BME 2315: Computational Biomedical Engineering



Python Bootcamp

Sarah Groves

UNIVERSITY of VIRGINIA

Adapted from Jeff Saucerman

Outline

- IDEs
- Command line operations
- Control structures
- Data structures
- Unit tests
- Debugging code
- Jupyter notebooks

Integrate Development Environment (IDE)

- A combination of a text editor and a Python (or other language) runtime implementation
- Some IDEs work for multiple languages (like VSCode), some only work for one or focus on one language (like PyCharm, Spider)
- You must tell your IDE where python is located on your computer, and may want to set up a virtual environment to do so
  - Some IDEs automatically find python on your computer

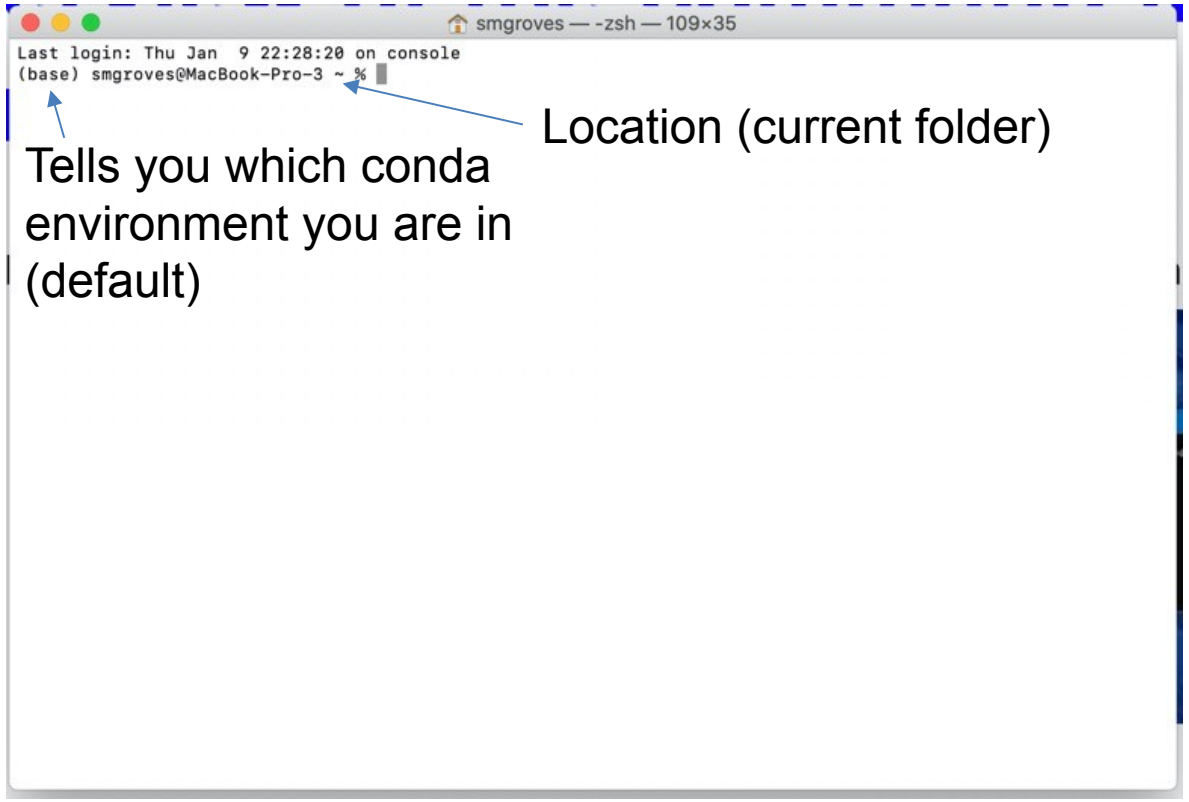
https://www.fullstackpython.com/development-environments.html
https://www.reddit.com/r/Python/comments/8n6cep/what_is_in_your_python_development_environment/

