Q1.Given the below function, please describe test scenarios for the below function. NOTE: When describing your scenarios, it's OK to use simple scenario descriptions for your tests.

def format\_zipcode(zip\_code): if len(zip\_code) <= 5: return '{:>05}'.format(zip\_code) if len(zip\_code) == 10: return str(zip\_code) if len(zip\_code) == 9: return '{}-{}'.format(zip\_code[:5], zip\_code[5:])

Ans:

We need to perform testing with various inputs

1. If the number of characters in input(zip\_code) is less than or equal to 5 , zipcode should be prefix with 0

Examples: print(format\_zipcode(“12345”)) - returns 12345

print(format\_zipcode(“1234”)) - returns “01234”

print(format\_zipcode(“1”)) - returns “00001”

print(format\_zipcode(“0”)) - returns “00000”

print(format\_zipcode(“a”)) - returns “0000a”

1. If the number of characters in input(zip\_code) is greater than 6 and less than 9 , that function should return None

Examples: print(format\_zipcode("123456"))- returns “None”

print(format\_zipcode("12345a7"))- returns “None”

print(format\_zipcode("123456a8"))- returns “None”

1. If the number of characters in input(zip\_code) is equal to 9 , that function should return “first5chracters - next4characters”

Examples: print(format\_zipcode("123456789"))- returns “12345-6789”

print(format\_zipcode("012345a78"))- returns “01234-5a78”

print(format\_zipcode("12345678a"))- returns “12345-678a”

print(format\_zipcode("12345-678"))- returns “12345—678”

1. If the number of characters in input(zip\_code) is equal to 10, that function should return same str

Examples: print(format\_zipcode("0123456789"))- returns “0123456789”

print(format\_zipcode("abcdefghij")) - returns “abcdefghij”

5.If the number of characters in input(zip\_code) is greater than 10, that function should return “None”

Examples :

print(format\_zipcode("0123456789123")) – returns “None”

print(format\_zipcode("abcdefghijxcdvaaaaaaaaac12345677889900000sfegfdfdv")) - return “None”

Other scenarios:

We also needs to perform the function parameters check

Ex1:

print(format\_zipcode()) - Should throw “TypeError”

TypeError: format\_zipcode() missing 1 required positional argument: 'zip\_code'

Ex2:

print(format\_zipcode("1234567891","scqcq”))- Should throw “TypeError”

TypeError: format\_zipcode() takes 1 positional argument but 2 were given

Q 2. Programming Challenge Description: Write a function that accepts two arbitrary strings and returns a new string containing only the unique characters present in both inputs. Input: The function must accept two string parameters. Output: The function must return a string.

Ans:

def return\_uniquevalues(str1,str2):

word1,word2=set(str1),set(str2)

common\_chars=word1.intersection(word2)

uniquechars\_str1=word1-common\_chars

uniquechars\_str2=word2-common\_chars

uniquechars\_str=''.join(uniquechars\_str1)+''.join(uniquechars\_str2)

return uniquechars\_str

Q3. Programming Challenge Description: Write 4 functions based on the following requirements

a. Write a function that accepts an integer and returns True if the input is between 4 and 10, inclusive; otherwise, return False

b. Write a function to test if a list contains any items. Return 'EMPTY' if it does not contain any items and 'NOT EMPTY' if it does.

c. Write a function that accepts a file name and a string and writes the string to the file with the given file name.

d. Write a function that accepts a list and doubles each value in the list. When no input parameter is provided, return an empty list.

Input: N/A Output: N/A

Ans:

a.Write a function that accepts an integer and returns True if the input is between 4 and 10, inclusive; otherwise, return False

Ans:

1st Option:

def function\_a(val):

val=int(val)

if val>=4 and val<=10 :

return True

else:

return False

2nd Option:

def function\_a(val):

val=int(val)

if val in range(4,11):

return True

else:

return False

b. Write a function to test if a list contains any items. Return 'EMPTY' if it does not contain any items and 'NOT EMPTY' if it does.

Ans:

def function\_b(list1):

if len(list1)>0:

return 'NOT EMPTY'

else:

return 'EMPTY'

c. Write a function that accepts a file name and a string and writes the string to the file with the given file name.

Ans:

Explanation: append mode will create the the file, if the specified file is not exits.

If the file exists , append mode will append the string content to the file .

with option allows to automatically release of resources (closes the file) after usage .

def function\_c(filename,str):

try:

with open(filename,'a') as file1:

file1.write(str+"\n")

except IOError :

print(filename+" is not writable or filepath is not valid")

d. Write a function that accepts a list and doubles each value in the list. When no input parameter is provided, return an empty list.

Ans:

def function\_d(\*args):

if len(args)==1:

list1=[x\*2 for x in args[0] ]

return list1

if len(args)==0:

list1=[]

return list1

Q4. Programming Challenge Description: Using any object oriented language, please define a class model to implement the following specification: A company has a staffing model which includes Employees, Contractors, and Temporaries. We want an object model which implements this structure. The expected behavior for the classes is documented below: Input:

Ans:

Staffing\_Model.py

class Employee():

"""This is employee clas"""

def \_\_init\_\_(self,firstname,lastname,pay\_rate,yearly\_vacation):

"""This is constructor method of employee"""

self.firstname=firstname

self.lastname=lastname

self.pay\_rate=pay\_rate

self.yearly\_vacation=yearly\_vacation

def get\_name(self):

"""This function returns the name of contractor"""

return self.lastname+","+self.firstname

def get\_pay\_rate(self):

