python Workshop

1st Session

BY-

CHANDAN MITTAL

STUDENT, CSE DISCIPLINE

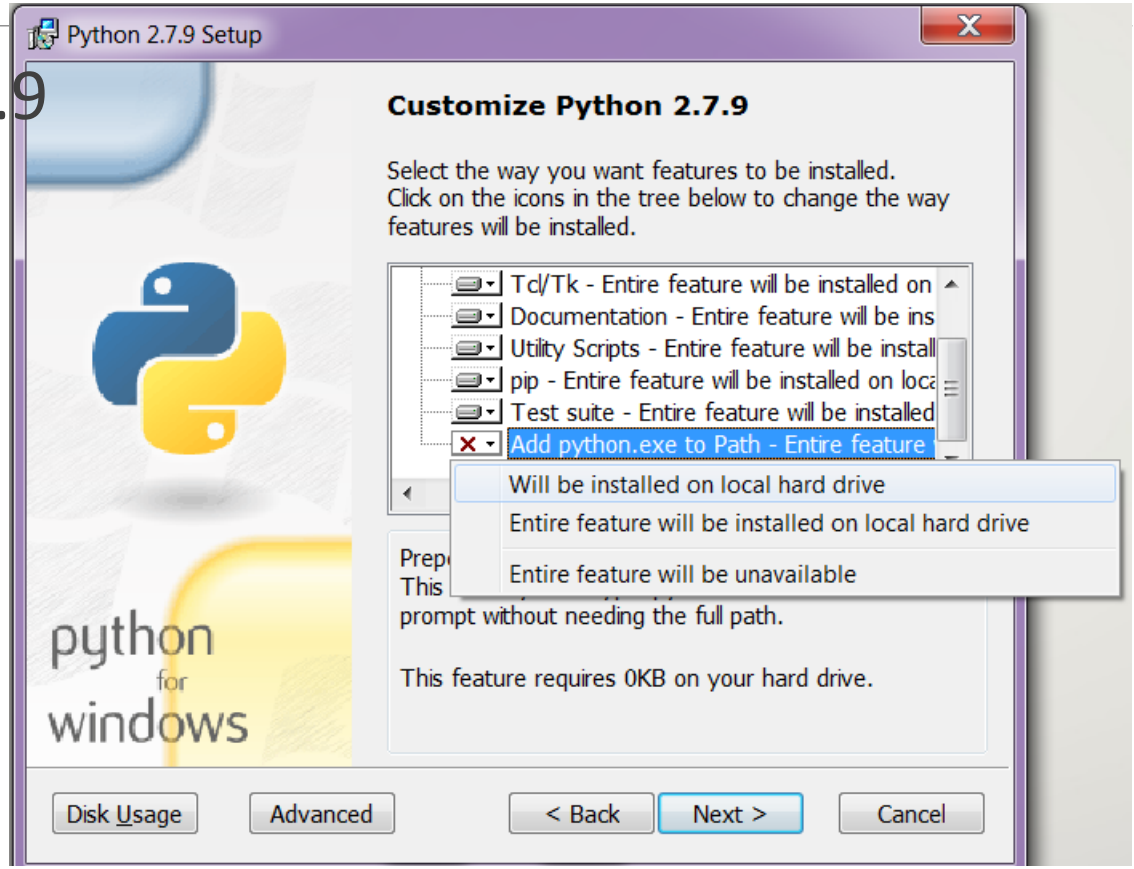
IIITDM JABALPUR

Why learn Python?

- Easy to Learn
- It looks like everyday English.
- Very Small length code.
- Very easy to use. Comes bundled with IDLE(interactive shell + IDE)
- The libraries, modules and frameworks
  - Scipy, NumPy, pyTk, pyQt, pyGame, beautifulsoup, selenium, mechanize, flask, django, web2py etc.