Anaconda installation

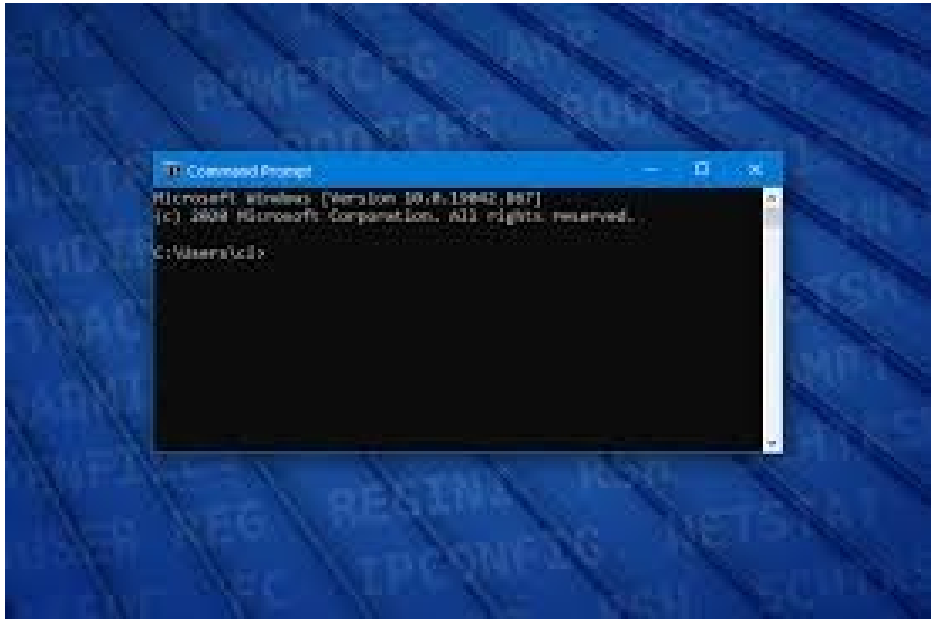
- You should all have Anaconda installed, which installed Python, some basic Python packages, and some applications (including some IDEs like Spyder)
- Anaconda allows you to set up *virtual environments* where you can keep all of the packages for a single project together (to avoid dependency errors, for reproducibility and reuse by other programmers)
  - The *base* environment is the default, where anaconda packages are installed
  - You can make as many environments as you want. This is useful for organizing projects.
  - We will practice making a conda environment for this class that you should **activate** for this class. After you are done working on code for this class, you should **deactivate** the environment. You will need to **activate** the environment when installing new packages or running Jupyter Notebook!
  - When working in an IDE, you can automatically tell the IDE to use Python from a specific environment.

Quick overview of the command line: don't be scared to use it!

Mac: Terminal



Windows: Command Prompt



Conda environment creation with the command line

```
#conda create --name [ENVNAME] python=3.9
conda create --name BME2315 python=3.9

conda activate BME2315

pip install scipy numpy ipykernel jupyterlab

conda list
```

Once a conda environment is activated, any packages installed will go to that location!

```
(base) smgroves@MacBook-Pro-3 ~ % conda activate BME2315

(BME2315) smgroves@MacBook-Pro-3 ~ % pip install scipy numpy

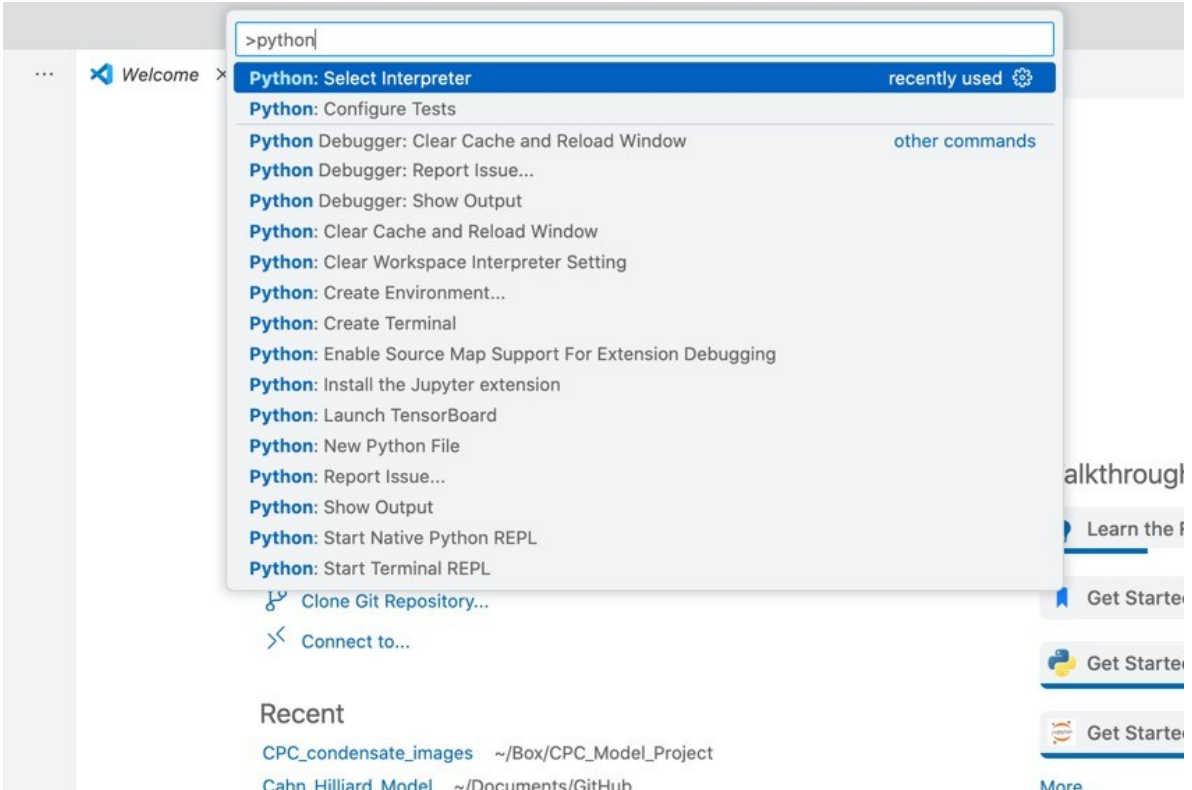
Collecting scipy
  Using cached scipy-1.13.1-cp39-cp39-macosx_10_9_x86_64.whl.metadata (60 kB)
Collecting numpy
  Downloading numpy-2.0.2-cp39-cp39-macosx_10_9_x86_64.whl.metadata (60 kB)
