WEEK1:-

1.Write a program to demonstrate different number data types in Python.

CODE:  
a=10

b=10.1

c=1+2j

print("Type!:",type(a))

print("Type2:",type(b))

print("Type3:",type(c))

OUTPUT:  
Type!: <class 'int'>

Type2: <class 'float'>

Type3: <class 'complex'>

2. Write a python program to Read input data from the Keyboard and perform different Arithmetic Operations on the numbers.

CODE:

#ALL arithematic operations addition (+), subtraction (-), multiplication (\*), division (/), modulo (%), and exponentiation (\*\*)

a = int(input("Enter First Number"))

b = int(input("Enter Second Number"))

print("Addition of ",a," and ",b,":",a+b)

print("Substraction of ",a," and ",b,":",a-b)

print("Multiplication of ",a," and ",b,":",a\*b)

print("Division of ",a," and ",b,":",a/b)

print("Modulo of ",a," and ",b,":",a%b)

print("Exponentiation of ",a," and ",b,":",a\*\*b)

OUTPUT:

Enter First Number10

Enter Second Number5

Addition of 10 and 5 : 15

Substraction of 10 and 5 : 5

Multiplication of 10 and 5 : 50

Division of 10 and 5 : 2.0

Modulo of 10 and 5 : 0

Exponentiation of 10 and 5 : 100000

3. Write a python program to declare variables and display types of respective variables.

CODE:  
a=10

b=10.1

c="Hello"

d=True

e=(0,1)

f=[0,1]

print("Type!:",type(a))

print("Type2:",type(b))

print("Type3:",type(c))

print("Type4:",type(d))

print("Type5:",type(e))

print("Type6:",type(f))

OUTPUT:

Type!: <class 'int'>

Type2: <class 'float'>

Type3: <class 'str'>

Type4: <class 'bool'>

Type5: <class 'tuple'>

Type6: <class 'list'>

4. Write a python program to convert integer type to float and vice versa.

CODE:

a=10.2

b=10

print("Converting int to float:",float(b))

print("Converting float to int",int(a))

OUTPUT:

Converting int to float: 10.0

Converting float to int 10

5. Write a python program to print the current date in the following format “Sun May 29 02:26:23 IST 2017”.

CODE:  
import time

ltime=time.localtime()

print(time.strftime("%a %b %d %H:%M:%S %Z %Y",ltime))

OUTPUT:

Sat Aug 31 06:19:22 UTC 2024

6. Write a program to read the following Employee data from the keyboard and print that data. Employee Details:

a. Enter Employee No [Data type: int]

b. Enter Employee Name [Data type: string]

c. Enter Employee Salary [Data type: int]

d. Enter Employee Address [Data type: string]

e. Employee Married ?[True|False]: [Data type: Boolean]

CODE:

Employee\_No = int(input("Enter Employee's Number:"))

Employee\_Name = input("Enter Employee's Name:")

Employee\_Salary = int(input("Enter Employee's Salary:"))

Employee\_Address = input("Enter Employee's Address:")

Employee\_Married = bool(int(input("Is employee married or not(0:False,1:True):")))

print("Employe No:",Employee\_No)

print("Employee Name:",Employee\_Name)

print("Employee Salary:",Employee\_Salary)

print("Employee Address",Employee\_Address)

print("Employee married:",Employee\_Married)

OUTPUT:

Enter Employee's Number:12

Enter Employee's Name:a ARUN

Enter Employee's Salary:125000

Enter Employee's Address:RAJAJIPPURAM

Is employee married or not(0:False,1:True):1

Employe No: 12

Employee Name: ARUN

Employee Salary: 125000

Employee Address RAJAJIPPURAM

Employee married: True

WEEK2:-

1.Write a Python program to find the longest increasing subsequence from a given list of numbers.

