(Guido Van Rossum in 1991) Date 22/March
Page No.

Data Analytic using Python :-

H.L.L / Interpreted Lang.

Python 1.0 -97 Dec [learn when I]

Features :-

→ Easy to code & learn → print ("Hello world")
→ Free & open source → python.org → Download, use, share.
→ Object oriented language → these class, object, polym.
→ GUI Programming Support →
→ High level language
→ Extensible language
→ Portable language → OS/MAC → same program run on other environment
→ Dynamic language
→ Interpreted language → H.L.L → Machine code
→ Expressive language → small code
→ Embeddable language
→ Integrated language
→ large standard library

Application

Console App. / Desktop Applic./

Data type

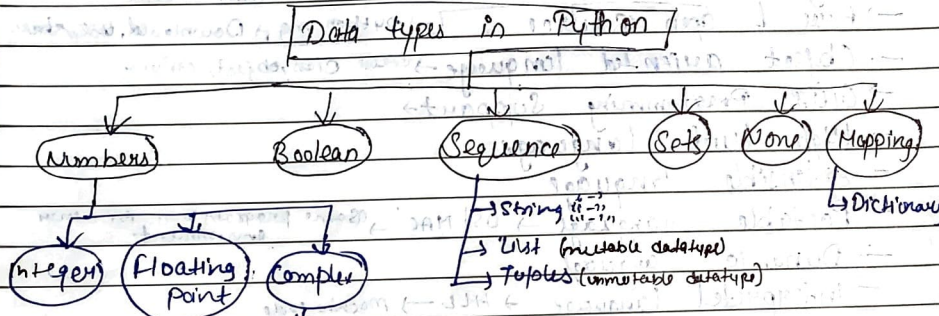
IDLE
Interactive mode → script mode

Date
Page No.

Data type in Python:-

Variable name → starts first character:

→ A-Z	0-9
→ a-z	
→ _	



List

a = [1, 2, 3, 4, 5]

23/May/22

Operators:-

Arithmetic Operators.

\+ num1 = 10 num = 20 num1 + num2 = 30

Str1 = "ABC" Str2 = "India", Str1 + Str2 = ABCIndia

Date
Page No.

Ex- Str(10)

o/p → '10'

(or)

int. -- Str -- (num)

num = 10

print (str(num))

Example ② Class Car():

def __init__ (self, windows, doors, engine type):

self.windows = windows

self.doors = doors

self.engine type = engine type

def drive(self):

print ("The person drives the car")

def __str__(self):

"This overrides the function of str ()"

return:

C = Car (4, 5, "Diesel")

print (C)

print (repr(C)) ← Direct call a function.

Date
Page No.

-	num1 - num2	-10	
*	num1 * num2	200	Str*2 ABCABC
/	num1 / num2	0.5	
%	num2 % num2	0	(divide karega aur decimal hata dega)
//(Flow division)	num1 // num2	0	
** (Exponent)	num3 ** num4	3**4	81

Relational operators :-

Operators — num1 = 3, num2 = 4, num3 = 10, num4 = 3

① == (equal to) num1 == num4 (3 == 4) — false

② != (not equal to) num1 != num2 — false

num1 != num2 — True

Str1 = "ABC", Str2 = "BCD"

Str1 != Str2

③ > num2 > num1 (True) , num3 > num4 (True)

④ < num2 < num3 (True) num1 < num4 (False)

Logical operators :-

num1 = 10 num2 = -20

① and (both cond^n same so true) num1 == 10 (T) and num2 == -20 (T)

num1 == 10 (T) and num2 == 10 (F)

② Or num1 >= 10 and num2 == -20 (T)

num1 >= 10 or num2 != -20 (T)

Date
Page No.

③ NOT not (num1==20) (True)
not (num1==10) (false)

Assignment Operator:-

a-=b
a=a-b

-=
+=
-=
*=
/=
%=
//=
**=

Membership Operators

① in L=[10, 20, 30, 40, "ABC", 40.5, 1] (0 1 2 3 4 5)
1 in L True
ABC in L False
'1' in L False

② not in

35 not in L - True

Expressions → Combination of operator, constants & variables.

a+b+20*30/20 a=10 → integer literal
b=30 String literal = "ABC"

literal - fixed value

numeric literal { Integer literal, Float literal, Complex literal }

Date
Page No.

Precedence Operators

( ) Highest Precedence
**
*/
+-

35+40*2+16/2+(4+3)

35+80+8+(7)
35+88+7
=130

35+40*2+16/2+7
35+80+16/2+7
35+80+8+7
115+15
130.