Using cached scipy-1.13.1-cp39-cp39-macosx_10_9_x86_64.whl (39.4 MB)
Downloading numpy-2.0.2-cp39-cp39-macosx_10_9_x86_64.whl (21.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.2/21.2 MB 31.8 MB/s eta 0:00:00
Installing collected packages: numpy, scipy
Successfully installed numpy-2.0.2 scipy-1.13.1
(BME2315) smgroves@MacBook-Pro-3 ~ % conda list
# packages in environment at /Users/smgroves/Documents/anaconda3/envs/BME2315:
#
# Name                    Version                   Build  Channel
bzip2                     1.0.8                hfdf4475_7    conda-forge
ca-certificates           2024.12.14           h8857fd0_0    conda-forge
libffi                    3.4.2                h0d85af4_5    conda-forge
liblzma                   5.6.3                hd471939_1    conda-forge
libsqlite                 3.47.2               hdb6dae5_0    conda-forge
libzlib                   1.3.1                hd23fc13_2    conda-forge
ncurses                   6.5                  h0622a9a_2    conda-forge
numpy                     2.0.2                    pypi_0    pypi
openssl                   3.4.0                hc426f3f_1    conda-forge
pip                       24.3.1             pyh8b19718_2    conda-forge
python                    3.9.21          h7fafba3_1_cpython    conda-forge
readline                  8.2                  h9e318b2_1    conda-forge
scipy                     1.13.1                   pypi_0    pypi
setuptools                75.8.0             pyhff2d567_0    conda-forge
tk                        8.6.13               h1abcd95_1    conda-forge
tzdata                    2024b                hc8b5060_0    conda-forge
wheel                     0.45.1             pyhd8ed1ab_1    conda-forge
(BME2315) smgroves@MacBook-Pro-3 ~ %
```