CODE:

n = int(input("Enter Number of elements:"))

arr = []

for i in range(n):

arr.append(int(input("Enter element:")))

ans\_list = []

for i in range (n):

temp\_list = []

lar = 0

for j in range (i,n):

if (lar<arr[j]):

temp\_list.append(arr[j])

lar = arr[j]

ans\_list.append(temp\_list)

print(ans\_list)

temp = 0

index = 0

for i in ans\_list:

if len(i)>temp:

temp = len(i)

ans = index

index +=1

print("Longest increasing subsequence:",ans\_list[ans])

OUTPUT:

Enter Number of elements:6

Enter element:50

Enter element:3

Enter element:10

Enter element:7

Enter element:40

Enter element:80

[[50, 80], [3, 10, 40, 80], [10, 40, 80], [7, 40, 80], [40, 80], [80]]

Longest increasing subsequence: [3, 10, 40, 80]

2. Create a Python script to generate a list that contains 25 elements and display the frequency of each item in a list

CODE:

import random

array = [random.randint(1, 100) for \_ in range(25)]

ans\_dict={}

for i in range(25):

ans\_dict[array[i]]=0

for i in range(25):

ans\_dict[array[i]]+=1

for i in ans\_dict:

print("Frequency of ",i," is:",ans\_dict[i])

OUTPUT:  
Frequency of 33 is: 1

Frequency of 97 is: 2

Frequency of 94 is: 1

Frequency of 40 is: 1

Frequency of 43 is: 1

Frequency of 85 is: 3

Frequency of 78 is: 2

Frequency of 80 is: 1

Frequency of 45 is: 1

Frequency of 79 is: 1

Frequency of 47 is: 1

Frequency of 63 is: 1

Frequency of 34 is: 1

Frequency of 37 is: 1

Frequency of 12 is: 1

Frequency of 20 is: 1

Frequency of 70 is: 1

Frequency of 11 is: 1

Frequency of 16 is: 1

Frequency of 3 is: 1

Frequency of 41 is: 1

3. Develop a Python program that constructs a list of 15 strings. It should then determine the count of strings in this list that have a minimum length of two characters and also start and end with identical characters. You can choose any specific list of strings for this task.

CODE:

import random

import string

# Function to generate a random string of given length

def generate\_random\_string(length):

letters = string.ascii\_letters # A-Z and a-z

return ''.join(random.choice(letters) for \_ in range(length))

string\_list = []

for \_ in range(15):

random\_string = generate\_random\_string(random.randint(1, 20))

string\_list.append(random\_string)

count = 0

for i in string\_list:

if len(i)>2:

if i[0]==i[len(i)-1]:

count+=1

print("Count required:",count)

OUTPUT:

Count required: 1

4. Develop a Python script that generates a list with 15 items and then eliminates any duplicates from that list.

CODE:

import random

list\_with\_duplicates = []

for \_ in range(15):

list\_with\_duplicates.append(random.randint(1, 100))

print("Original list with duplicates:")

for item in list\_with\_duplicates:

print(item, end=' ')

print()

unique\_list = []

for item in list\_with\_duplicates:

# Check if the item is already in the unique\_list

is\_duplicate = False

for unique\_item in unique\_list:

if item == unique\_item:

is\_duplicate = True

break

# If the item was not found in the unique\_list, add it

if not is\_duplicate:

unique\_list.append(item)

print("\nList after removing duplicates:")

for item in unique\_list:

print(item, end=' ')

print()

OUTPUT:

Original list with duplicates:

87 15 10 51 64 50 90 11 75 64 34 27 89 83 80

List after removing duplicates:

87 15 10 51 64 50 90 11 75 34 27 89 83 80

5. Develop a Python script that builds a list with 15 elements. This script will reposition the items in the list by doing a circular right shift. The number of positions shifted will be based on a user-specified value.

CODE:

arr = [i for i in range(1, 16)]

n = int(input("Enter number of positions to be shifted (n < 15): "))

print("Original List:", arr)

if 0 < n < 15:

# Perform the right circular shift in-place

# Slicing to achieve the right circular shift

length = len(arr)

arr[:] = arr[-n:] + arr[:-n] # Update the list in-place

print("Repositioned List:", arr)

else:

print("Invalid Input")

OUTPUT:

Enter number of positions to be shifted (n < 15): 3

Original List: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]

Repositioned List: [13, 14, 15, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

WEEK3:

1.Write a program that uses a 'while' loop along with a found flag to search through a list of powers of 2. Your program should identify the value corresponding to 2 raised to the fifth power (32).

