OJT - Python Exercise

1. What are the differences between list and tuple? Which is faster and why?

Ans:

List :  list in Python is used to store various types of data. A list can be defined as a collection of values or items of different types and separated by comma within a Square Bracket [ ]

Tuple: tuple in Python is used to store various types of data. A tuple can be defined as a collection of values or items of different types and separated by comma within a parentheses ( )

|  |  |
| --- | --- |
| List | Tuple |
| Store in a Square Bracket [ ] | Store in a parentheses ( ) |
| Mutable in nature | Immutable in nature |
| Consume more space or memory | Consume less space or memory |
| Slow in nature | Fast in nature |

Which is faster and why?

Ans: Tuple is faster than list because Tuples are immutable data types which are stored in a single block of memory fixed size memory so it doesn’t require extra space to store new objects but list are mutable data types and are allocated in two blocks where the fixed one with all object information and a variable sized block for the data.

1. What is the lambda function in python? Explain with examples.

Ans:

A lambda function is an anonymous function (defined without a name) that can take any number of arguments but, unlike normal functions, evaluates and returns only one expression.

Keyword : lambda

Example:

z=(lambda x: x \* 10 if x > 10 else ( x \* 5 if x < 5 else x))

print(z(12))

output: 120

Map Example: use to map the task to every individual element

Lst=[1,2,3,4,5]

r=(list(map(lambda x : x\*2,lst)))

print(r)

output: [2,4,6,8,10]

reduce Example: reduce sequence to find consolidated result

Lst=[1,2,3,4,5]

r=(reduce(lambda x,y : x+y,lst))

print(r)

output: 15

Example:

r=(lambda x, y,z : x+y-z)

print(r(12,2,1))

output: 13

filter Example:

lst=[1,3,4,5,6]

print(list(filter(lambda x: x >4,lst )))

out put: [5, 6]

**3 Monkey Patching**

Monkey-patching is a term that refers to modifying a class or module at a run time. In simple words, a class or module's work can be changed at the runtime.

This allows us to modify its behavior at run time.

It is use to modify and update the some part of code and working of class/module at run time without changing the whole code .

Example:

class main\_func:

def og\_func(self):

print ("og\_func() is being called by developer")

def monkey\_f(self):

print ("monkey\_f() is being called by developer ")

# replacing address of "og\_func" with "monkey\_f"

main\_func.og\_func = monkey\_f

obj = main\_func ()

# calling function "og\_func" whose address got replaced with function "monkey\_f()"

obj.og\_func()

4 PEP (Python enhancement protocol):

* The main purpose of PEP is to enhance the readability and consistency of code.
* A PEP is a design document providing information to the Python community, or describing a new feature for Python or its processes or environment.

**Value Error:**

Value Error is a built-in Python exception that is raised when the type of a passed argument to a function is incorrect or when a function is called with an argument that has an invalid value

1. Can we order a dictionary? How?

# 5. Can we order a dictionary? How?

# Yes, it is possible to order a dictionary in Python based on its keys or values.

# To order a dictionary by keys, you can use the sorted() function with the items()

# method of the dictionary as follows:

my\_dict = {'a': 3, 'c': 1, 'b': 2}

x=dict(sorted(my\_dict.items()))

print(x)    # {'a': 3, 'b': 2, 'c': 1}

#To order a dictionary by values, you can use the sorted() function with a lambda

# function that specifies the value to be sorted on:

y=dict(sorted(my\_dict.items(), key=lambda x : x[1]))

print(y)    #{'c': 1, 'b': 2, 'a': 3}

6.

Write a python program to right rotate a List by n Enter position to rotate list item: 3 Sample input: [10, 20, 30, 40, 50, 60, 70] Expected output: [50, 60, 70, 10, 20, 30, 40]

-------------------------------------------------------------------

l=[10,20,30,40,50,60,70]

n=int(input())

for i in range(n):

    x=l[0]

    for j in range(len(l)-1):

        l[j]=l[j+1]

    l[-1]=x

print(l)

7. Difference between append and extend operations of list

Append :

\*It adds 1 element at the end of the mentioned list

\* Increases length of list by 1

Extend:

\*It takes list as input

\*It adds all element of the list to the end of the mentioned list

\* Increases length of list by the list by number of elements present in input list

8 Create a dictionary where the key is an even number from the given list and the value

will be the occurrence of that element in the list. input= [1,2,3,2,4,2,4,7,8,4,5,8,6,9,2]

x={}

input= [1,2,3,2,4,2,4,7,8,4,5,8,6,9,2]

input=sorted(input)

for i in input:

    if i in x:

        continue

    else:

        if i%2==0:

            x.update({i:0})

print(x)

for key in x:

    x[key]=input.count(key)

# withot count built in........................................................

# for key in x:

#     count=0

#     for i in range(len(input)):

#         if input[i]==key:

#             count+=1

#         if input[i]==key+1:

#             break

#     x[key]=count

print(x)

9. Write a function **swap\_element** that contains **two args** which will be the position of

elements present in the list. The function must swap the elements present in those

positions.

