1. Sort the below list without using inbuilt function

I = [2,3,-5,-7,9,4,6,-1,-8,0]

for i in range(len(l)-1):

    for j in range(len(l)-1):

        if l[j+1]<l[j]:

            l[j],l[j+1]=l[j+1],l[j]

print(l)

**2. Define a function which returns a list contains only the palindrome strings from the list**

**of provided string elements**

**input : List of strings**

**output : List of palindrome strings**

# 2. Define a function which returns a list contains only the palindrome strings from the list

# of provided string elements

# input : List of strings

# output : List of palindrome strings

s=['hello','hannah','kayak','rotator','deed','hi','clear','wow']

def palindromes(s):

    l=[]

    for i in s:

        if i==i[::-1]:

            l.append(i)

    return l

print(palindromes(s))

**3. Define logic for identifying the even numbers and odd numbers from the given list and**

**generate a dictionary as follows**

**numbers = [4,5,7,2,9,8]**

**result = {“even”:[4,2,8],”odd”:[5,7,9]}**

# 3 Define logic for identifying the even numbers and odd numbers from the given list and

# generate a dictionary as follows

# numbers = [4,5,7,2,9,8]

# result = {“even”:[4,2,8],"odd":[5,7,9]}

n = [4,5,7,2,9,8]

def separator(n):

    d={'even':[],'odd':[]}

    for i in n:

        if i%2==0:

            d['even'].append(i)

        else:

            d['odd'].append(i)

    return d

print('result = ',separator(n))

**4.**

# Define a function which returns dictionary that stores the words and it’s length from

# the given text

# text = “Be happy”

# Expected Output : {"Be":2,”happy”:5}

text='Be happy'

def func(t):

    d={}

    l=[]

    t+=' '

    x=''

    for i in t:

        if i==' ':

            l.append(x)

            x=''

        else:

            x=x+i

    for i in l:

        d[i]=len(i)

    return d

print(func(text))

5

# 5.Let’s consider there is a list which contains usernames, You have received requirement

# to add one more username to the list (without using append and assignment approaches)

# input : [“user1”,”user2”]

# output : [“user1”,”user2”,”user3”]

l=["input1","input2"]

print(l)

l.extend(["input3"])

print(l)

6.

# 6.Define the logic for generating the email id based on username and department Get the username and department as a input and create a email id from it

# input :

# username : msys department: automation

# output:

# msys.automation@gmaiI.com

# Note : Generated email id should contain @ and should end with .com

username=input('username: ')

department=input('department: ')

def gmailgen(u,d):

    s='@gmail.com'

    return u+'.'+d+s

print(gmailgen(username,department))

7 In the given string, remove the special characters and add a space instead of that “Msys&Technologies@Chennai\*

# 7 In the given string, remove the special characters

# and add a space instead of that “Msys&Technologies@Chennai\*

import re

x="Msys&Technologies@Chennai\*"

op=re.sub(r'[^a-zA-Z0-9]',' ',x)

print(op)

#............or.................

# y=''

# for char in x:

#    if char.isalpha():

#     y+=char

#    else:

#     y+=' '

# print(y)

1. What is the return type of arbitrary positional arguments and arbitrary keyword arguments?

The return type of arbitrary positional arguments is Tuple

The return type of arbitrary keyword arguments is Dictionary

1. Given a string “abcde”, Display the output as “a1b2c3d4e5”

# 10.Generate a dictionary from the two given lists using dict function (without using for loops) and calculate the sum of all the values in the dictionary using reduce and anonymous concepts

# L1 = [“a”,”b”]    L2 = [1,2]

# Expected Output :

# data = {“a'.1, “b'.2} sum = 3

from functools import reduce

l1=['a','b']

l2=[1,2]

d=dict(zip(l1,l2))

sum=reduce(lambda x,y : x+y, d.values())

print('data =',d,'\n','sum=',sum)

1. Generate a dictionary from the two given lists using dict function (without using for loops) and calculate the sum of all the values in the dictionary using reduce and anonymous concepts

# 10.Generate a dictionary from the two given lists using dict function (without using for loops) and calculate the sum of all the values in the dictionary using reduce and anonymous concepts

# L1 = [“a”,”b”]    L2 = [1,2]

# Expected Output :

# data = {“a'.1, “b'.2} sum = 3

from functools import reduce

l1=['a','b']

l2=[1,2]

d=dict(zip(l1,l2))

sum=reduce(lambda x,y : x+y, d.values())

print('data =',d,'\n','sum=',sum)

