the opencv library:

1. Go to the terminal tab
2. Enter the command: pip3 install opencv-python

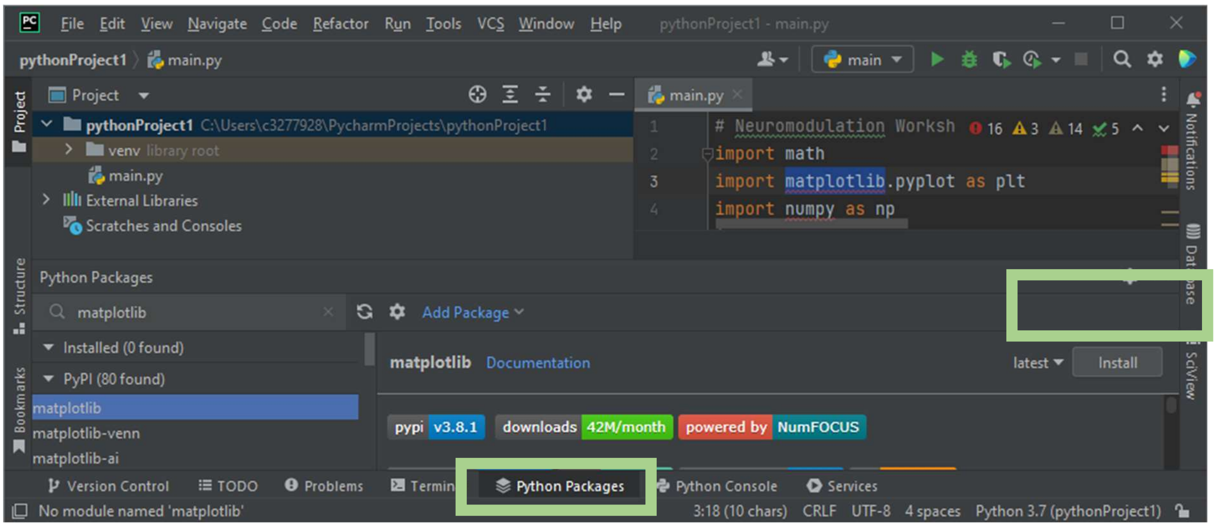


```
venv) PS C:\Users\c3277928\PycharmProjects\pythonProject1> pip3 install opencv-python
  Downloading opencv_python-4.8.1.78-cp37-abi3-win_amd64.whl.metadata (20 kB)
Requirement already satisfied: numpy>=1.17.0 in c:\users\c3277928\pycharmprojects\pythonproject1\venv\lib\site-packages (from opencv-python) (1.21.6)
Downloading opencv_python-4.8.1.78-cp37-abi3-win_amd64.whl (38.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 38.1/38.1 MB 73.0 MB/s eta 0:00:00
Installing collected packages: opencv-python
Successfully installed opencv-python-4.8.1.78
(venv) PS C:\Users\c3277928\PycharmProjects\pythonProject1>
```

- matplotlib,

- numpy,
- cv2,
- scikit-image,
- plotly,

and click the install button. Note that this may take a while to install.



```
# Neuromodulation Worksh
import math
import matplotlib.pyplot as plt
import numpy as np
```

No module named 'matplotlib'

Sometimes, **not required in this workshop**, different versions of libraries are needed to be compatible with code found on the internet. This can be done by selecting the dropdown menu next to the install button. You will often find in GitHub repositories what versions of python and libraries are required to run other peoples contributed code.

Module 5: Coding exercises

For the following 2 days we will be covering the exercises found on the GitHub Repository. For each exercise we hope to achieve the following outcomes:

1. Part One
   a. ‘Exercise1.py’: Simple arithmetic with numbers.
   b. ‘Exercise2.py’: Simple arithmetic with many numbers (vectors).
   c. ‘Exercise3.py’: Print to screen and write to a .txt file.
   d. ‘Exercise4.py’: read from a csv file and perform some statistical analysis.
   e. ‘Exercise5.py’: read from a csv file and plot the data.
   f. ‘Exercise6.py’: read from a csv file and create a histogram plot.
   g. ‘Exercise7.py’: read from a csv file and create a density plot.
   h. ‘Exercise8.py’: read from a csv file and re-create the original image.
2. Part Two
   a. ‘exe1.py’: display a jpg image.
   b. ‘exe2.py’: separate an RGB image.
   c. ‘exe3.py’: use preprocesses on the image to reduce noise.
   d. ‘exe4.py’: create a mask by thresholding the image.
   e. ‘exe5.py’: manipulate the mask image using morphological processes.
   f. ‘exe6.py’: label the masked image into different regions of interest (ROI).
   g. ‘exe7.py’: plot the labelled image.

Module 5: Important libraries and functions

A function is python is something that takes in an argument (input) and returns something back (output). A function may need many arguments and may return many outputs. Libraries contain pre-built functions that you can use. Please insert the following lines at the top of your code. This will install the libraries needed into your virtual environment.

```
import math
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import cv2
from skimage.measure import label, regionprops, regionprops_table
import plotly
import plotly.express as px
import plotly.graph_objects as go
from skimage import data, filters, measure, morphology
```

Functions typically are created to perform computationally quicker or to perform tasks. In the first exercise we use the 'print()' function. This function takes in a string (input) and prints that string to the console (as the output). Some other useful functions (highlighted in red) are listed below, with their outputs highlighted in blue, and there arguments highlighted in green.

- average = np.mean(array): the function np.mean takes in a value or many values and returns the mean of those values in the variable average.
- data = pd.read_csv('cell_properties.csv'): the function pd.read_csv takes in a string (a filename) and returns the result into a variable called data.
- image = plt.imread(filename): the function plt.imread takes in a string stored inside a variable filename and returns an image stored in the variable image

Note that the functions listed start with 'np', 'pd' or 'plt'. This is because they come from those libraries. *For example* ***np.mean()*** *comes from the* ***numpy*** *library which we have named as* ***np****.* Also, functions may have other input and output options available. You can check them out online at the library's documentation. For example, try:
https://numpy.org/doc/stable/reference/generated/numpy.mean.html