PYTHON FOR DATA SCIENCE — NOTES

BY SWOPNIM GHIMIRE

Comprehensive handwritten notes from my completed Python for Data Sclence course.

By Swopnim Ghimire

Python for Data Science

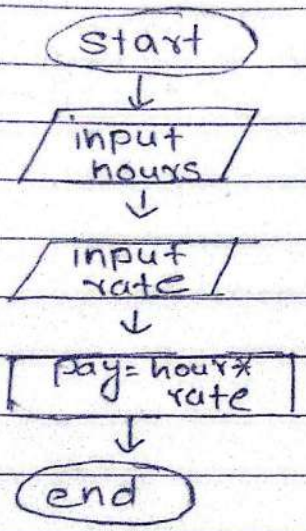
Barthwaj & Sanjay
Date:
Page:

focus: Problem Solving
Why python
Python (Zero to Detailed)
Data Science.

Problem Solving: finding Generic solution to complex problem.

Algorithm is step by step solution to a problem.

flowcharts



Pseudo Code

```
BEGIN
  input hour
  input rate
  pay = hour * rate
  print pay
END
```

* Pseudocode for making tea

```
PROGRAM make tea

put teabag in a cup
WHILE (water not boiling)
    boil water
ENDWHILE
Pour water in cup →  WHILE (Sugar needed)
                        add sugar
                     ENDWHILE
                     Serve and finish
```

(*) Pseudo code fror finding minimum number from a list of number.

Eg:- list l=[23,-4,0,73,-10,13]

• Search Minimum

```
searchMinfromlist(L,n)
    minValue = L[1] → assignment
    counter = 2
    while (counter<=n)
        V=L[counter]
        if (V<minValue)
            minValue = V & increament counter
        else
            pass
        endIf
        increament counter
    endWhile
    return minValue
endsearchMinfromlist
```

(*) Pseudo code for finding minimum number and the index of that number.

first visualize list L=[1,2,3,4,5]

- Search Minimum and it's index.

```
searchMinfromlist (L,n)
    minValue = L[1]
    Idx = 1
    counter = 2
    while (counter <= n)
        v = L[counter]
        if (v < minValue)
                minValue = v
                Idx = counter
        else
            pass
        endIf
        counter = counter + 1
    endwhile
    return minValue, idx
endsearchMinfromlist
```

Sarthaq & Sarjaq
Date:
Page:

(8) Pseudo code for sorting algorithm in asending order.

- Selection sort

```
sortlist (L,n)
L2 = [] → This is supposed to be the newlist
counter = 1
while (counter <= n)
    min value idx = searchMinfromlist (L,n)
    insertinL2 = min
    delete L[idx]
    n = n-1
endwhile
    return L2
endsortlist

searchMinfromList (L,n)
    minValue = L[1]
    idx = 1
    counter = 2
    while (counter <= n)
        V = L[counter]
        if (V < minvalue)
            minvalue = V
            idx = counter
        else
            pass
        end if
        counter = counter + 1
endwhile
    return minValue, idx
endsearchMinfromList.
```

↓ Here we are using the algorithm which we created previously.

Converting Pscudo code into python code

- Selection sort

```
def sortlist(L,n):
    L2=[]
    counter=0
    while(counter<=n):
        minvalue, idx= searchMinfromList(L,n)
        L2.append(minvalue)
        del L[idx]
        n=n-1
    return L2
```

used for defining a variable.

```
def searchMinfromList(L,n):
    minvalue=L[0]
    idx=0
    counter=1
    while(counter<n)
        V=L[counter]
        if(V<minvalue):
            minvalue=V
            idx=counter  → add counter too
        else:
            pass
    return minvalue, idx
```

Sarthaq & Sarjaq
Date:
Page:

⊛ Why chose Python for data Science.

Ans: It's Beginner friendly, versatile & flexable, Most powerful in Machine learning world and Most mature libraries around.

IDE (Integrated Development Environment)

Jupyter is the most popular IDE for Data science.
Download anaconda IDE and enter jupyter notebook to set up working environment.

You can just use ipython in annaconda to set up working environment too.

VARIABLE & OPERATOR

Variable is of different type like int, float and the variable is assigned a value.
They should be written in a descriptive way.

Multiple assignment a, b = 4, 5.0
(4 → int, 5.0 → float)

Declaring variable
eg:
x = 3 (%whos) tells abt all the variable in memory.
print(type(x))
result: <class 'int'>

Multiple assignment

```
a,b,c = 3, 4.5, 5
%whos (result: Variable type Data/Info)
a       int     3
b       float   4.5
c       int     5
```

c = 2+4j is complex data type.

del c is used to delete a variable here it's c.

Arithmetic operators

+ → Addition - = substraction
/ → division x %y = mod
* → Maultiplication // floor division ie 3.33=3
** → to the power of 2**3 = 8

These can be applied to different data type.

Variable cannot be declared using number and special symbols in the initial position but _a is a proper variable.

Type bool and comparisons.

==	True, if it is equal
!=	fo True, if not equal.
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to

Bool is a valid data type and it can be either true or false.