Input: [1,2,3,4,5,6,7,8] function: **swap\_element(arg1, arg2)**

# 9. Write a function swap\_element that contains two args which will be the position of

# elements present in the list. The function must swap the elements present in those

# positions.

# Input: [1,2,3,4,5,6,7,8] function: swap\_element(arg1, arg2)

def swap\_element(x,y):

    input[x],input[y]=input[y],input[x]

input=[1,2,3,4,5,6,7,8]

swap\_element(2,4)

print(input)

10. Write the output of the program:

match = ‘version’, input=’Upgraded\_image\_version\_8.0.4.3’

if match in input:

print(‘YES’)

else:

print(‘NO’)

match = 'version'

input='Upgraded\_image\_version\_8.0.4.3'

if match in input:

    print('YES')

else:

    print('NO')

11. Rewrite the program to get proper output

Match = 'version'

input=8

print(Match+input)

#  Rewrite the program to get proper output

# Match = 'version'

# input=8

# print(Match+input)

match='version'

input=8

print(match+str(input))

1. How is memory management done in python?

Memory management in Python involves a private heap containing all Python objects and data structures.

Interpreter takes care of Python heap and that the programmer has no access to it.

\* The allocation of heap space for Python objects is done by Python memory manager. The core API of Python provides some tools for the programmer to code reliable and more robust program.

\* Python also has a build-in garbage collector which recycles all the unused memory. When an object is no longer referenced by the program, the heap space it occupies can be freed. The garbage collector determines objects which are no longer referenced by the program frees the occupied memory and make it available to the heap

space.

13. Give a real time example for multithreading. Is it a good idea to use multi-thread to

speed your Python code?

'''Q13. Give a real time example for multithreading. Is it a good idea to use multi-thread to

speed your Python code?'''

import time

import threading

def print\_numbers():

    for i in range(1, 11):

        print('thread 1->',i)

        time.sleep(1)

def print\_letters():

    for letter in ['a', 'b', 'c', 'd', 'e']:

        print('thread 2 ->',letter)

        time.sleep(1)

# Create two threads

t1 = threading.Thread(target=print\_numbers)

t2 = threading.Thread(target=print\_letters)

# Start the threads

t1.start()

t2.start()

# Wait for the threads to finish

t1.join()

t2.join()

print("Done")

14.When do you use generators in python? Give an example

Generators are used when we want to produce a large sequence of values, but we don't want to store all of them in memory at once.

def count\_up\_to(n):

    i = 0

    while i <= n:

        yield i

        i += 1

for num in count\_up\_to(10):

    print(num)

15.Give the scenarios, when you will get ‘ValueError’

print(int('a'))

16. Write a program to multiply two given number without using “\*” operation and any in built

Function

# Write a program to multiply two given number without using “\*” operation and any in built

# function

def multiply(x,y):

    mi=min(x,y)

    ma=max(x,y)

    total=0

    for i in range(mi):

        total+=ma

    print(total)

multiply(10,2)

multiply(10,0)

17. Write a program to find the count of alphabet alone in the given alphanumeric string for

Ex1: **input**=’abb24ccc8ddbbca1’ **output**=’**a1b2**24**c3d2b2c1a1**1’

Ex2: **input** = ‘abc23’ **output**=’**a1b1c1**23’

#  Write a program to find the count of alphabet alone in the given alphanumeric string for

# Ex1: input=’abb24ccc8ddbbca1’ output=’a1b224c38d2b2c1a11’

# Ex2: input = ‘abc23’ output=’a1b1c123’

input='abb24ccc8ddbbca1'

# input='abc23'

op=''

x=''

for i in range(len(input)):

    if not input[i].isalpha():

        op=op+input[i]

    else:

        x=x+input[i]

        if input[i] != input[i+1]:

            op=op+x[0]+str(len(x))

            x=''

print(op)

18. Write a python program where for every two hours it prints the pattern without using

sleep function

\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*

\*\*\*\*\*\*

\*\*\*

\*

import math

import schedule

def pattern():

    print(

    '''\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*

\*\*\*\*\*\*

\*\*\*''')

schedule.every(2).hours.do(pattern)

while 1:

    schedule.run\_pending()

19. Write a program using decorators to print the traffic signal messages

Expected output -

RED : STOP

YELLOW : SLOW DOWN

GREEN : GO

The decorator should be working in this order

def trafficcolor(func):

    def message():

        color=input("Enter the color of traffic signal:").lower()

        if color in ("red","yellow","green"):

            func(color)

        else:

            print("invalid color")

    return message

@trafficcolor

def trafficsignal(color):

    if color=="red":

        print("STOP")

    elif color=="yellow":

        print("SLOW DOWN")

    else:

        print("GO")

trafficsignal()

20. Write a python program for sort the given below list based last character of each word

names\_list = ['Prabhu', Rahul', 'Arunesh, 'Sonali', 'Rakshit']

