Business Analytics Programming

Introduction (Class, Spyder, Numpy, Pandas)

Dr. Wajahat Gilani

Rutgers Business School

January 30, 2019

Bio - Dr. Wajahat Gilani

Academics (All Rutgers)

- BS Computer Science and BA Economics
- Master in Quantitative Finance (before the program split)
- Ph.D. in Optimal Investing in Illiquid and Incomplete Markets

Experience

- Merrill Lynch Investment Managers (BlackRock) - Front Office Application developer/Junior Quantitative Analyst
- JP Morgan Chase - Jumbo loans quant
- Libertas Partners - Desk Quant (ABS/CDO) "The Big Short"
- Libertas Partners - Desk Quant (ABS/CDO) "The Big Short"
- Citigroup - FX Derivative Quant
- Quant Consultant for Hedge Funds
- Private Investment Fund

What is Business Analytics Programming?

Data Science is primarily broken into 3 categories that tend to overlap.

- data wrangling - what data?
- business intelligence - what happened?
- business analytics - what will happen, predictive analytics

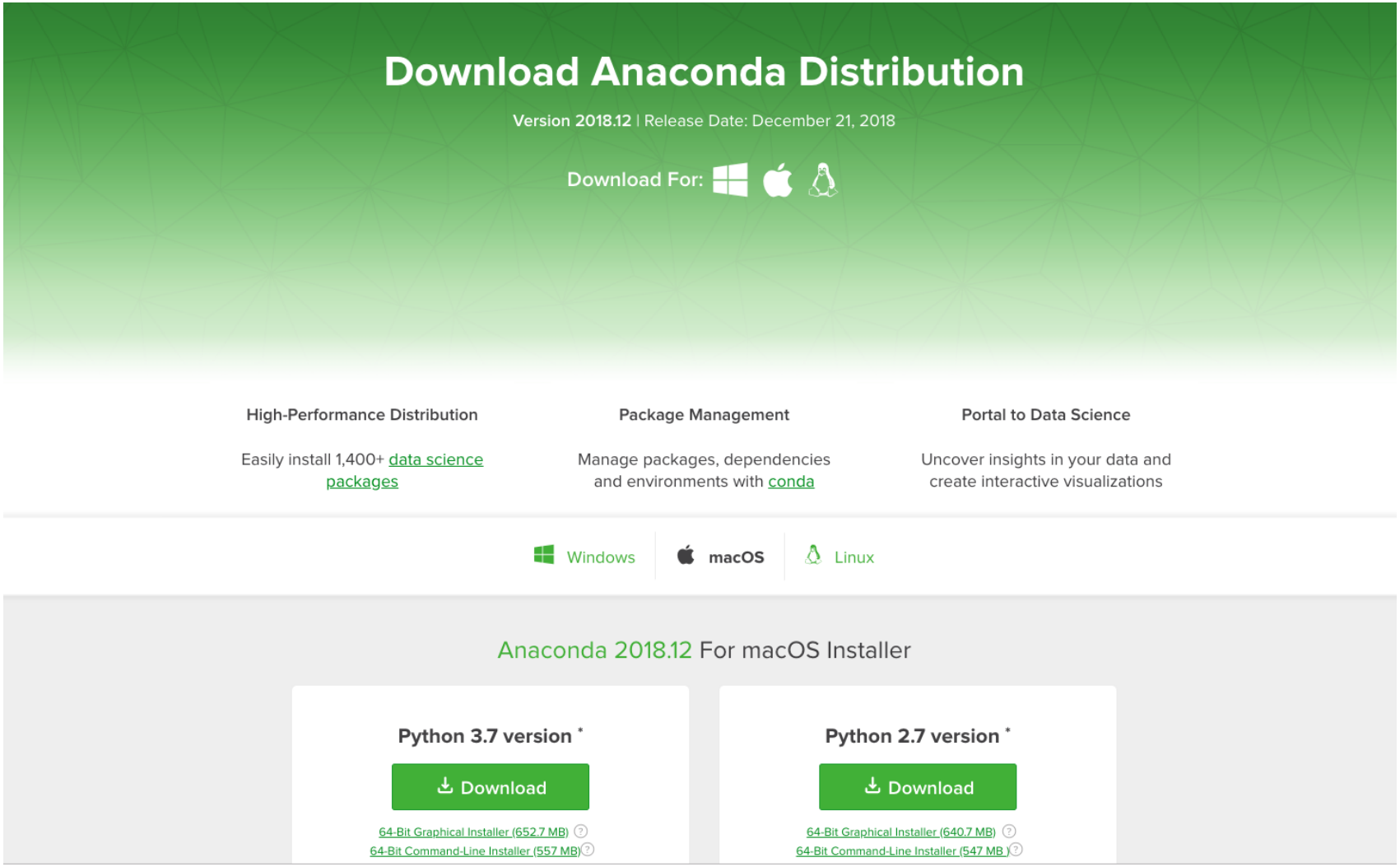
The classes that you are taking, are teaching you the tools to answer those questions, and again, they overlap.