CODE:

found = 0

array = []

index = 0

n = int(input("Length of array:"))

for i in range(n):

temp = int(input("Enter power of 2 to be added t0 list:"))

array.append(2\*\*temp)

while(index<len(array)):

if(array[index]==32):

found = 1

index+=1

if found:

print("Value present")

else:

print("Value not present")

OUTPUT:

Length of array:5

Enter power of 2 to be added t0 list:1

Enter power of 2 to be added t0 list:2

Enter power of 2 to be added t0 list:3

Enter power of 2 to be added t0 list:4

Enter power of 2 to be added t0 list:5

Value present

2. Write a program to calculate the distance covered by a robot after a series of movements on a plane. A robot starts at the origin point (0,0) in a 2D plane. It can move in four directions: UP, DOWN, LEFT, and RIGHT. The robot's movements are defined as follows:

a. UP - 5

b. DOWN – 3

c. LEFT - 3

d. RIGHT – 2

The number following each direction indicates the number of steps taken in that direction.

Implement a program that:

a. Tracks the robot's position after the given sequence of movements.

b. Calculates the distance between the robot's final position and the origin (0,0).

c. If the calculated distance is a floating-point number, round it to the nearest integer.

Constraints:

a. You are not allowed to use any external packages or libraries.

b. Only basic Python functionalities should be used.

CODE:

#Position represents (x,y) axis points:

posn = [0,0]

n = int(input("Enter number of sequence of movements to be done:"))

while(n):

dirn = input("Enter direction[UP,DOWN,LEFT,RIGHT]:")

if dirn == "UP":

posn[1] += 5

elif dirn == "DOWN":

posn[1] -= 3

elif dirn == "LEFT":

posn[0] -= 3

elif dirn == "RIGHT":

posn[0] += 2

else:

print("Invalid Input")

n-=1

print("Robots position after sequence of movements:",posn)

print("Distance between initial and final position of robot:",round(((posn[0]\*\*2)+(posn[1]\*\*2))\*\*(1/2)))

OUTPUT:

Enter number of sequence of movements to be done:2

Enter direction[UP,DOWN,LEFT,RIGHT]:UP

Enter direction[UP,DOWN,LEFT,RIGHT]:RIGHT

Robots position after sequence of movements: [2, 5]

Distance between initial and final position of robot: 5

3.Write a python program that can accept two strings as input and print the string with maximum length in console. If two strings have the same length, then the program should print all strings in one line.

CODE:

str1 = input("Enter first string:")

str2 = input("ENter second string:")

if (len(str1)>len(str2)):

print("String with max length:",str1)

elif(len(str2)>len(str1)):

print("String with max length:",str2)

else:

print("Both string are of equall legth:",str1,str2)

OUTPUT:

Enter first string:abc

ENter second string:rth

Both string are of equall legth: abc rth

4. Write a program that computes the net amount in a bank account based on a series of transactions provided as input.

Instructions:

a. Your program should accept a transaction log as input, where each transaction is either a deposit or a withdrawal.

b. The transaction log format is as follows:

c. D 100 indicates a deposit of 100 units.

d. W 200 indicates a withdrawal of 200 units.

e. The program should calculate the net amount in the bank account after processing all transactions.

