Python Tutorial

March 7, 2020

1 Installation Prerequsites

1.1 Anaconda Installation

Anaconda / Conda

- Platform that allows you to manage python versions and packages through “environments”
- Includes pip - python’s package manager that can be used to expand the capabilities of a base python environment
- Each environment can run different versions of python
- Each environment has a different set of installed packages

Install Link: https://www.anaconda.com/distribution/

- Download Anaconda Python 3.7 version
- Use default settings for installation
- If your username has a space in it, anaconda will warn that this might cause install issues

1.2 Create Conda Environment

Windows: Open “Anaconda Prompt” through Start menu

Mac: Open “Terminal” through spotlight search

Enter the following command to create a new environment named “py37” with python version 3.7

```
conda create --name py37 python=3.7
```

Activate the environment you just created

```
conda activate py37
```

Conda Cheet Sheet is a useful reference: http://bit.ly/2xknudL

1.3 Install Packages

Many features are already included in python, but many packages can be installed to expand the capabilities of python

These packages can be installed using pip - python’s package installer - or conda

1.3.1 jupyter-notebook

- a web-based interactive development environment (IDE) that allows you to interactively develop python scripts, analyze and visualize data

- A notebook is comprised of a series of cells; can either be a python code cell or a markdown text cell
- Can execute the contents of the active cell by clicking (Shift+Enter)
- Code cells are run in the order you execute them Install jupyter-notebook and add our conda environment as a python kernel (the python environment in which we'll run our code)

```
pip install jupyter
python -m ipykernel install --user --name=py37
```

1.3.2 Matplotlib

- A package for creating visualizations

```
pip install matplotlib
```

1.3.3 Pandas

- Used for working with datasets
- Load data into pandas DataFrame
- Can query large datasets

```
pip install pandas
```

1.3.4 Numpy

- Another useful tool for mathematical operations and analyzing data

```
pip install numpy
```

1.3.5 Sklearn

- Another useful tool for mathematical operations and analyzing data

```
pip install sklearn
```

1.4 Start-up jupyter

In Terminal/ Anaconda Prompt run:

```
jupyter-notebook
```

This will open the jupyter console in your web browser

1.4.1 Unix (mac terminal) Commands

Check current directory

```
pwd
```

List directory contents

```
ls
```

Change directory

```
cd < insert directory name without brackets >
```

Make new directory

```
mkdir < directory name >
```

2 Getting Started with python

Import packages that we'll be using

[1]:
```
import numpy as np
import pandas as pd
# iPython magic to allow interactive plots
%matplotlib notebook
# Import for 3D plots
from mpl_toolkits.mplot3d import Axes3D
# Plotting library
import matplotlib.pyplot as plt
# Colormap Library
from matplotlib import cm
import sklearn
```

2.1 Variables

2.1.1 Integers and Floating Point Numbers

[2]:
```
# Define two integers
num1 = 4
num2 = 3
total = num1 + num2
product = num1 * num2
ratio = num1 / num2

# int + float = float
float1 = .125
f1 = num1 + float1
```

[3]:
```
f1
```

[3]:
```
4.125
```

[4]:
```
num1
```

[4]:
```
4
```

[5]:
```
num1 += 2
```

[6]:
```
num1
```

[6]:
```
6
```

Printing

[7]:
```
print("sum:",total, " product:",product, "ratio:", ratio,"\n")
print("sum: {}\nproduct: {}\nratio: {:.3f}".format(total, product, ratio))
```

```
sum: 7  product: 12 ratio: 1.3333333333333333

sum: 7
product: 12
ratio: 1.333
```

2.1.2 Bool

[8]:
```
print("num1: ",num1, "\nnum1 > 3: ", num1 > 3)
```

```
num1:  6
num1 > 3:  True
```

2.1.3 Strings

https://www.w3schools.com/python/python_ref_string.asp

[9]:
```
first_half = "the quick brown fox"
second_half = "jumped over the lazy dog"
full_sentence = first_half + " " + second_half
```