Installing and Configuring Python: Windows

- Download IDLE for Python 2.7.9
- Install IDLE as you install any software
- Take care of enabling this feature
- This is setting the python path @Windows command prompt



Installing Python: Linux

- Python interpreter is installed by default in the debian systems.
- If you want to install IDLE in Debian/Ubuntu Linux, do
  - $ sudo apt-get install idle
- If you're using RHEL / CentOS / Scientific / Fedora Linux , type the following
  - # yum install python-tools
- How do I start IDLE?
  - $ idle
  - Or visit menu options Applications > Programming > IDLE

Help/Docs Available on the Internet

- For python2 https://docs.python.org/2/tutorial/index.html
- For python3 https://docs.python.org/3/tutorial/index.html
- MOOC: MITx 6.00.1x Intro to CS and Programming using Python @ edx.org
- Books:
  - Head First Python
  - Learning Python 2 O'Reilly and Learning Python 3 O'Reilly
  - Programming Python 2 O'Reilly and Programming Python 3 O'Reilly
  - O'Reilly Python CookBook
  - Gray Hat Python (Python Programming for Hackers and Reverse Engrs.

All the above pdf's are available at htpp://learncodeshare.wordpress.com.

How Python Program Runs

- Interpreter called IDLE
- Run your program line by line using Python Interpreter IDLE or on command line/terminal
- Execute your program in one go
- Write python program and save it to *filename*.py and execute
  - $ python *filename*.py ON LINUX terminal
  - C:\> python *filename*.py ON WINDOWS command prompt
  - Or open IDLE, press Ctrl + N which creates a new file
    - Write your python program in it,
    - Save it to disk as *filename*.py
    - And press F5. F5 is the run or execute command for IDLE editor.

First Program and Comments

>>> print ‘Hello World’

Hello World

>>> #this is a single line comment

>>> ‘’’ this is a multiple line comment

whatever you write here is a comment
this type of comment is started using atleast 3 single or double quote and ended using the same
This is the third line
‘’’

Variables and Data Types in Python

- What's so cool about Variables and Data Types in python?
- **You don't need to remember which variable had what data type.**
- Why??
- **Because you don't need to declare any.**
- Have a look at this:

```
>>> a = 3
>>> b = 'dakota johnson'
>>> c = 3+4j
>>> d = 3.1415
```

- No semi colon, no need to declare data type of variable before hand.

```
>>> type(a)
<type 'int'>
>> type(b)
<type 'str'>
>>> type( c )
<type 'complex'>
>>> type(d)
<type 'float'>
>>>a, b = b, a #swap is cool
```

Numbers and Floats

```
>>> a =
39781273823978128973189238972173293098120938218329030912839081238901820389210809283908210398298392108302109389021830981209832198392180320198301 8

>>> print a

39781273823978128973189238972173293098120938218329030912839081238901820389210809283908210398298392108302109389021830981209832198392180320198301 8

>>> b = 23.13432432432343123

>>> print b

23.1343243243                #by default in total 12digits are for float

>>> round(23.6543, 3)        #round to 3 decimal digits

23.654
```

Arithmetic and Logical Operators

```
>>> 2 + 3
5
>>> 2*3
 6
>>> 2/3
0
>>> 2 – 3
-1
>>> 3 % 2
1
```

```
>>> 2 + 3.0
5.0
>>> 2.0*3
 6.0
>>> 2/3.0
0.666666…
>>> 2 – 3.52
-1.52
>>> 3.2 % 2
1.20000…002
>>> 3 % 2.1
0.8999…
```

```
>>> 2 ** 4
16
>>> 3 << 3    #3 * 2^3
 8
>>> 10 >> 2  #10/(2^2)
2
>>> 1 << 3.2
error
>>> 10 >> 2.1
error
>>> 3 == 2
False
```

```
>>> 3 >= 2
True
>>> int(3.212)
 3
>>> float(3)
3.0
>>> from math import ceil
>>> ceil(3.42)
5
```

Strings

```
>>> a = ‘dakota johnson’
>>> print a
 dakota johnson
>>> len(a)
14
>>> a[0]
 ‘d’
>>> a = ‘234234’
>>> print a
‘234234’
>>> a[0]
‘2’
```

```
>>> a = “dakota johnson”
>>> print a
 dakota johnson
>>> a[0]
 ‘d’
>>> a = ‘234234’
>>> print a
‘234234’
>>> a[0]
‘2’
>>> len(a)
6
```

```
>>> a = “I don’t wana do B.Tech”
>>> print a
 “I don’t wana do B.Tech”
>>> a[2]
 ‘d’
>>> a = ‘I don\’t wana do B.Tech’
>>> print a
 “I don’t wana do B.Tech”
>>> a[2]
‘d’
>>> len(a)
22
```

