INTRODUCTION TO PYTHON

The language that meets all requirements
And the language which all requirements meet

Why python?

- Easy to learn; good code readability
- Open source, has a large community
- Large number of 3rd party libraries/packages
- ITS FREE!

If you see a slide in this colour scheme

Code it along with us

The obligatory hello world program

```
print('hello world')

# Trust us! That's it.



# And yeah, COMMENT!!
# Ignore hashtags at runtime
# Because this isn't your
# insta feed
```

Basics

- Datatypes
- Conditional Statements
- Loops
- Operators
- Lists
- Functions

Datatypes

What you must have learnt in **C**:

1) Integers & Floats
2) Floating points
3) Complex numbers
4) Strings

Datatypes

1) Integers & Floats
2) Floating points
3) Complex numbers
4) Strings

Code

```
#integer
a = 21

#float
b = 3.14159

#complex
c = 3 + 4j

#string
g = ‘hello world’
```

Datatypes:

```
x,y = 20,30

a = b = c = 1

name = “harambe”


print(type(x))
print(type(name))
```

Conditional Statements

1) if statement
2) if-else statement
3) if-elif-else statement

Code

```
a,b,c = 4,1,2


if a>3:
    print(“a is greater than 3”)


if b<3:
    print(“b is less than 3”)
else:
    print(“b is not less than 3”)
```

Conditional Statements

```
a = 4

if a>5:
    print(“a is greater than 5”)

elif a<5:
    print(“a is less than 5”)

else:
    print(“a equals 5”)
```

Loops

1) For loop
2) While loop

Code

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

#while loop
i = 0
while(i<10):
    print(i)
    i = i + 1
```

Example Code

```
for i in range(0,20,2):
    for j in range(1,10,1):
        print(i,j)


points = [0,4,5,10]
for i in points:
    print(i)
```

Operators - Arithmetic

1) Addition (+)
2) Subtraction (-)
3) Multiplication (*)
4) Division (/)
5) Modulus(%)
6) Exponent(**)

```
Code

a = 5
b = 6

#operations in order
a+b
b-2
a*4
b/3
a%b
a**2
```

Operators - Comparison

1) Check equality (==)
2) Check inequality(!= or <>)
3) Greater than (>)
4) Less than(<)
5) Greater than or equal to (>=)
6) Less than or equal to (<=)

Code

```
a = 5
b = 6

#operations in order
a==b
b!=2
a>4
b<3
a>=b
a<=2
```

Operators - Logical

Those C / C++ pain days of || and && and ! are gone!!

1) and
2) or
3) not

Code

```
a = 5
b = 6

if(a==b and b>5):
    print(‘something’)

if(a==5 or b==2):
    print(‘some other thing’)

if(not b==9):
    print(‘b is not equal to 9)
```

Lists

- **‘structures’ from C might ring a bell**

- Lists are written within square brackets
  a = [1,2,4,5]
- Lists can have multiple data types (including other lists)
  b = [‘name’ , 3 , 5.89, [1,2,3]]
- Each list element is accessed by its index (starting from zero)
  b[0]
- Multidimensional lists’ elements can be accessed by using either of
  b[2,0] or b[2][0]

Lists

```
ppap = [‘pen’, ‘apple’]

#add elements to list
ppap.append(‘pineapple pen’)
ppap.append(5)
ppap.append(67.89)
ppap.append(2e5)

#get length of the list
len(ppap)

#loop over a list
for i in ppap:
#For python3 users:
    print(i, ‘\t-->’, i*2)
#For Python2.7 users:
    print i, ‘\t-->’, i*2
```

Example Code:

```
a = [‘A’ ,’C’, ’C’, ’G’, ’C’, ’C’, ’G’, ’C’, ’T’, ’T’ ,’A’]

a.count(‘G’)

a.insert(3,’A’)
```

Just to put pseud on Python’s behalf

```
"hello"+"world"     "helloworld"        # concatenation
"hello"*3           "hellohellohello"   # repetition
"hello"[0]          "h"                 # indexing
"hello"[-1]         "o"                 # (from end)
"hello"[1:4]        "ell"               # slicing
len("hello")        5                   # size
"hello" < "jello"   1                   # comparison
"e" in "hello"      1                   # search
```

