**OJT**

1. 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] .

🡪 # 1st approach : Using list built-in methods (pop, insert).

list1 = [int(i) for i in input().split()]

position = int(input("Enter position to rotate list item: "))

length = len(list1)

position = position%length

for i in range(position):

    pop\_element = list1.pop()

    list1.insert(0, pop\_element)

print(f"Output : {list1}")

    # 2nd approach : Using slicing and concatenate operation.

list1 = [int(i) for i in input().split()]

position = int(input("Enter position to rotate list item: "))

length = len(list1)

position = position%length

rotate\_list = list1[-(position):] + list1[:-(position)]

print(f"Output : {rotate\_list}")

    # 3rd approach : Using loop concept.

list1 = [int(i) for i in input().split()]

position = int(input("Enter position to rotate list item: "))

length = len(list1)

position = position%length

for i in range(position):

    temp = list1[-1]

    for j in range(length - 1, 0, -1):

        list1[j] = list1[j - 1]

    list1[0] = temp

print(f"Output : {list1}")

2. Difference between append and extend operations of list ?

🡪  Both **append** and **extend** are list methods in Python that allow you to add elements to a list.       However, they differ in how they add elements to the list.

**# append** method adds a single element to the end of the list. It takes one argument and adds that argument to the end of the list.

 For example:

list1 = [10, 20, 30]

list1.append([1, 2, 3])

print(f"Output : {list1}") # Output : [10, 20, 30, [1, 2, 3]]

**# extend** method, on the other hand, adds multiple elements to the end of the list. It takes an iterable as an argument and adds each element from that iterable to the end of the list.

 For example:

list1 = [10, 20, 30]

list1.extend([40, 50, 60])

print(f"Output : {list1}") # Output : [10, 20, 30, 40, 50, 60]

**Note :  a.** The argument to **extend** must be an iterable (e.g., a list, a tuple, a set, or a string). If you pass a non-iterable argument to **extend**, you will get a **TypeError**.

**b.** The **append** method always adds a single element to the list, while the **extend** method can add multiple elements to the list at once.

3. 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]

🡪 # 1st approach : Using dictionary comprehension.

list1 = [int(i) for i in input().split()]

unique\_ele = set(list1)

even\_key\_dict = {i: list1.count(i) for i in unique\_ele if i%2 == 0}

print(f"Output : {even\_key\_dict}")

     # 2nd approach : Using loops.

list1 = [int(i) for i in input().split()]

even\_key\_dict = {}

for i in list1:

    if i%2 == 0:

        if i in even\_key\_dict:

            even\_key\_dict[i] += 1

        else:

            even\_key\_dict[i] = 1

print(f"Output : {even\_key\_dict}")

# 3rd approach : Using defaultdict and loop

from collections import defaultdict

list1 = [int(i) for i in input().split()]

even\_key\_dict = defaultdict(int)

for i in list1:

    if i % 2 == 0:

        even\_key\_dict[i] += 1

print(f"Output: {dict(even\_key\_dict)}")

4. 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)**

**🡪**

list1 = [int(i) for i in input().split()]

pos1 = int(input("Enter the position 1 : "))

pos2 = int(input("Enter the position 2 : "))

def swap\_element(list1, pos):

    pos1, pos2 = pos

    try:

        if pos1 != pos2:

            list1[pos1], list1[pos2] = list1[pos2], list1[pos1]

            return list1

    except IndexError:

        print("Error : Enter the position within the range!..")

print(f"Before swap : {list1}")

list1 = swap\_element(list1, (pos1, pos2))

print(f"After swap : {list1}")

5. 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’)

🡪 The output of the program will print **YES,** because the program checks if **match** present in the **input** using **in** operator, since **match** present in the **input**, the condition in the if statement returns **True**, so statements inside the **if**  will execute.

