Python Notes

Basics of Python programming language :

1- print (r "hello world \n hi")
output : hello world \n hi >> this is because using letter "r" before double quotes tells the program not to treat \n as a special command (starting new line)
 So, print("hello world \n hi")
output : hello world
          hi

2- How to add strings
----
x = "muhammed "
x + "khaled"
output : 'muhammed khaled'
 # or to add 2 string we use " ".join((x,y)) (note space string before join function to put a space between the 2 strings we want to join)
Example:-
x = "muhammed"
y = "khaled"
z = " ".join((x,y))
print (z)
output : muhammed khaled

3- We can also multiply strings in python
Example:-
x = "ali "
x * 5
output : 'ali ali ali ali ali'

4- slicing up strings in python
example:
x = "muhammed khaled"
x[3]
output : 'a'

5- to count the string characters from right to left use negative sign and start with -1
x[-1]
output : 'd'

6- We can also slice more than one character
example:-
x ="muhammed khaled"
x[2:7] >> stops right before 7th character which is 6th character
output : 'hamme'
another example:-
x[:4] >> starts from 0th character and stops before 4th character
output : 'muha'
another example:-
x[3:] >> goes to the end and starts at 3th character
output : 'ammed khaled'
another example:-
x[:]
output : 'muhammed khaled'

7- to calculate number of characters in a string we use len function
len ("jsadfhbasdhfvkf")
output : 15
another example:-
x ="lionel messi"
len(x)
output: 12

Comments in python
----
for single line of code, use #
for multiple lines of code, use ''' at the beginning and ''' at the end
'''
....
....
....
'''
or
....

----
to print 2 variables from different data types we use , instead of +
print("muhammed" + "khaled") output : muhammedkhaled
print (9 + 4) output : 13
print ("muhammed" + 9) output : error (You can't add 2 different data types with + sign)
print("muhammed" , 9) output : muhammed 9
we can convert a number to string using str function
example:-
x=3
print(str(x) + "muhammed") output : 3muhammed

8- Lists in python

-We can do operations on elements of a list using function like sum and len for example:
list = [1,4,7,45,88,54,567,3]
avg = sum(list) / len(list) #sum fn is used to add all elements of a list and len fn is used to count the number of the elements in a list
print(avg)
output:
96.125
-you can mix up different data types inside the same list. for example:
items = ['tuna' , 3 , '23 october']
 x = [22,9,7,10,30]
x[2]
output : 7
x[4] = 8
x
output : [22,9,7,10,8] >> New list x
x + [100,23,4]
output : [22,9,7,10,8,100,23,4] >> it's not permanent list, so if u type x and hit enter, the old list will be outputed
x
output : [22,9,7,10,8]
what we can do to assign x + [100,23,4] to a new list (for example list y)
y = x + [100,23,4]
output : [22,9,7,10,8,100,23,4] >> this is permanent

But now if we want to change a list permanently, we will use function called append

```
x.append(120)
x
output : [22,9,7,10,8,120] >> permanent list x
```

----

we can also slice up from lists just like strings with the same rules

----

example on slicing from lists

```
x[:2] = [0,0]
x
output : [0,0,7,10,8,120]
```

9- To remove elements from a list, we assign them to empty square brackets

example:

```
x[:2] = []
x
output : [7,10,8,120]
```

----

to remove the entire list

```
x[:] = []
x
output : [ ]
```

10-If statement

----

example:-

age = 27

if age < 21: (when you hit enter at this point it will go to the next line but shifted by a "tab" which means execute what in the next line if the if statement is true

```
                print("you can have beer!")
```

*the previous example if it's only one condition but if we want more than one we use elif and else in addition to if

example:-

name = "muhammed"

if name is "muhammed": (notice using keyword (is) instead of relational operators with strings)

```
                print("hi muhammed!")
elif name is "ali":
                print("hi ali!")
output : hi muhammed!
```

----

if
elif : as many as we want
else : at the end to make a default choice if all choices are false
So, syntax of if statement in python is

```
if ......... :
                ...................
elif ......... :
                ...................
else :
                ...................
```

Quiz :-

```
1- if none:
     print("hello")
output : nothing will be printed
```

```
2- for char in 'PYTHON STRING':
      if char == ' ':
      break

      print(char, end='')

       if char == 'O':
       continue
output : PYTHON (don't know why)
```

11- For Loop

```
players = ['messi' , 'ronaldo' , 'neymar', 'antoine' , 'ibra' ]
for i in players:  Note : in c++ for(int i=0 ; i <=4 ; i++) ( i is the counter
variable which will go through items in the players list)
                print(i)
```

