Python Workshop

Python...

- is an interpreted high-level programming language for general purpose programming
- was created by Guido van Rossum and first released in 1991
- has a design philosophy that emphasizes in code readability, easy use and a syntax which allows programmers use less lines of code than *Java* or *C++* to write a program
- has an expansive library of open source data analysis tools,web frameworks, and testing instruments which make its ecosystem one of the largest out of any programming community.

IEEE ranked Python as the #1 programming language in 2018 after ranking as the #1 language in 2017 and top 3 languages in 2016

Why to use Python in Machine Learning?

- Prebuilt Libraries:
  Python has a lot of libraries for every need of your ML project. Few names include Numpy for scientific computation, Scipy for advanced computing and Pandas for data science, PyBrain for machine learning (Neural Networks, Reinforcement learning)

- Support:
  Python is completely open source with a great community of active coders willing to help programmers in every stage of their developing cycle.

Variables

- counter = 100 # An integer assignment
- miles = 1000.0 # A floating point
- name = "John" # A string
- status = True # Boolean

Numbers

The interpreter acts as a simple calculator:

```
>>> (50 - 5*6) / 4
5.0
>>> 17 // 3 # floor division
5
>>> 17 % 3 # returns the remainder of division
2
>>> 5 ** 2  # 5 squared
25
```

Strings

```
>>> 'spam eggs‘ or “spam eggs”
spam eggs
>>> "\"Yes,\" they said.“
‘"Yes," they said.’
>>> ‘Py’ ‘thon’ or ‘Py’ + ‘thon’
‘Python’
>>> word = 'Python’    #a string is a list of characters
>>> word[5]  #character in position 5
‘n’
>>> word[-1]  #last character/ reverse count
‘n’
>>> word[1:3] #characters from position 1(included) to 3 (excluded)
‘yt’
```

Lists

- The most versatile variable is the *list*, which can be written as a list of comma-separated values (items) between square brackets.

- Lists might contain items of different types, but usually the items all have the same type.

Lists

```
>>> squares = [1, 4, 9, 16, 25]
>>> squares[-3:] #slicing returns a new list
[9, 16, 25]    #from index -3 or 2 to the end of the list
>>> squares + [36, 49, 64, 81, 100] #adding a list creates a new list
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
>>> squares.append(121) # add the square of 11 to the end of the list
>>> squares
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100,121]
>>> len(squares) # returns the length of the list
11
```

Nested Lists

```
>>> a = ['a', 'b', 'c']  #a list of strings
>>> n = [1, 2, 3]        #a list of integers
>>> x = [a, n]           #a list of lists
>>> x
[['a', 'b', 'c'], [1, 2, 3]]
>>> x[0]      #returns the 1st list
['a', 'b', 'c']
>>> x[0][1] #returns the 2nd value of the 1st list
'b'
```

Methods Used in Lists

- list.append(*x*)
- list.extend(*iterable*)
- list.insert(*i*, *x*)
- list.remove(*x*)
- list.pop([*i*])
- list.clear()
- list.index(*x*[, *start*[, *end*]])
- list.count(*x*)
- list.sort(*key=None*, *reverse=False*)
- list.reverse()
- list.copy()

Logical Operators

and - True if both operands are true

or - True if either of the operands is true

not - True if the operand is false

Bitwise:

& - Bitwise AND

| - Bitwise OR

~ - Bitwise NOT

If…elif

```
>>> x = 30
>>> if x < 0:                    #condition is false
        x = 0
        print('Negative changed to zero')
    elif x == 0:                 #condition is false
        print('Zero')
    elif x == 1:                 #condition is false
        print('Single')
    else:                        #condition is true
        print('More')
```

…prints *‘More’*

for loops

```
>>> words = ['cat', 'window', 'defenestrate']
>>> for w in words:                # for is iterating over a list
        print(w, len(w))
```

cat 3

window 6

defenestrate 12

while loops

```
>>> count = 0

>>> while (count < 9):
        print( 'The count is:' + count)
        count = count + 1

>>> print "Good bye!"
```

The count is: 0
The count is: 1
The count is: 2
The count is: 3
The count is: 4
The count is: 5
The count is: 6
The count is: 7
The count is: 8 # condition (9<9) is false
Good bye!

Inside the loop...

- break : exits the innermost enclosing for or while loop
- continue: The continue statement, also borrowed from C, continues with the next iteration of the loop

range() function

```
>>> range(5, 10)  # start=5 , stop=10 (not included)
5, 6, 7, 8, 9
>>> range(0, 10, 3)  # step=3
0, 3, 6, 9
>>> for i in range(3):      # range(3)==[0,1,2]
            print(i)
 0
 1
 2
```

Function example

```
>>>def mean(a,b):          # functions are defined using def and :
                           # add the name and the parameters
        mean=(a+b)/2
        return mean

>>> m=mean(5,8)           #function is called
>>> print(m)
6.5
```

Fibonacci Function

- Fibonacci: the **Fibonacci numbers** are the numbers characterized by the fact that every number after the first two is the sum of the two preceding ones:

  0, 1 , 1 , 2 , 3 , 5 , 8 , 13 , 21 , 34 , 55 , 89 , 144 , …

- **MiniTask**: Create a function called fib with argument the number n which will be the boundary where the series stop and print the result for n=2000.