How to enter the input & how to display the output :-

a=input("Enter your name")

O/P = '10'
type(a)
class(str)

a = int(input("Enter —"))
Float

Print("My name is deepak")
O/P ⇒ My name is deepak

Date
Page No.

Keywords :- reserved words that arent used as identifier.

Eg. False True, while, return, for, and, or, if — etc.

For naming variables:

(1) Not start with numeric digit as first letter.
(2) Special symbol not allowed
(3) Allowed variable name start with
    a-z
    A-Z
    _ Underscore

(4) Meaningful name should be such as current_salary, std_name.

(5) Constant should be write in capital letter such as PI = 3.14.

(<>) = (!=) = means not equal to

Relational Operator

Date
Page No.

Bitwise Operator :-

identify operator (is, is not)
membership operator (in, not in)

```
a = 10
if (type(a) is int)
    print("true")
else print("false")
```

```
a = 1.5
float
if (type(a) is not int)
    print("true")
else
    print("false")
```

Conditional Statement in Python :-

Syntax
```
If test condition :
    Statements
    num = 10
else If num > 0
    Print("number is positive")
```

```
if condition
    Statement 1
else if condition
    Statement 2
else Statement 3
```

```
if a > b and a > c:
    print("a is largest no:")
elif
```

print("A") if a>b else print("B")

Nested if statement

```
if condition:
    if condition:
        if condition
else.
```

```
num = -10
if num >= 0:
    if num == 0:
        Print("No is zero")
    else: Print("No is positive")
else:
    print("No is negative").
```

Date
Page No.

Pass statement

```
if condition:
    pass
```

Q. Write a program whether a given no. is even or odd?
Q. Write a program to print the largest of the 3 no.
Q. WAP to check whether a person is eligible to vote or not
Q. Print the following grade acc. to marks.

if mark >85 and mark <=100 = grade A+
>60 <=85 A
>40 <=60 B
>30 <=40 C
else fail

① if n%2==0:
    print("no is even");
else:
    print("No is odd")

② if a>b && a>c
    print("a is greater")
else b>a && b>c:
    print("b is greater")
else:
    print("c is greater")

③ if age <18:
    print("Not eligible for vote")
else:
    print("Eligible for vote")

④ if marks >85 && marks <=100:
    print("Grade A+")
elif marks >60 && marks <=85:
    print("grade A")
el if marks >40 && marks <=60:
    print("Grade B")
elif >30 && marks <=40:
    print("Grade C")
else:
    print("Fail");

Loops in Python:- For Loop
① for
② while
③ do-while

L=[10,20,30,40,50,60]
for i in L:
    Print(i) — print(i, end=" ")

10
20
30
40
50
60

Date
Page No.

Control Statement:-

- Break ✓
- Continue
- Pass

```
i = 0
while (i<=6)
    print(i)
    if i == 3:
        break
```

```
i=0
while (i<=6)
    print(i)
    i=i+1
else
print("No is larger than 6")
```

Q/ Write a program to print a table of 5.

```
n = int (input ("enter the number"))
l = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
for i in l
    c = n * i
    print(c)
```

Q WAP to print a sum of elements of a given list
l = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

```
Sum = 0:
for i in l :
Sum = sum + i
print (Sum)
```

Range function :-

range (start, stop, step-size)

range(10)
0, 1, 2, 3, 4, 5, 6, 7, 8, 9

$1 \frac{5}{2} = 2.5$ || $\frac{5}{2} \Rightarrow 2$

Date
Page No.

Q WAP to print even no. (with range function)

```
n = int (input ("enter the number"))
for i in range (0, n+1, 2)
    print (i)
```

Q/ WAP to print a table using with range function.

```
n = int (input ("enter the number"))
for i in range (1, 11, 1):
    c = n * i
print (n, "x", i, "=", c)
```

Q WAP to Check palindrome or not no. while loop.

```
Num = int ( input ("enter a value"))
temp = num
rev = 0
while (Num > 0):
    digit = num % 10
    revs = rev * 10 + dig
    num = num // 10
if (temp == revs)
    print ("This value is palindrome")
else
    print ("This is not a palindrome")
```

No switch case in python

#

```
num = int (input ("enter no for workday name"))
if (num==1)
    day = "Monday"
    print (day)
elif (num==2)
    day = "Tuesday"
    print (day)
elif (num==3)
    day = "Wednesday"
    print (day)
elif (num==4)
    day = "Thursday"
    print (day)

elif (num==6)
    day = "Saturday")
    print (day)

elif (num==7)
    day = "Sunday"
    print (day)
else
    print ("Invalid day number")
```



```
dict = {
    1: 'Sunday'
    2: 'Monday'
    3: 'Tuesday'
    4: 'Wednesday'
    5: 'Thursday'
    6: 'Friday'
    7: 'Saturday'
}

print ('Entered no. is 2 day of the week is', dict.get (2, 'invalid no'))
```

Date 4/4/22
Page No.