If we use and & or gate in the same line without bracket then and will execute first.

eg: True or false and false
result: True

find result of print((not(2!=3) & True) or (false and True))

result: False

*** Some useful function in python.

a) round() function rounds the input value to a specified number of places or to the nearest integer.

print(round(4.556)) res: 5
print(round(4.335)) res: 4
print(round(4.556789,2)) res: 4.56
print(round(4.556789,3)) res: 4.557

b) divmod(x,y) output the quotient and the remider in a tuple
(we will see what a tuple is)

divmod(27,5)
Quotient=5 reminder=2 (5,2) → tuple

print(divmod(34,9)) result(3,7)

G = divmore (24,5)
print (G) ——→ result (4,4)
type (G) → tuple Quotient → Remindey
G[0] is 4 ÷ Q
G[1] is 4 ÷ R

e) isinstance() returns True, if the first argument is an instance of that class. Multiple classes can also be checked at once.
eg:-
print(isinstance(1,int)) res:- True
print(isinstance(1.0,int)) res:- false
print(isinstance(1.5,float)) res:- True
print(isinstance(1.67,(int,float))) res:- True
print(isinstance(2+3j, complex)) res:- True

d) pow(x,y,z) x raised to the power y and reminder by z
ie $x^y \% z$
for just power use x**y, pow(x,y)
eg:-
pow(2,3,4) res:- $2^3 \% 4 = 0$

e) Input() a = input("Enter Something")
default value of a becomes string.
to convert a to int;
a = int(a)
type(a) result is integer.
or
a = int(input("Enter a number"))
but if user enters something else than no it shows error.

(*) If you know the function but aren't aware of the exact syntax then you can use functionname? or help(functionname)

eg: pow? help(pow)
input? help(input)

CONTROL FLOW
(IF-Elif-Else)

```
a=int(input())
b=int(input())
```

Task:
print only the greater number.

instead of task use if b>a

eg:

```
a = int(input())
b=int(input())
 if a>b:
   print(a)
   print("a is the greatesnumber")
 if b>a:
   print(b)
   print("b is the greater no")
```

→ use can use two if or else

```
else:
   print(b)
   print("b is greater than a")
```

eg:

```
a = 10
b = 15
if (a == b):
    print ("Equal")
elif else if (a > b):
    print ("a is greater")
else
    print ("b is greater")
```

* A grading system.

```
a = int (input ("Enter your Marks"))
if (a >= 85):
    print ("A Grade")
elif (a < 85) and (a >= 80):
    print ("A- Grade")
elif (a < 80) and (a >= 75):
    print ("B grade")
else (a < 75) and (a >= 70):
    print ("B- grade)
else
    print ("Below average")
```

instead of else

```
tet a = 10
if (a > 9):
    print ("A is greater than 9")
elif not (a > 10):
    print ("Else part")
```

(Nested If)

eg:-

```
x = int(input("Enter a number"))
if x>10:
    print("Your no is greater than 10")
    if x>20:
        print(" and also above 20 ")
    else:
        print(" but not above 20")
print("outside the loop")
```

In python Indentation matters very much.

#comment → single line comment

''' ''' → Multi line comment.

User will enter a no 238.915. Your task is to find out the integer portion before the point (in this case 238) and check if it's even no or odd no

```
a = 238.915
a = int(a)
if (a%2==0):
    print("a is even number")
else:
    print("a is odd number")
```

The question specifies just to take the integer portion so, if we use floor then the value will get rounded off so we can't use it.

Sarthaq & Sarjaq
Date:
Page:

Control flow (loops)

```
n = int(input('Max iterations: '))
i = 1
while (i <= n):
    print(i)
    i = i+1 (i += 1)
print('done')
```

printing natural no upto n

* Print all ~~even~~ natural number upto n.

```
n = int(input("Max iteration: "))
i = 1
while (i <= n)
    if (i % 2 == 0)
        print(i)
        print("is even number")
    else
        pass → used to say program to goon.
    i += 1 (i = i+1)
print("done")
```

Statements:

break inside a loop exists loop immediately & brings you outside the loop.

continue brings the next iteration loaded regardless of the next statement of the loop.

Sarthaq & Sarjaq
Date:
Page:

Eg:

```
i=1
while True:
    if i%.17==0:
        print('break')
        break
    else:
        i+=1
        continue
    print('I am inside the loop')
print('done')
```

Current flow (for loop)

eg: `l=[]` ← i value starts from 0 to 9

```
for i in range(10):
    print(i+1)
    l.append(i**2)
```

result:

l=[0,1,4,9,16,25,36,49,64,81]
if we print(l)

range(0, 10, 2) end point is not included.
start end Jump

eg:

```
for i in range(1,20,3):
    print(i+1)
    l.append(i**2)
```

result

1, 4, 7, 10, 13, 16, 19

l=[1, 16, 49, 100, 169, 256, 361]

No Copy

Control flow (else in for loops)

eg:

```
S = {"apple", 4.9, "cherry"}
for x in s:
    print(x)
else:
    print('loop completes its iterations')
print ("outside the loop")
```

result:

```
apple
4.9
cherry
loop completes with iteration
outside the loop
```

Case where this else doesn't execute.

```
i = 1
for x in s:
    print(x)
    i = i+1
        if (i==3):
            break
        else:
            pass
else:
    print("for loop executed successfully")
print("outside the loop")
```

result:

```
apple
4.9
outside the loop.
```

Sarthaq & Sanjay
Date
Page

Control flow (Exploring a Dictionary)

```
D={"apple":44,"cheery":"game"}
for x in D:
    print(x, D[x])
```

Result:

```
apple 44
cheery game
```

another eg:

```
D={"A":10,"B":-19,"C":"abc"}
for x in D:
    Print(x, D[x])
```

Result:

```
A 10
B -19
C abc
```

Sarthaq & Sarjaq
Date:
Page:

Q Given a list of numbers ie l=[1,2,4,-5,7,9,3,2], make another collist that contains all the items in sorted order from min to maximum. ie your result will be l2=[-5,1,2,2,3,4,7,9]

```
l=[1,2,4,-5,7,9,3,2]
for j in range(len(l))
        m=l[j]
        idx=j
        c=j
for i in range(j,len(l)):
    if l[i]<m:
        m=[[i]
        idx=c
        c+=1
    temp=l[j]
    l[j]=m
    l[idx]=temp
    print(l)
```

Q finding min value.

```
l=[1,2,4,-5,7,9,3,2]
m=l[0]
for i in l:
    if i<m:
        m=i
print(m)
```

Q for finding index

```
l=[1,2,3]
m=l[0]
idx=0
c=0
for i in l:
    if i<m:
        m=i
        c+=1idx=c
    c+=1
print(m,idx)
```

HD Copy

Sarthaq & Sarjaq
Date:
Page:

Functions

```
def printSuccess():
    print("The task was successful")
    print("Moving to the next text")
    print("Send me the next text")
```

This is the initial phase. of defining a function then we need to call it to execute the funation.

```
printSuccess()
```

function (Doc string) is on way of describing the function functionality.

```
def printsuccess2():
    """This function is just used to
    print the message hellow"""
    print("hellow")
```

PrintSuccess2 ?
You'll get details in Docstring

functions (input arguments)

```
def printmessage(msg):
    print(msg)
```

function is dynamic it can be uset to print anything.

eg:
```
def printMsg(msg):
    if isinstance(msg, string):
        print(msg)
    else:
        print("Your msg isn't string")
        print("Your msg datatype is",
              type(msg))
```

Eg:
```
printMsg("Hello world")
```
Result:

Hello world

```
PrintMsg(23)
```
Result:

Your msg isn't string
Your msg data type is: <class 'int'>

function (input arguments)

```
def mypow(a,b):
    """This function computes power just like
    builtin power function"""
    c = a**b
    print(c)
```

Input:

mypow(3,4) result: 81

```
def checkArgs (a,b,c):
    if isinstance (a,(int,float)) and
       isinstance (b,(int,float)) and
       isinstance (c,(int,float))
            print((a+b+c)** 2)

    else:
        print ("Error: Argument
                passed is Invalid")
```

input checkArgs(3,4,5) result:- 144

input checkArgs(3,4,"g") result:- Error

functions (Order of input arguments)

```
def f C.(2,(1,(3):
    print ((1,(2,(3)
```

f(2,3,"game") result 3,2,"game"

```
def f (a,b,c):
    print ("A is: ", a)
    print ("B is: ", b)
    print ("C is: ", c)
```

Input f (3,2,"bird" result:-

A is : 3
B is : 2
C is : bird

functions (variables inside)

```
def add(x,y):
    c=x+y  # I need value of c?
```

Input # The value of c is available just within the function so we use return

```
def add(x,y):
    return x+y
```

so, if we use `d=add(2,3)`
`print(d)`
it can print the sum but `c=x+y` cannot print the sum as the value of the variable is just available within a function.

If you return some value out of a function then it returns the value but if you don't pass any value then calling it prints 'none' and it's datatype becomes 'NoneType'.

Return can also be used as a break function code below doesn't run:

```
def r():
    a=5  b=7  d="something"
    return a,b,d
x,y,z=r()
print(x,y,z)
```

result: 5 7 something

Sarthaq & Sarjaq
Date:
Page:

functions (Variable number of input arguments)

```
def add (*args):
    sum = 0
    for i in range (len(args)):
        sum += args[i]
    return sum
```

add(3,4,5,6,7) result: 25

Dictionary

```
def f(**c):
    for x in c:
        print (x, c[x])
```

f(c1="A", c2="B", c3="C")

Eg:

```
def printAllVariableNamesAndValues (**args):
    for x in args:
        print ("Variable Name is:", x,
               "And Value is:", args[x])
```

printAllVariableNamesAndValues (a=3, b="B", c="CCC", y=6.7)

result:

Variable Name is: a And Value is: 3
" " : b " " " : B
" " : c " " " : CCC
" " : y " " " : 6.7

function (Default values)

```
def f(sum=0):
    print(sum)
```

f(3) returns 3 f() returns 0

If l=[1,2,3] and l2=l Then in python they don't share different memory location, it's just another variable name for the same memory location.

def f(l=[]) so at the time of every compilation default value is assigned and the default value doesnot changes.

Eg:
```
def gg(s=4):
    print(s)
input gg() res: 4
input gg(56) res: 56
```

Eg:
```
l=[1,2,3]
l2=l
l2[0]=-9
print(l)
```
result:
[-9,2,3]

Eg:
```
def ff(l=[1,2]):
    for i in l:
        print(i)
l2=[12,3,4]
ff()  res: 1 2
ff(l2) res: 12 3 4
```

Note: Even after multiple manupulation of data the default value remains same.

Sarthaq & Sarjaq
Date:
Page:

Modules

Modules is a python file that contains function that you want to use in several coding projects or the functions that you don't want to write.

D:/mymodules/myfuncs.py

```
def printMe (msg='No message
        was supplied'):
    print (msg)

def printlist (l=[]):
    for x in l:
        print(x)
```

```
import sys
sys.path.append ('D:/mymodules/')
import myfuncs as f
f.printMe ('hellow')
```

Create a file in jupyter named anything eg:- name be my_universal_functions.

```
def checkType def checkInfoNumeric(**args):
                .outputvalue=True
                    for x in args:
                        if isinstance
                        (x,(int,float
```

Download this file as .py file.

