Introduction to Python

Table of Contents

Python

What is Python

Python is a programming language that lets you work quickly and integrate systems more effectively. It's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. Python.org (https://www.python.org/):

Why should you know Python?

In academic and research we can use Python and its lots of labraries to do wide range of things from basic calculation to complex statistical methods including machine learning. Since it is open-source, it is free to use.

Where can I get it?

There are many ways to install Python and its libraries. However, we are going to guide you in a pretty stable way, Anaconda.

Anaconda Cloud

Anaconda is a platform for Python that bundles most of the essential packages required in research. You can download it from here (https://www.anaconda.com/download/). **Note that we will use Python version 3.x in this course.**

> Anaconda.org (https://anaconda.org/):
>
> Anaconda Cloud is where data scientists share their work. You can search and download popular Python and R packages and notebooks to jumpstart your data science work.

Package Managers

We do not need to install any additional packages in this class. However, when you need it in the future recall these two names: `pip` and `conda` . For more information you can check this (https://conda.io/docs/user-guide/tasks/manage-pkgs.html#installing-packages) out.

Jupyter Notebook

What is Jupyter Notebook?

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more. Here are a few examples of Jupyter Notebooks projects:

- Machine Learning (https://nbviewer.jupyter.org/github/rhiever/Data-Analysis-and-Machine-Learning-Projects/blob/master/example-data-science-notebook/Example%2520Machine%2520Learning%2520Notebook.ipynb)
- Signal Processing (https://www.gw-openscience.org/s/events/GW150914/GW150914_tutorial.html)

How to use it?

This section will list the most used command shortcuts. For full version of docs you can visit The Jupyter Notebook documentation (http://jupyter-notebook.readthedocs.io/en/latest/notebook.html).

- `arrows` : *move cursor*
- `enter` : *enter edit mode*
- `esc` : *quit edit mode*
- `y` : *change cell to code mode*
- `m` : *change cell to markdown mode*
- `ctrl + enter` : *run selected cell*
- `shift + enter` : *run selected cell and move cursor to the next cell*
- `h` : *open help panel, list of shortcuts*

Let's code

Introduction to Python

Here we will try to write some Python built-in functions to get how it works. This is brought from official Python website. For more information follow the official tutorial (https://docs.python.org/3/tutorial/).

In [ ]:
```
# print function
print("Hello, I'm Python!")
```

Numbers

In [ ]:
```
# arithmatic
print(1+3) # plus
print(2-1) # minus
print(3*6) # multiply
print(2**3) # power
print(17/3) # divide
print(17//3) # floor divide
print(10%3) #modulo
print(abs(-5)) #absolute
print(max(6,5)) #max value
```

In [ ]:
```
a = 3
print("a+2=",a+2)
```

In [ ]:
```
x = 1
y = 2
z = x+y
print(z)
print(max(x,y))
```

Strings

In [ ]:
```
phrase = "I am your father"
phrase1 = "Nooo!"
print(phrase.upper()+ phrase1.lower())
print(len(phrase))
print(phrase[5])
print(phrase[-1]) # last character
print(phrase[-2]) # second-last character
print(phrase[0:4])# characters from position 0 (included) to 4 (excluded)
print(phrase[10:]) # characters from position 10 (included) to the end
print(len(phrase)) # returns the length of a string
```

Lists

In [ ]:
```
fruits = ['Banana', 'Apple', 'Lime','Watermelon','Papaya','Banana']
print(fruits)
print(fruits[0]) # indexing returns the item
print(fruits[-1]) # last item
print(fruits[1:4]) # return new list contain items index 1 (included) to 4 (ex
cluded)
print(fruits[:3])
```

In [ ]:
```
fruits = ['Banana', 'Apple', 'Lime','Watermelon','Papaya','Banana']
fruits.pop() # remove last items of the list
print("pop:", fruits)
fruits.append('Pineapple') # add new items at the end of the list
print("append:", fruits)
fruits.insert(1, 'Mango') # insert item into the list
print("insert:", fruits)
fruits.remove('Apple')
print("remove:", fruits)
```

In [ ]:
```
cubes = [1, 8, 27, 65, 125]
cubes[3] = 64 # replace item at index 3
cubes
```

In [ ]:
```
letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
letters[2:5] = ['C', 'D', 'E'] # replace some values
letters
```