More conda commands

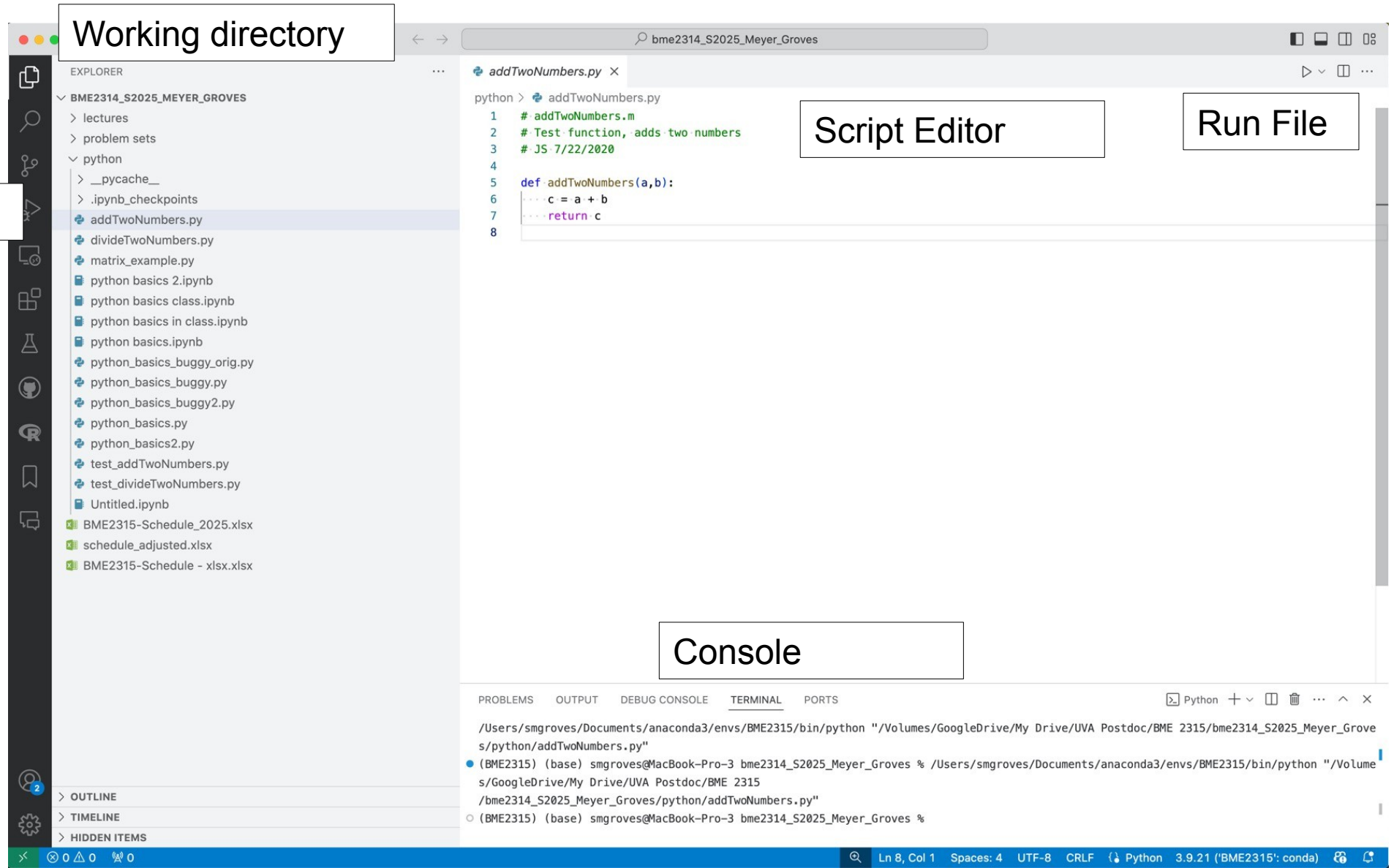
- **conda env list**: list all environments
- **conda list**: list all packages in current environment
- **conda env create --file environment.yml**: create an environment from a yml file (lists needed packages)
- **pip install [package]**: if in correct environment this should install a package in the conda environment
- **conda install [package]:** will install package in active conda environment