[10]:
```
full_sentence
```

[10]:
```
'the quick brown fox jumped over the lazy dog'
```

[11]:
```
first_half.replace("fox","bear")
```

[11]:
```
'the quick brown bear'
```

Check the docs to see if the function returns the result or sets the value in-place

[12]:
```
first_half
```

[12]:
```
'the quick brown fox'
```

[13]:
```
first_half[0]
```

[13]:
```
't'
```

[14]:
```
first_half[4:9]
```

[14]:
```
'quick'
```

[15]:
```
second_half[ second_half.find("the") + 4 : second_half.find(" dog") ]
```

```
[15]: 'lazy'
```

2.1.4 Lists, Tuples

https://www.w3schools.com/python/python_ref_list.asp

```
[16]: list_1 = [1,2,3,4,5]
      tup_1 = (6,7,8)
      list_2 = list(tup_1)
```

```
[17]: list_1 + list_2
```

```
[17]: [1, 2, 3, 4, 5, 6, 7, 8]
```

This is an example of an in-place function

```
[18]: list_1.reverse()
```

```
[19]: list_1
```

```
[19]: [5, 4, 3, 2, 1]
```

You can index a list and get a subsequence

```
[20]: list_1[3]
```

```
[20]: 2
```

```
[21]: list_1[:2]
```

```
[21]: [5, 4]
```

```
[22]: list_1[2:-1]
```

```
[22]: [3, 2]
```

Lists are mutable

```
[23]: list_1
```

```
[23]: [5, 4, 3, 2, 1]
```

```
[24]: list_1[2] = 55
```

```
[25]: list_1
```

```
[25]: [5, 4, 55, 2, 1]
```

Tuples are not mutable

[26]:
```
# try running this code block and if an error gets raised do that
try:
    tup_1[1] = 5
except:
    print("TUPLES ARE NOT MUTABLE")
```

```
TUPLES ARE NOT MUTABLE
```

2.1.5 Sets

[27]:
```
fruits = set(["apple", "orange", "banana"])
vegetables = set(["broccoli", "carrot", "lettuce", "olives"])
food_I_dont_like = set(["olives", "anchovy"])
```

[28]:
```
fruits
```

[28]:
```
{'apple', 'banana', 'orange'}
```

[29]:
```
fruits.union(vegetables)
```

[29]:
```
{'apple', 'banana', 'broccoli', 'carrot', 'lettuce', 'olives', 'orange'}
```

[30]:
```
vegetables.difference(food_I_dont_like)
```

[30]:
```
{'broccoli', 'carrot', 'lettuce'}
```

2.1.6 Dictionaries

[31]:
```
me = {"first name": "Dan", "last name": "Zeiberg", "age": 24}
pedja = {"first name": "Predrag", "last name": "Radivojac"}
```

[32]:
```
people = [me, pedja]
```

[33]:
```
people
```

[33]:
```
[{'first name': 'Dan', 'last name': 'Zeiberg', 'age': 24},
 {'first name': 'Predrag', 'last name': 'Radivojac'}]
```

2.1.7 Nump Arrays, Matrices

[34]:
```
arr_1 = np.array([6,2,6,8,1,6,22,6,8,999])
arr_2 = np.random.randint(0,25,size=10)
```

[35]:
```
arr_1
```

[35]:
```
array([  6,   2,   6,   8,   1,   6,  22,   6,   8, 999])
```

You can do arithmetic on arrays

```
[36]: arr_2 / arr_1
```

```
[36]: array([3.5       , 6.5       , 2.83333333, 2.375     , 1.        ,
              0.        , 0.95454545, 3.83333333, 0.875     , 0.01101101])
```

```
[37]: matrix_0 = np.array([[1,2,3],[4,5,6],[7,8,9]])
```

```
[38]: matrix_0
```

```
[38]: array([[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9]])
```

Indexing Matrices

```
[39]: matrix_0[1,2]
```

```
[39]: 6
```

Numpy allows you to draw from statistical distributions

```
[40]: matrix_1 = np.random.normal(loc=0, scale=1, size=[5,4])
```

```
[41]: matrix_1
```

```
[41]: array([[ 1.65423992,  1.71262782, -1.13080366,  0.39229579],
              [-1.28035338, -0.27232278, -2.24187914,  0.37101365],
              [ 0.84828532,  0.64919246, -1.16023924,  0.14783276],
              [ 0.03972101, -0.73081974, -0.63169612, -0.9884438 ],
              [ 0.14377593,  1.58346939,  1.01394705,  0.39110502]])
```