File handling

Text file -

- Store information in ASCII or unicode character.
- Each line of text is terminated by a special character known as EOL (End of line)
- Extension for text file is .txt
- Default mode of file

Binary file

- Contains information in the same format in which the information is held in memory.
- There is no delimiter for a line
- No translation are required
- More secure.

Diff b/w Textfile & Binary file

Textfile	Binary file
① Text file store information in ASCII character	① Binary files are used store data such as image audio/video
② Each line of text is terminated with a special character known as EOL	② There is no delimi in binary file.
③ Text file are easy to understand because these files are in human readable form	④ Binary files are difficult to understand.
④ text files are slower than binary files	④ Binary files are fast
⑤ text files are having extension .txt	⑤ Binary files are having extension .dat

Operations as a file

- open a file
- Create a file
- Read content of file
- write into a file
- Traverse the content of a file
- Append data into a file

open a file

```
File_object = open (<Filename>, <File mode>)
                or
File_object = open (<File name>)
```

where,

File object = File handle
Filename = Name of a file
file mode = mode of a file.

e.g. `f = open ("Ram.txt", 'w')`

'r' → read
'w' → writing

if both, w+r

DELTA Notebook

File Access Modes

b → binary
a → append
W → write
R → read

w
wb – read the content of binary file & pointer is come in beggining
w+ –
wb+ – read & write the file
w – write the content in the file
w+ – Read & write in to the text file
wb = write the content in the file
wbt =
a –
ab –
a+
ab+

fileobject = open ("filename", "access mode")

f = open ("myfile.txt", "r")

fileobject."close()"

Syntax:- With clause

with open(file_name, accessmode) as fileobject :

Example with open ("myfile.txt", "r+") as f:
content = f.read()

write():- For writing a single String
writeline():- For writing a sequence of Strings.

Example:- myobject = open ("myobject.txt", 'w')
myobject.write ("Hey I have started using file in python")
myobject.close()

Write lines (Multiple lines)

myobject = open ("myfile.txt", 'w')
lines = ["Hello everyone \n",
"This is second line\n",
"This is 3rd line\n",]
myobject.write(lines)
myobject.close

Hello everyone
This is second line
This is third line

Read content from a file:-

- read()
- readline()
- readlines()

f = open ("myfile.txt" 'r')
f.readlines (n/1)
NO argument
-negative value

Date 6/4/22
Page No.

Writting into a text file

- write()
- writeline()

Reading Contents of a file

- Read()
- Readline()
- readlines()

```
myobject = open("myfile.txt", 'w')
lines = ["Hello everyone\n"
         "writing multiline, string\n",
         "This is 3rd line\n"]
a = myobject.writelines(lines)
print(a)
myobject.close()
myobject = open("myfile.txt", 'r')
b = myobject.readlines()
print(b)
```

```
for i in b
print(i)
a=0
```

Display each word of a line repeatedly as an element of a list.

- By using split function.

```
myobject = open("myfile.txt", 'r')
print(myobject.readlines())
myobject.close()
myobject = open("myfile.txt", 'r')
d = myobject.readlines()
for line in d:
    words = line.split()
    print(words)
```


List -

Date
Page No.

Use of Split() and splitline() :-

```
f = open("myfile.txt", 'r')
d = f.readlines()
for line in d:
    word = line.split()
    print(word)
for line in d:
    word = line.splitlines()
    print(word)
```

myfile.txt

```
Hello everyone
This is 1st line
This is 2nd line
```

['Hello', 'everyone']
['This', 'is', '1st', 'line']
['Hello everyone']
['This is 1st line']

Tell() and Seek() :-

f.seek(offset, Refrence point)

0 - Beginning
1 - Current positive
2 - End of the file.

```
f = open("myfile.txt", 'r')
d = f.read(n)
print(f.tell())
f.seek(0)
f.seek(10)
print(f.read())
```

Date________
Page No.________

f.txt
["hello python"]

```
print ("learning to move file object ")
  f= open ("f.txt", "rt")
  str= f.read()
  print(str)
  print ("Position of file object is", f.tell())
  f.seek(2)
  print ("Now the file object is at position=", f.tell())

  file object.seek(5)
  print ("Now the file object is at", f.tell())
  str= f.read()
  print (str)
```

* isme jo display hoga wo haga python an. python se phle space.