Strings (continued..)

```
>>> #strip is cool
>>> a = ‘   Talk is cheap, show me the code.   ’
>>> a.rstrip()
‘   Talk is cheap, show me the code.’
>>> a.lstrip()
‘Talk is cheap, show me the code.   ’
>>> a.strip()
‘Talk is cheap, show me the code.’
>>> a
'    Talk is cheap, show me the code.    '
```

```
>>> #split is just awesome
>>> path = ‘home/user/folder/file’
>>>path.split(‘/’)
[‘home’, ‘user’, ‘folder’, ‘file’]
>>>  s = ‘2 3 4 5 6 7’
>>> #if split is supplied with no parameter, by default it splits using whitespaces
>>> s.split()
[2, 3, 4, 5, 6, 7]
```

Strings (continued…) [slicing]

```
>>> s = “chandan mittal” #0-based indexes
>>> #substring starting from index 1 to 9
>>> s[1:10]
‘handan mi’
>>> #step of 2 over substring from index 1 to 9
>> s[1:10:2]
‘hna i’
>>> s[::-1]          #reverse the string
‘lattim nadnahc’
>>> #reverse string and take a step of 2
>>> s[::-2]
‘lti anh’
>>> #reverse a substring
>>> s[1 : 10][ : : -1]
‘im nadnah’
```

Lists (like arrays)

```
>>> a = [2, 3, 4]    #a is list of 3 elements whose first element is 2 and last is 4
>>> print a
[2, 3, 4]
>>> print a[1]
3
>>> a[1] + a[2]
7
>>> #Unlike array, lists can have elements of different data types
>>> a = [‘eggs’, 2, ‘I don\’t like B.Tech’, true]
>>> print a[2]
I don’t like B.Tech
```

Lists (continued..)

```
>>> a = [2, 4, 2]
>>> a.append(5)
>>> a.count(5)
1
>>> a.sort()
[2, 2, 4, 5]
>>> a.pop()
5
>>> print a
[2, 2, 4]
```

```
>>> a.remove(2)
>>> print a
[2, 4]
>>> a.index(5)
error
>>> a.index(2)
0
>>> a.reverse()
>>> print a
[4, 2]
```

```
>>> #a.insert(index, value)
>>> a.insert(1, 3)
>>> print a
[4, 3, 2]
```

Lists (continued..)

```
>>> #Lists can be of any dimension
>>> a = [2, 3, ['hello', 5], ['world', 3], True]
>>> print a[0], a[2][0]
2 hello
>>> (a[1] == a[3][1]) == a[4]
True
```

```
>>> #initializing large lists
>>> b = [0]*10
>>> b
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
>>> a = [2]
>>> b = ['string']
>>> a + b   #Lists can be concatenated
[2, 'string']
```

Tuples

- Ordered collection of arbitrary objects
- Accessed by offset/index
- Immutable(don't support in-place changes)
- Fixed length, heterogeneous, arbitrarily nestable

```
>>> z = (1, 2, 'chandan')
>>> z[2]
'chandan'
>>> z[1] = True
error
```

```
>>> y = (9, 8, [2, 3, ['hello', 'world']])
>>> y[2][1]
3
>>> y[2][2][1]
'world'
>>> z + y
(1, 2, 'chandan', 9, 8, [2, 3, ['hello', 'world']])
```

Dictionaries or key-value pairs

```
>>> d1 = {}
>>> d2 = {'one' : 1, 'two' : 2}
>>> d2['two']
2
>>> d3 = {1 : 'one', 2 : 'two'}
>>> d3[1]
'one'
>>> len(d3)
2
```

```
>>> d4 = {'india' : {'agra': 'UP', 'indore': 'MP', 'jaipur': 'Rajasthan'}}
>>> d4['india']['indore']
MP
>>> d4['india']['indore'] = 'unknown'      #reassign the value
>>> del d['india]['indore']                #delete the key
>>> d['india']['indore']
KeyError
```