3 Pandas DataFrames

Load DataFrame from file

boston housing dataset: https://www.kaggle.com/puxama/bostoncsv

```
[42]: housing = pd.read_csv("../data/BostonHousing.csv")
```

```
[43]: housing
```

```
[43]:         crim    zn  indus  chas    nox     rm   age     dis  rad  tax  \
      0    0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296
      1    0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242
      2    0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242
      3    0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222
```

```
4    0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222
..       …     …     …     …     …     …     …       …   …   …
501  0.06263   0.0  11.93     0  0.573  6.593  69.1  2.4786    1  273
502  0.04527   0.0  11.93     0  0.573  6.120  76.7  2.2875    1  273
503  0.06076   0.0  11.93     0  0.573  6.976  91.0  2.1675    1  273
504  0.10959   0.0  11.93     0  0.573  6.794  89.3  2.3889    1  273
505  0.04741   0.0  11.93     0  0.573  6.030  80.8  2.5050    1  273

     ptratio       b  lstat  medv
0       15.3  396.90   4.98  24.0
1       17.8  396.90   9.14  21.6
2       17.8  392.83   4.03  34.7
3       18.7  394.63   2.94  33.4
4       18.7  396.90   5.33  36.2
..       …       …     …     …
501     21.0  391.99   9.67  22.4
502     21.0  396.90   9.08  20.6
503     21.0  396.90   5.64  23.9
504     21.0  393.45   6.48  22.0
505     21.0  396.90   7.88  11.9

[506 rows x 14 columns]
```

```
[44]: people_df = pd.DataFrame(people)
```

```
[45]: people_df
```

```
[45]:   first name  last name   age
      0        Dan    Zeiberg  24.0
      1    Predrag  Radivojac   NaN
```

```
[46]: # each element in the dataframe is uniquely identified by it's index value
      # Lets set the index for the people DataFrame to "last name"
      people_df = people_df.set_index("last name")
```

```
[47]: people_df
```

```
[47]:           first name   age
      last name
      Zeiberg          Dan  24.0
      Radivojac    Predrag   NaN
```

Get the age of the person represented by the first row

Can index a dataframe by row number

```
[48]: housing.iloc[0]["age"]
```

[48]:
```
65.2
```

Or you can index by the index (key) and column name

Get Dan's age

[49]:
```
people_df.loc["Zeiberg","age"]
```

[49]:
```
24.0
```

Extract Column of DataFrame, convert to numpy array, limit to first 10 values

[50]:
```
housing["age"].values[:10]
```

[50]:
```
array([ 65.2,  78.9,  61.1,  45.8,  54.2,  58.7,  66.6,  96.1, 100. ,
        85.9])
```