- data mining (clustering, classification trees, random forests)
- machine learning
- neural networks
- AI

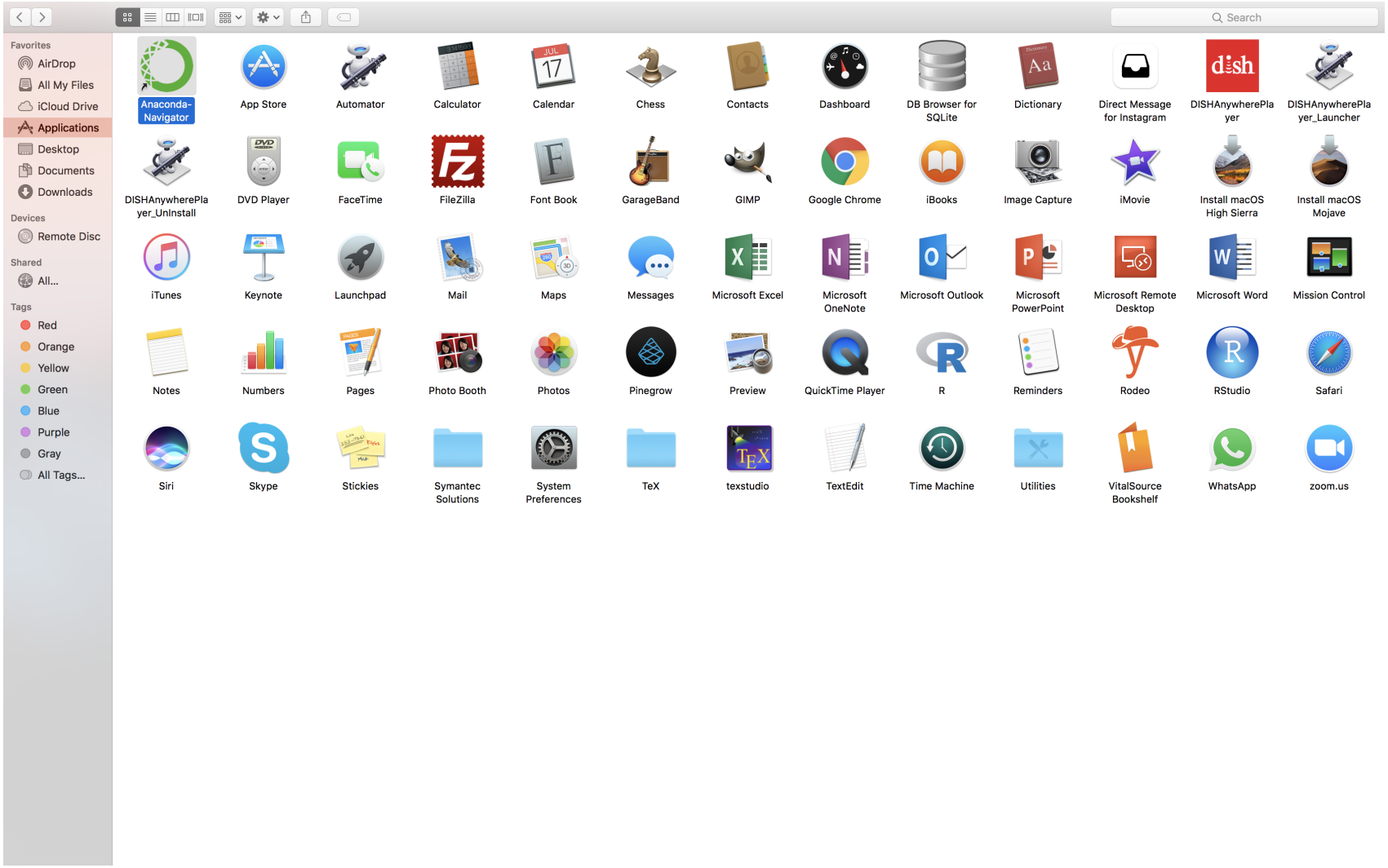
We are basically attempting to use python to analyze data and attempt to predict what will possibly happen in the future.