We can also slice a list during the loop

```
for i in players[:2]:
            print(i)
ouput:
messi
ronaldo
```

12-Range function

----

if we want to loop but not through an existing list, we use range keyword

```
for i in range(10):  ( range 10 means from 0 to 9 )
            print(i)
output:
0
1
2
3
4
5
6
7
8
9
```

```
for x in range(5,12): means from 5 to 11
for x in range(10,40,5): Note >> 10(start) , 40(end) , 5(increament)
            print(x)
output:
10
15
20
25
30
35
```

13- While Loop

----

example:

```
x = 5
```

```
while x < 10:
        print(x)
        x +=1 (if we didn't add this line the loop will run forever)
output:
5
6
7
8
9
```

Note : it's better to use for loop instead of while if you are iterating through a sequence (like: list) as for loop is more pythonic

----

14- Continue and Break

----

example:-

```
magicnumber = 26
for n in range(101):
        if n is magicnumber:
                print(n , "is the magic number!")
                break
        else:
                print(n)
output:
0
.
.
25
26
```

Continue keyword

Continue is used to skip a piece of code and not to execute it

for example:-

```
numberstaken = [10,34,5,35,7]
for n in range(40):
  if n in numberstaken:
    continue
  else:
    print(n)
```

output:

```
0
1
2
3
4
6
8
.
.
.
.
33
36
37
38
39
```

Random Note:
when we do the assignment a = 2, here 2 is an object stored in memory and a is the name we associate it with. We can get the address (in RAM)

of some object through the built-in function, id(). Let's check it.

```
# Note: You may get different value of id


a = 2

print('id(2) =', id(2))


# Output: id(2)= 10919424

print('id(a) =', id(a))
# Output: id(a) = 10919424
```

15- Functions

```
def functionname():
                ...........
                ...........
```

to call a function

functionname()

that's it

for example

```
def print():
                print("my first python function")
print()
```

output: my first python function

----

function with arguments

example:-

```
def area(x,y)
                return x*y
print(area(3,4))
```

output : 12

----

another example in using arguments

```
def BTC_to_USD(btc):
                usd = btc * 527
                print(usd , "dollars")
BTC_to_USD(3.85)
```

output: 2028.95 dollars

----

find factorial of a number using recursion function:-

example:-

```
def factorial(x):
   if x==1:
      return x
   else:
      return x*factorial(x-1)

print(factorial(5))
```

output : 120

----

cool example using functions

```
#this code is used to calculate dating limit age for different men ages
def dating_limit(age):
   girlsage = age/2 + 7
   return girlsage
listofages = [50,21,30,43,15]
for i in listofages:
   print(i , "years old")
   print("girls dating limit age is" , dating_limit(i))
```

Output:

```
50 years old
girls dating limit age is 32.0
21 years old
girls dating limit age is 17.5
30 years old
girls dating limit age is 22.0
43 years old
girls dating limit age is 28.5
15 years old
girls dating limit age is 14.5
```

----

example of using default arguments

```
def gender(sex = 'unknown'):
   if sex == 'm':
      sex = 'Male'
   elif sex == 'f':
      sex ='Female'
   print(sex)
gender('f')
gender('m')
gender()
```

Output:

```
Female
Male
unknown
```

16-Variable Scope

----

any defined variable inside a function can't be used anywhere else in the program and it's called local variable but if any variable is defined outside any function and above it, the function can access it and it's called global variable

----
--

Example on using default arguments

```
def eating(name = "muhammed" , action = "loves" , food = "pizza"):
   print(name,action,food)
eating()
eating("menna" , "loves" , "chocolate")
eating(food="crepe")
eating(action = "hates")
eating(name = "mohab")
```

Output:

```
muhammed loves pizza
menna loves chocolate
muhammed loves crepe
muhammed hates pizza
mohab loves pizza
```

Note : we can also enter arguments at any order we want in case we write the name of the variable like last 3 cases in above example

----
--

17-Flexible number of Arguments

We can also bulid a function that can take in number of arguments

Example:-

```
def add_numbers(*args):
   total =0
   for i in args:
      total +=i
   print(total)

add_numbers(3,4,63,75,23,6,4,24,67,333,56,87,33)
```

Output : 778

18-Unpacking Arguments

```
def health(age , apples , cigs):
   lifeexpectancy = (100-age) + (apples *3) - (cigs * 2)
   print(lifeexpectancy)
muhammedinfo = [21,20,0]
health(muhammedinfo[0] , muhammedinfo[1],muhammedinfo[2])
Output : 139
```