You can query a DataFrame

[51]:
```
housing[housing["age"] > 33]
```

[51]:
```
          crim    zn  indus  chas    nox     rm   age     dis  rad  tax  \
0      0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296
1      0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242
2      0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242
3      0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222
4      0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222
..         …     …     …    …     …      …     …     …   …   …
501    0.06263   0.0  11.93     0  0.573  6.593  69.1  2.4786    1  273
502    0.04527   0.0  11.93     0  0.573  6.120  76.7  2.2875    1  273
503    0.06076   0.0  11.93     0  0.573  6.976  91.0  2.1675    1  273
504    0.10959   0.0  11.93     0  0.573  6.794  89.3  2.3889    1  273
505    0.04741   0.0  11.93     0  0.573  6.030  80.8  2.5050    1  273

     ptratio       b  lstat  medv
0       15.3  396.90   4.98  24.0
1       17.8  396.90   9.14  21.6
2       17.8  392.83   4.03  34.7
3       18.7  394.63   2.94  33.4
4       18.7  396.90   5.33  36.2
..         …       …     …     …
501     21.0  391.99   9.67  22.4
502     21.0  396.90   9.08  20.6
503     21.0  396.90   5.64  23.9
504     21.0  393.45   6.48  22.0
505     21.0  396.90   7.88  11.9

[425 rows x 14 columns]
```

```
[52]: housing[(housing["age"] > 33) & (housing["tax"] <= 350)]
```

```
[52]:        crim    zn  indus  chas    nox     rm   age     dis  rad  tax  \
      0    0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296
      1    0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242
      2    0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242
      3    0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222
      4    0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222
      ..       …    …     …    …      …      …    …       …   …    …
      501  0.06263   0.0  11.93     0  0.573  6.593  69.1  2.4786    1  273
      502  0.04527   0.0  11.93     0  0.573  6.120  76.7  2.2875    1  273
      503  0.06076   0.0  11.93     0  0.573  6.976  91.0  2.1675    1  273
      504  0.10959   0.0  11.93     0  0.573  6.794  89.3  2.3889    1  273
      505  0.04741   0.0  11.93     0  0.573  6.030  80.8  2.5050    1  273

           ptratio       b  lstat  medv
      0       15.3  396.90   4.98  24.0
      1       17.8  396.90   9.14  21.6
      2       17.8  392.83   4.03  34.7
      3       18.7  394.63   2.94  33.4
      4       18.7  396.90   5.33  36.2
      ..       …       …    …     …
      501     21.0  391.99   9.67  22.4
      502     21.0  396.90   9.08  20.6
      503     21.0  396.90   5.64  23.9
      504     21.0  393.45   6.48  22.0
      505     21.0  396.90   7.88  11.9

      [202 rows x 14 columns]
```

3.1 Joining DataFrames

```
[53]: people_df
```

```
[53]:           first name   age
      last name
      Zeiberg           Dan  24.0
      Radivojac     Predrag   NaN
```

```
[54]: publications_df = pd.DataFrame([
          {"title": "Fast Nonparametric Estimation of Class Proportions in the␣
       ↪Positive-Unlabeled Classification Setting",
           "year": 2020,
           "first author": "Zeiberg"},
          {"title": "Prediction of boundaries between intrinsically ordered and␣
       ↪disordered protein regions",
           "year": 2003,
    "first author": "Radivojac"}])
```

[55]:
```
publications_df
```

[55]:
```
                                               title  year first author
0  Fast Nonparametric Estimation of Class Proport…  2020      Zeiberg
1  Prediction of boundaries between intrinsically…  2003    Radivojac
```

[56]:
```
people_and_publication = people_df.merge(right=publications_df, how="left",␣
 ↪left_on="last name", right_on="first author")
```

[57]:
```
people_and_publication
```

[57]:
```
  first name   age                                              title  year  \
0        Dan  24.0  Fast Nonparametric Estimation of Class Proport…  2020
1    Predrag   NaN  Prediction of boundaries between intrinsically…  2003

  first author
0      Zeiberg
1    Radivojac
```

[58]:
```
# remove the first author column
people_and_publication = people_and_publication.drop("first author",axis=1)
```

[59]:
```
people_and_publication = people_and_publication.rename(columns={"title":
 ↪"publication name"})
```

[60]:
```
people_and_publication
```

[60]:
```
  first name   age                                   publication name  year
0        Dan  24.0  Fast Nonparametric Estimation of Class Proport…  2020
1    Predrag   NaN  Prediction of boundaries between intrinsically…  2003
```

4 Conditionals and Loops

[61]:
```
randnum = np.random.choice([1,2,3,4])
if randnum == 1:
    print("chose first value")
elif randnum == 2:
    print("chose second value")
elif randnum == 3:
    print("chose third value")
else:
    print("choes fourth value")
```

```
chose second value
```

You can repeat a block of code using for loops and while loops

For loops

iterate over a list of values, this can be loop indices or other data

[62]:

```
for i in range(10):
    print(i)
```

```
0
1
2
3
4
5
6
7
8
9
```

[63]:

```
for person in people:
    print(person["first name"])
```

```
Dan
Predrag
```