Download Anaconda

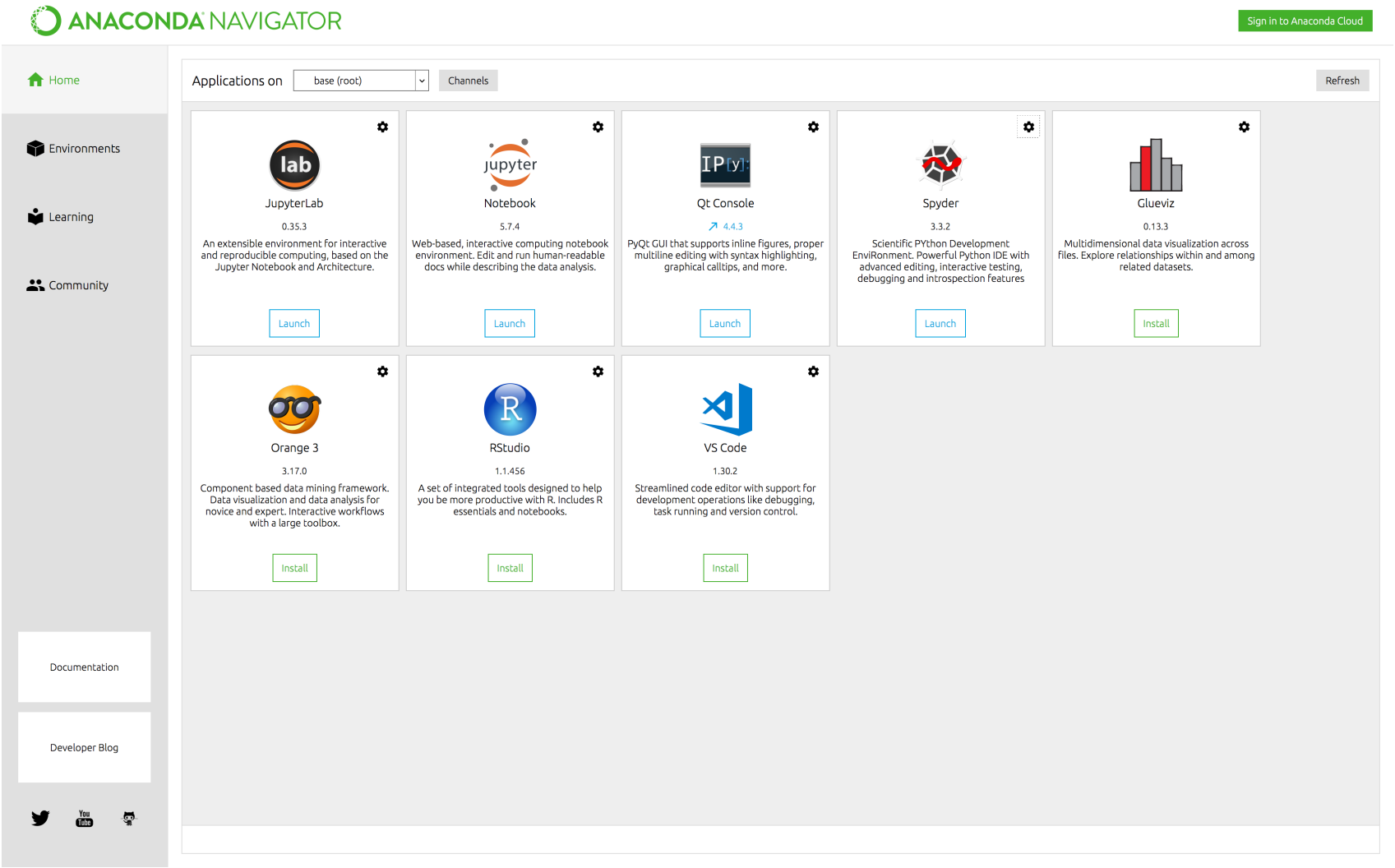
https://www.anaconda.com/download/ (python 3)



