Introduction to python

First tutorial session

Python lab sessions: administrative details

Lab sessions

1. ME and MS students: Monday and Thursday – 2 pm to 3:15 pm
2. Energy and economics: Monday and Thursday 3:45 to 5 pm
3. Be punctual; late comers may be denied entry to the lab
4. For every lab session, the groups are random – So, please keep both the afternoons free. We will post the session details in moodle
5. You are allowed to do the lab session only on the allotted day and time; all submissions should be only from the Bits and Bytes lab machines
6. Follow the instructions of TAs; any malpractice will be severely dealt with

In the lab...

1. Your conduct should be professional in the lab
2. Bits and Bytes lab: login to the student account
3. Create a directory and name it with you roll number on desktop; At the end of the lab, after uploading to moodle, delete directory; Do not store files anywhere else
4. You are allowed to login to moodle and use resources posted by us on moodle; Do not use internet, web or any other application
5. Invoke python from terminal and use notepad or terminal for writing scripts

Python: a brief introduction

Types of variables

1. Integer: int
2. Floating point: float
3. Complex number: j is the complex number (x + j y)
4. String: str
5. Boolean: bool

Syntax and style

1. White space: used to identify the beginning and end of code segments
2. No mixing of white space and tabs in a script
3. Be aware of magic words: __init__, __import__, __file__,...

Operators

Not comprehensive: please look up a textbook or manual

1. Assignment: = (and variations such as +=, -=, *=,/= ...)
2. Arithmetic: +, -, *, /, %, **, // (integer division)
3. Unary: ~ (invert the bits), - and +
4. Relational: ==, !=, > , <. >=, <=
5. Logical: and, or, not

Look up operator precedence!

Operators ...

1. Membership operators: in, not in
2. Identity operators: is, is not

Defining and calling functions

```
def SayHello():
    print(“Hi”)

def addition(val1, val2):
    return val1+val2

SayHello
addition(5,3)
```

Conditional: if / else-if / else

if *condition*:
    statements
elif *condition*:
    statements
else:
    statements

Nested loops: identified by the spaces / tabs

Loops: for and while

for *variable* in *list*:
    statements
while (condition):
    statements

continue and break: statements that are self-explanatory

An example

Function to calculate factorial of N

```
def factorial(N):
    fact = 1
    while(N>1):
        fact = fact*N
        N = N - 1
    return fact
```

factorial(5)

What happens if somebody gives factorial (0) or factorial(-1)? Are the results correct? Do we want to modify?

Second example

Function to calculate factorial of N

```
def factorial(N):
        fact = 1
        for i in range(N):
                fact = fact*i
        return fact
```

factorial(5)

Now what happens? Why?

How do we fix this?

Second example: modified

Function to calculate factorial of N

```
def factorial(N):
        fact = 1
        for i in range(1,N+1):
                fact = fact*i
        return fact
```

factorial(5)

Data creation: sets

Create sets:

```
OddNos = set([1,3,5,7,9])
Primes = set([2,3,5,7])

U = OddNos.union(Primes)
I = OddNos.intersection(Primes)
D = OddNos.difference(Primes)
```

Data creation: lists

Create sets:

```
OddNos = [1,3,5,7,9]
Primes = [2,3,5,7]

OddNos.extend(Primes)
Primes.append(OddNos)
```

Data creation: tuples

Create tuples:

```
ModelTuple = (1,3,5,(2,4,6,(4,9)))
ModelTuple.__add__((8,27))
```

Index data: dictionary

Index data:

MyDictionary = {“Odd”:(1,3,5,7),”Even”:(2,4,6,8),”Prime”:(2,3,5,7)}

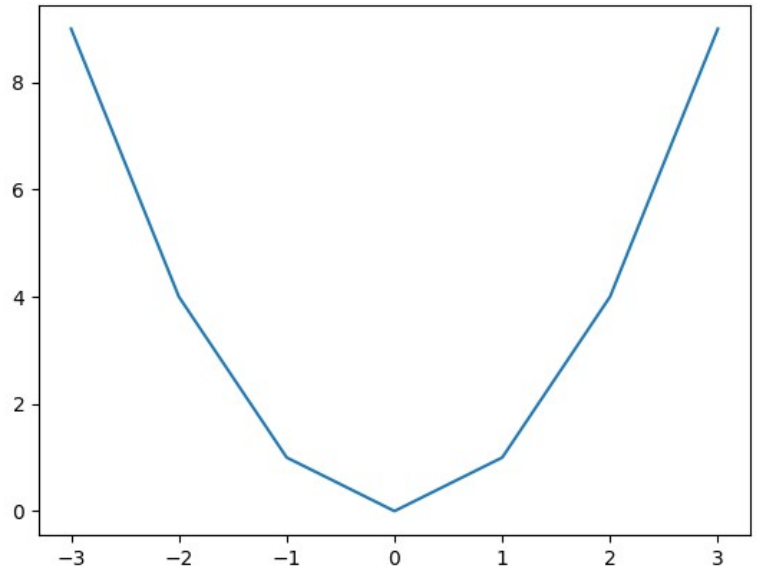
MyDictionary[“Odd”]

Working with libraries

```
import numpy

Vandermonde = numpy.array([[1,2,3],[4,5,6],[7,8,9]])
numpy.linalg.eig(Vandermonde)
numpy.linalg.inv(Vandermonde)

Import numpy as np
Vandermonde = np.array([[1,2,3],[4,5,6],[7,8,9]])
np.linalg.eig(Vandermonde)
np.linalg.inv(Vandermonde)
```

Third example

```
x = [-3,-2,-1,0,1,2,3]
y = [9,4,1,0,1,4,9]

import matplotlib.pyplot as plt
Or
from matplotlib import pyplot as plt

plt.plot(x,y)
plt.show()
```



Working with scripts

(1) Store the commands in a file with .py as extension. For example, here is parabola.py

```
x = [-3,-2,-1,0,1,2,3]
y = [9,4,1,0,1,4,9]
import matplotlib.pyplot as plt
plt.plot(x,y)
plt.show()
```

(2) In terminal, give the command

python3 parabola.py

(3) Once the script is fine and the figures are as expected, you can upload your script in moodle for evaluation.

(4) If needed, you can save the image and upload. From command line you can save the image using the command *plt.savefig("filename.jpg")*

Summary

In the laboratory session this week, you are expected to

(1) Implement a given algorithm or formula;
(2) Input or generate data and plot the same;
(3) Input data in appropriate form for further manipulation; and,
(4) Debug codes with errors.

Please see moodle for your lab session – day and time!

Thank you!!

ALL THE BEST!