PYTHON Short Notes (By Himanshu Gupta)

Swap Integers without additional variable:

a,b=b,a

Common Python Data Types:

1. Numbers
   - int
   - float
   - long ;Ex: a=100L
   - complex ;a=5.0+4.4j
2. String
   - ascii ;
   - unicode ;a=u'Ditiss' & a=unicode('Ditiss')
3. Lists ;a=[1,2,'a','b']
4. Tuple ;a=(1,2,'a')
5. Dictionaries ;a={1:'abc',2:'xyz',3:123}
6. Set ;a=set([1,2,3,3,4,2,1])
7. Boolean ;a=True/False

Example-1

```
#!/usr/bin/python -tt

import sys

def main():

        var1=input("Enter a value:")      (or)      var1=raw_input("Enter a value:")

        print 'value of var1 is:',var1

        print 'var1 is of type:',type(var1)

if __name__ == '__main__':

        main()
```

If Statement:

```
if check_condition 1:
        do_this
elif check_condition2:
        do_this
elif check_condition3:
        do this
else
        do this
```

For Loop:

Example:

```
>>>a='Hello World'
>>>for elem in a:
...     print elem,
output -> Hello World
```

Example:

```
>>>a='Hello'
>>>for elem in a:
...     print elem
output ->H
         e
         l
         l
         o
```

While Loop:

```
>>>while i<n:
...     print a[i]
...     i=i+3
...
0
3
6
9
12
15 ....
```

The abs() Function:

- Returns the absolute value of a number.

The round() Function:

- return the floating point value no. rounded to n digits after the decimal point.

- if n digits is omitted it defaults to zero.

Example: **a=20.4347973098** and print **round(a,4) = 20.4347**

repr() :- shows (internally) value. Example- >>>a='''hello''' and >>>print repr(a) = 'hello'

a[Start:End] :- End element not included in output.

Negative Indexing:- .... -5 -4 -3 -2 -1 **Note**- **S[:n]+S[n:]=S**

String Methods :

1. **s.lower() and s.upper()**
2. **s.strip()**
3. **s.isalpha()**
4. **s.isdigit()**
5. **s.isspace()**
6. **s.startswith('other') and s.endswith('other')**
7. **s.find('other')**
8. **s.replace('old','new')**
9. **s.split('delim')**
10. **s.center(width,[fillchar])** ;odd/even no-padding on left /right side

List Methods :-

1. **list.append(elem)**
2. **list.insert(index,elem)**
3. **list.extend(list2)**
4. **list.index(elem)**
5. **list.remove(elem)**
6. **list.sort()** **;order-**number->special char -> number string -> character
7. **list.reverse()**
8. **list.pop(index)**
9. **del ;** Example- del a[1]

id() - it shows the memory location where that data is stored.

SORTING :-

sorted (list) Function :

- It takes a list & returns a new list with the elements in sorted order. Original list is not changed.
- The sorted() function can be customised through optional arguments.

- The syntex **cmp**, **key** and **reverse** refers to optional named arguments.

reverse :

The sorted() optional argument **reverse=True** makes it sort backwards.

```
example-        a=[4,2,1,6]
                sorted (a,reverse=True)
                [6,4,2,1]
                sorted(a)
                [1,2,4,6]
```

Key :

The optional 'key' can be used which specifies a function that transforms each element before comparison.
The key function takes in 1 value and returns 1 value.
The returned "Proxy" value is used for the comparison within the sort.
Example 1 -

1. key=len
2. key=str.lower
3. key=str.upper
4. key=str.isupper
5. key=str.islower

Example 2 -

sort the list using the last character.

```
>>>a=['ccc','aaaz','d','bb']
>>>def Last(s):return s[-1]
>>>sorted (a,key=last)
['bb','ccc','d','aaaz']
```

cmp :

- Traditional two-argument comparision was done using the cmp function
- Its the Builtin comparision function for strings,int..etc.
- Takes 2 values from the list & returns negative/0/positive to indicate their ordering.

```
Example-        >>>cmp
                >>>a='a'
                >>>b='b'
                >>>cmp (a,b)     = -1
                >>>cmp (a,a) = cmp (b,b)  =0
                >>>cmp (b,a)     =1
```

Make a String out of a List :-

1. ':'.join(a)
2. '\n'.join(a)
3. ''.join(a)

Break Statement: Terminates the current loop & resumes at the next statement.
Continue Statement:

- Returns the control to thr begining of the while loop.

