Q1. Write a function greet that takes two arguments: name (a string) and greeting (a string with a default value of “Hello”). The function should return a greeting message .

def greet(name,greeting):

return f"hello {name}, {greeting}"

greet("bob","Good Morning")

Output : 'hello bob, Good Morning'

Q2.  Create a function create\_profile that takes arguments for name, age, and city as named arguments. The function should return a string like "Name: Alice, Age: 25, City: New York". Ensure that the age argument has a default value of 18.

def create\_profile(name, city,age=18):

return f"Name: {name}, Age: {age}, City: {city}"

create\_profile("Alice", "New York" )

Output : 'Name: Alice, Age: 18, City: New York'

Q3. Write a function sum\_numbers that takes any number of positional arguments (args) and keyword arguments (\*kwargs). It should:

* Return the sum of all \*args if they are numbers.
* Return a dictionary of all keyword arguments.

def sum\_numbers(\*args,\*\*kwargs):

*# taking the sum of only positional argument*

sums=sum(arg for arg in args if isinstance(arg, (int, float)))

return sums, kwargs

sum\_numbers(6,5,8,8,x=5,y=6)

Output : (27, {'x': 5, 'y': 6})

Q4. Write a function that uses map and a lambda to return a list where each element is squared.

def square\_list(numbers):

return list(map(lambda x: x\*\*2, numbers))

print(square\_list([2,5,6,7]))

Output : [4, 25, 36, 49]

Q5. Write a function that filters out all odd numbers from a list using filter and a lambda function.

def odd\_out(numbers):

odd = list(filter(lambda x: x%2 !=0, numbers))

print(odd)

odd\_out([1,2,3,44,5])

Output : [1, 3, 5]

Q6. Create a list comprehension that takes a list of numbers and returns a list of their squares.

numbers = [2,5,6,8,9]

squares = [x\*\*2 for x in numbers]

print(squares)

Output : [4, 25, 36, 64, 81]

Q7.  Use a list comprehension to create a list of even numbers from 1 to 20.

def even\_out():

even=[x for x in range(1, 21) if x % 2 == 0]

return even

even\_out()

Output : [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]

Q8. Write a function file\_operations that creates a directory named "test\_folder" using os.makedirs(). Then, pause the execution for 3 seconds using time.sleep(), and finally, delete the directory using os.rmdir().

import os

import time

def file\_operations():

os.makedirs("test\_folder")

time.sleep(3)

os.rmdir("test\_folder")

file\_operations()

Output :

Q9. Write a Python script that imports only sleep from the time module and renames it to pause. Use it to pause execution for 2 seconds and print "Paused execution...".

from time import sleep as pause

pause(2)

print("Paused execution")

Output : Paused execution

Q10.  Write a recursive function flatten\_list that can flatten a nested list of any depth using \*args and \*\*kwargs.

def flatten\_list(\*args, \*\*kwargs):

flat\_list = []

*# Recursive helper function to flatten each element*

def flatten(item):

if isinstance(item, list):

for sub\_item in item:

flatten(sub\_item)

else:

flat\_list.append(item)

*# Flatten all the positional arguments*

for arg in args:

flatten(arg)

return flat\_list

result = flatten\_list([1, [2, 3, [4, 5], 6], 7, [8, [9, 10]]])

print(result)

Output : [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]