XSEDE

Extreme Science and Engineering Discovery Environment

Python libraries for scientific computing

Presented by:

- Steve Lantz - steve.lantz@cornell.edu

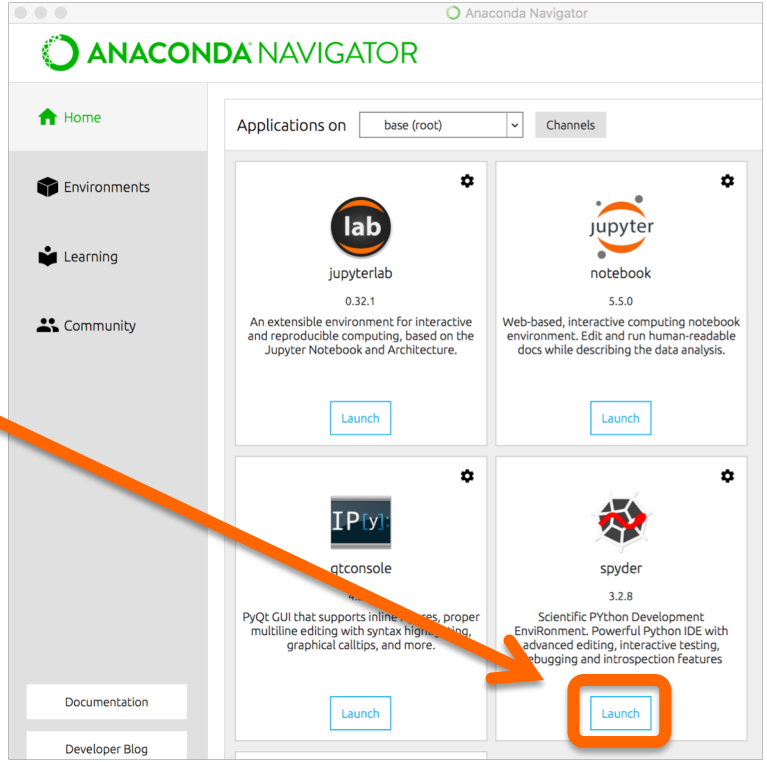
With thanks to:

- Roberto Camacho - rcamachobarranco@utep.edu

Anaconda orientation

- Anaconda Navigator
- The Spyder IDE
  - Panes and buttons
  - Built-in tools

- Download all the workshop files here: tinyurl.com/y3me2jpc

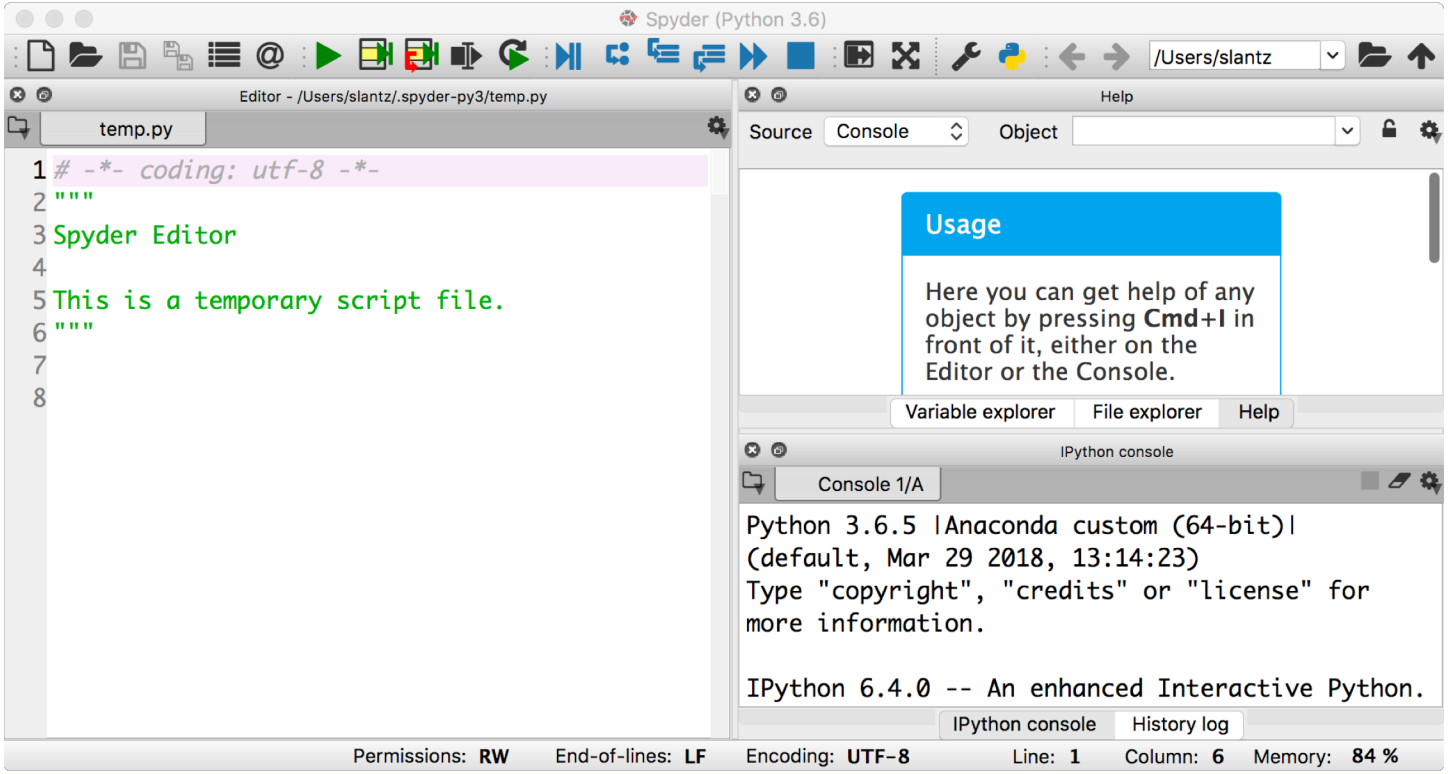


Getting started

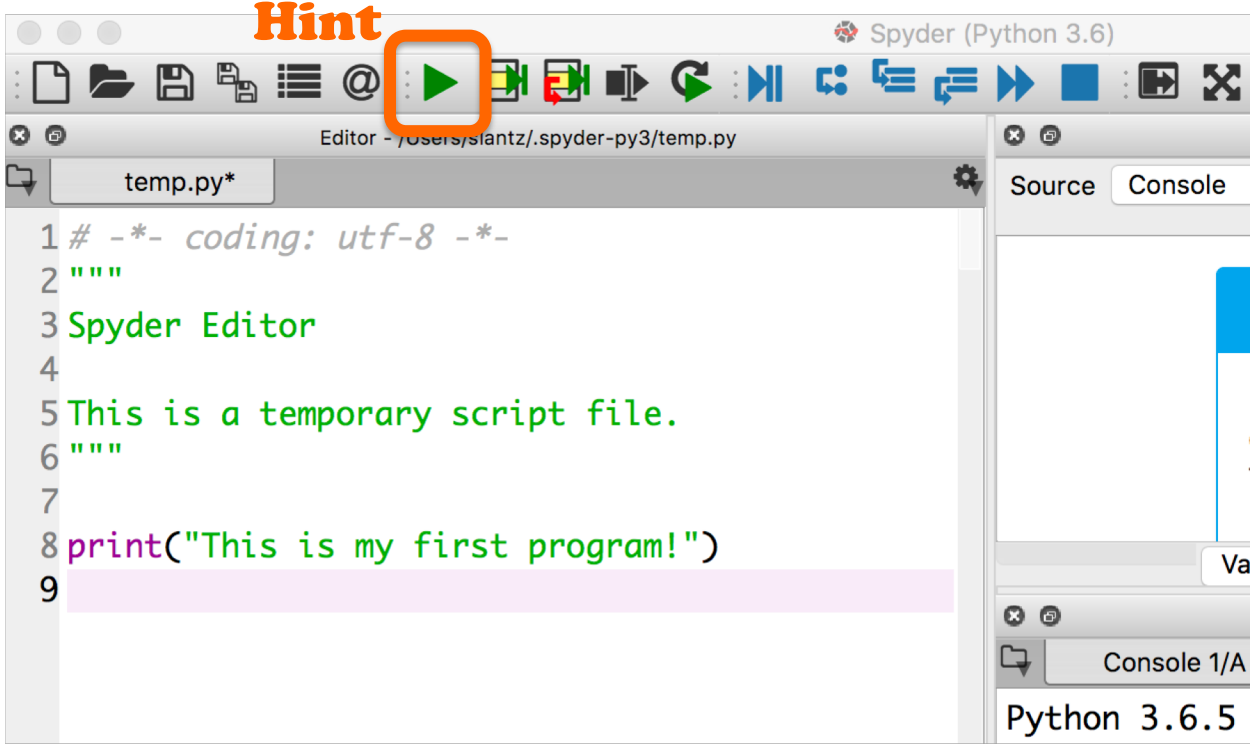
- You should have Anaconda installed and Spyder open