Loop over matrix values

[64]:

```
# Loop over each row
for r in range(matrix_1.shape[0]):
    # Loop over each column
    for c in range(matrix_1.shape[1]):
        print(matrix_1[r,c])
    print()
```

```
1.65423992031951
1.712627822242182
-1.1308036574487206
0.3922957878874731

-1.2803533825104598
-0.2723227779931885
-2.2418791430036276
0.3710136540528735

0.8482853165237828
0.6491924645783096
-1.1602392378006166
```

```
0.14783275760652517

0.039721006981576455
-0.7308197379929903
-0.6316961216431078
-0.988443795217655

0.14377593402483918
1.5834693931259611
1.0139470508658355
0.39110501887619814
```

While Loops

Continue executing a block of code while the specified condition is false

[65]:

```
values = [1,7,1,6,888,221,5]
idx = 0
while values[idx] < 50:
    idx += 1
print("first big number is ",values[idx])
```

```
first big number is  888
```

5 Functions

[66]:

```
def add(a,b):
    return a+b
```

[67]:

```
add(4,5)
```

[67]:

```
9
```

[68]:

```
def factorial(x):
    if x > 0:
        return x * factorial(x-1)
    elif x == 0:
        return 1
    else:
        raise Exception("Input must be non-negative")
```

[69]:

```
factorial(5)
```

[69]:

```
120
```

6 Classes

[70]:

```python
class Person:
    def __init__(self, name, hair_color, eye_color):
        self.name = name
        self.hair_color = hair_color
        self.eye_color = eye_color

    def introduce(self):
        print("Hi, my name is {}".format(self.name))

    def converse(self, other_person):
        print("Hi {}, how are you today?".format(other_person.name))
```

[71]:

```python
dan = Person("Dan", "brown", "green")
emily = Person("Emily", "blond", "blue")
dan.introduce()
emily.converse(dan)
```

```
Hi, my name is Dan
Hi Dan, how are you today?
```

7 Plotting

[72]:

```python
ph = pd.read_csv("../data/pH-example.txt")
```

[73]:

```python
ph
```

[73]:

```
     time      v
0    0.00  7.000
1    0.01  6.999
2    0.02  6.998
3    0.03  6.997
4    0.04  6.996
..    …      …
96   0.96  6.991
97   0.97  6.990
98   0.98  6.989
99   0.99  6.988
100  1.00  6.987

[101 rows x 2 columns]
```

[74]:

```python
fig,ax = plt.subplots(1,1)
ax.plot(ph["time"], ph["v"])
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
[74]: [<matplotlib.lines.Line2D at 0xa218ec978>]
```

```
[75]: fig, ax = plt.subplots(1,1)
      hist = ax.hist(housing["age"],bins=30)
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

3D-Plot

```
[76]: from scipy.stats import multivariate_normal as mvn
```

```
[77]: grid = np.zeros((20,20))
      for i in range(grid.shape[0]):
          for j in range(grid.shape[1]):
              grid[i,j] = mvn.pdf([i,j],[9,9],[[5,0],[0,5]])
```

```
[78]: fig = plt.figure()
      ax = fig.gca(projection='3d')
      X,Y = np.meshgrid(list(range(grid.shape[0])), list(range(grid.shape[1])))
      surface = ax.plot_surface(X,Y,grid, cmap=cm.coolwarm)
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

8 Example

```
[79]: ph_arr = ph.values
```

```
[80]: delta = ph_arr[1:,1] - ph_arr[:-1,1]
```

```
[81]: fig, axes = plt.subplots(2,1, figsize=(6,5))
      # adjust vertical spacing between subplots
      plt.subplots_adjust(hspace=0.3)
      axes[0].plot(ph_arr[:,0],ph_arr[:,1],'--o', linewidth=2, markersize=2,␣
       ↪label="pH")
      axes[0].set_ylabel("pH")
      axes[1].plot(ph_arr[1:,0],delta, color="red", label="$\Delta$ pH")
```

```
axes[1].set_ylabel("$\Delta$ pH")
xlab = axes[1].set_xlabel("Time, h")
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

[ ]:

```

```