Fibonacci Function Solution Code

```
>>>def fib(n):
        a, b = 0,1    #both variables are defined
        while a < n:
            print(a, end=' ')
            a, b = b, a+b
>>> fib(2000)
```

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597

Lambda Function

- A lambda function is a small anonymous function.
- A lambda function can take any number of arguments, but can only have one expression.

```
>>>def myfunc(n):
        return lambda a : a * n
>>>mytripler = myfunc(3)
>>>print(mytripler(11))
33
```

Data Structures

- Lists
- Sets #keep only unique values ,are unordered

```
>>> {1,2,3,1,2,1,2,3,3,3,3,2,2,2,1,1,2}
{1, 2, 3}
```

- Dictionaries # assign a key to an item

```
>>> d = {'key1':'item1','key2':'item2'}
>>> d['key1']
'item1'
```

- Tuple #unchangable

```
>>> thistuple = ("apple", "banana", "cherry")
```

Input/output

```
x = input(‘Enter your name: ’)
print('Hello, ' + x)
```

***easy typecasting int/float(input(...))

```
>>>x = 3
>>>print('The number is %d' %(x))
>>>print("the number is " + str(x))
>>>'The number is {}'.format(x)
```

The number is 3

Numpy

- NumPy (or Numpy) is a Linear Algebra Library for Python, the reason it is so important for Data Science with Python is that almost all of the libraries in the PyData Ecosystem rely on NumPy as one of their main building blocks.
- $conda install numpy *in Anaconda Prompt*
- import numpy as np

Numpy methods

- np.array(my_list) # creates an array from a list
- np.arange(n1,n2)
- np.zeros(n) / np.zeros((n1,n2))
- np.ones(n)
- np.linspace(0,10,3)

  array([0., 5., 10.])
- np.eye(4)

  array([[1., 0., 0., 0.],
  [0., 1., 0., 0.],
  [0., 0., 1., 0.],
  [0., 0., 0., 1.]])

Numpy Methods

- arr.max(),arr.min(),arr.argmax(),arr.argmin()
- arr.T # transpose of arr
- np.random.rand(n)
- np.random.randn(n)
- np.random.randint(1,100,n)

```
>>>x = np.random.rand(6)
```

```
>>>x.reshape(2,3)
```

array([[0.50503179, 0.08251868, 0.48370013],
[0.97738639, 0.55120198, 0.28890533]])

NumPy

```
>>> arr = np.arange(1,11)
>>> arr
array([ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
>>> arr > 4
array([False, False, False, False, True, True,
   True, True, True, True], dtype=bool)
>>> arr[arr>2]
array([ 3, 4, 5, 6, 7, 8, 9, 10])
```

NumPy Math

- Basic mathematical functions operate elementwise on arrays : + , - , * , /

```
>>>x = np.array([[1,2],[3,4]])
>>>y = np.array([[5,6],[7,8]])
>>> x*y
array([[ 5, 12],
       [21, 32]])   #Elementwise multiplication
```

- Matrix multiplication

```
>>> x.dot(y)
>>> np.dot(x,y)   #same result
```

Numpy Basic Statistics

- np.mean(array) #returns the mean of the array
- np.median(array) #returns the median of the array
- np.std(array) #returns the standard deviation of the array
- np.var(array) #returns the variance of the array
- np.corrcoef(x_array,y_array) #returns the correlation coefficient matrix of the two arrays
- np.random.normal(mean, std, values) #generates an array with a normal distribution
- np.sum(axis)

Numpy Practice

There is a group of 100 IEEE Students of Auth Sub-Branch and we would like to measure their BMI(Body Mass Index).

- Create 2 arrays of random heights and weights

  Height range [155,195](cm) *Weight range[50,100](kg)*

- Calculate BMI of each student and save it in an array

  BMI = Weight(kg) / (Height(m)^2)

- Find the number of overweight students

  (25<=BMI<=30)

- Print the standard deviation of non-overweight students

  (BMI<25)

Solution Code

```
import numpy as np

Weight = np.random.randint(50,100,100)

Height = np.random.randint(155,195,100)

Height = Height*0.01

BMI = Weight / (Height**2)

print("BMI array:")

print (BMI)

num_of_overweight = len(BMI[(BMI>=25) & (BMI<=30)])

print()

print("The number of overweight is :" + str(num_of_overweight))

non_overweight = BMI[BMI<25]

print()

print("The std of non-overweight is :" + str(np.std(non_overweight)))
```

Pandas

- Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language
- Using Pandas, we can accomplish five typical steps in the processing and analysis of data— load, prepare, manipulate, model, and analyze
- $conda install pandas
- >>>import pandas as pd

Pandas Dataframes Exercise

There are 10 IEEE Student who want if their weight is not normal.

- Import the csv file.
- Calculate and save the BMI in a new column.

  BMI = Weight(kg) / (Height(m)^2)

- Change the indexes to the IEEE student names.
- Find and remove the normal weight students

  20<BMI<25 *#df.index[i],df.ColumnName*

Thank you for your time! :)