Getting your IDE to use the right Python

- You probably have multiple versions of Python installed: type where python in command line to see all of them
- Command palette: Command + Shift + P
- Python: Select Interpreter
- Find the newly made virtual env

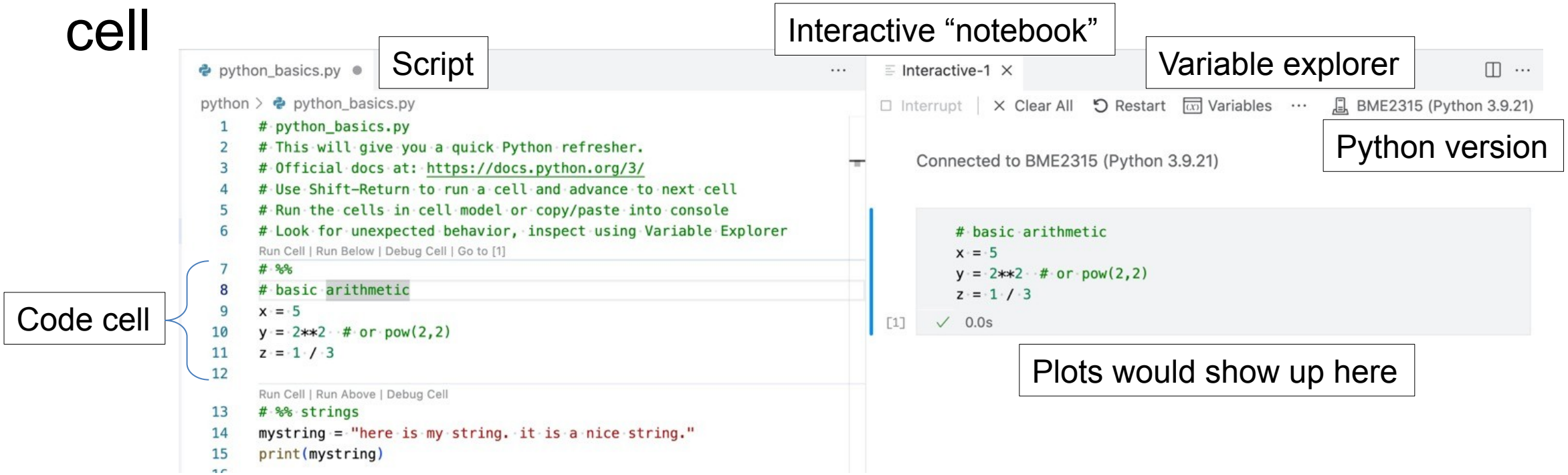


VS Code IDE


Working directory
Debug
Script Editor
Run File
Console
bme2314_S2025_Meyer_Groves
EXPLORER
BME2314_S2025_MEYER_GROVES
lectures
problem sets
python
__pycache__
.ipynb_checkpoints
addTwoNumbers.py
divideTwoNumbers.py
matrix_example.py
python basics 2.ipynb
python basics class.ipynb
python basics in class.ipynb
python basics.ipynb
python_basics_buggy_orig.py
python_basics_buggy.py
python_basics_buggy2.py
python_basics.py
python_basics2.py
test_addTwoNumbers.py
test_divideTwoNumbers.py
Untitled.ipynb
BME2315-Schedule_2025.xlsx
schedule_adjusted.xlsx
BME2315-Schedule - xlsx.xlsx
OUTLINE
TIMELINE
HIDDEN ITEMS
addTwoNumbers.py
python > addTwoNumbers.py
1 # addTwoNumbers.m
2 # Test function, adds two numbers
3 # JS 7/22/2020
4
5 def addTwoNumbers(a,b):
6 c = a + b
7 return c
8
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python
/Users/smgroves/Documents/anaconda3/envs/BME2315/bin/python "/Volumes/GoogleDrive/My Drive/UVA Postdoc/BME 2315/bme2314_S2025_Meyer_Groves/python/addTwoNumbers.py"
(BME2315) (base) smgroves@MacBook-Pro-3 bme2314_S2025_Meyer_Groves % /Users/smgroves/Documents/anaconda3/envs/BME2315/bin/python "/Volumes/GoogleDrive/My Drive/UVA Postdoc/BME 2315
/bme2314_S2025_Meyer_Groves/python/addTwoNumbers.py"
(BME2315) (base) smgroves@MacBook-Pro-3 bme2314_S2025_Meyer_Groves %
Ln 8, Col 1
Spaces: 4
UTF-8
CRLF
Python
3.9.21 ('BME2315': conda)

Interactive mode in Python

- #%% will make code cells that can run in interactive mode
- Hover over cell to Run or Debug or click Shift + Enter to run cell



https://code.visualstudio.com/docs/python/jupyter-support-py

Python_basics.py

- Arithmetic
  - +, -, *, /
  - ** = power
  - % = remainder; // = divide (discard remainder)
- Relational
  - <, <= : is less than
  - >, >= : is greater than
  - == : is equal to
  - != : is not equal to
- Logical
  - and
  - or
  - Not
- Assignment
  - = : assign a variable
  - A += B : add B and reassign new sum to A
  - -=, *=
- Identity
  - Is
  - Is not
- Membership
  - In
  - Not in

Python_basics.py: basic control structures

- **While** loop: Loop until the condition is not met
  - These can be dangerous and lead to never-ending code! **Make sure** you have a break condition that will be met for all inputs.
- **For** loop: Loop through a set number of iterations defined in condition
- **If/elif/else:** If a condition is met, do an action.
- **Break**: exit the loop if the condition is met
- **Continue**: skips the current iteration and proceeds to the next iteration of the loop
- **Pass**: does nothing (placeholder)

```
#%% nested loop.
for i in range(0,4):
    for j in range(0,4):
        print('Counter = (', i,',', j,')')


#%% while loop
i = 0
while i < 5:
    print('Counter = ', i)
    i += 1

#%% conditional expressions
i == 4
i <= 5
i != 5
(i == 4) and (i <= 5)
(i == 4) or (i <= 5)

#%% create functions

def addTwoNumbers(a,b):
    c = a + b
    return c

# single line function definition
def multiplyTwoNumbers(a,b): return a*b

seven = addTwoNumbers(3,4)
six = multiplyTwoNumbers(2,3)
```

Open the Lockbox Challenge!

Reminder:

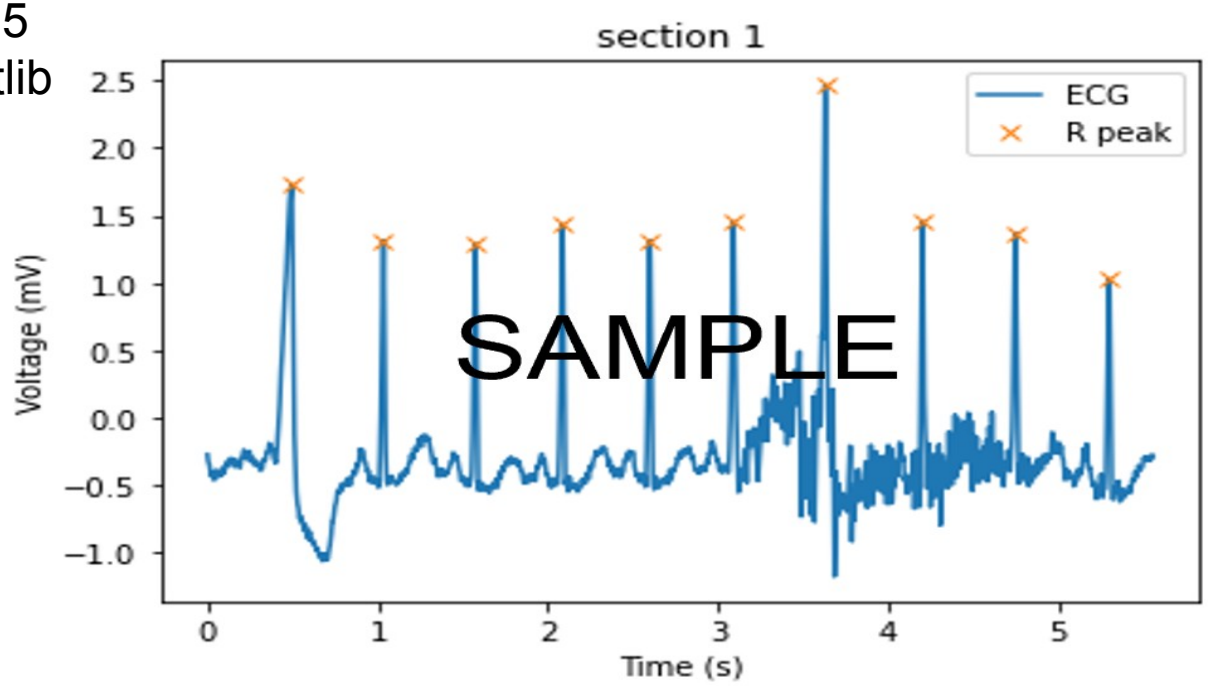
- Arithmetic
  - +, -, *, /
  - ** = power
  - % = remainder; // = divide (discard remainder)
- Relational
  - <, <= : is less than; less than or equal to
  - >, >= : is greater than; greater than or equal to
  - == : is equal to
  - != : is not equal to
- Logical
  - (A) and (B)
  - (A) or (B)
  - Not A
- Assignment
  - = : assign a variable
  - A += B : add B and reassign new sum to A
  - -=, *=
- Identity
  - Is
  - Is not
- Membership
  - In
  - Not in

- Step 1: Solve your problem (without a computer) and find others who have the same answer as you.
- Step 2: Get 4 new problems and solve as a group (without a computer). Go to smgroves.github.io/codebreak and enter your answers as a code.
- Step 3: If you unlocked the Final Challenge page, follow the instructions on the page.
- Goal: Open the lockbox!!

Problem Set 2 due 1/28

1) Programming challenges: writing functions
2) Fibonacci sequence
3) Plotting and analyzing electrocardiogram (ECG) data

Conda activate BME2315
Pip install numpy matplotlib
Pip install seaborn
Pip install pandas
Conda deactivate



Data structures: Lists, arrays, etc.

- List
  - Built into Python
  - Can store elements of different types
  - Dynamic size (can easily change size)
  - Best for 1D lists of things
- Array
  - Use **numpy**
  - Store a single datatype
  - Fixed size (not a good idea to change its size)
  - More memory efficient (contiguous memory)
  - Can be multidimensional
- Dataframes
  - Use **pandas**
  - Best for 2D data, especially with named columns and rows
- Sets
  - Built into Python
  - Only has unique elements
  - Unordered
- Dictionary
  - Key-value pairs; both keys and values can be any data type!
  - Ordered (as of Python 3.7)

Python_basics2.py: arrays etc.

```
# python_basics2.py
# This will give you a quick overview of numpy and plotting
# Official docs at: https://docs.python.org/3/
# Official Numpy tutorial at: https://numpy.org/doc/stable/user/quickstart.html

#%% creating matrices and array indexing using Numpy
import numpy as np

a = np.array([1,2,3,4,5,7])
b = np.array([[6,4],[5,1],[3,8]])
c = np.array([[1,2,3],[4,5,6],[7,8,9]],dtype=float)

# inspecting your array
a.dtype
numrow, numcol = b.shape
c.ndim

# array indexing
a[0:2]
a[-1]   # last element
a[::2]  # every other element
b[2,0]
b[:,1]  # second column
idx = np.array([1,3,4])
a[idx]
```

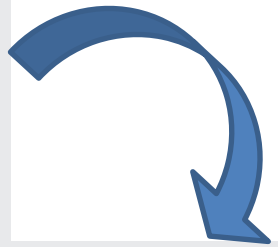
List comprehension: the Pythonic way to code

- Make a list based on the values of an existing list using for loops, if statements, etc. **all in one line**!

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
newlist = []

for x in fruits:
  if "a" in x:
    newlist.append(x)

print(newlist)
```



```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]

newlist = [x for x in fruits if "a" in x]

print(newlist)
```

Plotting: matplotlib and seaborn

```
Run Cell | Run Above | Debug Cell
# %% mathematical functions and plotting using Matplotlib
import matplotlib.pyplot as plt

x = np.arange(0, 10, 0.1)
y = np.sin(x) * np.exp(-0.2 * x)

plt.plot(x, y, "o-")
plt.xlabel("x")
plt.ylabel("y")
```

```
# %% seaborn
import seaborn as sns

sns.lineplot(data=df, x="A", y="B", hue="C", style="C", markers=True)
plt.show()
```

Unit Tests

- Checks code in small chunks (units) for errors
- Always test that your code works in known conditions before applying it to unknown conditions (e.g. real data)
- Can compare computed answer to:
  - Exact analytical solution when available
  - Numerical solution obtained by hand
  - When answer is unknown, convergence to a stable answer

Example unit tests