But, if we have a lot of people who want to test their life expectancy, it will take too long to enter arguments in the function call each time so we can use a simple trick here.

```
 health(muhammedinfo[0] , muhammedinfo[1],muhammedinfo[2])
**OR** health(*muhammedinfo)
```

So we can do this

```
def health(age , apples , cigs):
   lifeexpectancy = (100-age) + (apples *3) - (cigs * 2)
   print(lifeexpectancy)
muhammedinfo = [21,20,0]
health(*muhammedinfo)
Output : 139
```


19-Sets

```
footballers = {'messi' , 'ronaldo','kaka','robben','iniesta','kaka'}
print(footballers)
Output : {'messi', 'ronaldo', 'iniesta', 'kaka', 'robben'}
```

2 notes in the output :

A) the output is not the same order of the set footballers, Because sets don't care about the order of elements in it.

B) if there's an element which is repeated in the set, it doesn't appear more than one time in output (kaka in this set).

20-Dictionary

Dictionary in real world consists of words and definitions of these words.

Dictionary in python consists of keys(words) and values(definitions).

example:-

```
classmates = {'tony' : 'cool guy but nosey' , 'emma' : 'sits behind me' ,
```

'david' : 'loves football and pizza' } #this how to create a dictionary with keys and values.
print(classmates['tony']) #output will be : cool guy but nosey) we asked for the meaning of word(key) which is tony and
output will be the definition(value) of tony which is cool guy but nosey.

Now, we can loop through our dictionary instead of repeat the same line of code many times with different keys
#we will need 2 counters in for loop for keys and values (k for keys and v for values)
example:-

```
for k , v in classmates.items(): #note how we loop through a dictionary called classmates, we use a function called items for it.
    print(k , ":" , v)
```

output:

```
tony : cool guy but nosey
emma : sits behind me
david : loves football and pizza
```

21-Modules

If you want to use a function you created in different program or a set of functions, You can throw these functions in a file, then include that file in any other
program to use these functions anytime instead of recreate them again.
steps to use modules
1-create a new file and write the function that you are going to use it over and over again in different programs
2-go to the program you are working on and in the first line of code use keyword (import) to import the module in which the function we want is there
example:-

```
import yourfilename
yourfilename.yourfunction()
```

and now you did called a function in a module outside the program you are working on instead of creating the same function.

----

some notes on modules
1-modules are like libraries but libraries are bigger
2- there are many predefined modules that you can use like random module which has a lot of functions that we can use
3-you can download modules from the internet and use them in your program

----

random module

```
import random
x = random.randrange(1,1000)
print(x)
output : a random number between 1 and 1000
```

Notes on Modules

1- If you want to download modules from the web go to file > settings > project interpreter
2-urllib.request module defines functions and classes which help in opening URLs (mostly HTTP) in a complex world —
basic and digest authentication, redirections, cookies and more.

22-Download an image from the internet using a function

----

example:-

```
def download_image(url):
   name = "rubik's cube.jpg"
   #now we want to use a function in urllib.request module to download the image and this function takes 2 arguments (urlretrieve(url,image name))
   urllib.request.urlretrieve(url , name)
download_image("http://www.hbc333.com/data/out/193/47081647-random-picture.png")
```

and BOOOOOOOOOOOOOOOOM,the image is added to a new file

Files in Python

- File is a named location on disk to store related information. It is used to permanently store data in a non-volatile memory (e.g. hard disk).
- Since, random access memory (RAM) is volatile which loses its data when computer is turned off, we use files for future use of the data.
- When we want to read from or write to a file we need to open it first. When we are done, it needs to be closed,
 so that resources that are tied with the file are freed.
- Hence, in Python, a file operation takes place in the following order.
   1-Open a file
   2-Read or write (perform operation)
   3-Close the file
- >>> f = open("test.txt")   # open file in current directory
>>> f = open("C:/Python33/README.txt") # specifying full path where f is the file object.
- We can specify the mode while opening a file. In mode, we specify whether we want to read 'r', write 'w' or append 'a' to the file.
We also specify if we want to open the file in text mode or binary mode.
The default is reading in text mode. In this mode, we get strings when reading from the file.
On the other hand, binary mode returns bytes and this is the mode to be used when dealing with non-text files like image or exe files.
- We need to be careful with the 'w' mode as it will overwrite into the file if it already exists. All previous data are erased, so we use 'a' instead of 'w'
to append to existing data.

