THEORY QUESTIONS

Q.1. What is the difference between a function and a method in Python?

Function- it is a standable block of reuseable code which can be used independently.
it can be used inside or outside of a class.

# example-
print('hello welcome')
len('pwskills')
type('pwskills')

Method- it is a function which is associated with a class
it is only used inside a specific class

example-

# string method-
s='rupak'
s.upper()   # all will be in upper case
s.title()   # first letter will be in upper case

# list method-
l=[1,5,6,7,8,4]

l.append(100)   # add 100 to the list
l.pop(3)   # remove the 3rd index element
l

Q.2. Explain the concept of function arguments and parameters in Python.

Parameter- a parameter is the variable define within the parentheses during the function definition.

Argument- an argument is a value that is passed to a function when it is called.

example-

```
def even(n):                    # n is a parameter
    if n%2==0:
        print("this is even no ")
    else:
        print('this is odd number')

even(7)                         # 7 is an argument
```

Q.3. What are the different ways to define and call a function in Python?

1. function without argument:
```
def test():
  print("hello this is a function without argument")
```

```
test()

# 2. function with argument
def greeting(name):
  print("hello " , name,"this is a function with argument")

greeting(’rupak’)


# 3. function with return statement:
def details(name,age):
  return f"{name} is {age} years old"
details(’avi’,29)


# 4. pass named argument:
def add(a,b):
  return a+b

add(b=5,a=6)

# 5. default argument:
def add(a,b=10):
  return a+b

add(1)
add(9,10)


# 6. taking input from the user:
def multiply():
  a=int(input(’enter the 1st number’))
  b=int(input(’enter the 2nd number’))
  return a*b

# multiply()

# 7. variable length argument
def sales(*amount,name=’poly diagnostic’):
    return amount,name

sales(200,300,600)

# 8. key word argument
def marks_in_subject(**kwargs):
   marks_list= []
   for subject,marks in kwargs.items():
    marks_list.append(marks)

   return marks_list

marks_in_subject(a=20,b=30,c=90)
```

```
# Q.4. What is the purpose of the ‘return‘ statement in a Python function?

#  By default function returns "nonetype" ,so when we use Functions in other operations
(like-concatination)
#  it will get error.
#  But return statement maintain/retain the data type of an argument . Thus the previous problem will not
occur.
#  this is the purpose of return statement.

# example-
def test():
   print(’hello test’)

type(test())

def test1():
   return "hello test1"

type(test1())

# Q.5. What are iterators in Python and how do they differ from iterables?

# ITERABLE= DATA STRUCTURE THAT CAN PRODUCE ITS ELEMENT ONE BY ONE.
# ITERATION= PROCESS OF PRODUCING THE ELEMENT .
# ITERATOR= IT IS A ITERABLE OBJECT ANYTHING WHICH WE CAN CONVERT IN ITERATOR
OBJECT.
#          IT IS A INTERMEDIATE STEP TO DO THE ITERATION.

# list ,tuple,string are the iterable

# example-
# A STRING
s=’pwskills’

# CREATE ITERABLE OBJECT
iter(s)                    # this is iterator*****

# convert iterable object into a list:
list(iter(s))

# next :
a=iter(s)
next(a)
next(a)
next(a)
next(a)
next(a)


# Q.6. Explain the concept of generators in Python and how they are defined.

# GENERATOR FUNCTION- A FUNCTION WHICH DOES NOT RETURN A SINGLE VALUE ,
# ISTEAD RETURNS ITERATOR OBJECT

# generator function
```

```
def sq_generator(n):
  for i in range(n):
    yield i**2


# create a generator object
gen_object=sq_generator(10000000000000)


# now use next:
next(gen_object)


# Q.7. What are the advantages of using generators over regular functions?

# IN NORMAL FUNCTION ,WHEN IT IS CALLED IT PERFORM ALL OF THE OPERATION AND SHOW THE DIRECT RESULT OF IT'S ARGUMENT.
# ON THE OTHER SIDE GENERATOR FUNCTION CREATE RETURNS A ITERATOR OBJECT , FORM WHICH WE CAN FETCH THE OUTPUT ONE BY ONE.
# SO WHEN THE OUTPUT INVOLVES LARGE CALCULATION (LERGE NUMBER OF OUTPUT) NORMAL FUNCTION FAILS,BUT GENERATOR FUNCTION
# SHOW THE OUTPUT ONE BY ONE. THIS IS THE ADVANTEAGE OF GENERATOR FUNCTION.

# example-
# fibonacchi function:
def fib(n):
  a=0
  b=1
  for i in range(n):
    yield a
    a, b=b,a+b


# create a generator object
obj=fib(100000000000000000)

#use next
next(obj)
next(obj)
next(obj)
next(obj)
next(obj)




# Q.8. What is a lambda function in Python and when is it typically used?

# DEFINITION-
#  A lambda function in Python is a small, anonymous function defined using the lambda keyword.
#  Unlike regular functions defined using def, a lambda function is defined in a single line of code
#  and can have any number of arguments but only one expression.
#  The expression is evaluated and returned when the lambda function is called.
```

```