- The continue statement can be used in both while and for loops.
- Rejects all the remaining statements in the current iteration of the loop.

Pass statement:

- Used when a statement is required syntactically but you do not want any command or code to execute.
- The pass statement is a null operation - nothing happens when it executes.

Questions-

1. Build a list of the squares of the integers from 0 to 10.
2. Build a list of the lengths of the names in the list.
   names=['Tinku','Pinky','Gabbar']
3. Build a list of all possible pair of webserver & browser.
   webservers=['apache','iis','nginx']
   browsers=['firefox','ie','chrome']

4. Create a list of the squares of the integers from 0 to 100 where the last digit of square is 9.

List Comprehension:

- Also called mapping lists.
- A compact way to write an expression that expends to a whole list.

Example-

-
  ```
  >>>nums= [1,2,3,4]
  >>>squares = [n*n for n in nums]
  >>>print squares
  [1,4,9,16]
  ```
- List comprehension with Strings
  ```
  >>>strs=['hello','and','goodbye']
  print strs
  ['hello','and','goodbye']
  ```
  - change to uppercase with '!!!' append
  ```
  >>>strs_new=[s.upper()+'!!!' for s in strs]
  >>>print strs_new
  ['HELLO!!!','AND!!!','GOODBYE!!!']
  ```
- If condition with List comprehension
  ```
  >>>nums=[2,8,1,6]
  >>>small =[n for n in nums if n<=2]
  >>> print small
  [2,1]
  ```

Python Dictioanry Data Type :-

- Key value Pair (un-ordered)
- Also known as hash tables
- keys are unique & immutable objects
- value can change

Strings - Immutable
Lists - Mutable
Tuple - Immutable

Value lookup:-

- looking up a value which is not in the dict throws a keyerror.
- use the 'in' to check if the key is in the dict.
- can also use **dict.get(key)** which returns the value or none if the key is not present.

  Example-

```
>>>print dict.get('a')
>>>print dict.get('X' , 'can return a custom message like this')
```

dict.keys() and dict.values() :

Returns the list of the keys or values explicitly.

Dictionary inside Dictionary :

```
>>>aa={1:{3:'oneone'},2:{4:'twotwo'}}
>>>print aa
{1:{3:'oneone'},2:{4:'twotwo'}}
print aa[1][3]
twotwo
print aa[2][4]
twotwo
print aa[1][1]
keyerror : 1
```

Deleting an Item from Dictionary :

```
>>>d1 = {'count':2,'cartoon':'superman'}
>>>d1
{'count':2,'cartoon':'superman'}
>>>del d1['cartoon']
>>>d1
{'count':2}
```

Dict formatting :

```
>>>d1={}
>>>d1 ['cartoon']='chota Bheem'
>>>d1 ['count'] = 2
>>> print d1
{'count':2,'cartoon':'superman'}
>>>string1='I want %(count)d copies of %(cartoon)s'%d1
>>>print string1
I want 2 copies of chota bheem.
```

items() :-

- returns a list of (key,value) tuples
- most efficient way to examine all the key value data.

<u>SET DATA</u> :

```
>>>set A=set([1,2,3,4,3,2,1])
>>>print set A
set([1,2,3,4,3,2,1])
>>>print type(set A)
<type 'set'>
```

#Union of two sets

```
set A| set B
```

#Intersection of two sets

```
set A & set B
```

#Difference Between two sets

```
set A - set B       ;elements present in set A but not in set B
set B - set A       ;elements present in set B but not in set A
```

<u>To Read and Write a File Using Python</u> :

1. Read:-

```
#nano p1.py                 ;make a file first
```

1st Method:-

```
#!/usr/bin/python -tt
import sys
def print_file (filename):
            f=open(filename,'rU')
            lines=f.readlines()
            print type(lines)
            print lines
            f.close()
def main():
            print_file (sys.argv[1])

if __name__ == '__main__':
            main()
```