In [ ]:
```
# concatenated (glued together) with the + operator
fruits = ['Banana', 'Apple', 'Lime','Watermelon','Papaya','Banana']
vegetables = ['Tomato','Broccoli']
print(fruits+vegetables)
```

While statements

In [ ]:
```
# Fibonacci series:
# the sum of two elements defines the next
a, b = 0, 1
while a < 10:
    print(a, end = ' ')
    a, b = b, a+b
```

If statements

In [ ]:
```
x = -1

if x < 0:
    x = 0
    print('Negative changed to zero')
elif x == 0:
    print('Zero')
elif x == 1:
    print('Single')
else:
    print('More')
```

For statements

In [ ]:
```
words = ['cat', 'window', 'defenestrate']
for w in words:
    print(w, len(w))

# Loop over a slice copy of the entire list.
# if length of being looped word longer than 6
# insert this word to index 0 of the list
for w in words[:]:
    if len(w) > 6:
        words.insert(0, w)
print("inserted list:", words)
```

In [ ]:
```
for i in range(5):
    print(i)

a = ['Mary', 'had', 'a', 'little', 'lamb']
for i in range(len(a)):
    print(i, a[i])
```

In [ ]:
```
for n in range(2, 10):
    for x in range(2, n):
        if n % x == 0:
            print(n, 'equals', x, '*', n//x)
            break
    else:
        print(n, 'is a prime number')
```

In [ ]:
```
for num in range(2, 10):
    if num % 2 == 0:
        print("Found an even number", num)
        continue
    print("Found a number", num)
```

Defining functions

In [ ]:
```
# Python 3: Fibonacci series up to n
def fib(n):
    a, b = 0, 1
    while a < n:
        print(a, end=' ')
        a, b = b, a+b
    print()

fib(1000)
```

In [ ]:
```
def ask_ok(prompt, retries=4, reminder='Please try again!'):
    while True:
        ok = input(prompt)
        if ok in ('y', 'ye', 'yes'):
            return 'YES'
        if ok in ('n', 'no', 'nop', 'nope'):
            return 'NO'
        retries = retries - 1
        if retries < 0:
            return 'invalid user response'
        print(reminder)

ask_ok('Do you really want to quit?: ') # giving only the mandatory argument
```

In [ ]:
```
ask_ok('OK to overwrite the file?: ', 2, 'Come on, only yes or no!') # giving
 all arguments
```

Mathematics and Statistics packages

In research we will inevitably have to do some statistics. It is good to know how we can do mathematics and statistics operation with Python. As we are using Anaconda which already installed most required packages, so you do not need to install additional package here.

- The packages we will learn today are `numpy` , `pandas` and `matplotlib` .

NumPy

NumPy is a python library to help dealing with mathematics especially array. As it can do so many things, the examples listed below may not be able to cover all. For more information, you can visit [NumPy's official tutorial (https://docs.scipy.org/doc/numpy-1.15.0/user/quickstart.html)](https://docs.scipy.org/doc/numpy-1.15.0/user/quickstart.html).

In [ ]:
```
import numpy as np

# create array of range [0, 15)
a = np.arange(15)
print("basic array:", a)

# reshape array
print("\nreshaped:", a.reshape([3, 5]))

# or do it at once
print("\nall at once:", np.arange(15).reshape([3, 5]))
```

In [ ]:
```
# create array from list
mylist = [1,2,3,4]
print("array from list:", np.array(mylist))

# create 2d array
print("\n2d array:", np.array([mylist, mylist]))
print("\ncheck dimension:", np.array([mylist, mylist]).shape, "<- 2 rows 4 col
umns")
```

In [ ]:
```
# create array of ones
print("array of ones:", np.ones([3,3]))

# create array of zeros
print("\narray of zeros:", np.zeros([5,5]))

# some constants
print("\npi:", np.pi)
print("e:", np.e)

# basic arithmetics
print("\nsum, 1+2+3+4+5:", np.sum([1,2,3,4,5]))
print("power, 2^8:", np.power(2, 8))
```

Pandas

pandas is an easy-to-use python library for data structures and data analysis. pandas implements a number of statistical functions that can be used in research. You can learn more about pandas in these pandas's official quick start tutorials:

- 10 mins tutorials (https://pandas.pydata.org/pandas-docs/stable/10min.html#min)
- pandas tutorials (https://pandas.pydata.org/pandas-docs/stable/tutorials.html)

In [ ]:
```
import pandas as pd
#create dataframe
df_for_scatter = pd.DataFrame(np.random.rand(5,2), columns=['col1','col2'])
df_for_bar = pd.DataFrame(np.random.rand(5, 1), columns=['col1'])
df_for_line = pd.DataFrame(np.random.rand(100, 1), columns=['col1'])

print("scatter dataframe:\n",df_for_scatter,"\n")
print("bar dataframe:\n",df_for_bar,"\n")
print("line dataframe:\n",df_for_line,"\n")
```

In [ ]:
```
# we can also see dataframe rendered as HTML
df_for_scatter
```

In [ ]:
```
# create new dataframe by concatenate existing ones
new_df_concat_down = pd.concat([df_for_scatter[3:], df_for_scatter[3:]]) # use
[3:] to shorten
# notice index start from 3 instead of 0
new_df_concat_down
```

In [ ]:
```
new_df_concat_side = pd.concat([df_for_scatter, df_for_scatter], axis=1)
new_df_concat_side
```

Matplotlib

Matplotlib is a popular Python 2D plotting library. You can do most of data visualization using Matplotlib. pandas plotting methods also use Matplotlib as a based tool.

- Pyplot tutorial by matplotlib.org (https://matplotlib.org/tutorials/introductory/pyplot.html)
- Matplotlib tutorial by Nicolas P. Rougier (https://www.labri.fr/perso/nrougier/teaching/matplotlib/)

In [ ]:
```
import matplotlib.pyplot as plt
%matplotlib inline

plt.plot(df_for_line)
plt.title('Line plot')
plt.figure()

plt.scatter(df_for_scatter.col1, df_for_scatter.col2)
plt.title('Scatter plot')
plt.figure()

plt.bar(df_for_bar.index, df_for_bar.col1)
plt.title('Bar plot')
plt.figure()

from skimage import data

plt.imshow(data.chelsea())
```

Simple Linear Regression : Basic of Machine Learning and Statistics

y = ax + b

A linear function has one independent variable and one dependent variable. The independent variable is x and the dependent variable is y.

- a is the constant term or the y intercept. It is the value of the dependent variable when x = 0.
- b is the coefficient of the independent variable. It is also known as the slope and gives the rate of change of the dependent variable.

“The scenario is you are a HR officer, you got a candidate with 5 years of experience. Then what is the best salary you should offer to him?”

source: towardsdatascience (https://towardsdatascience.com/machine-learning-simple-linear-regression-with-python-f04ecfdadc13)

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy import stats
from sklearn.metrics import r2_score
import seaborn as sns
color = sns.color_palette()
sns.set()

sns.set(color_codes=True)
# Importing the dataset
df = pd.read_csv('./datasets/salary_data.csv')
x = df['YearsExperience']
y = df['Salary']*0.45

df
```

In [ ]:
```
plt.xlabel('YearsExperience (yr)')
plt.ylabel('Salary (Baht)')
plt.title('Salary Data on Scatter Plot')
plt.scatter(x,y)
plt.show()
```

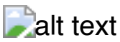
In [ ]:
```
stdy = y.std()
meany = y.mean()
plt.hist(y)
plt.xlabel('Salary (Baht)')
plt.ylabel('Quantity')
plt.title('Salary Distribution Histogram')
print ("The std of y is: " + str(stdy) + " The mean of y is: " + str(meany))
```

In [ ]:
```
stdx = x.std()
meanx = x.mean()
plt.hist(x)
plt.xlabel('YearsExperience (yr)')
plt.ylabel('Quantity')
plt.title('YearsExperience Distribution Histogram')
print ("The std of x is: " + str(stdx) + " The mean of x is: " + str(meanx))
```

Parametric and Nonparametric Statistics

Parametric tests

assume underlying statistical distributions in the data. Therefore, several conditions of validity must be met so that the result of a parametric test is reliable. For example, Student's t-test for two independent samples is reliable only if each sample follows a normal distribution and if sample variances are homogeneous.
The advantage of using a parametric test instead of a nonparametric equivalent is that the former will have more statistical power than the latter. In other words, a parametric test is more able to lead to a rejection of H0. Most of the time, the p-value associated to a parametric test will be lower than the p-value associated to a nonparametric equivalent that is run on the same data.

 alt text

Nonparametric tests

do not rely on any distribution. They can thus be applied even if parametric conditions of validity are not met. Nonparametric tests are more robust than parametric tests. In other words, they are valid in a broader range of situations (fewer conditions of validity).

Machine Learning

- **numpy.poly1d** : one-dimensional polynomial class
- **numpy.polyfit** : Least squares polynomial fit

In [ ]:
```
p = np.poly1d(np.polyfit(x,y,1))
p
```

In [ ]:
```
plt.xlabel('YearsExperience (yr)')
plt.ylabel('Salary (Baht)')
plt.title('Salary Data on Scatter Plot - Linear Regression')

