Date
Page No.

output :-
1 : 3
2 : 2
3 : 2
4 : 2
5 : 2

9 l = ["Rajesh Rana", 21, 8.82]
t = tuple(l)
print(t)
output :- ("Rajesh Rana", 21, 8.82)

~~l = "Rajesh Rana"~~

10 l = [1, 3, 4, 10, 2, 5]
print(l.sort())
output :- [1, 2, 3, 4, 5, 10]

unit - 2

statements are of 3 types :-
→ sequential statements
→ selection control statements
→ Loop control statements.

selection control statements :-
if statement :
It can used in 4 ways :
1) if statement,
2) if ... else,
3) elif
4) nested if.

Date
Page No.

Syntax of if :

```
if (condition):
    statement
```

EX1 :-

```
x=20
y=10
if (x>y):
    print("%d is big" %x)
    (or)
    print(f"{x} is big")
```

output :- 20 is big

syntax : f"some text {variable} some text"

f before the string tells python : this is a formatted string
Anything inside {} will be replaced by the value of the variable
No need for % formatting anymore.

Syntax of if-else :

```
if (condition)
    statement 1
else:
    statement2
```

ex1 :

```
x= int (input ("enter a number"))
y= int (input ("enter a number"))
```

Date
Page No.

```
if (x > y):
    print ("x is big", x)
else:
    print (f"{y} is big)
```

output :-

Enter a number 10
enter a number 20
20 is big

EX 2 :-

```
x = input ("Enter an alphabet: ")
if x in "aeiou":
    print ("vowel")
else:
    print ("consonant")
```

output :-

Enter an alphabet : R
consonant

Q A variable location is bank and if location is market print "purchase vegetable" other print "withdraw money"

Ans

```
loc = "bank"
if loc == "bank"
    print ("withdraw money")
else:
    print ("purchase vegetable")
```

Date
Page No.

Syntax of elif :-

```
if (condition 1):
    statement 1
elif (condition 2):
    statement 2
else:
    statement 3
```

EX :-

```
a = int (input ("Enter a number :"))
b = int (input ("Enter a number :"))
c = int (input ("Enter a number :"))
if (a>b and a>c):
    print (f"{a} is greater")
elif (b>c):
    print (f"{b} is greater")
else:
    print (f"{c} is greater")
```

output :-

Enter a number : 10
Enter a number : 30
Enter a number 20
30 is greater.

Q wap to input a digit within 0 to 6. Display week day.
Ans EX 0 for sunday, 1 for monday etc (use elif ladder)

```
a = int (input ("Enter a number :"))
if a == 0:
    print ("It is sunday")
elif a == 1:
    print ("It is monday")
```

Date
Page No.

```
elif a==2:
    Print ("It is tuesday")
elif a==3:
    Print ("It is wednesday")
elif a==4:
    Print ("It is thursday")
elif a==5:
    Print ("It is Friday")
elif a==6:
    Print ("It is saturday")
else:
    Print ("Invalid day")
```

output:-

Enter a number : 3
It is wednesday

7 Write to input an arithmetic operator symbol and two integers calculate and display the result as per the given operation (use elif ladder)

Ans

```
a = int (input ("Enter 1st number"))
b = int (input ("Enter 2nd number"))
o = input ("Enter an operator (+, -, *, /, %)")
if o == "+":
    print (f"{a}+{b} = {(a+b)}")
elif o == "-":
    Print (f"{a}-{b} = {(a-b)}")
elif o == "*":
    print (f"{a}*{b} = {(a*b)}")
elif o == "/":
    Print (f"{a} / {b} = {(a/b)}")
elif o == "%":
    print (f"{a} % {b} = {(a%b)}")
```

Date
Page No.

output:-
Enter 1st number : 2
Enter 2nd number : 3
Enter an operator (+, -, *, /, %) : +
2+3=5

sort

List = [2.5, 3.5, 10.0]
List 1 = [2, 4, 6]
List 2 = ["APP", "APS", "APPL"]
List.sort()
List2.sort()
list2.sort()
Print (list)
Print (list1)
Print (list2)

Q WAP to check whether a given year is leap year or not.