"""This function returns pay\_rate in hourly basis"""

return float(self.pay\_rate)

def get\_yearly\_vacation(self):

"""This function returns number of days yearly vacation"""

return int(self.yearly\_vacation)

class Contractor(Employee):

"""This is contractor clas inherited from employee"""

def \_\_init\_\_(self,firstname,lastname,pay\_rate,agencyname):

"""This is constructor method of Contractor"""

yearly\_vacation=0

super().\_\_init\_\_(firstname,lastname,pay\_rate,yearly\_vacation)

self.agencyname=agencyname

def get\_name(self):

"""This function returns the name of contractor"""

return super().get\_name()+"[C]"

def get\_agency(self):

"""This function returns the name of contractor agency """

return self.agencyname

class Temporary(Employee):

"""This is temporary clas inherited from employee"""

def \_\_init\_\_(self,firstname,lastname,pay\_rate,agencyname):

"""This is constructor method of Temporary"""

yearly\_vacation=0

super().\_\_init\_\_(firstname,lastname,pay\_rate,yearly\_vacation)

self.agencyname=agencyname

def get\_name(self):

"""This function returns the name of temporary employee"""

return super().get\_name()+"[T]"

def get\_agency(self):

"""This function returns the name of temporary agency """

return self.agencyname

Test Example:

e=Employee('user',"test",12,12)

print(e.get\_name())

print(e.get\_pay\_rate())

print(e.get\_yearly\_vacation())

e1=Contractor('user1',"test1",24,"ABC")

print(e1.get\_name())

print(e1.get\_pay\_rate())

print(e1.get\_yearly\_vacation())

print(e1.get\_agency())

e2=Temporary('user2',"test2",36,"ABCD")

print(e2.get\_name())

print(e2.get\_pay\_rate())

print(e2.get\_yearly\_vacation())

print(e2.get\_agency())

Q5. Programming Challenge Description: Given the API in the Input Description, write a script that successfully creates a new widget, confirms the creation of the widget, and then updates the description of the widget. The script should then confirm that the updated widget is in the full list of all widgets. Finally, delete the widget and confirm the deletion via status code.

Ans:

#!/usr/bin/env python

import requests

import json

from requests.packages.urllib3.exceptions import InsecureRequestWarning

requests.packages.urllib3.disable\_warnings(InsecureRequestWarning)

def get\_all\_widgets(url,headers):

print("url is "+url)

response=requests.get(url,headers=at\_headers,verify=False,timeout=60)

if response.status\_code==200:

print(response.json())

return response.json()

elif response.status\_code==400:

print("get all widgets call failed ")

return 400

def create\_widget(url,headers,payload):

response = requests.post(url, data=json.dumps(payload), headers=headers, verify=False)

if response.status\_code==201:

print(response.json())

widgetid = response.json()["widget\_id"]

print("Created widget with id: " + widgetid)

return widgetid

elif response.status\_code==400:

print("create widget operation failed")

return 400

def getwidget(url,widgetid,at\_headers):

if widgetid=="":

print("widget id is not sepcified.please specify the widget id")

return

else:

get\_url=url+widgetid

print(get\_url)

response=requests.get(get\_url,headers=at\_headers,verify=False)

if response.status\_code==200:

print(response.json())

elif response.status\_code==400:

print("specified widget is not exists")

return response

def updatewidget(url,widgetid,at\_headers,payload):

if widgetid=="":

print("widget id is not sepcified.please specify the widgetid")

return

else:

update\_url=url+str(widgetid)

print(update\_url)

response=requests.patch(update\_url,headers=at\_headers,data=json.dumps(payload),verify=False)

print(response.status\_code)

print(response.json())

if response.status\_code == 201:

print("widget updation is successful")

return 0

elif response.status\_code == 400:

print("specified widget is not updated ")

return 1

def delete\_widget(url,widgetid,at\_headers):

delete\_url=url+str(widgetid)

print(delete\_url)

response=requests.delete(get\_url,headers=at\_headers,verify=False)

if response.status\_code == 204:

print("widget deletion is successful")

return 0

elif response.status\_code == 400:

print("specified widget is not exists in rackspace")

return 1

def main():

url="https://dev.rackspace.example.com/widgets"

token="xxxxxxxx"

at\_headers = {

"Content-Type": "application/json",

"X-Auth-Token": token

}

payload = {

"widget\_name": "Test Widget1",

"description": "This is Testwidget"

}

update\_payload={

"description": "A new description"

}

###creation of new widget

widgetid\_or\_code=create\_widget(url,at\_headers,payload)

widget\_id=""

if widgetid\_or\_code != "400":

print("created wiget id is:"+widgetid\_or\_code)

widget\_id=widgetid\_or\_code

###updation of widget with widegt\_id got from POST response

update\_code=updatewidget(url,widget\_id,at\_headers,updatepayload)

####Validation of updated widget in list of all widgets

if update\_code == "0":

if ( get\_all\_widgets(url,at\_headers) != "400" ):

response\_json=get\_all\_widgets(url,at\_headers)

if any(x['widget\_id'] == widget\_id for x in response\_json):

print("widget with"+ widget\_id +"is updated and exists in list of widgets")

else:

print("widget with"+ widget\_id +"is not exists or timesout")

####deletion of widget

if delete\_widget(url,widget\_id,at\_headers) ==0:

print("deletion of widegt is success")

else :

print("widegt is not exists to delete ")

if \_\_name\_\_=='\_\_main\_\_':

main()