Binary Files :-

Pickle
- dump - write into a binary file
- load - read data from

```
import pickle
print (" working with binary files")
  f= open ("e.dat", "ab")
  rec no=1
print ("Enter record of Student")
print ()
while true:
  print ("Read no :", recno)
  roll no = int(input ("enter rollno"))
```

DELTA Notebook

Date________
Page No.________

```
Name = input ("enter name of student")
Class = input ("enter your class")
sdata = [rollno, name, class]
pickle.dump (sdata, f)
  ans = input ("Do you want to enter record (Y/N)")
  recno = recno +1

if ans.lower() == 'n'
  print ("record entry over")
  print()
  break
f.close()
print ("Now reading data from binary file")
  print ()
read rec =1
  try:
  with open ("e.dat", "rb") as f:
  while true:
    sdata = pickle.load(f)
    print ("record no is", readrec)
      print (sdata)
      read rec = read rec +1
    Except EOF Error
      pass
      f.close()
```

→ with → we use with bcz we dont have need to close the file.

pass → to (gado) kaha prog. run hojaye.

DELTA Notebook

with clause- automatically close. Date____ Page No.____

CSV File :- (Comma Separated values)

- Easier to create
- Preferred import & export format for databases and spreadsheets.
- Capable of storing large amount of data.

import CSV
⇒ reader()
⇒ writer()

Open a CSV file

⇒ with open ("Filename.csv", "access mode") as fileobject
⇒ File object = open ("P.csv", "w")
  fileobject.close()

writing in a CSV file:-

write object = CSV.writer() [object written return karta hai usi ko yaha se file mai write karke lo]
• writeobject.writerow() — Single
• writeobject.writerows() — Multiple

```
import csv
f= open ("Student.csv", "w", newline='')
Stuwriter = csv.writer(f)
Stuwriter.writerow (['Roll no', 'Name', 'Marks'])
record = []
while true:
    print ("Enter Student Record")
    r = int (input ("Enter Roll no of student"))
    n = input ("enter name of student")
```


Date____ Page No.____

```
t= int (input ("enter marks of student"))
rec = [r,n,t]
record.append(rec)
```

Ex/Alt: [Stuwriter.writerow(rec)]

```
Ch = input (" Do you want to enter more records")
if Ch == 'n' or Ch == 'N':
    break
for i in record:
    Stuwriter.writerow(i)
f.close()
```

Reading Content of a csv file :-

Name	Roll no	marks
-	-	-
-	-	-

```
with open ("student.csv", 'r') as f:
    fobj = csv.reader(f)
    for i in fobj:
        print(i)
```

[next. f(obj) → next skip
for i in fobj:
    if i[2] ≥ 500
    Print(i)]

Read a file as dictionary

```
import csv
with open ("student.csv", 'r') as f:
    csvf = csv.dictReader (f)
    for row in csvf:
        print (dict (row))
```

[Roll no: 1, 'Name': 'Ram', "Marks": 800]

Date
Page No.

Functions

1. Built in functions
2. User defined functions

random (0.0 - 1.0) 0.9

Function — Function Definition
Function Calling

⑥ No Argument, No return

```
def function_name():
    ==
    ==  } body of function
```
} function definition.

function calling

```
def printline():
    print("This is a body of a function")
printline()
```

② No argument, return something :-

```
def add():
    a = int(input("enter 1st no"))
    b = int(input("enter 2nd no"))
    c = a+b
    return c
d = add()
print("sum of 2 no.'s is", d)
```


Date
Page No.

③ Argument Passing, no return :-

```
def add(c,d):        → parameters (formal parameters)
    e = c+d
    print("sum of two number is", e)

a = int(input("enter 1st no"))
b = int(input("enter 2nd no"))
add(a,b)        → arguments (actual parameters)
```

*. Calling ke time jo variable call hoga usko argument kehte h.