Ans a = int (input ("Enter a year : "))
if (a%4==0 and (a%400==0 or a%100!=0)):
    print (f"{a} is a leap year")
else:
    print (f"{a} is not a leap year")

output :-
Enter a year : 2004
2004 is a leap year.

Date
Page No.

Shorthand if:
If we have only one statement to execute then we can put it in the same line as:
Ex:
a=20; b=10
if (a>b): Print("%d is big" %a)

Short hand if--else:-
we can write if--else also in same line as:
ex: find greatest among two numbers
a = int(input("enter 1st no"))
b= int(input("enter 2nd no"))
print("a") if (a>b) else Print("b")

Q write to greatest among 3 numbers using short hand if:-
Ans
a= int(input("Enter 1st number"))
b= int(input("Enter 2nd number"))
c = int(input("Enter 3rd number"))
if (a>b and a>c): print("%d is big" %a)
if (b>a and b>c): Print(f"{b} is big")
if (c>a and c>b): print(f"{c} is big")

print("A") if (a>b and a>c) else Print("B") if (b>c) else Print("C")

Date
Page No.

Python Loops:-

It is having two loop controls. they are:- • while loop
• For loop.

while loop:- we can execute a set of statements as long as the condition is true.

Syntax:

```
while (condition):
    statement block
```

Q wAP to print natural number 1 to 10.

Ans
```
i= 1
while (i<=10):
    Print (i, end=", ")
    i=i+1
```

output:- 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Q wAP to print the even numbers between 1 and 10.

Ans
```
i= 1
while (i<=10):
    if (i%2==0):
        Print (i, end=", ")
    i=i+1
```

Q wAP to print the odd numbers between 1 to 10

Ans
```
i=1
while (i <=10):
    if (i%2!=0):
        print (i, end=", ")
    i=i+1
```

Date
Page No.

```
name = ""
while (name != "durga"):
    name = input ("enter name:")
print ("thanks for confirmation")
```

```
i = 1
while (i <= 10):
    print (f"n x {i} = {n*i}")
    i = i+1
```

infinite loops:

If the condition given in while will never become false then it will never terminate.

Ex 1:

```
while (1):
    print ("never ends")
```

ex 2:

```
v = 2
while (v != 3)
    i = int (input ("enter a no"))
    print ("the no is", i//2)
```

12-12-25

Key=str

Date
Page No.

sort():-
List_name.sort(key=None, reverse=False)
sort() sorts the list in-place, meaning it changes the original list and does not return a new list.
It works only on lists.

Parameters:-
1. key (optional)
A function that decides the sorting logic.

2. reverse (optional, default=False)
· If False → ascending order
· If True → descending order
l=[3,1,4,2]
l.sort()
print(l)
#output:[1,2,3,4]
words=["cat","elephant","dog"]
words.sort(key=len)
print(words)
#output ["cat","dog","elephant"]

l=[3,1,4,2]
l.sort(reverse=True)
print(l)
#output: [4,3,2,1]

cannot sort mixed datatypes directly with sort()
Can sort mixed datatypes only if you convert them with a key function.

Date
Page No.

#1. Sorting integers
numbs = [5, 2, 9, 1]
numbs.sort()
Print("Sorted integers:", numbs)
o/p: sorted integers: [1, 2, 5, 9]

#2 Sorting strings
words = ["banana", "apple", "kiwi"]
words.sort()
Print("Sorted strings:", words)
#o/p: sorted strings: ["apple", "banana", "kiwi"]

#3 Sorting strings by length
words_len = ["apple", "kiwi", "banana"]
words_len.sort(key=len)
Print("Strings sorted by length:", words_len)
o/p: Strings sorted by length: ["kiwi", "apple", "banana"]

#4 Sorting tuples
tuples = [(3, 2), (1, 4), (2, 1)]
tuples.sort() #sorts by first element
Print("Tuples sorted by first value:", tuples)
o/p" Tuples sorted by first value: [(1, 4), (2, 1), (3, 2)]

#5 Sorting a set (use sorted())
S = {4, 1, 7, 2}
Print("sorted set:", sorted(S))
o/p: Sorted set: [1, 2, 4, 7]

Date
Page No.