**11.**

**add\_sub.py**

def addition(v1,v2):

    return v1+v2

def substraction(v1,v2):

    return v1-v2

**mul\_div.py**

def multiplication(v1,v2):

    return v1\*v2

def division(v1,v2):

    return v1/v2

**Main.py**

# 11.Define Calculator logic in such a way that addition and subtraction functions should be in

# one python file and multiplication and division should be in another python file,

# Access these functions and utilize them inside the main file.

from add\_sub import \*

from mul\_div import \*

v1=int(input('val 1:'))

v2=int(input('val 2:'))

op=input('operator:')

if op=='+':

    print(addition(v1,v2))

elif op=='-':

    print(substraction(v1,v2))

elif op=='\*':

    print(multiplication(v1,v2))

else:

    print(division(v1,v2))

**12.**

# 12.Solve the following scenarios

# •Let’s assume that there is a tuple containing username, You have got a requirement to add password also into it.

# Input : (“user1”)

# Output : (“user1”,”password1”)

input=('user1',)

op=input+('password',)

print(op)

print(type(op))

**B**

# •Below logic is failing with an error message, Instead of auto generated Error, Please display

# the user defined message saying “Error : Cannot concatenate String and Number”

# print(“msys” + 2007)

try:

    # print('msys')

    print("msys"+2007)

except TypeError as e:

    print('Cant concatinate a string with an integer')

else:

    pass

finally:

    print("Done")

13.Let’s assume there is function defined which expects only list as an argument, However there is chance that sometimes function will be called with different type of data, Now Fix this scenario to handle the other types without breaking the code

* + Scenario 1: If the argument provided is a list then, Print inside the function as “valid argument “
  + Scenario 2: if the argument provided is a different data type, then Print a message saying “ invalid argument, You have provided data type (str/int) “

# 13.Let’s assume there is function defined which expects only list as an argument, However there

# is chance that sometimes function will be called with different type of data, Now Fix this

# scenario to handle the other types without breaking the code

# Scenario 1: If the argument provided is a list then, Print inside the function as

# “valid argument“

# Scenario 2: if the argument provided is a different data type, then Print a message saying

# “ invalid argument,

#  You have provided data type (str/int) “

l=[]

s=''

t=12

def func(l):

    if type(l)==list :

        print("valid argument ")

    else:

        print(f'You have provided {type(l)}')

func(l)

func(t)

func(s)

1. Define a function which can read json file and displays the data present in it to the console in dictionary format

Note : Please create .json file and store the sample data in it and read the json file, display the data in dictionary format

1. Define logic for generating the random password with the provided length as an input

**16.**

# 16.Let’s consider there are two files, one file contains testnames, other file contains testnames and status for each one. Generate dictionary with key’s as testname and value as status

# Input :

# FiIe1.txt:

# test1 test2

# File2.txt:

# test1-pass test2-fail

# Output :

# { "test1" : "pass", "test2" : ”fail”}

with open('file1.txt','r') as f1:

    testcase=f1.read().split()

print(testcase)

d={}

with open('file2.txt','r') as f2:

    l= f2.read().split()

    for line in l:

        # print(line.split('-'))

        test,status=line.split('-')

        if test in testcase:

            d[test]=status

print(d)

**17.**Define the function which returns the counts of saturdays part of a year (year is an input [Ex: 2022])

# .Define the function which returns the counts of saturdays part of a year

# (year is an input [Ex: 2022])

import datetime

def count\_saturdays(year):

    saturdays = 0

    for month in range(1, 13):

        for day in range(1, 32):

            try:

                date = datetime.date(year, month, day)

                if date.weekday() == 5:

                    saturdays += 1

            except ValueError:

                pass

    return saturdays

print(count\_saturdays(int(input("enter any year: "))))

**18.**

# 18.Write sample code for reproducing the below errors and print the user defined messages with the use of exception handling concept

# •IndexError,TypeError,AttributeError,ValueError

a=[1,2,3]

try:

    x='hello'

    y=x[1]

    # y=x[10]           #index error

    # y+=1              #type error

    # print(int('a'))   #value error

    # print(a.upper())     # attributeerror

except IndexError as e:

    print('index error')

except TypeError as e:

    print("It is a type error")

except ValueError as e:

    print(" wrong value is given for typecasting")

except AttributeError as e:

    print("it is an attribute error")

else:

    print(y)

finally:

    print("Done")

**19**

# 19.Define a generator to print the numbers between o to n (including)

# which are divisible by 5 and should be even

def gen(n):

    for i in range(n):

        if i%2==0 and i%5==0 and  i!=0:

            yield i

n=int(input("enter the value: "))

g=gen(n)

for i in g:

    print(i,end=' ')