④ Argument Passing, return something :-

```
def add(c,d):
    e = c+d
    return e
a = int(input("enter 1st no"))
b = int(input("enter 2nd no"))
e = add(a,b)
print("addition of two no is", e)
```

Passing default argument :-

```
def myfun(x, y=50):
    print("x:", x)
    print("y:", y)
myfun(10,20)
```

Date________
Page No.________

Passing variable length argument

[*- lgane u multiple values aeen kr skta h]

```
def my fun (*argv)
    for i in argv:
        print (i).
myfun( 'Hello', 'welcome', 'to', 'python')
```

O/P: Hello
welcome
to
python

Q/ what the o/p?

①
```
def myfun (x):
    x=[20,30,40]
L = [10,11,12,13,14,15]
myfun (L)
print(L)
```

O/P → 10,11,12,13,14,15

②
```
def my fun(x):
    x[0]=20
L= [10,11,12,13,14,15]
my fun (L)
print (L)
```

O/P → 20,11,12,13,14,15

③
```
def my fun(x):
    x=20
    print ("x is", x)
x =30
my fun (x)
print ("x is", x)
```

O/P → 20, 30

Date________
Page No.________

Recursive Functions :-

```
def t-rec (k)
    if (k>0)
        result = k + t-rec (k-1)
        print (result)
    else
        result=0
    return result
print ("This is recursive function")
t-rec (6)
```

result = 6 + t-rec(5)
5 + t-rec(4)
4 + trec (3)
3+ trec(2)
2+ trec(1)
1 + trec(0)
0

6+18 ⇒ 21

Function as object

- A function is an instance of the object type
- → You can store the function in a variable
- → You can pass the function as parameter to another function.
- → You can return a function form form a function.

```
def cap(text): return text.upper()
print (Cap ('Hello'))
    Big = Cap
    print (Big ('Hello'))
```

How a function passed as a argument:-

```
def cap(text):
    return text.upper()
def small (text):
    return text.lower()
```

```
def alphabet(func):
    letter = func("Hi, I am created by a function passed as an argument in")
    print(letter)
alphabet(cap)
alphabet(small)
```

Function can return another function.

```
def create-adder(x)
    def adder(Y)
        return x + Y
    return adder
adder_15 = create-adder(15)
print add_15(10)
```

```
def disp():
    def show():
        return "show function"
    print("Disp function")
    return show
u-sh = disp()
print(u-sh())
```


List functions

- append()
- insert()
- extend()
- Methods to process lists - in, test. in
- len()
- max()
- min()
- +
- *
- indexing
- Remove.

Ex

```
L = []                          []
L.append(4)                     [4]
L.append(2)                     [4,2]
L.append(3)   1,2,3             [4,2,3]
for i in range(1,4)
    L.append(i)                 [4,2,3,1,2,3]
L.append([5,6])                 [4,2,3,1,2,3,(5,6)]
L.append([10,11,12])            [4,2,3,1,2,3,(5,6),(10,11,12)]
```

Date 19/Apr/22
Page No.

```
def printname (firstname, last name):

print (f "Welcome" {firstname} {last name})
      format specifier
print name ("James", "watt")
printname (firstname = "James", lastname = "watt")
printname (firstname = "Watt", first name = "James")
```

Ist Way

```
def sum(a,b):
    print (a+b)
def sum (a,b,c)
    print(a+b+c)
def sum (a,b,c,d)
    print(a+b+c+d)

Sum(10,20)
Sum(10,20,30)
Sum (10,20,30,40)
```

IInd way (Variable length non-keyword argument)

```
def add (*arg) :
    print (sum(arg))
add()
add(12)
add (12,13)
add(12,13,14)
add (10,10,20,20)
```

Variable length keyword Argument :-

```
def myfun (** kwarg):
  for key, value in kwarg.items()
      print ("%.s = %.s" %. (key, value))
myfun (a="Krishan", b="Ram", c="Mohan")
```

O/p → a = Krishan
b = Ram
c = Mohan

Date
Page No.