- We are going to use Spyder to write and run our program
  - Spyder = **S**cientific **PY**thon **D**evelopment **E**nvi**R**onment
  - Spyder checks your program for syntax errors (spelling, punctuation…)
  - Spyder lets you see the variables that are defined and their values
  - Spyder includes a debugger (pdb) to help you find problems

The Spyder window



First program: how to run it



First program: what's in it

```
# -*- coding: utf-8 -*-
"""
Spyder Editor
This is a temporary script file.
"""
```

The text between """ is a docstring ➔ it helps you to know what the code does, but it's not executed. For commenting code, use #; all of text to the right of the # symbol will be a comment

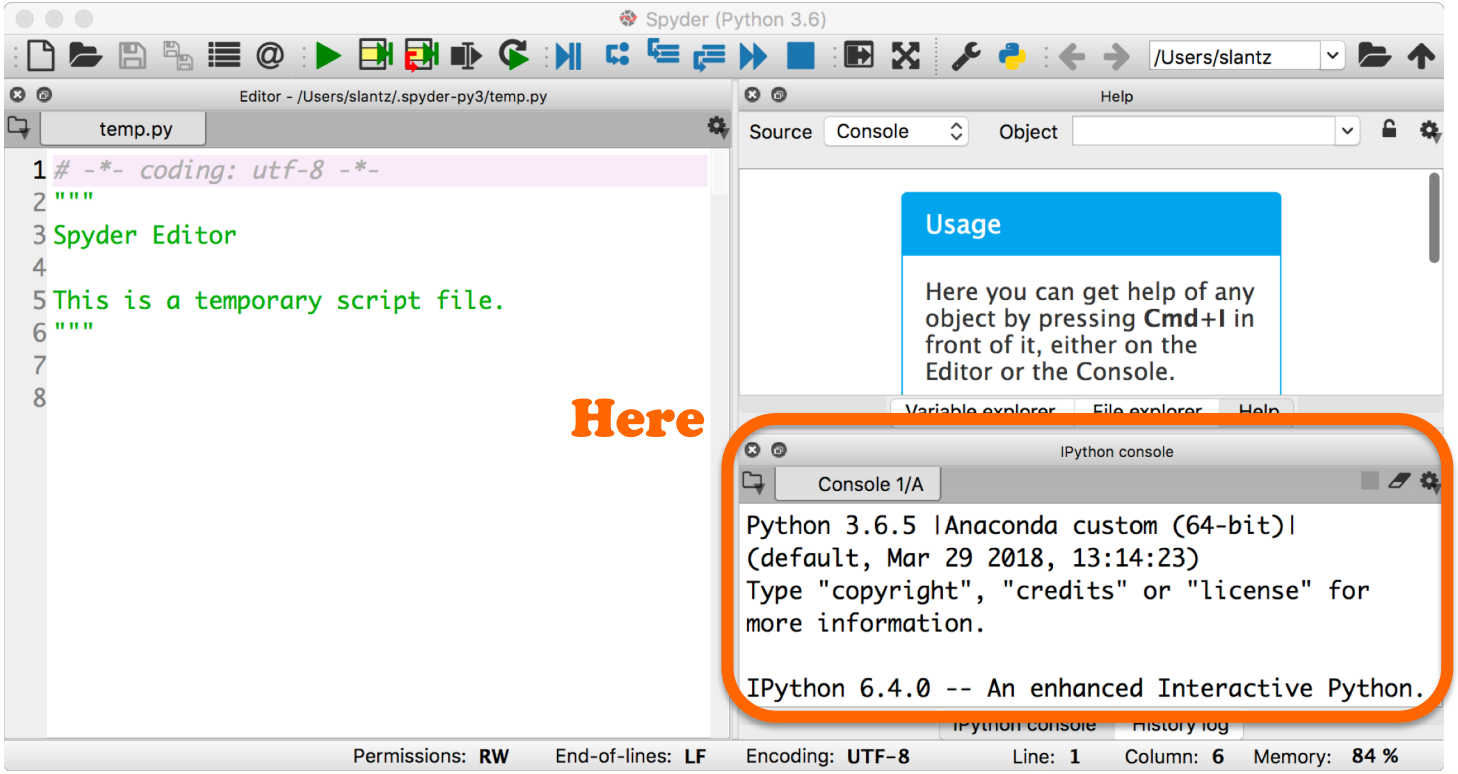
```
print("This is my first program!")
```