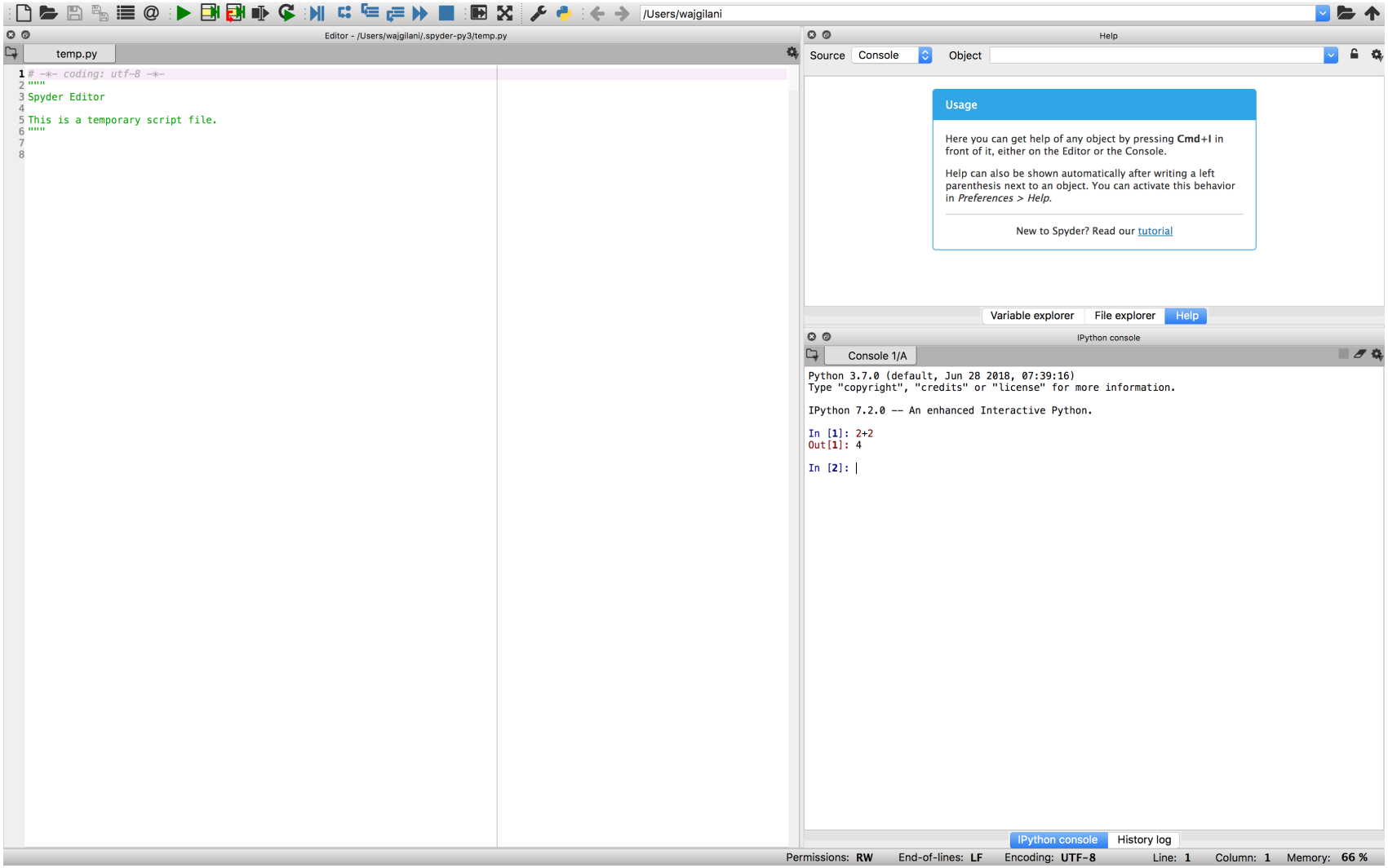
Open Anaconda Navigator



Open Spyder


ANACONDA NAVIGATOR
Sign in to Anaconda Cloud
Home
Environments
Learning
Community
Documentation
Developer Blog
Applications on
base (root)
Channels
Refresh
JupyterLab
0.35.3
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
Launch
Notebook
5.7.4
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch
Qt Console
4.4.3
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.
Launch
Spyder
3.3.2
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features
Launch
Glueviz
0.13.3
Multidimensional data visualization across files. Explore relationships within and among related datasets.
Install
Orange 3
3.17.0
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.
Install
RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.
Install
VS Code
1.30.2
Streamlined code editor with support for development operations like debugging, task running and version control.
Install