~~#6. Sorting dictionary by values~~
~~print("dictionary sorted by values:", sorted(d, key=d.~~

#6. Sorting dictionary (keys)
d = {"k3": 3, "k1": 1, "k2": 2}
print("Dictionary keys sorted:", sorted(d))
o/p: Dictionary keys sorted: ['k1', 'k2', 'k3']

#7 Sorting dictionary by values
print("Dictionary sorted by values", sorted(d, key=d.get))

o/p:
Dictionary sorted by values: ['k1', 'k2', 'k3']

#8. Sorting mixed datatype using key=str(allowed)
mixed = [10, "apple", 5, "cat"]
mixed_sorted = sorted(mixed, key=str)
print("mixed datatypes sorted using key=str",
mixed_sorted)

o/p mixed datatypes sorted using key=str:
[10, 5, "apple", "cat"]

#9. Sorting numeric strings + integers using key=int
(only if convertible)
mixed_numeric ["10", "5", 20, 3]
mixed_numeric_sorted = sorted(mixed_numeric, key=int)
print("Mixed numeric data sorted using key=int",
mixed_numeric_sorted)

o/p:
mixed numeric data sorted using key=int:
[3, "5", "10", 20]

Date
Page No.

sorted():

sorted(iterable, key = None, reverse = False)

sorted() returns a new sorted list from the items in any iterable (list, tuple, string, dictionary, set, etc.).
It does NOT modify the original iterable.

parameters:-

1. iterable

The sequence or collection you want to sort.

Ex:

```
numbers = [5, 2, 9, 1]
Print (sorted(numbers))
o/p : [1, 2, 5, 9]
```

2. key (optional)

A function that decides the sorting logic.

Examples:

```
sorted ["apple", "banana", "kiwi"], key = len)
#Sorts based on length of each word
o/p : ["kiwi", "apple", "banana"]
```

3. reverse (optional, default = False)

if False -> ascending order
if True -> descending order

Examples :-

```
print(sorted ([5, 2, 9, 1], reverse = False))
o/p : [1, 2, 5, 9]
Print (sorted ([5, 2, 9, 1], reverse = True))
o/p : [9, 5, 2, 1]
```

Date
Page No.

Using of else with while loop:-
We can write else block for while loop. Here the else block will be executed when the while loop condition becomes false.

EX1:

```
i=1
while (i<=5):
    print(i) # 1 2 3 4 5
    i=i+1
else:
    print("loop ended")
```

- when a while loop is terminated with break statement then in that instance the else block will not be executed.

EX2:

```
i=1
while (i<=5):
    print(i)
    i=i+1
    if (i==3):
        break
else:
    print ("loop ended")
```

#-

Date
Page No.

The break statement :-
Example
Exit the loop when i is 3 :-

```
i=1
while i<6:
    print(i)
    if i==3:
        break
    i+=1
```

The continue statement
with the continue statement we can stop the current iteration, and continue with the next :-
Example
continue to the next iteration if i is 3 :-

```
i=0
while (i<6):
    i+=1
    if (i==3):
        continue
print(i)
```

Q wap to test a number is Prime number or not :-
Ans

```
n=int(input("Enter a number"))
i=2
flag=0
while (i<=n**0.5):
    if (n%i==0):
        flag=1
        break
    i+=1
```

:)-1

Date
Page No.

```
If (flag==0):
    Print (f"{n} is a Prime number")
else:
    Print (f"{n} is not a prime number")


i=2
while i<num:
    if num%i==0:
        Print (num, "is not a prime number")
        break
    i+=1
else:
    print (num, "is a prime number")
```

Q WAP to test a number is Armstrong or not

Ans
```
n=int(input("Enter a number:"))
count = [redacted] len(str(n))
res=0, temp=n
while(n!=0):
    res+=((n%10)**count)
    n=n//10
If (res==temp):
    Print (f"{temp} is an armstrong number")
else:
    print (f"{temp} is not an armstrong number")
```

Date
Page No.