# USES-
# 1.  Lambda functions are often used for short-term operations, where defining a full function would be unnecessarily verbose.
# 2.  often used as arguments to higher-order functions like map(), filter(), and reduce().

# example-1
# (add a number  to every number of a list)
# list
l=[3,6,8,9,10]

# map function;
list(map(lambda x: x+5,l))

# example-2(sum)
from functools import reduce

# list
l=[1,2,3,4,5]

#reduce function
reduce(lambda x,y : x+y,l)

# example-3 (finding even numbers)
l=[1,2,4,5,200,101,5,7,90,201,99]

# filter function
list(filter(lambda x : x%2==0,l))


# Q.9 Explain the purpose and usage of the ‘map()‘ function in Python.

# PURPOSE-
# in normal function for iteration, loop is used for several operarion and length of the function defintion
# could be larger ,to nullify the problem  map()  function is used,
# it takes two  argument -map( function/method , iterable )

# USES-
# map function is use exclusively with the iterable-like(sting,tuple,list)

# example- (add a number  to every number of a list)
# list
l=[3,6,8,9,10]

# map function;
list(map(lambda x: x+5,l))


# Q.10. What is the difference between ‘map()‘, ‘reduce()‘, and ‘filter()‘ functions in Python?

# 1. map()-Used for applying a function to every item of an iterable and returning a list of the results.
#    functionality- Transforms each element in the iterable independently.
#    return type- Returns a map object (an iterator).
```

```
# example-
# list
l=[1,2,3,4,5]

# square every element of a list
list(map(lambda x : x**2,l))


# 2. reduce()- Used for performing a cumulative computation to sequential pairs of items in an iterable,
#    reducing it to a single value.
#    functionality- Combines elements in the iterable to produce a single cumulative value.
#    return type-  Returns a single value.

# example-
from functools import reduce

# function-(sum)
l=[1,2,3,4,5]

# reduce function
reduce(lambda x,y : x+y,l)



# 3. filter()- Used for selecting items from an iterable based on a function that returns a boolean value.
#    functionality- Filters elements in the iterable based on a condition.
#    return type-  Returns a filter object (an iterator).

# example- (find positive numbers)
l=[-1,-3,4,-5,-7,90,30,-50]

# filter function
list(filter(lambda x : x>0,l))


# Q.11.  Using pen & Paper write the internal mechanism for sum operation using  reduce function on this
given
#        list:[47,11,42,13].



# PRACTICAL QUESTIONS:

# Q.1. Write a Python function that takes a list of numbers as input and returns the sum of all even
numbers in

# define the function
def even_sum(l):
  sum=0
  for i in l:
      if i%2==0:
        sum = sum + i
  return sum

# list-
```

```
l=[5,6,3,7,9,10]

# call the function-
even_sum(l)


# Q.2 Create a Python function that accepts a string and returns the reverse of that string.

# define the function
def reverse_string(s):
   s1=s[::-1]
   return s1

# string
s= 'pwskills'

# call the function
reverse_string(s)

# Q.3  Implement a Python function that takes a list of integers and returns a new list containing the
squares of
#       each number.

# define the function
def sq_list(l):
    sq =[]
    for i in l:
        sq.append(i**2)
    return sq

# list-
l=[5,6,3,7,9,10]

# call the function
sq_list(l)


# Q.4. Write a Python function that checks if a given number is prime or not from 1 to 200.




# Q.5. Create an iterator class in Python that generates the Fibonacci sequence up to a specified number
of
# terms.

# fibonacchi function: (consider as a class)
def fib(n):
  a=0
  b=1
  for i in range(n):
    yield a
    a, b=b,a+b
```

```
# this is the object of the Fibonacci sequence up to a specified number
obj=fib(100000000000000000)


# now show the output one by one
next(obj)
next(obj)
next(obj)
next(obj)
next(obj)


# Q.6. Write a generator function in Python that yields the powers of 2 up to a given exponent.

# define the function
def sq_generator(n):
  for i in range(n):
      yield i**2

# create the generator object
o=sq_generator(20000000)

# fetch the numbers
next(o)
next(o)
next(o)
next(o)

# Q.7. Implement a generator function that reads a file line by line and yields each line as a string.


# Q.8. Use a lambda function in Python to sort a list of tuples based on the second element of each tuple.

# list if tuples
l=[(2,5,6),(1,2,8,),(4,3,9,4),(2,4),(2,3),(2,1)]

# create the function
sorted(l,key=lambda x : x[1])

# show the result
result=sorted(l,key=lambda x : x[1])
result


# Q.9. Write a Python program that uses ‘map()‘ to convert a list of temperatures from Celsius to
Fahrenheit.

# list of celsius temperature
c=[100,40,90,70]

# map function to convet the celsius temperature to f:
list(map(lambda celsius : (celsius * 9/5)+32, c))


# Q.10. Create a Python program that uses ‘filter()‘ to remove all the vowels from a given string.
```

```
# string
s=’pwskills’

# list of vowels
vowels=[’a’,’e’,’i’,’o’,’u’]

# filter function
result=filter(lambda x : x not in vowels,s)

# now show the result
print(’’.join(result))
```