CODE:

transactions= []

Balance = 0

while (True):

transaction= input("Entry(done to stop)")

if transaction.lower() == "done":

break

else:

transactions.append(transaction)

for transaction in transactions:

parts = transaction.split()

if len(parts)!=2:

continue

else:

transaction\_type,ammount = parts

if transaction\_type == "D":

Balance+= int(ammount)

elif transaction\_type == "W":

Balance-= int(ammount)

print("Net Account Balance:",Balance)

OUTPUT:

Entry(done to stop)D 500

Entry(done to stop)W 200

Entry(done to stop)done

Net Account Balance: 300

5. Write a python program to implement the binary search which searches an item in a sorted list. (Do not use any Packages)

CODE:

array = [1,2,3,4,5,6,7,8,9,10]

print("Array:",array)

low,high = 0,(len(array)-1)

ele = int(input("Enter element to be searched:"))

posn = -1

while (low<=high):

mid = (low+high)//2

if (array[mid]==ele):

posn = mid

break

elif ele<array[mid]:

high = mid-1

else :

low = mid+1

if (posn==-1):

print("Element not found")

else:

print("Element found at position:",posn)

OUTPUT:

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

Enter element to be searched:4

Element found at position: 3

6. Write a Python 3 program that demonstrates how to calculate the summation of all possible combinations of elements in a list of tuples. Use nested for loops to iterate through the tuples and generate the combinations.

CODE:  
list\_of\_tuples = [(1,2),(3,4),(5,6)]

print("List of tuples:",list\_of\_tuples)

total\_sum = 0

for i in range(len(list\_of\_tuples)):

for j in range(len(list\_of\_tuples[i])):

for k in range(len(list\_of\_tuples)):

if i!=k:

for l in range(len(list\_of\_tuples[i])):

combination\_sum = list\_of\_tuples[i][j]+list\_of\_tuples[k][l]

total\_sum+=combination\_sum

print("Summation of all possible combinations:",total\_sum)

OUTPUT:

List of tuples: [(1, 2), (3, 4), (5, 6)]

Summation of all possible combinations: 168

7. Write a Python3 program that demonstrates how to convert a tuple into a list, where each element in the list is the succeeding element of the original tuple.

CODE:

original\_tuple = (10, 20, 30, 40, 50)

for i in range(len(original\_tuple)):

if i + 1 < len(original\_tuple):

succeeding\_elements\_list[i] = original\_tuple[i + 1]

else:

succeeding\_elements\_list[i] = None

print("Original tuple:", original\_tuple)

print("List of succeeding elements:", succeeding\_elements\_list)

OUTPUT:

Original tuple: (10, 20, 30, 40, 50)

List of succeeding elements: [20, 30, 40, 50, None]

WEEK4:-

Exercise 1: Customer Directory Management System

Objective

The goal of this case study is to develop a Python program that manages customer

information for a telecommunications company. This involves creating a function to handle

customer data, extracting relevant information, and printing the data in various formats.

Requirements

1. Function Definition:

A. Define a function named tel\_directory that takes a list of dictionaries representing

customer information. Each dictionary contains:

a. customer\_id (Unique identifier for the customer)

b. customer\_name (Name of the customer)

c. Subscription\_type (Type of subscription: "prepaid" or "postpaid")

2. Data Input:

B. Input at least 10 customer records using the tel\_directory function.

3. Data Extraction:

C. Extract data from the list of dictionaries and create a list of lists. Each list should

be structured based on different key combinations.

4. Output:

D. Print the extracted data in three different combinations of key fields:

a. Combination 1: A list containing [customer\_id, customer\_name,

Subscription\_type]

b. Combination 2: A list containing [customer\_id, customer\_name]

c. Combination 3: A list containing [customer\_name, Subscription\_type]

Exercise 2: Enhancing the Customer Directory Management System

Add a new function named Search\_Customer to the existing Customer Directory

Management System. This function should:

1. Function Purpose:

A. Search for a customer by their name within the directory.

2. Function Details:

A. The function should take the customer\_name as input.

B. If the customer is found in the directory, display their information.

C. If the customer is not found, print a message indicating that the customer is not in

the directory.