HD Copy

Sarthaq & Sarjaq
Date:
Page:

```
def checkIfnotNumeric(*args):
    for x in args:
        if not (isinstance(x,(float,int))):
            return false
    return True

def addAllNumerics(*args):
    s=0
    for x in args:
        s+=x
    return s

myName = "Python Course"
```

After downloading the file create a folder in a drive and paste the downloaded file there. eg:
creating a folder in win C named ABC and then pasting the file there.
Now,
in jupyter;

```
import sys
sys.path.append('C:/ABC/')

import my_universal_functions as myfs.
myfs.addAllNumerics(2,3,4,6)
.myfs.checkIfnotNumeric(1,3,A)
myfs.checkIfnotNumeric(1,2,3)
myfs.myname
```

result: ~~myfs.~~
15 false True 'Python Course'

HD Copy

Sarthaq & Sarjan
Date:
Page:

If you have to import a part of the function and give it a new name then we use something like

```
import sys
sys.path.append('C:/ABC/')

from my_universal_function import
    addAllNumerics as sum.

Sum(3,4,5,6)  result: 18
```

Practice function

Q """ Give a list of no ie [1,2,4,-5,7] make another list that contains all the items in sorted order from min to max. ie your result will be another list ie [-5,1,2,4,7] """

```
def findMin(L):
    m=[L[0]]
    idx=0
    counter=0
    for x in L:
        if x<m  if counter<m
                    m=counter
                    idx=
```

```
def findMin():
    m=L[0]
    idx=0
    i=0
    for x in L:
        if x<m:
            m=x
            idx=i
        else:
            pass
        i+=1
    return m, idx
```

This function defines minimum for a single data it find the lowest value and return it with its index. so, everytime same value is returned.

eg:-

```
a,b=findMin([2,3,4,0,9])
print(a,b)
```

result 0,3

```
def swapvalues (L, idx1, idx2):
    tmp=L[idx1]
    L[idx1]=L[idx2]
    L[idx2]=tmp
    return L

L=[2,3,6,7]
L2=swapvalues(L,1,3)
print(L2)
```

result:-

[2,7,6,3]

Sarthaq & Sarjan
Date:
Page:

function to find different min in diff cases.

```
def findMin(L, startIndx):
    m = L[startIndx]
    idx = startIndx
    for i in range(startIndx, len(L)):
        x = L[i]
        if x < m:
            m = x
            idx = i
        else:
            pass
    return m, idx

def checkIfNotNumeric2(L):
    for x in L:
        if not(isinstance(x,(int,float))):
            return False
    return True

def sortList(L):
    if not(checkIfNotNumeric2(L)):
        print("Error: No numeric val")
        return
    else:
        c = 0
        for x in L:
            m, idx = findMin(L, c)
            c += 1
    return L
```

Ho Copy

Input :
```
l2 = sortlist([2,3,6,5,4,-1])
print(l2)
```
result :
```
[-1,2,3,4,5,6]
```

DATA STRUCTRUE IN PYTHON

STRING

"python" & 'python' both are valid string.

```
a = "Hello"
b = "world"
c = a + " " + b
d = a + "messed up" + b
print(c)  res: Hello World
print(d)  res: Hellomessedup world.
```

This is called concatination of string.

You can convert a integer to string like this :

```
a = 15
a = str(a)
print(type(a))  res: str
```

you can not concat an integer with a string you need to ~~concat~~ convert the int into string to concat.

String (Multiline String)

```
multilinestring = """This is a very
                     Good course
                     That Iam learning
                     from"""
print(multilinestring).
```

~~result: This is avery Good course that Iam learning from.~~

result:
This is a very
Good course
That Iam learning
from.

```
print("""Hey mate!
         Hi buddy,
         Hope you're okay""")
```

result:

Hey mate!
Hi buddy
Hope you're okay

(Indexing & Slicing)

```
a= "Game of programming"
print(a[3:8]) → access from start
print(a[-8:-3]) → access from end
```

result:
e of p
snsl

- −8 ↑ Starting point (inclusive)
- −3 ↑ Ending point (Exclusive)

a="Abra ka Dabra"
print(a[0,5,2]) ie 0,2,4
result: AYL

If you dont mention the start [:5] it starts from initial point and if you don't mention end it starts from the goes till the end.

If you want to totally reverse the strind do [::-1]

functions of string

a=" A lot of spaces at start"
b=a.strip()
print(b.lower())
print(b.upper())
print(a.replace(" ", ","))

String to list

l="game,and,no".split(",")
#returns list
result=[game, and, no]
l[0]=game l[1]=and l[2]=no

Sarthaq & Sarjaq
Date:
Page:

To find if a substring is at a string or not

print ("at" in "andraatlas")
res: True
print ("lol" in "lotesh")
res: false

print ("we are learning \"string\" here")
res: we are learning "string" here

print ("We are \n now on newline")
res: We are
now on new line

print ("Hellow \t world")
res: Hellow world

To print c:\name\drive

print (r "c:\name\drive")

Sarthaq & Sarjaq
Date:
Page:

Data Structure

List	ordered	changable	Duplicates
Tuple	ordered	unchangable	No Dup
Set	unordered	addable/removable	No dup
Dictionary	unordered	changable	No dup

Eg:-

```
l = [1, 3, 4.9, "name", 3]
T = (1, 3, 4.9, "name", 3)
S = {1, 3, 4.9, "name", 3}
D = {23: "twothree", 'B': 43, 'C': 'CCD'}
```

Accessing
for list use index l[0], l[1] etc
for tuple use index T[0], T[1] etc
for set use x ins ie 3 ins S, 4 ins S
↳ It returns either True or false
for Dictionary Use D["val"] eg:- D[23]
D['B']

The properties of string like slicing and indexing applies to list and tuple.

```
l[1:3]   res:- [3, 4.9]
l[::-1]  res:- [3, "name", 4.9, 3, 1]
T[:3]    res:- (1, 3, 4.9)
```

Tuple is immutible you cannot change it.