23- How to read and write files:

- To create a file, first we need to create a file object
so to create a file, just do the following line of code
fw = open('filename.txt' , 'w') # fw is the file object which you name whatever you want and assign it to a function called open and name your file with an extension .txt
and the other argument in open function which has value w means

write to the file
Now we are ready to write to the file
we will use the file object we created with name fw with different functions to do with the file

```
fw.write('writing some shit to my first python file \n')
fw.write('my second line in a file ')
```

and we can write whatever we want in our file using write function after our file object.
Now after we are done with our file, we want to close it so we will use close function

```
fw.close()
```

----
we write to a file in above section, Now we want to read from a file, How?
we will need a new file object, let's name it fr refers to file read like fw: file write

```
fr = open('sample.txt' , 'r')
```

to read some data from a file we need to store this data in a variable because in python you can't deal with data inside a file directly
so you need to assign it to variables to deal with it
example:

```
text = fr.read()
```

this line of code will read the entire data in the file and assign it to variable named text
Now we can manipulate the data assigned to variable text and do whatever we want with it like just print it out

```
print(text)
```

let's write the full source code to create,write,read, and close a file

```
'''
fw = open('sample.txt' , 'w')
fw.write('writing some shit to my new text file using python language\n')
fw.write('i love this fuckin\' awesome language')
fw.close()
fr = open('sample.txt' ,'r')
text = fr.read()
print(text)
fr.close()
'''
```

We can read a file line-by-line using a for loop. This is both efficient and fast.

f is file object

```
>>> for line in f:
...     print(line, end = '')
...
This is my first file
This file
contains three lines
```

24- How to read from text file on the internet after downloading it

----

first of all we need to search the internet for a file let's say .csv format and copy its url

assign this url to a variable

x =
'http://real-chart.finance.yahoo.com/table.csv?s=GOOG&d=2&e=8&f=2015&g=d&a=2&b=27&c=2014&ignore=.csv'

then we want to make a function to download this file like we created a function before to download an image

----

Steps to manipulate data from a file on the internet

1.) define a function which takes LINK ADDRESS(url) as parameter

2.) we use module 'urllib' which is divided into parts-->request,error,parse so, 'from urllib import request' or 'import urllib.request'

3.) now we use 'urlopen(url)' to set up a connection just like we used 'urlretrieve()'while downloading image "to retrieve data".

4.)The connection to that webpage will be stored in 'response'

5.) response[establishes the connection].[this dot says to do] read()[reading] -> response.read()

6.) Since we dont know which format it is in,we convert everything into a string using--> str()

7.)After conversion now everything will be single line with \n like--> blah blah blah \n blah blah blah \n blah

8.) Now we split them at \n escaping their special meaning ->

split("\\n") and store it in a list(lines) lines = ["blah blah blah" , "Blah blah blah ", "blah"]//something like this
9.) Now we have to write it onto a file.
10.)using loop copy all the lines.
11.) Now we need actual '\n'(new line after writing each line from 'lines') so we use -> (line + '\n')
12.) fw.close()
13.) Call the function

Source code for previous section steps:

```
from urllib import request # or import urllib.request

goog_url =
"http://real-chart.finance.yahoo.com/table.csv?s=GOOG&d=2&e=8&f=2015&g=d&a=2&b=27&c=2014&ignore=.csv"

def download_stock_data(csv_url):
  response = request.urlopen(csv_url)
  csv = response.read()
  csv_str = str(csv)
  lines = csv_str.split("\\n")
  dest_url = r'goog.csv'
  fx = open(dest_url, "w")
  for line in lines:
     fx.write(line + "\n")
  fx.close()

download_stock_data(goog_url)
```

25- Difference between syntax error and exception:-
----
syntax error : is an erroe in the programmer syntax
Exception happens when the user enters wrong input like if i asked the user about his age and i made a placeholder of type integer to store the age
the user will enter, But the user entered characters instead of his age since he's an idiot, the program my crash and we want to avoid that

So there're 2 ways to ask the user for input, both are correct but one of them is better to handle exception problem

1st way

```
x = int(input("what's your age? \n"))
print(x)
```

output: what's your age?

user inputs 20 for example the program will work perfectly

user inputs some crap that's not integer the program will crash and shows value error whic type of exception problem

----

2nd way (better one)

```
while True:
   try:
      x = int(input("what's your age, buddy?\n"))
      print(18/x)
      break
   except ValueError:
      print("make sure you entered your age, idiot!")
   except ZeroDivision Error:
      print("zero is not allowed to be entered, try again another number!")
```

this code will repeat the question to the user if he doesn't enter his age as an integer instead of making the program crash.