```python
# test_addTwoNumbers

from addTwoNumbers import addTwoNumbers

def test1():
    computed = addTwoNumbers(3,4)
    expected = 7
    assert computed == expected
    print("Test 1 completed.")

def test2():
    computed = addTwoNumbers(5,8)
    expected = 13
    assert computed == expected
    print("Test 2 completed.")

test1()
test2()
```

```python
# test_divideTwoNumbers

from divideTwoNumbers import divideTwoNumbers

def test1():
    computed = divideTwoNumbers(3,4)
    expected = 0.75
    assert computed == expected        # tests for equal
    print("Test 1 completed.")

# When exact solutions cannot be provided, check if the difference between
# computed and expected results are within a tolerance.

def test2():
    computed = divideTwoNumbers(2,3)
    tol = 1e-3
    expected = 0.6666
    error = abs(computed - expected)
    assert  error < tol # checks if answer within a specified tolerance
    print("Test 2 completed.")


test1()
test2()
```

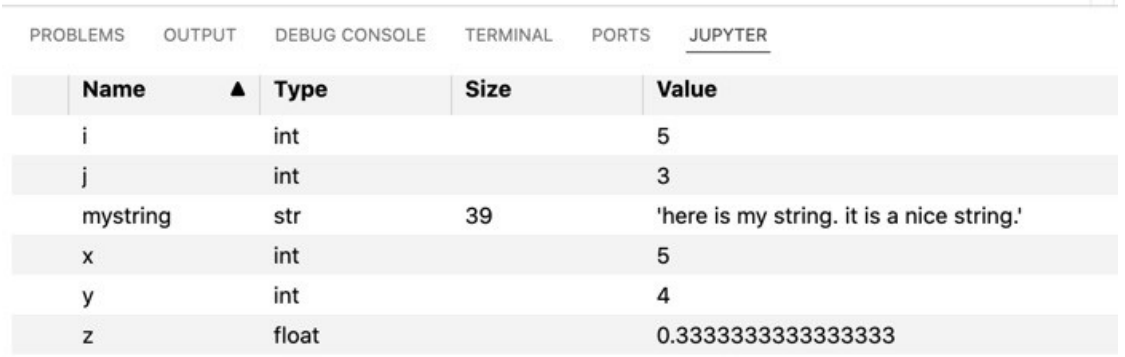
Unit Tests

Requirements of a test function

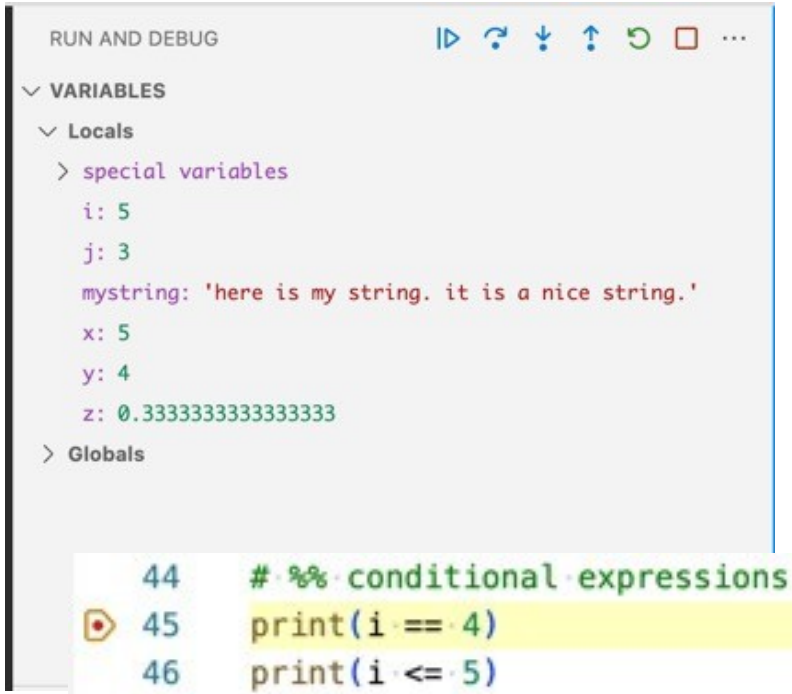
- Unit test filename starts with “test_”
- Unit test functions (inside unit test file) start with “test_”
- Test functions have no input arguments
- Test functions evaluate a Boolean expression using *assert*

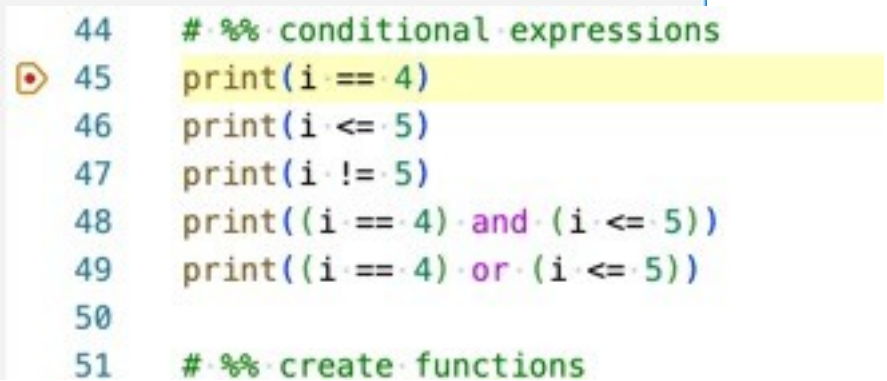
Debugging Code

- For simple programs, may be able to debug by adding print statements