# Write a python program for sort the given below list

# based last character of each word names\_list = ['Prabhu', Rahul', 'Arunesh, 'Sonali', 'Rakshit']

l=['Prabhu', 'Rahul', 'Arunesh', 'Sonali', 'Rakshit']

d={}

for i in range(len(l)):

    d.update({l[i][-1]:l[i]})  #This makes d={'u': 'Prabhu', 'l': 'Rahul', 'h': 'Arunesh', 'i': 'Sonali', 't': 'Rakshit'}

    l[i]=l[i][-1] # This makes l=['u', 'l', 'h', 'i', 't']

l.sort()

for i in range(len(l)):

    l[i]=d[l[i]]

print(l)

21. How do you open a file of large size, say around 10GB? So that program should not

Crash

22. Write a function where month and year are taken as arguments which returns the output

with all the dates of saturdays occuring the month

import calendar

def satCounter(year ,month):

    days=[0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]

    sat=[]

    d=days[month]

    if month==2 and calendar.isleap(year):

        d=29

    for day in range(1,d+1):

        sat.append(calendar.weekday(year,month,day))

    # print(sat)

    return f"number of saturdays are {sat.count(6)}"

year=int(input("Enter the year"))

month=int(input("Enter month 'in numbers':"))

print(satCounter(year,month))

23. Find the highest sum of the string by removing the duplicates for each iteration

input=’1211’

# 23. Find the highest sum of the string by removing the duplicates for each iteration

input='1211'

sum=0

if input:

    sum=int(input[0])

for i in range(1,len(input)):

    if input[i]==input[i-1]:

        continue

    else:

        sum+=int(input[i])

print(sum)

# or

# def highest\_sum(string):

#     prev\_char = ''

#     sum = 0

#     for char in string:

#         if char == prev\_char:

#             continue

#         else:

#             sum += int(char)

#             prev\_char = char

#     return sum

# print(highest\_sum(input))

24. Write a python script to copy files from a directory **D1** based on timestamp(current\_date)

to another directory **D2** and delete the source directory **D1**. Whenever the script is called

this program must run.

25. Write a program to send a mail notification to customers regarding the arrival of goods

on a daily basis. The admin email has a separate domain email address owned by your

company.Do not forget to add cc candidates in customer’s mail.

26. You are given a string **S**. Your task is to find the indices of the start and end of string **k** in

**S**

The first line contains the string **S**.The second line contains the string **k**.

Print the tuple in this format: **(start \_index, end \_index).** If no match is found, print (-1,

-1).

Sample Input

Sample Output

aaadaa

aa

(0, 1)

(1, 2)

(4, 5)

# 26. You are given a string S. Your task is to find the indices of the start and end of string k in

# S The first line contains the string S.The second line contains the string k.

# Print the tuple in this format: (start \_index, end \_index). If no match is found, print (-1,

# -1).

# Sample Input Sample Output

# aaadaa

# aa

# (0, 1)

# (1, 2)

# (4, 5)

s='aaadaa'

k='aa'

def substring(s,k):

    size=len(k)

    l=[]

    for i in range(len(s)-size+1): #length-size+1 because we are slicing from given index to given+size index

        if k==s[i:i+size]:

            l.append((i,i+size-1))

            # print(s[i:i+size])

    return l

print(substring(s,k))

27. Write a Python class to check the validity of a string of parentheses, '(', ')', '{', '}', '[' and '].

These brackets must be closed in the correct order, for example "()" and "()[]{}" are valid

but "[)", "({[)]" and "{{{" are invalid

# 27. Write a Python class to check the validity of a string of parentheses, '(', ')', '{', '}', '[' and '].

# These brackets must be closed in the correct order, for example "()" and "()[]{}" are valid

# but "[)", "({[)]" and "{{{" are invalid

input=["()", "()[]{}", "[)", "({[)]" ,"{{{" ]

paren={')':'(','}':'{',']':'['}

def parentheses(x):

    l=[]

    for i in x:

        if i not in paren:

            l.append(i)

        else:

            if l and paren[i]==l[-1]:

                l.pop()

            else:

                return False

    # print(l)

    if not l :

        return True

    else:

        return False

# Testing each input.....................

for i in input:

    print(i,parentheses(i))

    print()

28. Write a Python program to remove the parenthesis area in a string **using Regular**

**Expression**

Sample data : ["example (.com)", "MSys", "github (.com)", "keka (.com)"]

Expected Output:

Example

MSys

github

Keka

# Write a Python program to remove the parenthesis area in a string using Regular

# Expression

# Sample data : ["example (.com)", "MSys", "github (.com)", "keka (.com)"]

# Expected Output:

# Example

# MSys

# github

# keka

input= ["example (.com)", "MSys", "github (.com)", "keka (.com)"]

def func(s):

    for i in s:

        if '(' not in s:

            print(s)

            break

        else:

            index=s.index('(')

            print(s[:index])

            break

for i in input:

    func(i)

29. Write a regular expression to find the html tags that are more than 4 letters.

Note: Html tags can be found inside <> characters and closing html tags can be found in

the same format after / character. </>

i.e.: <param> </param>

30. How does the context manager work in python? Explain the internal methods. Write a

custom sample context manager.