Types of Exception:-

1-Value Error is one of exception types and happen when the user enters data that's not convertable to integer or data type
you defined for the entered data
2-ZeroDivision Error is when the user inputs zero and operation in his inputs contains dividing by zero like the code above if user
enters zero it will be 18/0 so it's zerodivison error so we add another except condition

----

26-Classes and Objects:-

We can think of class as a sketch (prototype) of a house. It contains all the details about the floors, doors, windows etc.
Based on these descriptions we build the house. House is the object. As, many houses can be made from a description,

we can create many objects from a class. An object is also called an instance of a class and
the process of creating this object is called instantiation.
EXAMPLE:-

```
class ENEMY:
    life = 4

   def attack(self):
      print("ouch!")
      self.life -=1

   def checklife(self):
      if self.life <=0:
         print("i'm dead")
      else:
         print(str(self.life) + " life left")
         -------------
```
Now we made our class then we want to deal with functions inside it by creating an object called for example enemy1
```
enemy1 = ENEMY()
enemy1.attack()
output: ouch!
enemy1.checklife()
output: based on conditions
```
----
each object is a copy of the class and it's independent from other objects, which means if we created
a new object called enemy2 and we attack enemy1, enemy2 won't be affected
example based on class above
```
enemy1 = ENEMY()
enemy2 = ENEMY()
#we created 2 objects enemy1 and enemy2
enemy1.attack()
enemy1.attack()
enemy1.checklife()
enemy2.checklife()
```

```
output:
ouch!
ouch!
2 life left
4 life left  #notice here enemy2 still has 4 lifes (not affected by
attacking enemy1)
```

27-Init(constructor in C++)

```
class Tuna:
   def __init__(self): #this how to define an init in python 2 underscores
before init and 2 underscores after init
         print("blablabla")

   def swim(self):
      print("i'm swimming")


tuna1 = Tuna()
tuna1.swim()

output:
blablabla
i'm swimming
```

----

example on classes with non empty init

```
class Enemy:
   def __init__(self , x):
      self.energy = x
   def getenergy(self):
      print(self.energy)

hitler = Enemy(100)
mosoleni = Enemy(90)
hitler.getenergy()
mosoleni.getenergy()
```

output:
```
100
90
```

28-Class variables VS Instance variables

Class variables : any time you create an object of type class x for example, it will have the variables in the class by default
for example:
```
class girl:
    gender ='female'
#so if we created object called girl1 for example,by default it will be female
```
so any object(girl) is a female

Instance variable:
```
class girl:
    gender ='female'

    def __init__(self , name):
          self.name  = name
#here variable called name will have different value for different objects(girls)
```

Example on class and instance variables:

```
class Girl:

   gender = 'female'

   def __init__(self, name):
      self.name = name

r = Girl('Rachel')
s = Girl('Stanky')
print(r.gender)
print(s.gender)
print(r.name)
print(s.name)
```

output:

```
female
female
Rachel
Stanky
```

29- Inheritance:

```
class parent():
   def lastname(self):
      print("khaled")

class child(parent):
   def firstname(self):
      print("muhammed")


ob = child()
ob.firstname()
ob.lastname()
```

output:

```
muhammed
khaled
```

Note: the child class can overwrite a function from the parent class, How?

```
class child(parent):
   def firstname(self):
      print("muhammed")
   def lastname(self):
      print("elsisy")
```

output:

```
muhammed
```

elsisy
So child class can't just inherit but it can also overwrite what it inherited from the parent class
----

30-Multiple Inheritance:-

Now, We can also inherit not just from only one class but we can inherit from more than one class in our baby class :D
example:

```
class Mario():

   def move(self):
      print('I am moving!')


class Shroom():

   def eat_shroom(self):
      print('Now I am big!')


class BigMario(Mario, Shroom):
   pass

bm = BigMario()
bm.move()
bm.eat_shroom()
```

output:

```
I am moving!
Now I am big!
```

31- Threading

```
import threading

class BuckysMessenger(threading.Thread):
  def run(self):
    for _ in range(10):
      print(threading.currentThread().getName())

x = BuckysMessenger(name='Send out messages')
y = BuckysMessenger(name='Receive messages')
x.start()
y.start()
```

output:

```
Send out messages
Send out messages
Send out messages
Send out messages
Receive messages
Send out messages
Receive messages
Send out messages
Receive messages
Send out messages
Receive messages
Send out messages
Receive messages
Send out messages
Receive messages
Send out messages
Receive messages
Receive messages
Receive messages
Receive messages
```