HD Copy

Date:
Page:

Adding element:

for list L=L+["game"] or L[2]="orange"
for Tuple Immutable T3=T1+T2,
for set S.add("newItem") #individual
S.update({"hi","she"}) #multiple
for Dictionary D["val"]=newValue
D["newValue"]="newVal"

Deletion of element
for list del L[1] or del L
for Tuple Immutable delT
for set S.Remove("banana")/delS
for Dictionary del D["val"] delD

L=[1,2,3]
L2=L+[4,5,6]
print(L2) res: [1,2,3,4,5,6]

T1=(1,2,3)
T2=(4,5,6)
T3=T1+T2
print T3 res: (1,2,3,4,5,6)

L2.append(6.8)
prin(L2) res [1,2,3,4,5,6,6.8]

S={1,3,4,"name"}
S.add(56) res: {1,3,4,"name",56}
S.update({23,"he"}) res: {1,3,4,"name",56,23,"he"}

D={ 23: "twothree", "B": 43 }

It's possible to concat two dictionary but not using plus operater rather you use update.

D2={"B": 'BB', "C" = "CC"}

D.update(D2)
print(D)
res:
{ 23: 'twothree', "B": 43, "B": "BB", "C" = "CC"}

Copy function

In pytho when you assign a variable to another variable it doesn't copy all the content rather just simply points to the same memory location.
If you change any content of l2 then l1 gets affected too.

l1=[1,2,3]
l2=l1
l2

to fix it you should call copy function

l2=l.copy
l2[2]="hey" #change just happens in l2, l1 remains same.

Sarthaq & Sarjna
Date:
Page:

Same concept goes for the Set & Dictionary.

Since the content of the tuple can't be changed so copy function isn't available for it.

```
l=[1,2,3,4,5]
l3=l[1:3]   res: l3=[2,3]
```

Now l3 is a new ~~string~~ List.

```
l3[0]="Three"
print(l3)   res: l3=[0,3]

l2=l.copy()
s2=s.copy()
d2=d.copy()
```

for Query

```
help(L.append)
l.clear?
l.pop?
l.reverse() / l[::-1]
```

HD Copy

Karthaq & Sanjay
Date:
Page:

```
In D2 = {'A': L, 'B': T, 'C': S, 'D': D3
print D2['A']
result: [6.8, 'you', 6, 'are', 'how',
         3, 'four point nine', 3, 1]
print['A'][3]
  result
print D2['A'][3]  result: are
K=D2['D']
print(K)

for x in k:
    print(x, k[x])
result:
  23 - twothree
  B    43
newkey newValue
  y      YY
  z      10

L3=[L, T, D, 23, "game"]
type(L3[2])
    result: dict
L3=[x**2 for x in range(10)]
L3
 [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

S3={x**2 for x in range(2, 2013)}
S3
 {4, 25, 64, 121, 196, 289}
```

HD Copy

Problem of Data structures:

""" Lets say you are a teaches and you have different student records containing id of a student and the list in each subject where diff students have taken different number of subjects. All these records are in hardcopy. You want to enter all the data in computer and want to compute the average marks of each student & display"""

```
def getDatafromUser():
    While true
    D = {}
    while True:
        StudentID = input("Enter
            Student ID: ")
        marklist = input("Enter the
            marks by comma seperate:"
        morestudents = input(
            "Enter yes/No for adding
            more students:")
        if studentID in D:
            print(studentID,
                "is alread inserted")
        else:
            D[studentID] =
                marklist.split(",")
        if morestudents.lower() ==
            "no":
            return D
```

```
def getAvgMarks(D):
    avgMarksDict = {}
    for x in D:
        l = D[x]
        s = 0
        for marks in l:
            l = D[x]
            s = 0
            for marks in l:
                st = int(marks)
            avgMarks[x] = s/len(l)
    return avgMarks

avgM = getAvgMarks(studentData)

for x in avgM:
    print("Students:", x, "got avg Marks
        as:", avgM[x])
```

NumPy (python library)

It's very fast. You can import it as:

```
import numpy as np

a = np.array([1,2,3,4,5])
b = np.array([1,6,9,12,7])
print(a)
print(b)
```

You can use both [] () to define an array. Here type of a and b is numpy.ndarray

You can define the type of the array

```
a = np.array([1,2,3], dtype = 'f')
b = np.array([4,5,6], dtype = 'f')

a.dtype   res: dtype('int32')

b.dtype   res: dtype('float32')
```

Sarthaq & Sarjaq
Date:
Page:

Numpy (Dimensions)

```
import numpy as np
a = np.array([[1,2,30],[4,5,6]])
a.ndim # this gives the dimensions
```

res ÷ 2

a[0,2] res ÷ 30

This defines which array to chose

This defines which component of array is chosen.

~~B = np.array~~

when you're defining one dimensional array the number of element should remain consistant.

Defining three dimensional array.

```
B = np.array([[[1,2,3],[4,5,6],[7,8,9]],
              [[-1,-2,-3],[-5,-6,-7],[-8,-9,-10]]])
```

B.ndim (result = 3)

B[1, 0, 2]

Defines whether it's first or second group of array

chose a particular array within the group

choose the index.