Q WAP to check whether the string is palindrome or not.

```
Char= String input (" enter a string"))
temp = str
  temp = char
  rev = 0
  while (char > 0):
    Value = char % 10
    revs = rev * 10 + value
    char = char
```

```
st = input (" Enter String ")
No = "Palindrome" if st = st[:: -1]
                              reverse
  else
     rs = "Not palindrome"
```

O/P	I/P
L.append([8,20,3])	⇒ [10,20,30,40,[8,20,3]]
L.insert (2, [8,20,3])	⇒ [10,20,[8,20,3],30,40,[8,20,3]]
L.extend ([8,20,3])	⇒ [10,20,[8,20,3],30,40,[8,20,3], 8,20,3]

1. in operator — 2 in L1
2. Not in operator — 2 not in L1
3. len() — print(len(L1)) → 3
4. min() — print (min (L1)) → 1
5. max() — print (max (L1)) → 3
6. + operator
7. * operator
8. index (ele, beg, end) {element, beginning, end}
9. Count ()

L1 = [1,2,3]
L2 = [4,5,6]
L3 = L1 + L2 o/p
print(L3) ⇒ [1,2,3,4,5,6]
L4 = L1*3
print(L4) [1,2,3,1,2,3,1,2,3]

Lis = [2,1,3,5,4,3]
print(lis.index(3,3,6)) 5 o/p
print (list (count(3)) ⇒ 2 o/p

Date
Page No.

List as Stack

```
Stack = ["Ram", "Priya", "Komal"]
Stack.append ("Mohan")
Stack.append ("Raman")
print(Stack)
print(Stack.pop())
print(Stack)
print(Stack.pop())
print(Stack)
```

List As Queue

```
q = ["A", "B", "C", "D"]

q.append ("E")
q.append ("K")
print(q)
print(q.pop(0))
print(q)
print(q.pop(0))
print(q)
```


Date
Page No.

Queues can be implemented by deque :-

```
import deque
queue = deque (["Ram", "Tarun", "Mohan"])
print (queue)
queue.append ("Komal")
print (queue)
print (queue.popleft())
print (queue.popleft())
print (queue)
```

List as Matrix :-

```
A = [ [1,4,5,12], [-5,8,9,0], [-6,7,18,19])
print ("A =", A)
print ("A[1] =", A[1])
print ("A[1][2] =", A[1][2])
print ("A[0][-1] =", A[0][-1])
  C = []
for Row in A
    C.append (row[2])
print(C)
Matrix Multiplication
X = [[12,7,3], [4,5,6], [7,8,9]]
Y = [[5,8,12], [6,7,3,0], [4,5,9,1]]
result = [[0,0,0,0], [0,0,0,0], [0,0,0,0]]
for i in range (len(x))
for j in range (len(y[0])):
for k in range (len(y)):
result[i][j] += X[i][k] * Y[k][j]
for r in result:
    print (r)
```

agar , comma lgahuah tor ye tuple h ni toh ne,

Date
Page No.

Tuples

```
emp = ()
print (emp)
t = 1,2,3
t1 = (1,2,3,34,5)
t2 = (1)
t3 = "Ram", "Mohan"
print(t1 + t3)
t4 = (T1, T3)
```

Deleting a tuple

```
t5 = (0,3)
del(t5)
print(t5)
```

Printing length of a tuple

```
print(len(t1))
```

Converting list to tuple

```
l = [1,2,3]
print (tuple (l))
```

tuples in a loop

```
tup = "geet",
n=5
for i in range (n))
  tup = (tup,)
  print(tup)
```

use cmp(), max(), min()

```
tup1 = ('python', 'language')
tup2 = ('Code', '1')
if (cmp (tup1, tup2)! = 0):
  print(Not the same)
  else
  print (Same)
```


Date 25/4/22
Page No.

Shallow and Deep Copy:-

```
L1 = [10,20,30,40]
L2 = L1
[10,20,30,40] print(L2)
[10,20,30,40] print (L1)
L1 [2] = 100
[10,20,100,40] print (L1)
[10,20,100,40] print (L2)
```

Shallow Copy:-

```
import copy
L1 = ["Name", "Rollno", "Class"]
L2 = Copy. Copy(L1)
```

↑ module ↑ function

Example of Shallow Copy:-

```
Old_list = [[1,2,3],[4,5,6],[7,8,9]]
new_list = Copy.copy (old_list)
print ("old list is", old_list]       print(id (old_list))
print ("New list is", new_list]       print(id (new_list))
old_list.append([4,4,4])
print("old_list :", old_list)         [ ——— [4,4,4]]
print ("new_list :", new_list)        [ ——— ]
old_list [1][1] = 'AA'
print (old list)                      [ —— [4,AA,6] —— ]
print (new list)                      [ —— [4,AA,6] —— ]
```