Exercise 3: Enhancing the Customer Directory Management System

Add a new function named Search\_subscription to the existing Customer Directory

Management System. This function should:

1. Function Purpose:

A. Search for customers based on their subscription type.

2. Function Details:

A. The function should take the Subscription\_type ("prepaid" or "postpaid") as input.

B. Display the information of all customers who have the specified subscription type.

C. If no customers have the given subscription type, print appropriate message.

CODE:

#Function to take input for data of directory:

def tel\_directory(customer\_info):

"""data\_values = int(input("Enter no.of data elements(>10):"))

if (data\_values<10):

data\_values = 10

for i in range(data\_values):

customer\_info[0][i] = int (input(f"Enter {i+1} customer id:"))

customer\_info[1][i] = input(f"Enter {i+1} customer name:")

Subscription\_type = input(f"Enter {i+1} subscript(prepaid or postpaid):")

if (Subscription\_type.lower() == "prepaid" or Subscription\_type.lower() == "postpaid"):

customer\_info[2][i] = Subscription\_type.lower()

else:

customer\_info[2][i] = None

print("\n")"""

customer\_info = [{0:1,1:2,2:3} , {0:"A",1:"B",2:"C"} , {0:"prepaid",1:"postpaid",2:"prepaid"}]

data\_values = 3

return customer\_info,data\_values

#function to Search for a customer by their name within the directory:

def search\_customer(customer\_array,data\_values):

customer\_find = input("Enter Customer Name to find its details:")

flag = 0

for data in range(data\_values):

if (customer\_array[1][data].lower() == customer\_find.lower()):

flag = data

if (flag):

print(f"\nCustomer Found:-\nCustomer Id:{customer\_array[0][data-1]}\nCustomer Name:{customer\_array[1][data-1]}\nSubscription type:{customer\_array[2][data-1]}\n")

else:

print("\nCustomer is not in the directory.")

#function to Search for customers based on their subscription type:

def Search\_subscription(customer\_array,data\_values):

customer\_subscription = input("Enter subscription type(Prepaid or postpaid):").lower()

flag = 0

print(f"\n\nCustomer Id:\tCustomer Name: (With subscription type-{customer\_subscription})")

for data in range(data\_values):

if (customer\_array[2][data] == customer\_subscription):

print(f"{customer\_array[0][data]}\t{customer\_array[1][data]}")

flag = 1

if (flag == 0):

print(f"{None}\t{None}")

#creating list of dictionaries:

customer\_id = {}

customer\_name = {}

Subscription\_type = {}

customer\_info = [customer\_id,customer\_name, Subscription\_type]

customer\_directory,length = tel\_directory(customer\_info)

#creating list of lists with customer\_id at 0 , customer\_name at 1 , Subscription\_type at 2

customer\_array = [[],[],[]]

#Filling values in list of lists from list of dictionaries

for data in range(length):

customer\_array[0].append(customer\_directory[0][data])

customer\_array[1].append(customer\_directory[1][data])

customer\_array[2].append(customer\_directory[2][data])

print("Data (Combination1):\n")

print("Customer\_id\tCustomer\_name\tSubscription\_type\n")

for data in range(length):

print(f"{customer\_array[0][data]}\t{customer\_array[1][data]}\t{customer\_array[2][data]}\n")

print("\n")

print("Data (Combination2):\n")

print("Customer\_id\tCustomer\_name\n")

for data in range(length):

print(f"{customer\_array[0][data]}\t{customer\_array[1][data]}\n")

print("\n")

print("Data (Combination3):\n")

print("Customer\_name\tSubscription\_type\n")

for data in range(length):

print(f"{customer\_array[1][data]}\t{customer\_array[2][data]}\n")

print("\n")

#calling the function to search customer:

search\_customer(customer\_array,length)

#calling the function to search customers with particular subscription type:

Search\_subscription(customer\_array,length)

OUTPUT:

Data (Combination1):

Customer\_id Customer\_name Subscription\_type

1 A prepaid

2 B postpaid

3 C prepaid

Data (Combination2):

Customer\_id Customer\_name

1 A

2 B

3 C

Data (Combination3):

Customer\_name Subscription\_type

A prepaid

B postpaid

C prepaid

Enter Customer Name to find its details:b

Customer Found:-

Customer Id:2

Customer Name:B

Subscription type:postpaid

Enter subscription type(Prepaid or postpaid):prepaid

Customer Id: Customer Name: (With subscription type-prepaid)

1 A

3 C