2 Test a number is Palindrome or not
2 Test a string is palindrome or not

Python For Loops

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

The syntax of for loop in Python is given below.

```
for iterating_var in sequence:
        statement(s)
```

Example-1

Print each fruit in a fruit list.

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    print(x)
```

Example-2

```
>>> # Measuring each strings:
words = ['cat', 'window', 'defenestrate']
for w in words:
    print(w, len(w))
```

Example-3

```
tuple_1 = (1, 2, 3)
for item in tuple_1:
    print(item)
```

Date
Page No.

Example 4
mylist1 = [(1,2), (3,4), (5,6)]
for (a,b) in mylist1:
    print(a)
    print(b)

The print(a) statement prints the first element of each tuple.
The print(b) statement prints the second element of each tuple.

Example 5:-
mylist1 = [(1,2,3), (4,5,6), (7,8,9)]
for a,b,c in mylist1
    print(b)
# 2 5 8

Example 6
dict1 = {'K1': 100, 'K2': 200, 'K3': 300}
for i in dict1:
    print(i)

6-1-26

Date
Page No.

Q WAP to create 2 list and create a 3rd list which should contain the data of list 1 and list 2

Ans
```
l1 = [1,2,3,4,5]
l2 = [6,7,8,9,10]
l3 = l1+l2
print(l3)
```

Q WAP to take a single digit number from the keyboard and print its value in english words

Ans
```
l= [zero, one, two, three, four, five, six, seven, eight, nine]
n= int (input("Enter a single digit number:-"))
print ("It is :-", l[n])
```

using elif :-
```
n=int (input ("Enter a single digit number:-"))
if n==0:
    print ("It is zero")
elif n==1:
    print("It is one")
elif n==2:
    print ("It is two")
elif n==3:
    print ("It is three")
elif n==4:
    print ("It is four")
elif n==5:
    print ("It is five")
elif n==6:
    print ("It is six")
```

Date
Page No.

```
elif n==7:
    print ("It is seven")
elif n==8:
    print ("It is eight")
elif n==9:
    print ("It is nine")
else:
    print ("It is not a valid number")
```

Q WAP to print the characters of a given string.

Ans
```
s=input("Enter a string")
for i in range(len(s)):
    print(s[i])
```

```
for i in s:
    print(i)
```

Q WAP to print characters present in the string index wise.

Ans
```
s=input("Enter a string")
for i in range(len(s))
    print(s[i], i)
    print(f"The character present in {i} index is {s[i]}")
    print("The character present in", i, "index is", s[i])
```

Date
Page No.

The range() function

To loop a set of statements for a specified number of times, we can use the range() function.

By default the starting value is 0, and increments by 1 (by default), and ends at a specified number.

Syntax:-

Range (starting value, last value, change in value)

EX1:-

```
for x in range(6):
    print (x)
```

output:- 0, 1, 2, 3, 4, 5

EX2:-

Using the start parameter and increment value:-

EX: Increment the sequence with 3 (default is 1):

```
for x in range (2, 30, 3):
    print (x)
```

output:- 2, 5, 8, 11, 14, 17, 20, 23, 26, 29

Ex 3:-