Assigment operator use kr the the ve changes karte ha the.
→

Date ________
Page No. ________

Deep Copy :-

→ Data ki copy hoti h usme newly copy mai works krte h. Original mai reflect nhi hote.

```
import copy                     ↗BB
old_list = [['A', 'B', 'C'], [100, 200, 300, 400], [3, 4]]
new_list = copy.deepcopy(old_list)
print(old_list)
print(new_list)
print(id(old_list))
print(id(new_list))
old_list[1][0] = 'BB'
print(old_list)
print(new_list)
new_list.append([10, 20, 40, 50])
print(old_list) → [['A', 'B', 'C'], ['BB', 200, 300, 400], [3, 4]]
print(new_list) [['A', 'B', 'C'], ['100', '200', '300', '400'], [3, 4],
                 [10, 20, 40, 50]]
```

Dictionary :-

```
D = { 1 = "Ram",
      2 = "Komal",
      3 = "Krishna",
      4 = "Madav",
    }
```

⇒ Print all ~~keyword~~ key names of dictionary

```
for x in D:
    print(x)
```

⇒ print all values of dictionary

```
for x in D:
    print(D[x])
```

⇒ Values() function is also used to return values

```
for x in D.values():
    print(x)
```

DELTA Notebook

Date ________
Page No. ________

⇒ print both key & values of Dictionary

```
for i, j in D.items():
    print(i, j)
          ↓  ↓
        key values
```

⇒ access individual item

① print (D[1])
② print (D.get(1))

⇒ Change value

D[1] = "Mohan"

⇒ Add item

D[4] = "Mehal"

⇒ Removing item

D.pop(1)

D.popitem( )

⇒ Remove all item with the specified key name

del D[1]

⇒ Delete Dictionary completely

del D

⇒ Empty the Dictionary

D.clear()

DELTA Notebook

Date
Page No.

① Update Method of dictionary :-

```
A= [1: "Ram", 2:"Mohan", 3: "Komal"]
A.update ([4: suman"])
```

② From key method of Dictionary

Syntax :- dict.fromkey (keys, value)

```
X= ("key1", "key2", "key3")
y = 0
D = dict.fromkey (x, y)
print (D)
```

③ Ordered dictionary in python

items method + Getting the keys from dict

```
print ("This is a Dict")
d = {}
d['a'] = 10
d['b'] = 20
d['c'] = 30
For key, value in d.items ()
    print (key, value)
od = Ordered Dict()
od['a'] = 20
od['b'] = 30
od['c'] = 40
```


Date
Page No.

Deleting & Re-entering element into ordered list

```
od = Ordered Dict()
od['a'] = 1
od['b'] = 3
od['c'] = 4
od['d'] = 5
od.
```

Conversion of list & string into dictionary

```
def convert (list)
    it = [list[i] : list[i+1] for i in range (0, len(list), 2)]

list = ['a', '1', 'b', '2', 'c', '3']
print (convert (list))
```

String Conversion :-

By using eval method

```
String = "{'A': 8, 'B': 14, 'c': 15}"
D = eval (string)
print (D)
print (D['A'])
print (D['c'])
```

By Using Generator expression :-

String = "A-B, B-14, C-15" (key, value)

Dict = dict(i.strip(), j.strip()
for p,j in (element.split('-'))
for element in string.split(","))

for (i,j in ((element.split('-') for element in string.split(','))

Set

- Set is unordered collection of item.
- Every element in set is unique & immutable.
  - no ordered collection
  - no indexing
  - only element are immutable.
- List is not stored in Set.

Update()

$l_1 = [1,2,3]$	$l_1 = [1,2,5]$
$l_2 = [3,4,5]$	$l_2 = [1,2,3,4,5]$
$l_3 = [10,11,12]$	$s_1 = \{'a', 'b', 'c'\}$
	$s_2 = set(l_2)$ $S_2 = \{1,2,3,4,5\}$
$s_1 = set(l_1)$	$s_3 = set(l_3)$ $S_3 = \{1,2,3\}$
$s_2 = set(l_2)$	$s_2.update(s_1)$
$s_1.update(s_2)$	print(s_2)
print(s_1)	$s_2.update(s_1)$
$s_1.update(l_3)$	d = {6: 'six', 7: 'seven', 8: 'eight'}
print(s_1)	$s_2.update(s_2)$

* when we use dictionary method with update fn they take only key values.

adding & Removing element from set :-

① add()	s = {'a', 'b', 'c', 'd', 'e'}
② clear()	s.add('f')
③ discard()	print(s)
④ pop()	s.discard('a')
⑤ remove() → give error	s.remove('e')
remove('A')	s.pop()
discard('a')	s.clear()
	print(s)

- difference()

The difference b/w the two set's in python is equal to the difference b/w the no. of elements in two sets. The function difference() returns a set that is the difference b/w two set.