32- unpack list or tuples:-

----

unpack list means to make elements of this list seperated in a single variable each
for example:

```
def first_last(grades):
  first , *middle , last = grades #here we have a list named grades with x
```

number of elements and we unpacked this list to 2 variables and a list called *middle

```
  avg = sum(middle) / len(middle)
  print(avg)

x = [6,7,9,4,7,0,5,3,5,7]
first_last(x)
first_last([45,33,66,88,33,55,2,7,33])
```

33-Zip function:

----

zip function is used to tie up two lists together for example:

```
first = ['Bucky', 'Tom', 'Lionel']
last = ['Roberts', 'Hanks', 'Messi']

names = zip(first, last)
#names will be like this >> [('Bucky' , 'Roberts') ,
('Tom','Hanks'),('Lionel','Messi')]
for a, b in names:
print(a, b)
```

output:

```
Bucky Roberts
Tom Hanks
Lionel Messi
```

34-lambda function

```
answer = lambda x: x*7
print(answer(7))
```

output:

```
49
```

35-Max and Min and sorting dictionaries:

----

first of all in dictionary keys is a list and values is another list

```
menu = {'fish' : 50 , 'chicken' : 30 , 'meat' : 55 , 'chocolate' : 38 ,
'cheesecake' : 25}
print(menu.keys())
output :
dict_keys(['fish', 'chicken', 'meat', 'chocolate', 'cheesecake'])
print(menu.values())
output:
dict_values([50, 30, 55, 38, 25])
```

----

to get max or min or to sort a dictionary, we use zip function to tie up keys and values in a zipped list first

```
zip(menu.values() , menu.keys())
```

if we write values list first and keys second, sorting and finding max and min will be based on values list, and vice versa.

example on max,min,sorted a dictionary

```
menu = {'fish' : 50 , 'chicken' : 30 , 'meat' : 55 , 'chocolate' : 38 ,
'cheesecake' : 25}
print(min(zip(menu.values() , menu.keys())))
print(max(zip(menu.values() , menu.keys())))
print(sorted(zip(menu.values() , menu.keys())))
output:
(25, 'cheesecake')
(55, 'meat')
[(25, 'cheesecake'), (30, 'chicken'), (38, 'chocolate'), (50, 'fish'), (55,
'meat')]
```

36-Structs:

Struct in python is when you take in type of data whether it's number,lists,strings or anything, and convert it into bytes format
Note : to determine the size of a data type (in C++ we used sizeof function but in python we use calcsize function)
example

```
from struct import *
print(calcsize('i')) # integer
print(calcsize('f')) # float
print(calcsize('d')) # double
print(calcsize('iiiii')) # 5 integers
```

output:
```
4
4
8
20
```

```
from struct import *
packed_data = pack('iif' , 6 ,4, 3.4)  #pack function used to store data in bytes format
print(packed_data)
original_data = unpack('iif' , packed_data) #unpack function is used to get back the original data as 2 integers and 1 float
print(original_data)
```
 output:
```
b'\x06\x00\x00\x00\x04\x00\x00\x00\x9a\x99Y@'  #our numbers in bytes format
(6, 4, 3.4000000953674316)
```

37-Map function in python

Map function is used to apply a function through each element of a list directly without the need of for loop and
it takes parameters whic are function name and list name
for example:
```
income = [10,34,20,44]
def double_income(dollars):
   return dollars *2

x = list(map(double_income , income))  #list function is used to put results in a new list called x
print(x)
```
output:
```
[20,68,40,88]
```

38-Bitwise Operators:

AND Gate
```
a = 50      #1 1 0 0 1 0
b = 25      #0 1 1 0 0 1
```

```
c = a & b  #0 1 0 0 0 0
Binary Right shift
x = 240       #11110000
y = x >>2     #00111100    #shift x by 2 bits at  a time to the right
print(y)
output : 60
```

39-Finding Largest or Smallest Items:

```
import heapq
grades = [133,556,33,678,444,744,754,335,843]
print(heapq.nlargest(n, listname))
# n  : if we want largest 2 numbers for example n =2 and so on
print(heapq.nlargest(3,grades))
output:
[843, 754, 744]
print(heapq.nsmallest(3,grades))
output:
[33,133,335]
```