Sarthaq & Sarjaq
Date:
Page:

B=[[[1,2,3],[3,4,5],[6,7,8]],[[-1,-2,-3],[-4,-5,-6],[-7,-8,-9]]])

B.shape res: (2,3,3)

- There are 2 2-d array
- There are 3 1-d array
- There are 3 element in 1-d array.

C.shape[0] res: 3

3 1d array

C.shape[1] res: 3

A=np.array([27])
A.ndim res: 1
B=np.array(3)
B.ndim Yes: 0

B.size res: 18
B.nbytes res: 72

Sarthaq & Sarjaq
Date:
Page:

Numpy (np.arrange
np.random.permutation
np.reshape)

A = np.arange(100)
[0, 1, 2, 3 . . . . . 97, 98, 99]
A = np.arrange(20, 100)
[20, 21, 22 . . . . . 97, 98, 99]
A = np.arrange(20, 100, 3)
[20, 23, 26, 27 . . . . . 92, 95, 98]

Iteration : process of executing a block of code repetedly.

print(rang(10))
range(0, 10)
print(list(range(10)))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

np.random has a lot of package one of them is permutation.

A = np.random.permutaion(np.arrange(10))
print(A)
[3, 9, 5, 7, 6, 4, 2, 1, 0, 8]

A = np.random.randint(20, 30)
res : 27 # returns random no within range.
type(A) is integer.

np.reshape

```
D = np.arange(100).reshape(4,25)
D.shape
    (4,25)
```

D will be 4 by 25 matrix with 4 rows and 25 columns. This will be a two dimensional array.

```
D = np.arange(100).reshape(4,5,5)
D.shape
    (4,5,5)
```

np.zeros?
np.ones?

```
A = np.random.rand(1000)
print(A)
```

#prints 1000 random numbers between the range 0-1
(we'll import plotting package)

```
import matplotlib.pyplot as plt
plt.hist(A)
```

(you get a uniform graph

```
plt.hist(A, bins=100)
```

↓ 100 individual datas.

```
B = np.random.randn(1000)
plot.hist(B, bins=200)
```

creates a ∩ this natured curved.

Sarthaq & Sanjay
Date:
Page:

rand → uniform distribution
randn → standard normal distribution,

You can also create random matrix array using.

```
x = np.random.rand(2,3)
print(x)
res:- [[[0.44, 0.53, 0.26], [0.32, 0.42, 0.5]
]]
x.ndim   res:- 2
```

and,

```
c = np.random.rand(2,3,4,2)
c.ndim   res:- 4
```

NUMPY (SLICING)

```
a[1:5] # index 1 till 5 but not 5
a[:5] # index 0 till 5 but not 5
a[2:] # index 2 till end including last element.
a[::-1] #from end till start (reverses the array)
a[::-2] #from start till and every other element.
a[::2,:]  ?

A = np.arange(1000)        #imp
b = A[3:10]
print b        [3 4 5 6 7 8 9]
b[0] = -1200
p              [-1200 4 5 6 7 8 9]
```

Hb Copy

Sarthaq & Sarjaq
Date:
Page:

Changing the element of b also changes the element of a as despite of having different variable name they share same memory location.

If you dont want that to happen then

```
b=A[3:10].copy()
```

Then if you change any element in b then the original value of A isnot altered.

```
b=A[3:10].copy()
```

```
A[::5]
    array([ 0  5  10 15 .....
            ... 85 90 95]

A[::-5]
    array([99 94 89 84 ....
           ... 14  9  4]
A
  array([0,1,2,3,4 ..... 97 98 99])
```

```
# idx=np.argwhere(A==3)[0][0]
  idx  res:3
  A[idx] = 400
```

Here we found the index of array.

Sarthaq & Sarjaq
Date:
Page:

A = np.round(10*np.random.rand(5,4))

array([[3., 6., 10., 4.]
[4., 5., 3., 7.]
[1., 3., 4., 5.]
[4., 9., 8., 7.]
[2., 3., 5., 4.]])

A[1,2] ie 3

A[1,:] # accessing the whole 2nd row.
array([4., 5., 3., 7.])

A[:,1] # accessing the whole 2nd col
array([6., 5., 3., 9., 3.])

A[1:3, 2:4] row 1 to row 3
array([[3,7], colum 2 to 4
[4,5]]) (end not included)
End point is
Exclusive.

A.T (gives the transpose of the matrix)

import numpy.linalg as la
la.inv(np.random.rand(3,3))

(returns the inverse of the matrix)

HD Copy

Sarthaq & Sarjan
Date:
Page:

A
array([[3., 6., 10., 4.],
[6., 4., 8., 8.],
[5., 2., 7., 8.],
[5., 10., 8., 10.],
[4., 1., 5., 7.]])

for sorting each column individually.

A.sort(axis=0)

A
array([[3., 1., 5., 0.],
4., 2., 7., 4.],
5., 4., 8., 7.],
5., 6., 8., 8.],
6., 10., 10., 10.]])

for sorting each row individually

A.sort(axis=1)

A
array([[0., 1., 3., 5.],
[2., 4., 4., 7.],
[4., 5., 7., 8.],
[5., 6., 8., 8.],
[6., 8., 10., 10.]])

Sarthaq & Sarjaq
Date:
Page:

Numpy (More Indexing)

A[index_array]

A[[1,4,6]] # index 1, 4 and 6 elements

a[[True, false, True, True, false]]

Assuming array has 5 element, the above returns all the elements corresponding to True Index.

If you want to get all the element that are less than 8 in the index_array then we use :

A[a<8]

A[a<8 & a>4] # difference between (and, &) ?