Example
Set A = {'00', 20, 30, 40, 50}
Set B = {20, 30, 40, 90, 100}
A-B
B-A = {90}

```
print(A.difference_update(B))
print(A.difference_update(A))
```

The difference update method helps in an in-place of way of differentiate the set. if A & B are two sets the set difference method will get the (A-B) & will return a new set ; the set difference update method modify the existing set.

Date
Page No.

Frozenset()

The frozenset is an inbuilt function in python which take as iterable object as input & make them immutable, simply it freezes the iterable object & make them unchangeable.

Syntax:-

```
frozenset (iterable_object_name)
S = [1,2,3,4,5,6,7]
fs = frozenset(s)
print (fs)
```

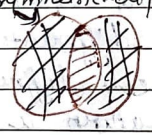
Intersection

Intersection function return a new set with an element that is common to all sets.

```
A = [100, 20, 30, 40, 50]
B = [100, 30, 80, 40, 60]
C = [200, 400, 30, 80]
print(A.intersection(B))
print(B.intersection(A,C))
print(A&B)  [By using & you perform intersection]
print(A&B&C)
```

Symmetric-difference -

```
print(A.symmetric_diff)
```



Date
Page No.

Isdisjoint() function:

is disjoint() function check whether the two sets are disjoint or not if it is disjoint then it returns true otherwise it will return false.
Two sets are said to be disjoint when there intersection is null.

Ex:- A = [4,5,6,8]
B = [10,9,20]
print(A.isdisjoint(B))
⇒ True

Lambda Function :-

Lambda function is used to define anonymous function anonymous function are function that have no name. A lambda function can take any no. of argument but can only have one expression.

Syntax Lambda argument : Expression

① x = lambda a : a + 10
print(x(5))

② x = lambda a,b : a * b
print(x(5,6))

③ Addition of 3 no.s
x = lambda a,b,c : a + b + c
print(x(5,6,2))

Date
Page No.

Filter Function :-

The element of particular

The Filter function returns an iterator where the items are filtered through a function to test if the item is accepted or not.

Syntax filter (function, sequence)

function - Tests if each element of a sequence is true or not.

Sequence - which need to be filtered
it is a collection of element which need to be filtered which can be list, set, tuple or container of any iterator, this function can return an iterator that already filtered.

Example - Syntax filter (function, sequence)

Example

```
ages = [5,12,17,18,24,32]
def myfun(x)
    if x > 18:
        return true
    else :
        return false
adults = list (filter (myfunc , ages))
for x in adults
    print (x).
```

o/p :-

```
18
24
32
```


Picture card matching game work

Date
Page No.

Map Function -

A Map function execute a specified func for each item in a iterator the item is send to the func as parameters.

Syntax :- Map (func, seq)

Example :-

```
ages = [5, 12, 17, 18, 24, 32]
if  x < 18,
    Return False
else:
    return true
def myfunct(x):
    return x * x
```

Same functionioning by using lambda function :-

```
ages = [5, 12, 17, 18, 24, 32]

adults = filter (lamda a: a>18 , ages)
Square = list (map (lambda a: a*a , adults))
print (square)
```

Reduce Function

It is used to apply a particular function passed in its arguments to all of the list elements mentioned in the Sequenced passed along.
This function is defined in Functool module.
↓
import this tool.

Date
Page No.

Example

```
# from functools import reduce :-

def myfunc(a,b):
    return a*b
val = [1,2,3,4,5]
add = reduce(myfunc, val)
print("Addition is", add)
#
```

Same function with lamba function :-

```
from functools import reduce
val = [1,2,3,4,5]
add = reduce(lambda a,b: a+b, val)
print("Addition is", add)
```


Date 9/May/22
Page No.

List Comprehension.

LC is an elegant way to define and create list in python we can create list just like mathematical statements and in one line only.
A LC is generally consist of 3 parts.

- Output expression
- input Sequence
- A variable representing member of input sequence
- An optional predicate part

Example:-

```
list = [x**2 for x in range(1,11) if x%2==1]
```

where

x**2 is output expression
range(1,11) - input sequence
x is variable that represent member of list.

Create a list contain sequence of odd numbers from 1 to 10 using for loop without list comprehension.

```
list = []
for x in range(1,11)
    if x%2==1
    list.append(x**2)
print list
```

```
l = []
for i in range(1 to 11)
    if x%2==1:
        a = i*i
    l.append(a)
```