xp = np.linspace(0, 11, 100)
plt.plot(xp,p(xp),c = 'r')
plt.scatter(x,y)
plt.show()
```

In [ ]:
```
r2_train = r2_score(y, p(x))
print('The rsquared value is: ' + str(r2_train))
```

- **scipy.stats.linregress** : Calculate a linear least-squares regression for two sets of measurements.

In [ ]:
```
slope, intercept, r_value, p_value, std_err = stats.linregress(x,y)
intercept,slope
```

In [ ]:
```
def linefitline(b):
    return intercept + slope * b

line = linefitline(x)
plt.scatter(x,y)
plt.xlabel('YearsExperience (yr)')
plt.ylabel('Salary (Baht)')
plt.title('Salary Data on Scatter Plot - Linear Regression')
plt.plot(x,line, c = 'g')
plt.show()
```

In [ ]:
```
r2_lin = r_value * r_value
print('The rsquared value is: ' + str(r2_lin))
```

In [ ]:
```
shuffled = df.sample(frac=1).reset_index(drop=True)
shuffx = shuffled['YearsExperience']
shuffy = shuffled['Salary']
trainX = shuffx[:20]
testX = shuffx[20:]
trainY = shuffy[:20]
testY = shuffy[20:]
plt.scatter(trainX, trainY)
plt.xlabel('YearsExperience (yr)')
plt.ylabel('Salary (Baht)')
plt.title('Training Data - Salary Data on Scatter Plot')
plt.show()
plt.scatter(testX, testY)
plt.xlabel('YearsExperience (yr)')
plt.ylabel('Salary (Baht)')
plt.title('Testing Data - Salary Data on Scatter Plot')

plt.show()
```

In [ ]:
```
p1 = np.poly1d(np.polyfit(trainX, trainY, 1))
xp = np.linspace(0, 10, 100)
axes = plt.axes()
axes.set_xlim([0,10])
plt.scatter(trainX, trainY)
plt.xlabel('Age (yr)')
plt.ylabel('Length (cm)')
plt.title('Training Linear Regression')
plt.plot(xp, p1(xp), c='r')
plt.show()
```

In [ ]:
```
xp = np.linspace(0, 10, 100)
axes = plt.axes()
plt.scatter(testX, testY)
plt.xlabel('Age (yr)')
plt.ylabel('Length (cm)')
plt.title('Testing Linear Regression')
plt.plot(xp, p1(xp), c='r')
plt.show()
```

In [ ]:
```
r2_train = r2_score(trainY, p1(trainX))
print('The rsquared value is: ' + str(r2_train))
```

In [ ]:
```
r2_test = r2_score(testY, p1(testX))
print('The rsquared value is: ' + str(r2_test))
```

FAQs

- **How should I search when I am stuck?**
  - You can use google with specific keywords. For example, if you are doing some table processing using `pandas` , you should include `pandas` in keyword instead of `python` .
- **What sources from search results should I set higher priority to check out?**
  - This can be various, but first top focus should include **StackOverflow**, **GitHub**, and official sites of Python or packages in question.

Mini challenge

- Click here to open mini challenge notebook. (MiniChallenge.ipynb)

Assignment

- Click here to open assignment notebook. (Assignment.ipynb)

Learning Materials

- Official Python tutorial (https://docs.python.org/3/tutorial/)
- Chula's Python 101 (https://www.cp.eng.chula.ac.th/books/python101/)