Spyder IDE


/Users/wajgilani
Editor - /Users/wajgilani/.spyder-py3/temp.py
temp.py
-*- coding: utf-8 -*-
"""
Spyder Editor
This is a temporary script file.
"""
Help
Source
Console
Object
Usage
Here you can get help of any object by pressing Cmd+I in front of it, either on the Editor or the Console.
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.
New to Spyder? Read our tutorial
Variable explorer
File explorer
Help
IPython console
Console 1/A
Python 3.7.0 (default, Jun 28 2018, 07:39:16)
Type "copyright", "credits" or "license" for more information.
IPython 7.2.0 -- An enhanced Interactive Python.
In [1]: 2+2
Out[1]: 4
In [2]:
IPython console
History log
Permissions: RW
End-of-lines: LF
Encoding: UTF-8
Line: 1
Column: 1
Memory: 66 %

Spyder IDE - .py file

python File Edit Search Source Run Debug Consoles Projects Tools View Help

lab1.py

```
1 # -*- coding
2 """
3 Spyder Edit
4
5 This is a te
6 """
7
8
9
```

New file...	⌘N
Open...	⌘O
Open last closed	⇧⌘T
Open recent	▶
Save	⌘S
Save all	⌥⌘S
Save as...	⇧⌘S
Save copy as...	
Revert	
Print preview...	
Print...	
Close	
Close all	⇧⌘W
File switcher...	⌘P
Symbol finder...	⌥⌘P
Restart	⌥⇧R

Spyder IDE - Console

IPython console

Console 1/A

```
Python 3.7.0 (default, Jun 28 2018, 07:39:16)
Type "copyright", "credits" or "license" for more information.

IPython 7.2.0 -- An enhanced Interactive Python.

In [1]: 2+2
Out[1]: 4

In [2]:
```

Spyder IDE - Console

/Users/wajgilani/Documents/BAP/BAP Spring 2019

Editor - /Users/wajgilani/Documents/BAP/BAP Spring 2019/lab1.py

lab1.py

```
1 # -*- coding: utf-8 -*-
2 """
3 Spyder Editor
4
5 This is a temporary script file.
6 """
7
8
9 print(2+2)
```

Help

Source Console Object

Usage

Here you can get help of any object by pressing **Cmd+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

Variable explorer File explorer Help

IPython console

Console 4/A

```
Python 3.7.0 (default, Jun 28 2018, 07:39:16)
Type "copyright", "credits" or "license" for more information.

IPython 7.2.0 -- An enhanced Interactive Python.

In [1]: runfile('/Users/wajgilani/Documents/BAP/BAP Spring 2019/lab1.py', wdir='/Users/
wajgilani/Documents/BAP/BAP Spring 2019')

In [2]: runfile('/Users/wajgilani/Documents/BAP/BAP Spring 2019/lab1.py', wdir='/Users/
wajgilani/Documents/BAP/BAP Spring 2019')

In [3]: runfile('/Users/wajgilani/Documents/BAP/BAP Spring 2019/lab1.py', wdir='/Users/
wajgilani/Documents/BAP/BAP Spring 2019')
4

In [4]:
```

IPython console History log

Permissions: RW End-of-lines: LF Encoding: UTF-8 Line: 9 Column: 11 Memory: 63 %

Spyder IDE - Console

/Users/wajgilani/Documents/BAP/BAP Spring 2019

Editor - /Users/wajgilani/Documents/BAP/BAP Spring 2019/lab1.py

lab1.py

```
1  # -*- coding: utf-8 -*-
2  """
3  Spyder Editor
4
5  This is a temporary script file.
6  """
7
8  import numpy as np
9  import pandas as pd
10
11 x=[4,5,6]
12 print(x)
13
14 a=5
15
```

Variable explorer

Name	Type	Size	Value
a	int	1	5
x	list	3	[4, 5, 6]