```
'single quotes'  """triple quotes"""  r"raw strings"
```

Functions

A function is a block of organized, reusable code that is used to perform a single, related action.

Code

```
def isDivisibleBy(dividend,divisor):
    if(dividend%divisor==0):
        return 1
    else:
        Return 0


print(isDivisibleBy(4,2))
```

Another Example:

```
#Recursive Factorial Function

def fact(n):
    if(n<=1):
        return 1

    else:
        return n*fact(n-1)
```

Intermediate topics

- Dictionaries
- Opening files
- Regular Expressions
- Numpy arrays
- Searching with Scipy
- Matplotlib

Dictionaries

Ever wanted to access items in an array the other way round? i.e give the name and find the index?

Dictionaries are meant just for that. Its an associative list where your index can be a string

Dictionaries

```
a = dict(one=1, two=2, three=3)

b = {'one': 1, 'two': 2, 'three': 3}

c = dict(zip(['one','two','three'],[1,2,3] ))

d = dict([('two', 2), ('one', 1), ('three',3)])

e = dict({'three': 3, 'one': 1, 'two': 2})

#All are equivalent
#a['one'] gives 1
```

Example Code:

```
data['a']=1  # updates if 'a' exists, else adds 'a'

# OR
data.update({'a':1})

# OR
data.update(dict(a=1))

# OR
data.update(a=1)
```

Inverting Dictionary

```
A = {'Kim': 1, 'was': 2, 'robbed!!': 3}

inv_map = {v: k for k, v in A.items()}
# works iff the dictionary is invertable
```

Opening files

```
f = open('workfile','r+')
for line in f:
    a=line.split(' ')
```

split() splits a string at every occurrence of the argument. If no argument provided, it will split at all commonly occurring whitespace breakpoints like ‘ ‘, ’\n’, ’\t’, ’,’ , etc

Numpy Arrays

- Very useful Python Library
- Handles arrays(lists) using C backend, and so much much faster
- Can handle any n-dimensional array, and quickly do operations on them
- Convert any list to numpy array:
  A = [[1,2],[3,4]]
  B = np.array(A)

```
Code

import numpy as np

np.array()
np.zeros((100,100,3),np.uint8)
#gives an array of size 100x100x3
#of type 8bit int
```

Numpy Arrays

The basics

```
import numpy as np

# np.arange can take float steps too!!
a=np.arange(30,dtype=np.uint8)
print(a)

# Reshape into 5x6 matrix
a=a.reshape(5,6)
print(a)

# Flatten out an n-dim. array
a=a.ravel()
print(a)
```

Scipy

Scipy has good searching options

For example, if you want only elements greater than 0

```
Code

import scipy
i,j = scipy.where(A>0)
B = A[i,j]
#where A is 2D

#Similarly u can do it across arrays
i = scipy.where(A[:,0]>A[:,1])
```

Numpy Arrays

Thresholding

```
import numpy as np
from scipy import *

a=np.arange(30,dtype=np.uint8)
b=a.reshape(5,6)
i,j=where(b>15)

#i,j are lists containing indices of
#all those elements which are >15
#Set them all to 100
b[i,j]=100

#This can also be done in one line as
b=a.reshape(5,6)
b[b>15]=100
```

Matplotlib

```
import numpy as np
from matplotlib import pyplot as
plt

x=np.arange(1,10,.1)
y=x**2
plt.plot(x,y)
plt.scatter(x,y)
plt.show()
```

3D Plots

```
from pylab import *
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(111,
projection='3d')
z=arange(1,50,.1)
x=z*cos(z)
y=z*sin(z)
ax.plot(x,y,z)
plt.show()
```

Things you should check out by yourself

Cuz we said so :P

- Turtle library
- Regular Expressions
- Plotting capabilities of matplotlib
- Python packaging index (pypi)

And that’s it. Hope you liked it :-)

(Don’t ask about the distorted smiley….blame the font)