if elif else

```
>>> #program to check if number is positive, negative or zero
>>> n =  0.21
>>> if n < 0:
...         print ‘negative’
...     elif n == 0:
...         print ‘zero’
...     else:
...         print ‘positive’
```

```
>>> a, b, c = 4, 3, 5
>>> if a > b:
...         if a > c:
...             print str(a) + ‘ is largest’
...         else:
...             print str(c) +  ‘ is largest’
...     else:
...         if b > c:
...             print str(b) + ‘is largest’
...         else:
...             print str(c) + ‘ is largest’
```

```
>>> left, right = 2, 5
>>> x = 7
>>> if x <= right and x >= left:
...         print ‘yes in range’
...     else:
...         print ’not in range’
>>> #what does it prints??
>>> if left or 10%2:
...         print ‘either one is True’
...     else:
...         print ‘both False’
```

Input Output

```
>>> s = input()
Hello
>>> print s
‘Hello’
>>> n = input()
23.43
>>> print n
23.43
>>> x = input()
12323131231232132144142434234324324324234 23
>>> print x
12323131231232132144142434234324324324234 23
```

```
>>> s = raw_input()
Hello
>>> print s
‘Hello’
>>> n = raw_input()
23.43
>>> print n
23.43
>>> x = raw_input()
12323131231232132144142434234324324324234 23
>>> print x
12323131231232132144142434234324324324234 23
```

Input Output (continued..)

```
>>> x, y = ‘a’ , ‘b’
>>> print x, y
a b
>>> print x+y
ab
>>> import sys
>>> sys.stdout.write(‘hello world\n’)
hello world
>>> sys.stdout.write(x+y + ‘\n’)
```

```
>>> sys.stdin.readline()
Heyy
‘Heyy\n’
>>> a = sys.stdin.readline()
Heyy
>>> print a
Heyy
```

Functions: Simple Library Functions

```
>>> s = 'Don\'t think. Just do it.'
>>> len(s)
24
>>> j in s
Error, j not defined
>>> 'j' in s
False
>>> 'J' in s
True
```

```
>>> s.find('o') #find index of 1st instance
1
>>> s.find('o', 1)
1
>>> s.find('o', 2)
19
>>> s.find('o', 19)
19
>>> s.find('o', 20) #not found
-1          #returns -1 if not found
>>>#index returned is for string starting from 0
```

Simple Library Functions (continued..)

```
>>> a = [3, 2, 1]
>>> sorted(a) #returns variable
[1, 2, 3]
>>> #a still remains same
>>> a
[3, 2, 1]
>>> a.sort()
>>> a #a gets sorted forever
[1, 2, 3]
```

```
>>> bin(5)
'0b101'
>>> ord('a')
97
>>> x = 2
>>> b = range(3, 8)
>>> if x not in b:
...         print 'not in b'
...   else:
...         print 'yes'
```

Loops

```
>>> b = [2, 3, 2, 1]
>>> 3 in b
False
>>> 1 in b
True
>>> n = 5
>>> range(n)
[0, 1, 2, 3, 4]
>>> xrange(n)
error
```

```
>>> for i in range(n):
        print i
>>> for x in b:
        print x
>>> i = 0
>>> i++ #gives error
>>> while i < n:
        print i
        i = i + 1
```

```
>>> import math
>>> n = 20
>>> temp = n
>>> root = int(math.ceil(math.sqrt(n)))
>>> for i in range(2,  root+1):
...         if temp%i == 0:
...                 print str(i) + 'is prime factor of ' + str(n)
...                 while temp%i == 0:
...                         temp = temp/i
>>> if temp > 1:
...         print str(temp) + 'is a prime factor of ' + str(n)
```

User defined functions

```
>>> #recursive fn to find Fibonacci number
>>> #fn return type is not declared
>>> def fib(n):          #argument datatype not declared
...     if n <= 1:
...         return n
...     else:
...         return fib(n-1) + fib(n-2)
>>> print fib(5)
5
>>> print 'sum of first 5 fibo numbers is ' + str(fib(5+2))
13
```

```
>>> #fn returning multiple values returns a tuple
>>> def arithmetic(a, b):
        return a+b, a-b, a*b, a/b
>>> arithmetic(2, 3)
(5, -1, 0, 6)
>>> def f(a, b):
        return range(a) + range(b)
>>> f(2, 3)
[0, 1, 0, 1, 2]
```