Variable explorer | File explorer | Help

IPython console

Console 4/A

```
In [6]: runfile('/Users/wajgilani/Documents/BAP/BAP Spring 2019/lab1.py', wdir='/Users/
wajgilani/Documents/BAP/BAP Spring 2019')

In [7]: runfile('/Users/wajgilani/Documents/BAP/BAP Spring 2019/lab1.py', wdir='/Users/
wajgilani/Documents/BAP/BAP Spring 2019')

In [8]: runfile('/Users/wajgilani/Documents/BAP/BAP Spring 2019/lab1.py', wdir='/Users/
wajgilani/Documents/BAP/BAP Spring 2019')

In [9]: runfile('/Users/wajgilani/Documents/BAP/BAP Spring 2019/lab1.py', wdir='/Users/
wajgilani/Documents/BAP/BAP Spring 2019')
[4, 5, 6]

In [10]: runfile('/Users/wajgilani/Documents/BAP/BAP Spring 2019/lab1.py', wdir='/Users/
wajgilani/Documents/BAP/BAP Spring 2019')
[4, 5, 6]

In [11]: runfile('/Users/wajgilani/Documents/BAP/BAP Spring 2019/lab1.py', wdir='/Users/
wajgilani/Documents/BAP/BAP Spring 2019')
[4, 5, 6]

In [12]: runfile('/Users/wajgilani/Documents/BAP/BAP Spring 2019/lab1.py', wdir='/Users/
wajgilani/Documents/BAP/BAP Spring 2019')
[4, 5, 6]

In [13]: runfile('/Users/wajgilani/Documents/BAP/BAP Spring 2019/lab1.py', wdir='/Users/
wajgilani/Documents/BAP/BAP Spring 2019')
[4, 5, 6]

In [14]: runfile('/Users/wajgilani/Documents/BAP/BAP Spring 2019/lab1.py', wdir='/Users/
wajgilani/Documents/BAP/BAP Spring 2019')
[4, 5, 6]

In [15]: runfile('/Users/wajgilani/Documents/BAP/BAP Spring 2019/lab1.py', wdir='/Users/
wajgilani/Documents/BAP/BAP Spring 2019')
[4, 5, 6]

In [16]:
```