- Interactive Mode Variable Explorer helps you see current variable states.
- Cell mode runs code in chunks
- Debugger mode lets you run code interactively by adding breakpoints



Name	Type	Size	Value
i	int		5
j	int		3
mystring	str	39	'here is my string. it is a nice string.'
x	int		5
y	int		4
z	float		0.3333333333333333





Jupyter Notebooks (Jupyter Lab)

Combines text descriptions, graphs, and working code in the same document.

It is best for telling a “code story” from data input to data output.

Runs within your browser, enables you to dynamically change the code and get new results.

Highly useful for documenting what the code does.

**Not recommended for writing complex functions that may require debugging. Instead I recommend Jupyter notebooks for documenting working code to share the outputs.*

Good Tutorial: https://www.youtube.com/watch?v=jZ952vChhuI

jupyter python basics in class Last Checkpoint: 07/28/2023 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykerne

Run Markdown

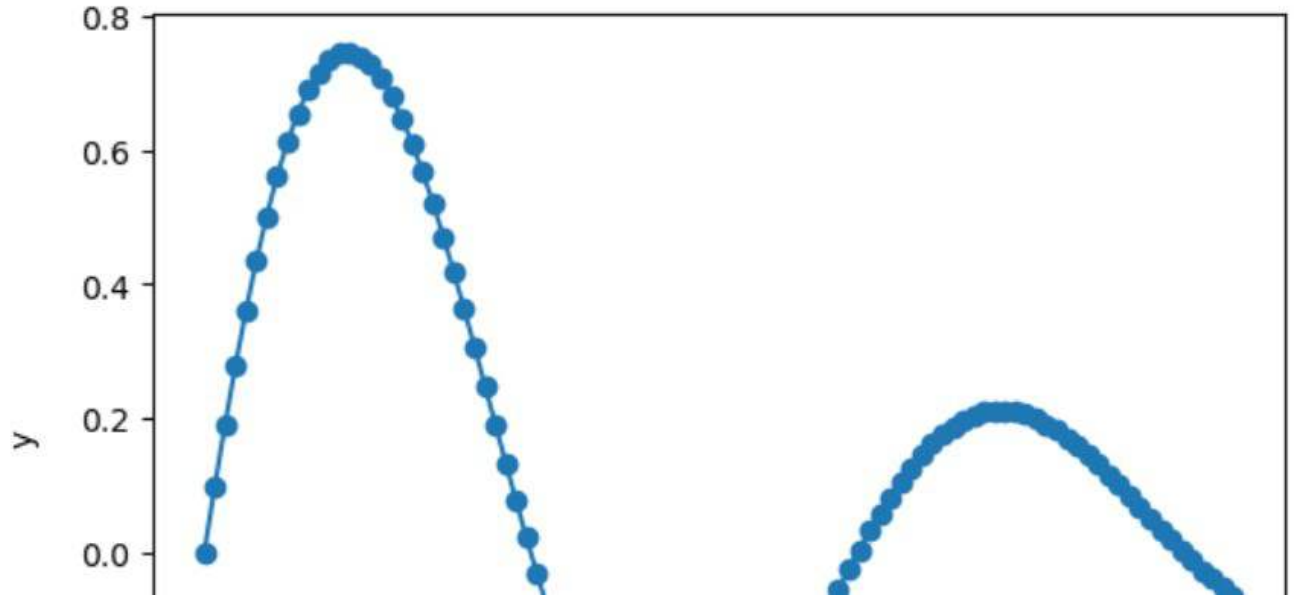
Plot a function using matplotlib

In [3]:
```
import matplotlib.pyplot as plt

x = np.arange(0,10,0.1)
y = np.sin(x)*np.exp(-0.2*x)

plt.plot(x,y,'o-')
plt.xlabel('x')
plt.ylabel('y')
```

Out[3]: Text(0, 0.5, 'y')



Problem Set 2 due 1/28

1) Programming challenges: writing functions
2) Fibonacci sequence
3) Plotting and analyzing electrocardiogram (ECG) data