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

🡪

#1st approach.

import os

import time

def read\_large\_file(file\_path):

    with open(file\_path, 'rb') as file:

        while True:

            data = file.read(100)  # Read 100B at a time

            if not data:

                break

            yield data

#  to check the file size

file\_size\_bytes = os.path.getsize("1GB.bin")

print(file\_size\_bytes)

for  piece\_of\_bytes in read\_large\_file("1GB.bin"):

    time.sleep(1)

    print(piece\_of\_bytes)

#2nd approach

import time

import os

def read\_large\_file(file\_path, f\_size):

    with open(file\_path, 'rb') as file:

        current\_size = file.tell()

        while current\_size != f\_size:

            data = file.read(100)  # Read 100 Bytes data at a time

            yield data

            current\_size = file.tell()

#  to check the file size

file\_size\_bytes = os.path.getsize("1GB.bin")

print(file\_size\_bytes)

for  piece\_of\_bytes in read\_large\_file("1GB.bin", file\_size\_bytes):

    time.sleep(1)

    print(piece\_of\_bytes)

7. Write a function where month and year are taken as arguments which returns the output with all the dates of Saturdays occurring the month.

🡪 1st approach

def is\_leap\_year(year):

    if year%100 == 0:

        if year%400 == 0:

            return True

        return False

    elif year%4 == 0:

        return True

    return False

def list\_of\_saturday(year, month):

    month\_code = {1:(1,31), 2:(4,28), 3:(4,31), 4:(0,30), 5:(2,31), 6:(5,30), 7:(0,31), 8:(3,31), 9:(6,30), 10:(1,31), 11:(4,30), 12:(6,31)}

    century\_code = {15:0, 16:6, 17:4, 18:2, 19:0, 20:6, 21:4, 22:2}

    day\_code = {0:'sat', 1:'sun', 2:'mon', 3:'tue', 4:'wed', 5:'thu', 6:'fri'}

    mc = month\_code[month][0]

    cc = century\_code[year//100]

    num\_years\_completed = year%100

    num\_of\_leap\_year = num\_years\_completed//4

    num\_of\_days\_in\_month = month\_code[month][1]

    leap\_year = is\_leap\_year(year)

    if leap\_year and month == 2:

        num\_of\_days\_in\_month += 1

    diff = 0

    if leap\_year and month in (1, 2):

        diff = 1

    res = []

    for day in range(1, num\_of\_days\_in\_month+1):

        val = day + mc + cc + num\_years\_completed + num\_of\_leap\_year - diff

        if day\_code[val%7] == 'sat':

            res.append(day)

    return res

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

month = int(input("Enter the month : "))

print(f"list of staurday in {month}/{year} :",list\_of\_saturday(year, month))

#2nd approach

import calendar

# mon - 0, tue - 1, wed - 2, thu - 3, fri - 4, sat - 5, sun – 6

def list\_of\_saturday(month,year):

    num\_days = calendar.monthrange(year, month)[1] # (2, 31)

    saturday = []

    for day in range(1,num\_days+1):

        weekday = calendar.weekday(year, month, day)

        if weekday == 5:

            saturday.append(day)

    return saturday

list = list\_of\_saturday(3,2023)

print(list)

#3rd approach

import calendar

cal = calendar.Calendar()

def list\_of\_saturday(month,year):

    saturday = []

    for day in cal.itermonthdays2(year, month):

        print(day)

        if day[0] != 0 and day[1] == 5:

            saturday.append(day[0])

    return saturday

print(list\_of\_saturday(3,2023))

8. Find the highest sum of the string by removing the duplicates for each iteration input=’1211’.

🡪

1st approach

string = input("Enter the string : ")

unique\_elements = set([int(i) for i in string])

print(f"output : {sum(unique\_elements)}")

2nd approach

from collections import defaultdict

unique\_elements = defaultdict(int)

for i in string:

    unique\_elements[int(i)] += 1

print(f"output : {sum(dict(unique\_elements).keys())}")

9. 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.

🡪

import os

import shutil

import datetime

D1 = ".\\D1\\"

D2 = ".\\D2\\"

for filename in os.listdir(D1):

    src\_filepath = os.path.join(D1, filename)

    des\_filepath = os.path.join(D2, filename)

    # os.rename(src\_filepath, des\_filepath)

    shutil.copy2(src\_filepath, des\_filepath)

# os.rmdir(D1)

shutil.rmtree(D1)

# to get the time-stamp of the file

modified\_time = os.path.getmtime(des\_filepath) # it returns unix time that measured in seconds

print(modified\_time)

date\_time = datetime.datetime.timestamp(modified\_time) # it returns the date-time object

print(date\_time)

10. 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.