1. Define sample code to achieve the following OOPs concepts
   * Inheritance
   * Method Overriding
   * Encapsulation
   * Method overloading
   * Abstraction
2. Imagine a scenario where you are required to fetch the employee names who joined after 02 Sep 2022 and location is Hyderabad

empIoyee\_ data = {

“priya”:{

“location” : “Hyderabad” “joining\_date '. “05/09/2022”

“mahi”. (

“raja”: (

“location” : “Bangalore” “joining\_date '. “20/02/2023”

“location” : “Hyderabad” “joining\_date '. “14/10/2022”

“prabhu”:(

“location” : “Bangalore” “joining\_date '. “02/01/2023”

22.Define the logic for verifying whether URL is Valid or Invalid Requirements for Valid URL

* + Should not contain any Special characters [,\*,&,%,$,#,@,!] and Spaces
  + Should start with https://

Input : URLs will be stored inside a file , read the URLs from the input file [input.txt]

Output : Generate a .txt file which contains the information whether URL is valid or not ( URL, Status [valid/invalid])

Example:

Input

Input.txt [text file]

https://m http s://m

Output:

1. https://m, valid
2. http s://m, invalid

Note: Define the logic with different approaches [1. Using RegEx 2. Without RegEx]

with open('input.txt', 'r') as file1:

    urls = file1.readlines()

with open('output.txt', 'w') as file:

    for url in urls:

        if url.startswith('https://'):

            if any(char in url for char in ',\*&%$#@! ') or ' ' in url:

                file.write(f'{url}, invalid\n')

            else:

                file.write(f'{url}, valid\n')

        else:

            file.write(f'{url}, invalid\n')

f=open('output.txt')

print(f.read())

23.Write a logic for calculating the time taken for executing the python function

import time

starttime=time.time()

def lst(l):

    for x in range (l):

        print("running")

lst(22)

stoptime=time.time()

print("time taken for exicuting program is : ",stoptime-starttime)

1. Define a logic for identifying the different files (In different format:.csv, .txt) which are part of a directory

Input : You can create a directory and create the files with two different formats (Manually for the input)

Output : Create two different directories and store this files separately based on the extension

Example :

Input:

Assume file1.csv, file2.txt, file3.csv, file4.txt are present inside a directory (Any name)

Output:

CSV - [Directory with the name CSV]

file1.csv

file3.csv

TXT - [Directory with the name TXT] file4.txt

fiIe2.txt

import os

import shutil

files = [f for f in os.listdir() if '.jpg' and '.csv' in f.lower()]

for file in files:

    if file.endswith('.jpg'):

        new\_path1 = 'txt\_d/' + file

        os.rename(file, new\_path1)

    else:

        new\_path = 'csv\_d/' + file

    os.rename(file, new\_path)

**25**

# 25.Define a logic to print the combinations from the two the below input data Input :

# 'Department“. ['Bakkt’, 'Cisco'],

# ’Team’    : [’Red’, ’Yellow’, ’Black’]

# Output :

# { ’Department“. ‘Bakkt’, ’Team“. ’Red’},

# { ’Department“. ‘Bakkt’, 'Team“. ’Yellow’}

# { ’Department“. ‘Bakkt’, 'Team“. ’Block’}

# { ’Department“. ‘Cisco’, ’Team“. ’Red’}

# { ’Department“. ‘Cisco’, ’Team“. ’Block’}

# { ’Department“. ‘Cisco’, ’Team': ’Yellow’}

department=['Bakkt', 'Cisco']

team=['Red','Yelow','Black']

d={}

for i in department:

    d['department']=i

    for j in team:

        d['team']=j

        print(d)

**26**

#    26.Print the pattern Pattern for the input : 3

#   \*1

#   21\*

#   \*123

n=int(input('enter a number:'))

x=''

for i in range(1,n+1):

    x+=str(i)

    if i%2 != 0:

        print('\*'+x)

    else:

        print(x[::-1]+'\*')

1. Write a python-selenium script to get the distance between Chennai and Bangalore using google-map
2. Define division logic which should also handle the the scenario if input argument (second argument) is 0, Use the decorator concept to include this validation before proceeding further on the actual functionality
3. Find the element in a list using Binary Search Algorithm and return a tuple containing the element and its index.
4. Read data from json file [data.json] and generate an excel report with the provided data

data.json

"company\_name"MSYS"

"employees" : [

( "name”:"Mahi", "location":"Chennai" },

( "name":"Raj", "location":"Bangalore" }

Excel Sheet data should be displayed as shown below

|  |  |
| --- | --- |
| Company Name Msvs | |
|  | |
| **Employees** | |
| **Name** | **Location** |
| Mahi | Chennai |
| Raj | Bangalore |