2nd Method:

```
#!/usr/bin/python -tt
import sys
def print_file (filename):
            f=open(filename,'rU')
            string=f.read()
            print type (string)
            print string
            f.close()
def main():
            print_file (sys.argv[1])

if __name__ == '__main__':
            main()
```

Write :-

```
#!/usr/bin/python -tt
import sys
def cat(filename):
        f=open(filename,'W')
        for val in range(6)
                f.write (str(val)+'\n')
        f.close()
def main():
        cat (sys.argv[1])

if __name__ == '__main__':
        main()
```

Print :-

```
#!/usr/bin/python -tt
import sys
def print_file (filename):
        f=open(filename,'rU')
        lines=f.readlines()
        print lines
        f.close()
def main():
        print_file (sys.argv[1])

if __name__ == '__main__':
        main()
```

Count Lines :-

```
#!/usr/bin/python -tt
import sys
def print_file (filename):
        f=open(filename,'rU')
        lines=f.readlines()
        print 'Total no of lines:', len(lines)
        print 'Ignoring empty lines'
        count=0
        for line in lines:
                if line != '\n'
                        count = count+1
        print count
        f.close()
def main():
        print_file (sys.argv[1])

if __name__ == '__main__':
        main()
```

Count Words :-

```
#!/usr/bin/python -tt
import sys
def print_file (filename):
	f=open(filename,'rU')
	lines=f.read()
	count=0
	for line in lines:
		if (line == '\n' or line == ' '):
			count = count+1
	print count
	f.close()
def main():
	print_file (sys.argv[1])

if __name__ == '__main__':
	main()
```

Count Characters :-

```
#!/usr/bin/python -tt
import sys
def print_file (filename):
	f=open(filename,'rU')
	lines=f.read()
	print len(lines)
	f.close()
def main():
	print_file (sys.argv[1])

if __name__ == '__main__':
	main()
```

Palindrome **:-**

```
import sys

def main():

    my_str = raw_input("Enter a string: ")
    a=my_str.upper()
    b=a[::-1]
    print b
    if a==b:
        print("It is palindrome")
    else:
        print("It is not palindrome")

if __name__ == '__main__':
    main()
```

Armstrong :

```
#!/usr/bin/python -tt
def main():
        print 'Check whether a number is armstrong or not'
        num1=raw_input('Enter a number to test ')
        leng=len(num1)
        sum=0
        for var in num1:
                sum=sum+int(var)**leng
        if str(sum)==num1:
                print 'it is a armstrong'
        else:
                print 'it is not a armstrong'

if __name__ == '__main__':
main()
```

<u>Fibonacci series</u> :

```
def fibo(n):
        if n <=1:
                return n
        else:
                return(fibo(n-1)+fibo(n-2))
def main():
        nterms=int(input("how many terms you want.."))
        if nterms==0:
                print "please enter positive number."
        else:
                for i in range(nterms):
                        print(fibo(i))
if __name__=='__main__':
        main()
```

<u>To find out Current Directory of File</u>:

```
#!/usr/bin/python -tt
import os
def main():
    print 'Current Working Directory :',os.getcwd()
if __name__ == '__main__':
      main()
```

<u>To find out all Folders & files inside given folder(path)</u>:-

```
#!/usr/bin/python -tt
import sys
import os
def list(dir):
    filenames=os.listdir(dir)
    print filenames
    print '---------'
    for file in filenames:
        #if file.startswith('m'):
        #if file.endswith('.pdf')
            print file
def main():
    list(sys.argv[1])
if __name__ == '__main__':
      main()
```

<u>To Find out absolute path and relative path</u> :-

```
#!/usr/bin/python -tt
import sys
import os
def list (dir):
    filenames = os.listdir(dir)
    print '---------------------'
    for file in filenames:
        print 'filename:',file
        path=os.path.join(dir,file)
        print 'Relative path:',path
        print 'Absolute path:',os.path.abspath(path)
        print '++++++++++++++'
def main():
    list(sys.argv[1])
if __name__ == '__main__':
        main()
```

<u>To Check if A file/folder Exists:</u>

```
>>>import os
>>>os.path.exists ('/root/py')
True/False
```

&

```
>>>os.path.isfile('one.txt')
>>>os.path.isdir('one')
```

<u>To make a Directory:-</u>

```
>>>os.mkdir('path')
```

<u>To remove a Directory:-</u>

>>>**os.rmdir**('Directory_path')
(**or**)
>>>**os.removedirs**('dir/dir1/sub_dir')

<u>os.removedirs(path)</u> :-

- Remove directories recursively
- Works like rmdir() except that if the leaf directory is successfully removed, removedirs() tries to successively remove every parent directory mentioned in path untill an error is raised (Which is generally ignored, because it means that a parent directory is not empty)
- Rasies OS Error if the leaf directory could not be successfully removed.