🡪

import smtplib

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

email\_from = 'jc.cgs.999@gmail.com'

email\_password = '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'

email\_subject = 'Daily Goods Arrival Notification'

email\_body\_text = 'Hello,\n\nThis is to inform you that new goods have arrived in our store today.\n\nThank you for your continued patronage.\n\nBest regards,\nSender'

email\_body\_html =  '<p>This is an <b>HTML</b> message</p>'

recipients = ['moinalikhandevopser@gmail.com']

cc\_recipients = ['afwaan3@gmail.com','meghumeghu3@gmail.com']

msg = MIMEMultipart()

msg['From'] = email\_from

msg['To'] = ', '.join(recipients)

msg['Cc'] = ', '.join(cc\_recipients)

msg['Subject'] = email\_subject

msg.attach(MIMEText(email\_body\_text, 'plain'))

msg.attach(MIMEText(email\_body\_html, 'html'))

server = smtplib.SMTP('smtp.gmail.com', 587)

server.starttls()

server.login(email\_from, email\_password)

server.sendmail(email\_from, recipients + cc\_recipients, msg.as\_string())

print('Email notification sent successfully!')

server.quit()

11. Write a program to extract the words starting with lowercase letter present in the list. [‘Nissan’, ‘maruti’, ‘hyundai’, ‘Volkswagen’, ‘audi’].

🡪

1st approach

def extract\_string\_startswith\_lowercase(list\_):

    a = ord('a')

    z = ord('z')

    res = []

    for string in list\_:

        val = ord(string[0])

        if val >= a and val <= z:

            res.append(string)

    return res

list\_ = [s for s in input().split()]

print(extract\_string\_startswith\_lowercase(list\_))

2nd approach

lowercase\_words = [word for word in list\_ if word[0].islower()]

print(lowercase\_words)

12. Write a program using dictionary comprehension which will contain the key value pair of i:i\*\*2. Value of ‘i’ will start from 1 and will go up to 10.

🡪

dict\_ = {i : i\*\*2 for i in range(1, 10)}

print(dict\_)

13. Take the input marks from user using input() function and find out the grade of the students. Note the grading will be in this manner – (100 – 91) – A1, (90-81) – A2, (80-71) – B1, (70-61) – B2, (60- 51) – C1 (50-40) – C2 and less than 40 students will ‘Fail’. Also make sure user should input the marks in the range 0<=marks<=100, if user will input some other marks in should print invalid marks.

🡪

1st approach

def find\_grade(marks):

    grade = {4 : "C2", 5 : 'C2', 6 : "C1", 7 : "B2", 8 : "B1", 9 : "A2", 10 : "A1"}

    if marks < 40:

        return 'Fail'

    key = marks//10

    if marks%10 > 0:

        key += 1

    return grade[int(key)]

marks = float(input("Enter the marks : "))

print(f"Your grade is : {find\_grade(marks)}")

2nd approach

def find\_grade(marks):

    if marks >= 91 and marks <= 100:

        grade = "A1"

    elif marks >= 81 and marks <= 90:

        grade = "A2"

    elif marks >= 71 and marks <= 80:

        grade = "B1"

    elif marks >= 61 and marks <= 70:

        grade = "B2"

    elif marks >= 51 and marks <= 60:

        grade = "C1"

    elif marks >= 40 and marks <= 50:

        grade = "C2"

    else:

        grade = "Fail"

    return grade

marks = int(input("Enter your marks: "))

print("Your grade is:", find\_grade(marks))

14. Given a list [1,2,1,5,9,10,2,2,7,5,3,10,8,9,15,17,21,16,17,90] find the difference between the length of the list and the count of unique elements in the list.

🡪

list\_ = [int(i) for i in input().split()]

result = len(list\_) - len(set(list\_))

print(result)

15. In the given String -- ‘MsYs TecHNOlogiEs iS a gREat place To woRk’ find the count of lowercase and uppercase letters.