```
for x in range (100, 0, -5):
    print (x)
```

output:- 100, 95, 90, 85, 80 - - - - 10, 5

Date
Page No.

Q wap to display odd number from 0 to 20 using range()

Ans
```
for i in range(1,20,2):
    print(i)
```

For even:-

Ans
```
for i in range(2,21,2):
    print(i)
```

zip():- this method is used with lists to combine multiple lists together.

Ex:-
```
mylist1 = [1,2,3]
mylist2 = ['a','b','c']
for item in zip(mylist1, mylist2):
    print(item)
```

o/p:-
```
(1, 'a')
(2, 'b')
(3, 'c')
```

The pass statement:-

for loops cannot be empty, but if you for some reason have a for loop with no content, put the pass statement to avoid getting error.

Ex:-
```
for x in range(1,100,3):
    pass
```

Ex:-
```
for x in [0,100,2]:
    pass
```

Date
Page No.

Nested for loop in python :-

The syntax of the nested for loop in python is given below.

```
for iterating_var1 in sequence:
    for iterating_var2 in sequence:
        #block of statements
    #other statements
```

Ex 1 :- generate a shape of triangle using * :-

```
n = int(input("enter the no.of rows:"))
i,j = 0,0
for i in range(n):
    print()
    for j in range(i+1):
        print("*", end=" ")
```

Note :- what does print("*", end=" ") mean?
Normally, print() adds a newline (\n) after printing. So every print appears on new line.
You are telling python:
Print "*"
Do not move to a new line, instead, add a space after printing. So multiple prints will stay on the same line.

Q WAP to print all the words that have even number of characters from a sentence.

Ans
```
s = Input("Enter a sentence")
for i in s.split():
    if len(i)%2==0:
        print(i)
```

Date
Page No.

Python functions :-
A function can be defined as an organised block of codes which is written once and called many times.
Its main advantage is code reusability.
Types of functions :-
→ Buil in functions : They are the library functions which we are using such as : abs(), max(), min() etc.

→ User defined functions : They are defined by programmer

How to create a function :-
def keyword is used.

Syntax :-
def <function name> (parameters) :
    statements in body
    return (expression)

- Here parameters are the variables to be passed.
- return is used to return a value.
- The body statements are to be written with proper indenting.

Parameters :- Using these we pass input data to a function. The parameters are actual parameters and formal parameters.
The actual parameters :- are the variables we represent in a function calling statement.
The formal parameters :- are the variables which we represent in the function body definition.

Date
Page No.

EX1:-

```
def func():
    print("hello Python")
func()
```

EX2:-

```
def add(a,b):  #formal parameters
    c=a+b
    return(c)
z=add(10,20) #actual parameters
print(z)
```

function calling statement:-

A function must be defined before calling it.
Else Python will give error.

EX1:-

```
func()
```

Return statement:-

Return statement is used to come out of function and return back to the place where it was called. Syntax:

Return <expression list>

EX:-

```
return x
```

when there is no return value then the return statement returns None object.

Date
Page No.

Ex:-
```
def greet(x):
    print("Good morning", x)
    return
print(greet "sam")
```

Ex2:- a function to find absolute value.
```
def absvalue(num):
    if num >=0:
        return num
    else:
        return (-num)

print(absvalue(-2))
```

This function returns the absolute value of a number.
Absolute value:
A positive number stays positive.
A negative number becomes positive.

Returning multiple values from a function:-
In other languages like C, C++ and java, function can return atmost one value.
But in Python, a function can return any number of values.
Eg1:
```
def sum_sub(a,b):
    sum = a+b
    sub = a-b
    return sum, sub
x, y = sum_sub(10,20)
print("The sum is:", x)
print("The subtraction is:", y)
```

Date
Page No.

Q wap to define a calculator function by accepting two parameters and should perform addition, subtraction, multiplication and division.

Ans
```
def calc (a,b):
    sum=a+b
    sub=a-b
    mult=a*b
    div=a/b
    return sum, sub, mult, div
x, y, z, a = calc (10,20)
print ("The sum is :-", x)
Print ("The subtraction is :-", y)
print ("The multiplication is :-", z)
print ("The division is :-", a)
```

(or)
```
t = calc (10,20)
for i in t:
    Print (i)
```

The variables can be local and global.

Local variables:
-> The local variables defined inside a function and are not visible outside.
-> The scope and life time is local.

Global variables:
-> A variable written outside having a global scope.
-> To get accessibility we need to use a keyword global.

Date
Page No.

Ex1:-