<u>To check the permission:-</u>

```
#!/usr/bin/python
import os,sys
def main():
        loc=sys.argv[1]
        ret=os.access(loc,os.F_OK)
        print "F_OK_return value %s" %ret
        ret=os.access(loc,os.R_OK)
        print "R_OK_return value %s" %ret
        ret=os.access(loc,os.W_OK)
        print "W_OK_return value %s" %ret
        ret=os.access(loc,os.X_OK)
        print "X_OK_return value %s" %ret

if __name__ == '__main__'
        main()
```

<u>To copy a File using Python</u> :-

```
>>>import shutil
>>>import os
>>>shutil.copy ('/root/py/one.py','/root/py/one/one.py')
>>>os.path.exists ('/root/py/one/one.py')
True
```

<u>Note</u> :-

- **os.remove() -** remove a file
- **os.rmdir() -** remove an empty directory
- **shutil_rmtree -**delete a directory and all its contents

<u>Factorial</u> :-

```
#!/usr/bin/python -tt
def main():
        print 'Calculation Of factorial'
        a=raw_input('Enter the number ')
        alist=range(1,int(a)+1)
        print alist
        prd=1
        for var in alist:
                prd=prd*var
        print 'The factorial of the number',a ,'is',prd
if __name__ == '__main__':
main()
```

To Identify Duplicate Files With Different Hashing: -

```
#!/usr/bin/python -tt
import hashlib
import os
def main ():
        listapp={}
        listdupfiles={}
        dir='/root/py'
        for name in os.listdir(dir):
                if os.path.isfile(os.path.join(dir,name)):
                        if hashlib.md5(open(os.path.join(dir,name)).read()).hexdigest() not in listapp:
                                listapp[hashlib.md5(open(os.path.join(dir,name)).read()).hexdigest()]=name
                        else:

                listdupfiles[hashlib.md5(open(os.path.join(dir,name)).read()).hexdigest()]=name
        print 'Original Files : ',listapp
        print 'Duplicate Files : ',listdupfiles

if __name__ == '__main__':
        main()
```

Sorting Based On Length :-

```
        #!/usr/bin/python -tt
        def main():
                print 'Calculation Of Duplicate Number'
                print 'Please enter the words with space between them'
                string_input=raw_input()
                input_list=string_input.split()
                input_list=[x for x in input_list]
                print input_list
                print sorted(input_list,key=lambda s:len(s))

        if __name__ == '__main__':
        main()
```

To find The Subdomain from Source code page of a website page(www.checkpoint.com) :-

First download the source code.

```
        #wget www.checkpoint.com

        #!/usr/bin/python -tt
        import re
        f=open('index.html','rU')
        var=set(re.findall('[\w.]*\.checkpoint.com',f.read()))
        for i in var:
                print i
```

<u>To Read a Webpage Content</u> :-

```
#!/usr/bin/python -tt
import urllib
url='https://en.wikipedia.org/wiki/OpenStack'
uf=urllib.urlopen(url)
print type(uf)
print uf.read()
print type(uf.code)
info=uf.info()
print info.gettype()
print info.getsubtype()
print uf.geturl()
print uf.headers
```

<u>To Find The Duplicate Numbers</u> :-

```
#!/usr/bin/python -tt
def main():
        print 'Calculation Of Duplicate Number'
        string_input=raw_input()
        input_list=string_input.split()
        input_list=[int(x) for x in input_list]
        print input_list
        dup_list=[]
        for var in input_list:
                if input_list.count(var) > 1 :
                        if(var not in dup_list):
                                dup_list.append(var)
        print 'The duplicate entry is',dup_list
if __name__ == '__main__':
main()
```