🡪

1st approach

string = input("Enter the string : ")

res = {'lower\_count' : 0, 'upper\_count' : 0}

for char in string:

    if char.isupper():

        res['upper\_count'] += 1

    elif char.islower():

        res['lower\_count'] += 1

print(res)

2nd approach

lower\_count = sum([1 for char in string if char.islower()])

upper\_count = sum([1 for char in string if char.isupper()])

print("Number of lowercase letters:", lower\_count)

print("Number of uppercase letters:", upper\_count)

16. You are given a string and width. Your task is to wrap the string into paragraph of width in reverse order. Blank spaces should be ignored.

for e.g.: i/p - first line contains a string with blank spaces - Hello, welcome to this organisation. the second line contains the width - 4

o/p : lleH ew,o mocl tote osih nagr tasi .noi

🡪

def wrap\_string(string, width):

    string = string.strip()

    if len(string) == 0 or width == 0 or width > len(string):

        return []

    res = ""

    paragraph = []

    for char in string:

        if char != " ":

            res += char

            if len(res) == width:

                paragraph.append(res[::-1])

                res = ""

    return paragraph

string = input("Enter the string : ")

width = int(input("Enter the width : "))

for para in wrap\_string(string, width):

    print(para)

17. Find the palindrome words with the count value from the given string. Output should be in form of dict. key will be palindrome word and value will be number of occurrences.

i/p given a string - Nittin & his mom went to a park last friday. His Mom observed that the weather was too cool. Nittin also met his sis. If we reverse the number 1331 then we also get 1331.

o/p - {'nittin': 2, 'mom': 2, 'sis': 1, '1331': 2}

🡪

from collections import defaultdict

def find\_palindrome\_word(string):

    result = defaultdict(int)

    string = string.lower()

    sentences = string.split('.')

    for sentence in sentences:

        words = [w for w in sentence.split() if len(w)>1]

        for word in words:

            if word[::-1] == word:

                result[word] += 1

    return dict(result)

string = input("Enter the string : ")

print(find\_palindrome\_word(string))

18. create 2 dictionaries as follows: dict1 = {'name': 'Msys', 'Place': 'Pune'} dict2 = {'EmpID': 0001, 'Salary': 50000} Perform following operations:

a. create single dictionary by merging dict1 & dict2

b. update the salary to 10%

c. update age to 35

d. extract & print all the values & keys separately in tuple.

e. delete the element with key 'Age' & print the dictionary elements.

🡪

dict1 = {'name': 'Msys', 'Place': 'Pune'}

dict2 = {'EmpID': 2982, 'Salary': 50000}

# a. create single dictionary by merging dict1 & dict2

for key, val in dict2.items():

    dict1[key] = val

print(dict1)

# b. update the salary to 10%

sal = dict1['Salary']

sal = int(sal + (sal\*(10/100)))

print(f"Salary before increment : {dict1['Salary']}")

dict1['Salary'] = sal

print(f"Salary after increment : {dict1['Salary']}")

# c. update age to 35

dict1['age'] = 35

print(dict1)

# d. extract & print all the values & keys separately in tuple.

print("Keys : ",tuple(dict1.keys()))

print("Values : ",tuple(dict1.values()))

# e. delete the element with key 'Age' & print the dictionary elements.

# del(dict1['age'])

print(dict1.pop('age'))

for key, value in dict1.items():

    print(key, " --> ", value)

19. You have given a string str1 = "abcbaefabcabchijkl" your task is to find the combination of given word without repetition, present in the string , given word 'abc' .

o/p = 7, explanation : abc, cba, cba, bca, acb cab, bac

🡪

import itertools

def find\_combination\_of\_word(string, word):

    combinations = set([''.join(p) for p in itertools.permutations(word)])

    count = 0

    for combination in combinations:

        if combination in string:

            count += 1

    return count

string = input("Enter the string : ")

word = input("Enter the word : ")

print(find\_combination\_of\_word(string, word))

20. Given an Integer n, count the total number of times 1 is appearing in all non-negative integers less than or equal to n.

Ex : n = 13, output should be 6 method – 1 is coming 6 times starting from number 0 till 13 in ‘1’, ‘10’, ‘11’, ‘12’, ‘13’. Also note 1 is coming 2 times in 11. That is why 6 is the output.

🡪

def count\_num\_of\_one(n):

    num = 0

    res = 0

    while num <= n:

        str\_num = str(num)

        if '1' in str\_num:

            res += str\_num.count('1')

        num+=1

    return res

n = int(input("Enter the number :"))

print(count\_num\_of\_one(n))