print() is a built-in function in Python 3. This is the code that will be executed

What happens when you run your program?

- There is a python interpreter
- The interpreter understands code in Python (the language)
- It converts Python code to something that a microprocessor understands
- Any microprocessor that can run a Python 3 interpreter will be able to run your program!

The IPython console (pane)



What is IPython?

- IPython = Interactive Python
- It's an enhanced interface to the standard python interpreter
  - IPython calls the interpreter a "kernel"
- A Jupyter notebook is in turn a fancy interface to IPython
  - Allows IPython to run in a browser window
  - Code blocks are interspersed with formatted text
- Spyder is just a different enhanced interface to IPython

Review of libraries and modules

- Python includes many external libraries or modules that provide additional functionality
  - Mathematical functions: math, NumPy, SciPy
  - System interaction: os
  - Plotting: matplotlib
- You can also define your own modules
  - Helpful to group a lot of functionality together and reuse it

import

- To use a module, you have to tell Python that you want it:

```
import math
```

- After that, you have access to math functions by entering **math + “.” + function name**

```
math.floor(math.pi)
math.sqrt(9)
```

from X import Y

- If you know that you only need one part of the module, like the “floor” function from math, just import it like this:

```
from math import floor
```

  - Now you can call `floor(3.14)`
  - But you can’t call `floor(math.pi)`

import X as Y

- Very often you will see code with something like:

```
import numpy as np
```

- The “as” name is usually a short form of the module name
- It is often clearer and definitely easier to type!

```
vect = np.linspace(1.,4.,6)
```

Handy libraries for scientific computing

- NumPy offers fast multi-dimensional arrays, and more
  - Trig functions, curve fitting, sorting, searching, Fourier transforms
  - Most routines rely on optimized libraries coded in C and Fortran

- SciPy extends NumPy to include higher-level capabilities
  - Optimization, integration, interpolation, signal/image processing
  - Similar functionality to MATLAB or Octave

The role of Anaconda

- The Anaconda distribution comes with NumPy and SciPy
- It comes with many more libraries as well
  - matplotlib: provides powerful plotting tools like MATLAB's
  - pandas: Python Data Analysis Library, some similarities to R
  - SymPy: symbolic mathematics (think of mistake-free algebra)
- Plus it has the conda package manager to help you get additional packages
- But you don't have to depend on Anaconda for *everything*…

You can create your own module

- Create a .py file with various functions that you need:

```
def function1(…):
def function2(…):
```

- Save it to my_module.py and import it from another .py file:

```
import my_module
```

- You should now be able to call: `my_module.function1(…)`

Is it a module, or is it a program? Yes!