```
def myfun():
    x=10
    print("values inside", x)

x=20
myfun()
print("value outside", x)
```

o/p:

values inside 10
value outside 20

a.1.26

Arguments:-

An argument is a value that is passed to a function when it is called. It might be a variable, value or object passed to a function or method as input.

Function arguments:-

They are 4 types:-

1. positional arguments or required arguments
2. keyword arguments
3. default arguments
4. arbitrary arguments (variable length args and keyword arguments)

1. Positional arguments or required arguments:-

When you call a function with the required no. of arguments to be passed without using keywords, they are called positional arguments. Positional arguments must be in the correct order.

Date
Page No.

Ex:-
```
def area (l,b):
    print ("Area of a rectangle =", l*b)
#function calling statement
area (10,20)
```

O/P:-
Area of a rectangle = 200

Ex2:-
```
def fun(s1,s2):
    print (s1+s2)
fun ("Python", "language")
fun ("Python")#, "language")
```

output:-
Pythonlanguage
TypeError: fun() missing 1 required positional argument: 's2'

② Keyword Arguments:-
Keyword Arguments is an argument passed to a function or method which is preceded by a keyword and equal to sign(=).
The order of keyword argument with respect to another keyword argument does not matter because the values are being explicitely assigned.

```
def student (name, rank, point):
    print ("Student name =", name)
    print ("Student rank =", rank)
    print ("Student points =", point)
```

Date
Page No.

```
greet (name = "sree", rank = 8, point = 6)
greet (rank = 6, point = 7, name = "ravi")
```

Output :-

```
student name = sree
student rank = 8
student points = 6
student name = ravi
student rank = 6
student points = 7
```

Ex2 :-

```
def nameAge (name, age):
    print ("Hi, I am", name)
    print ("My age is", age)
nameAge (name = "prince", age = 20)
nameAge (age = 20, name = "prince")
```

Output :-

```
Hi, I am prince
My age is 20
Hi, I am prince
My age is 20
```

(3) Default Arguments

In Python, we can provide default values to function arguments. We use the = operator to provide default values. If the function is called without an argument, it uses the default value:

Date
Page No.

Ex:-

```
def add_numbers(a=7, b=8)
    sum = a+b
    print('sum:', sum)
# function call with two arguments
add_numbers(2,3)
#function call with one argument
add_numbers(a=2)
#function call with no arguments
add_numbers()
```

O/P:-

```
sum = 5
sum = 10
sum = 15
```

(9) Arbitrary arguments:-

In Python, arbitrary arguments (also known as variable-length arguments) allow a function to accept any number of inputs. This is achieved using two special syntaxes: *args for positional arguments and **kwargs for keyword arguments. The single ~~astrik~~ asterisk (*) before a parameter name in a function definition collects any number of non-keyword (positional) arguments into a tuple inside the function. The conventional name is args, but any valid variable name can be used (e.g. *numbers, *values).

```
def add_all_numbers(*args):
    """Calculates the sum of an arbitrary number of inputs."""
    total = 0
    for num in args:  # args is a tuple
        total += num
    return total
```

Date
Page No.

#call the function with a varying number of arguments.
print(add-all-numbers(10,20,30))
print(add-all-numbers(1,2,3,4,5))

output:-
60
15

Ex2:-
def total(*args):
    s=0
    for num in args:
        s=num
    print("sum is:", s)

total(20,30)
total(2,3,4,5)

output:-
sum is:50
sum is:14

Real life examples: shopping bill:-
def shopping-bill(*prices):
    print("Prices:", prices)
    print("Total =", sum(prices))
shopping-bill(120,350,80)
shopping-bill(200,150)

output:-
Prices: (120,350,80)
Total = 550

Prices: (200,150)
Total = 350

Date
Page No.

Using Loop with *args :-