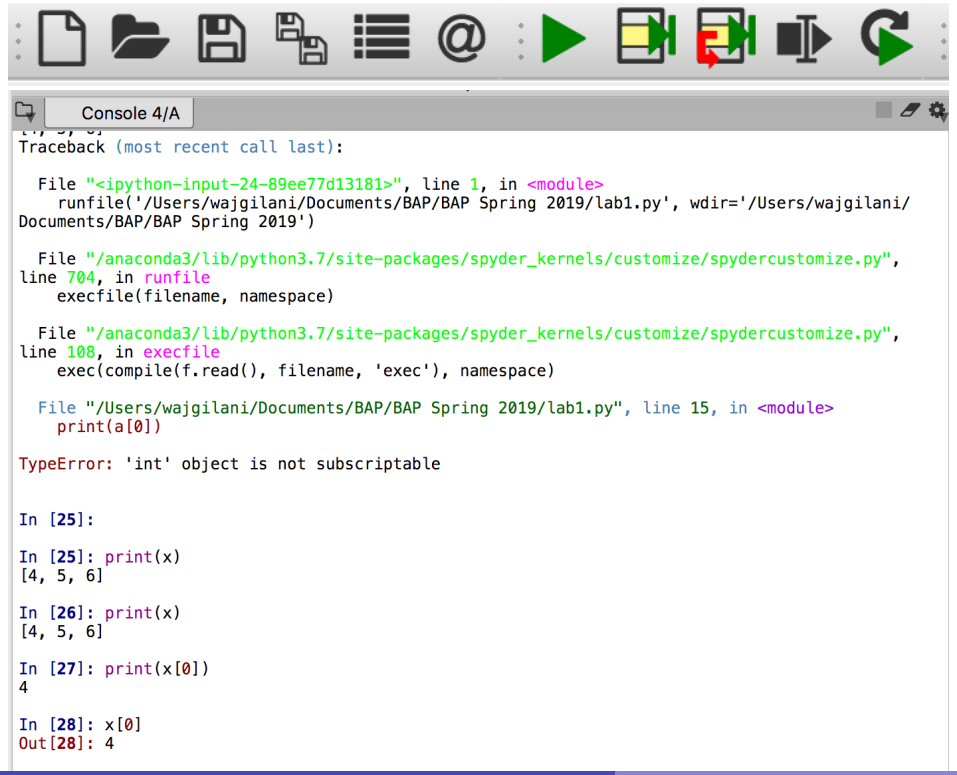
IPython console | History log

Permissions: RW End-of-lines: LF Encoding: UTF-8 Line: 15 Column: 1 Memory: 60 %

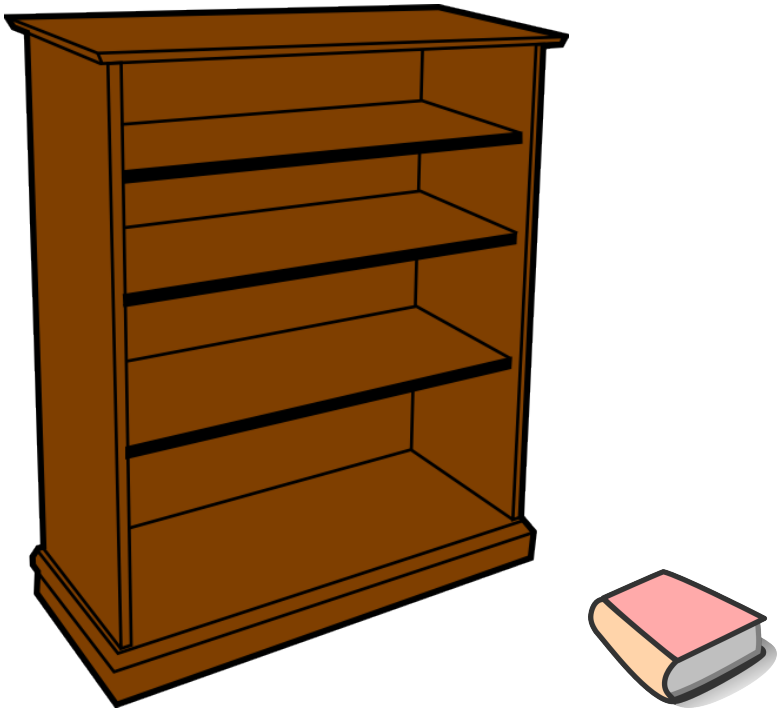
Variables and Lists

```
1  x=[4,5,6]
2  print(x)
3
4  a=5
5  print(a[0])
6  print(x)
7
```

What happens when you run line 5? line 6?



Container vs Variable



```
x=[4,5,6]
a=5
```

The book is a, the shelf is x, its a data container. Meaning the list holds data for you to use.

Spyder IDE - Cells

Editor - /Users/wajgilani/Documents/BAP/BAP Spring 2019/lab1.py

lab1.py*

```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.p
"""

import numpy as np
import pandas as pd

x=[4,5,6]
print(x)

a=5
print(a[0])
#%%
print(x[0])
```

Now just press command (control) and return (enter). Have as many cells as you want, or none at all, its up to you.

Lists

A list is a collection which is ordered and changeable. In python lists are written with square brackets.

```
x=[4,5,6]
blackpanther=['wakanda','forever','money']
avengers=['iron man',5,12]
```

Access list items by referring to the index number:

```
print(x[0])
print(blackpanther[1])
print(avenger[2])
```

Change the value of a specific item using the index number:

```
avengers[1]='captain america'
print(avenger[1])
```

You can loop through the items using a for loop:

```
for x in avengers:
  print(x)
```

Lists (Continued))

To determine if a specified item is present in a list use the in keyword:

```
blackpanther=['wakanda','forever','money']

if 'killmonger' in blackpanther:
  print('yes')

if 'wakanda' in blackpanther:
  print('yes')

if 'killmonger' in blackpanther:
  print('yes')
else:
  print('no')
```

For those of new to programming, the if/else keywords you see are conditional statements.