A[a<8 and a>4]

*, A = np.arange(100)
B = A[[3,5,6]]
B[0] = -4
Print(B) res: ([-4,5,6])

(= A[A<40]
print(() ie [0,1,2.....37,38,39]

Sarthaq & Sarjeq
Date:
Page:

B = A[(A<40) & (A>30)]
B
array([31, 32, 33, 34, 35, 36, 37, 38, 39])

and : It's used when both of the sides is one objects and has one value either true or false.

& : When the left side and the right side can be arrays & each element of that array can be true or false.

&, and # |, or # ~, not

NUMPY (Broadcasting)

A = A + 5

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ → Normally you can't do this but here it's converted to $\begin{bmatrix} 5 & 5 \\ 5 & 5 \end{bmatrix}$

A = A + $\begin{bmatrix} 1 \\ 2 \end{bmatrix}$

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ → This is turned to $\begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix}$

NUMPY

```
np.hstack
np.vstack
np.sort

A = np.round(10*np.random.rand(2,3))
A
array([[2., 7., 5.],[0., 0., 7.]])

At(np.arange(2).reshape(2,1))
array([[4., 7., 5.],
       [1., 1., 8.]])

B = np.round(10*np.round.rand(2,2))
A
array([[2., 7., 5.],
       [0., 0., 7.]])
B
array([[4., 2.],
       [8., 1.]])
C = np.hstack((A,B))
C
array([[2., 7., 5., 4., 2.],
       [0., 0., 7., 8., 1.]])

A = np.random.permutation(np.arange(10
[0,1,4,3, 7,8,6,5,2,9]
A.sort()
[0,1,2,3,4,5,6,7,8,9]
np.sort(A)
```

Sarthaq & Sarjan
Date:
Page:

Numpy Seed: Ufun(s)

```
B = np.random.rand(1000000)
%timeit sum(B)          → 300 m/s
%timeit np.sum(B)       → 3 m/s
def mysum: %% %timeit
    S=0
    for x in b:          } → 320 m/s
        S+=x
def mySum(G):
    S=0
    for x in G:
        S+=x
    return S

%timeit mySum(B)
```

Pandas

```
import pands as pd
print (pd.__version__)
A = pd.Series([2,3,4,5], index=['a','b',
    'c','d'])
type (A.values)
                res: numpy.ndarray
type(A)
                res: pands.core.series.series
A.index
  Index(['a','b','c','d'], dtype=
                'object')
A['a']    res: 2
A['a':'c']  res:  a  2
                  b  3
                  c  4

grades_dict = {'A': 4, 'A-': 3.5, 'B': 3,
    'B-': 2.5, 'B': 23
grades = pd.series (grades_dict)

marks_dict = {'A': 85, 'A-': 80, 'B': 75
  'B-': 70, 'D': 653
marks = pd.Series (marks_dict)
print(grades.valueres.)
 array [4., 3.5, 3., 2.5]
print
```

```
marks_dict = {'A': 85, 'B': 75, 'C': 65,
              'D': 55}
marks = pd.series(marks_dict)
marks
A    85
B    75
C    65
D    55
dtype: int64

marks[A]      res = 80
maks[0:2]     res = A  85
                    B  75
                    datatype: int64

D = pd.Dataframe({'Marks': marks,
                  'Grades': grads3})
print(D)
   marks  Grades
A    85     4.0
B    75     3.5
C    65     3.0
D    55     2.5

print(D.T)
          A     B     C     D
Marks   85.0  75.0  65.0  55.0
Grades  4.0   3.5   3.0   2.5
```

Sarthaq & Sarjaq
Date:
Page:

```
D.values
array([[ 85.,  4.],
       [ 75.,  3.5],
       [ 65.,  3.],
       [ 55.,  2.5]])

D.values[2,0]   res: 65.0
D.columns
    Index(['Marks', 'Grades'],
          dtype='object')

D['scaledMarks'] = (D['Marks']/90)*100
D
```

	Marks	Grades	scaledMarks
A	85	4.0	94.44
B	75	3.5	83.33
C	65	3.0	72.22
D	55	2.5	61.11

```
del D['scaledMarks']
D
```

	Marks	Grades
A	85	4.0
B	75	3.5
C	65	3.0
D	55	2.5

```
G = D[D['Marks'] > 70]
G
```

	Marks	Grades
A	85	40
B	75	3.5

HD Copy

Sarthaq & Sarjan
Date:
Page:

Pandas (NaN)

```
pd.Dataframe([{'a':1,'b':2},
              {'b':3,'c':4}])
```

	a	b	c
0	1.0	2	NaN
1	NaN	3	4.0

```
A= Pd.Dataframe([{'a':10,'b':15},
                 {'a':13,'c':9}])
```

	a	b	c
0	10	15	NaN
1	NaN	13	9

```
A.fillna(6.8)
```

	a	b	c
0	10	15	6.8
1	6.8	13	9

A.dropna? (Remove missing value)

Pandas (Indexing)

```
data = pd.Series(['a','b','c'],
                 index=[1,3,5])
data[1] # explicit index, use loc instead
data[1:3] # implicit index, use iloc instead.
rs.iloc[2:3]
```

HD Copy

eg:- A=pd.series(['a','b','c'], index=[1,3,5])

A[1] res:- (a)

A[1:3] res:- 3 b
5 c
dtype: object

A.loc[1:3]
res:- 1 a
3 b
dtype: object

A.iloc[1:3]
res:- 3 b
5 c
dtype: object

D result:-

	Marks	Grades
A	85	4.0
B	75	3.5
C	65	3.0
D	55	2.5

D.iloc[2,:]
Marks : 65.0
Grades : 3.0
Name : C, dtype: float64

D.iloc[::-1,:] res

	Marks	Grades
D	55	2.5
C	65	3.0
B	75	3.5
A	85	4.0

Sarthaq & Sarjaq
Date:
Page:

Pandas (CSV file)

```
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer
# df = pd.read_csv('covid_19_data.csv')
df = pd.read_csv('E:/covid/covid_19_data.csv')
df.head()
    first few records is shown
df.head(10)
    # first 10 records
df.drop(['SNO', 'Last Update'],
        axis=1, inplace=True)
# deletes the SNO and last update
from the CSV file and inplace=
True assures that it's to be done
in the original ~~vari~~ CSV file.
# Now if you call df.head() it
won't show SNO and it will
also not display the last update
df.rename(columns={'Observation Date'
:'Date', 'Province/State':'Province',
'Country/Region':'Country'},
inplace=True)
```

HD Copy

Santhaq & Sanjay
Date:
Page:

```
df['Date'] = pd.to_datetime(df['Date'])
df.head() # to see the result
```

Initially 1/22/2020 — New date formate 2020-01-22

```
# Use this to change any date formate to pandas date formate.
df.describe() # gives you details about the data
df.info() # more information about the file and all.
df.fillna('NA') # All missing values will be filled with NA.
df.info() # you'll see no null values.
```

Q Group by country and show country, Confirmed cases, death and recovered by their total number.

```
df2 = df.groupby('Country')['Country', 'Confirmed', 'Deaths', 'Recovered']
      .sum().reset_index()
df3 = df2[df2['Confirmed'] > 100]
```

Hb Copy

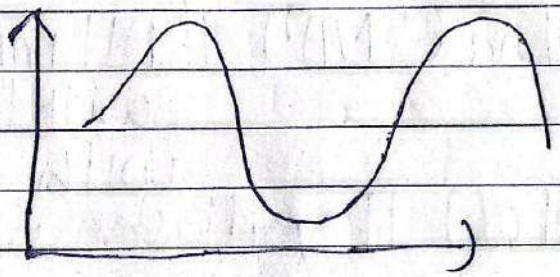
Sarthaq & Sarjaq
Date:
Page:

Matplotlib

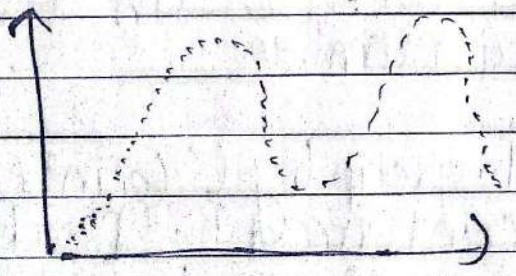
```
import matplotlib.pyplot as plt.
x=np.linspace (0,10,100)
plt.plot (x, np.sin(x));
```

another

```
import matplotlib.pyplot as plt
x=np.linspace (0,10,1000)
y=np.sin(x)
plt.plot (x,y)
```



```
plt.scatter (x,y)
```



more clear visualization on

```
plt.scatter (x[::10], y[::10])
```

you can also give a color = 'red' just use comma and add it.

Some things to do:

```
plt.plot (x, x+0 , 'g' , ) # solid green
plt.plot (x, x+1 , '--c' ) # dashed cyan
plt.plot (x, x+2 , '-.k' ) # dashed cyan
plt.plot (x, x+3, ':r' ) # dotted red
```

Eg:

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer

df = pd.read_csv ('E:/covid/covid-19_data.csv')
df.head(50)

df.drop(['SNO', 'Last Update'], axis=1, inplace = True)
df.rename ( columns = {'ObservationDate': 'Date', 'Province/State': 'State', 'Country/Region': 'Country'})

df['Date'] = pd.to_datetime(df['Date'])
imputer = SimpleImputer(strategy = 'constant')
df2 = pd.DataFrame(imputer.fit_transform(df), columns = df.columns)
```

HD Copy

Sarthaq & Sarjaq
Date:
Page:

```
df3 = df2.groupby(['Country', 'Date'])
[['Country', 'Date', 'Confirmed',
'Deaths', 'Recovered']].sum().
reset_index()

df3.head(20)

countries = df3['Country'].unique()
len(countries)  res: 14

for idx in range(0, len(countries)):
    c = df3[df3['Country'] == countries
    [idx]].reset_index()
plt.scatter(np.arrange(0, len(c)),
c['Confirmed'], color='blue', label
='Confirmed')
plt.scatter(np.arrange(0, len(c)),
c['Recovered'], color=green,
label='Recovered
plt.scatter(np.arrange(0, len(c)),
c['Deaths'], color='red', label=
'Deaths')

plt.title(countries[idx])
plt.xlabel('Days since the first
'subject')
plt.ylabel('Number of cases')
plt.legend()
plt.show()

# This is for each country individually
```

Hb Copy

If we want overall trend, the overall growth and the overall deaths of the world then we can do:

```
df4 = df3.groupby(['Date'])[['Date',
 'Confirmed', 'Deaths', 'Recovered']].
 Sum().reset_index()

 C = df4
 plt.scatter(np.arrange(0, len(C)),
  C['Confirmed'], color='blue', label='Con
  ormed')
 plt.scatter(np.arrange(0, len(C)),
C['Recoverded'], color='green', label=
 'recovered')

  plt.scatter(np.arrange(0, len(C)),
   C['Deaths'], color='Red',
   label='Deaths')
   plt.title('World')
   plt.xlabel('Days since the first
                 Suspect')
   plt.ylabel('Numbers of cases')
  plt.legend()
  plt.show()
```