```
def show_names (*names):
    for name in names:
        print(name)
show_names ("sree", "Ravi", "goutham")
```

Output :-

Sree
Ravi
Goutham

**kwargs (Arbitrary ~~Keyword~~ Keyword Arguments)

The double asterisk (**) before a parameter name collects any number of keyword arguments (arguments passed as key=value) into a dictionary inside the function. The conventional name is kwargs, but other names work too (e.g. **details).

**kwargs is always a dictionary
order of output follows the order of arguments passed.

EX :-

```
def fun (**kwargs) #inside the function kwargs becomes a dictionary.
    for k, val in kwargs.items():
        print (k, "=", val) #Prints each key with its
                            corresponding value.

fun (s1="Python", s2='is', s3='Awesome')
```

OUTPUT :- s1 = Python
s2 = is
s3 = Awesome

Date
Page No.

Real-life examples of **kwargs in Python.

Student Details

```
def student_details(**kwargs):
    for key, value in kwargs.items():
        print(key, ':', value)

student_details(
    name = "Rajesh",
    age = 21,
    course = "Python",
    email = "Raj@gmail.com"
    city = "Bhadrak"
)
```

output :-

```
name : Rajesh
age : 21
course : Python
email : Raj@gmail.com
city : "Bhadrak"
```

Real life example of cab booking :-

```
def cab_booking(**info):
    for k, v in info.items():
        print(k, ":", v)
cab_booking(
    passenger = "Rahul",
    pickup = "Railway station",
    drop = "Airport",
    ac = True,
    payment = "UPI"
)
```

Date
Page No.

Anonymous function :-

Lambda function

A lambda function is a small anonymous function (function without a name).

Its main purpose is one time usage or instant.

Syntax :-

Lambda input arguments : Expression

Normal function	Lambda function
def square(n): return n*n print(square(4)) O/P : 16	s = lambda n : n*n print(s(4)) O/P : 16
def add(a,b): return a+b add(10,5)	add = lambda a,b : a+b print(add(10,5))

def biggest(a,b):
  if a>b:
  return a
  else:
  return b
print(f"The biggest of {a} and {b} is {biggest(a,b)}")
	biggest = lambda a,b : a if a>b else b
	print("Biggest :", biggest(a,b))

Note :- A lambda function internally returns the value of the expression written in it. Therefore, we are not required to write the return statement explicitely. Lambda functions can contain only a single expression.

Date
Page No.

We can lambda functions with map(), filter() and reduce()

1. map() :- map() applies a function to each element of an iterable (list, tuple, etc).
2. A new sequence is created from the existing data. From each element of sequence a new sequence is generated.

Syntax:- map(function, sequence)

Ex:-
```
l = [1, 2, 3, 4, 5]
l1 = list(map(lambda n: n*n, l))
Print(l1)
```

O/P :- [1, 4, 9, 16, 25]

```
names = ["python", "java", "c"]
upper_names = list(map(lambda x: x.upper(), names))
print(upper_names)
```

O/P :- ["PYTHON", "JAVA", "C"]

map() can be applied on multiple sequences also.

```
l1 = [1, 2, 3, 4, 5]
l2 = [5, 10, 15, 20, 25]
l3 = list(map(lambda x, y: x*y, l1, l2))
Print(l3)
```

Output :-

[5, 20, 45, 80, 125]

Note :- Here x values are taken from l1 and y values are taken from l2.

Date
Page No.

Note:- If no. of elements are more in the 1st list than second one than they are ignored and common elements only executed, vice-versa.

Filter():-

filter() selects elements from an iterable based on a condition.

i.e It filters the values from the given sequence based on some condition.

Syntax: filter (function, sequence)

Ex:- Print even nos.

```
numbers = [1,2,3,4,5,6,7,8,9,10]
even_numbers = list (filter(lambda x: x%2==0, numbers))
print (even_numbers)
```

O/P:- 2,4,6,8,10