Lists (length,add,insert)

To determine how many items a list has, use len():

```
blackpanther=['wakanda','forever','money']

print(len(blackpanther))
```

To add an item to the end of the list, use append():

```
blackpanther=['wakanda','forever','money']
blackpanther.append('vibranium')
print(blackpanther)
```

To add an item at the specified index, use the insert() method:

```
blackpanther=['wakanda','forever','money']
blackpanther.insert(1,'vibranium')
print(blackpanther)
```

Lists (remove,pop,del)

The remove() method removes the specified item:

```
blackpanther=['wakanda','forever','money']
blackpanther.remove('forever')
print(blackpanther)
```

To pop() method removes the specified index (or last item if index is not specified):

```
blackpanther=['wakanda','forever','money']
blackpanther.pop()
print(blackpanther)
```

To del keyword removes the specified index, also it can just delete the whole list:

```
blackpanther=['wakanda','forever','money']
del blackpanther[0]
print(blackpanther)

del blackpanther
print(blackpanther)
#you will get an error
```

Lists (clear,list)

The clear() method empties the list:

```
blackpanther=['wakanda','forever','money']
blackpanther.clear()
print(blackpanther)
```

You get an empty box [].
The list() constructor to make a list:

```
blackpanther=list(('wakanda','forever','money'))
print(blackpanther)
```

Notice the double parenthesis.

Math on Lists

Lists aren't great for vector based mathematics:

```
a=[1,2,3]
b=[4,5,6]
c=a+b
print(c)
```

Enter the Numpy

The fundamental library needed for scientific computing with Python is called NumPy. This Open Source library contains:

- a powerful N-dimensional array object
- advanced array slicing methods (to select array elements)
- convenient array reshaping methods

and it even contains 3 libraries with numerical routines:

- basic linear algebra functions
- basic Fourier transforms
- sophisticated random number capabilities

NumPy can be extended with C-code for functions where performance is highly time critical. In addition, tools are provided for integrating existing Fortran code. NumPy is a hybrid of the older NumArray and Numeric packages, and is meant to replace them both.

Numpy - Creating Arrays

```
1  import numpy as np
2  #just run this once per program
3
4  #convert list to an array
5  l=[1,2,3]
6  a=np.array(l)
7  print(a)
8
9  a.shape #(3,)
10 print(a.dtype)
11
12 #create an array directly
13 b=np.array([4.0,5.0,6])
14 print(b.shape) #(3,)
15 print(b.dtype)
16
17 a[1]='cat'
```

What happens in line 17?

Numpy - Slicing

```
#convert list to an array
l=[1,2,3]
a=np.array(l)
print(a[1])
```

Answer: 2

```
print(a[0:1])
```

Answer: 1

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

Answer: [1,2]

```
print(a[0:3])
```

Answer: [1,2,3]

```
print(a[0:4])
```

Answer: [1,2,3]

```
print(a[1:1])
```

Answer: []

Numpy - Slicing (Continued)

```
#convert list to an array
l=[1,2,3]
a=np.array(l)
print(a[1:2])
```

Answer: 2

```
print(a[1:3])
```

Answer: [2,3]

```
print(a[1:4])
```

Answer: [2,3]

```
print(a[:2])
```

Answer: [1,2]

Numpy - Generation Function

```
x=np.arange(0,10,1)
print(x)
```

0	1	2	3	4	5	6	7	8	9

x

```
y=np.arange(0,10,5)
print(y)
```

0	5

y

Numpy - Generation Function (Continued)

```
z=np.linspace(0,5,5)
print(z)
```

0.	1.25	2.5	3.75	5.

z

```
w=np.arange(3, 7, 0.5)
print(w)
```

3.	3.5	4.	4.5	5.	5.5	6.	6.5

w

Numpy - Arrays Keep Their Type

```
d = np.arange(5)
d[1]=9.7
print(d)
```

0	9	2	3	4

d

```
e=d*0.4
print(e)
```

0.	3.6	0.8	1.2	1.6

e

Numpy - Arrays Keep Their Type (Continued)

```
d = np.arange(5)
d[1]=9.7
print(d)
```

0	9	2	3	4

d

The right way is to create an array that holds the data type *floats*.

```
d = np.arange(5,dtype=np.float)
d[1]=9.7
print(d)
```

0.	9.7	2.	3.	4.

d

Numpy - Math

```
a = np.arange(1,4)
b = np.arange(4,7)
c=a+b
print(c)
```

Answer: [5,7,9]

```
e = np.array([1,2,3])
f=e+5
print(f)
```

Answer: [6 7 8]

```
g = np.array([1,2,3])
h = np.array([1,2,3,4])
i=g+h
print(i)
```

Answer: Error!!