- Your module can include a "**main**" program
- You protect it by putting all its statements inside an if block
  - If the file is run as a program, the **main** code is executed
  - If the file is imported as a module, the **main** code is not executed

```
if __name__ == "__main__":
    first_statement_of_your_program
```

Objects, classes

Everything in Python is an *object* belonging to a *class*

- ***Objects*** are entities that combine data together with the possible actions that can be taken on the data
- ***Class*** is the special name given to the type of an object; a class's definition includes both ***attributes*** (data members) and ***methods*** (actions)

Libraries are a way of giving you access to new classes

Example: objects of class *list*

- List is a class that lets you put subscripts on variables
- Each item in a list object is assigned a position (index)
- You can access each index using [ ]; first index is 0
- Items in the list can be of different type

```
mylist1 = ["first", "second", 3]
mylist2 = [1, 2, 5.0, 6.0]
```

Special methods for list objects

- You can use *indexing* to assign a new value to a list item:

```
mylist2[0] = 10
print(mylist2)
```

- You can also use *slicing* (note, a slice begins *at* the first index and ends *before* the second index):

```
print(mylist2[0:2])
```

The “in” keyword for lists

- You can check if a value exists in a list (True/False answer)

```
1 in mylist2
```

- You can (of course) iterate over the elements of a list

```
for x in mylist2:
    print(x + 0.5)
```

Standard methods for lists (dot notation)

- It's possible to add new items to a list:

```
my_list.append(new_item)
```

- Or find the position of the first occurrence of an item in a list:

```
my_list.index(item)
```

- Or count how many times a given item appears in the list:

```
my_list.count(new_item)
```

Functions for lists

- You can get the number of items in a list

```
len(mylist2)
```

- You can find the maximum and minimum values in a list

```
max(mylist), min(mylist)
```

- These functions are defined for virtually all sequence-type objects, e.g., strings

How are NumPy array objects different from lists?

- NumPy arrays can have multiple dimensions
- Indexing and slicing work similarly; so does “in”
- Methods and functions are different
- 1D NumPy arrays have the most similarity to lists
  - It is easy to convert a list to a 1D array
  - Either a list or a 1D array can be used in a plot
  - But a 2D array is *not* a list of lists
- We will see the differences in example programs today

matplotlib

- **matplotlib** is the most popular plotting library in Python

```
import matplotlib.pyplot as plt
```

- You normally plot lists (try this):

```
myx = [1,2,3,4]
myy = [2,3,4,5]
plt.plot(myx, myy)
```

- But a 1D NumPy array will work just as well

Exercise: plotting lists and arrays

- Construct lists and arrays of 10 numbers
  - Use the list() constructor, with a call to range() as input
  - Use the np.array() constructor in the same way
  - Use the output the np.arange() function
- Plot these lines against each other using matplotlib
- Change the colors and symbols; update the plot
  - See the tables on the next slide

Changing the plot style

- `plt.plot(myx, myy, c=color, s=symbol)`
- Combine colors and markers to create different styles

Symbol	Color
'r'	red
'b'	blue
'g'	green
'c'	cyan
'm'	magenta
'y'	yellow
'k'	black
'w'	white

Symbol	Marker/style
'o'	Use circles
'^' 'v' '<' '>'	Use triangles
's'	Use squares
'*'	Use stars
'-'	Single dashes
'--'	Double dashed line

Should we program with lists or arrays?

- Both!
  - "Plain Python" is useful and good to know
  - But if you have a *Need for Speed*…
  - It's good to know NumPy, SciPy, and friends
- A code can evolve based on requirements
  - If a code is slow, try better methods
  - Try to make the best use of your hardware
  - Ultimately, try using better hardware

Introduction to Monte Carlo methods

- Obtaining numerical results though random sampling
- It works much like an opinion poll
  - With enough samples, you can estimate the real answer
- Application examples:
  - Numerical integration
  - Optimization (finding the best solution)
  - Simulating a probabilistic process
- Named for a casino in Monaco

Our quest for today: π

- Randomly sample points in a square
- Some lie inside the inscribed circle
- Number is proportional to area
- Ratio of areas is $\pi / 4$
- This lets us compute π
- We do it by numerical integration
- Convergence is slow