Q7. Write a function that accepts two strings and returns the indices of all the occurrences of the second string in the first string as a list. If the second string is not present in the first string then it should return -1.

Solution:

```
def is_equal(counter_one):
        counter_two = 0
        limit = counter_one + len(string_two)
        for i in range(counter_one, limit):
                if string_one[i] != string_two[counter_two]:
                        return False
                counter_two += 1
        return True



def is_present():
        len_first = len(string_one)
        len_second = len(string_two)
        result = []
        counter_one = 0
        while counter_one < len_first:
                if is_equal(counter_one):
                        for i in range(counter_one, counter_one + len_second):
                                result.append(i)
                counter_one += 1
        if len(result) == 0:
                return -1
        return result
```

```
string_one = input("Enter first string: ")
string_two = input("Enter second string: ")
print(is_present())
```

Output 1:

Enter first string: Hello, this is Harsh

Enter second string: Harsh

[15, 16, 17, 18, 19]

Output 2:

Enter first string: Hello, I am Harsh from Harshland.

Enter second string: Harsh

[12, 13, 14, 15, 16, 23, 24, 25, 26, 27]

Q8. WAP to create a list of the cubes of only the even integers appearing in the input list (may have elements of other types also) using for loop and list comprehension.

Solution:

```
# using for loop
def even_cubes(input_list: list):
        cube_list = []
        for i in input_list:
                if type(i) is int and i % 2 == 0:
                        cube_list.append(i**3)
        return cube_list



# using list comprehension
def even_cubes_list_comprehension(input_list: list):
        cube_list = [i**3 for i in input_list if type(i) is int and i%2==0]
        return cube_list

values = [3, 4, 3, 5, 2, 4, 5, 5, 6, 3, (3, 2, 46, 2)]
print(even_cubes(values))
print(even_cubes_list_comprehension(values))
```

Output:

[64, 8, 64, 216]

[64, 8, 64, 216]

Q9. WAP to read a file and

a. Print the total number of characters, words and lines in the file.
b. Calculate the frequency of each character in the file. Use a variable of dictionary type to maintain the count.
c. Print the words in reverse order.
d. Copy even lines of the file to a file named ‘File1’ and odd lines to another file named ‘File2’.

Solution:

```
file = open("sample_file.txt", "r")


# 1. Total number of characters, words and lines in file.
file_data = file.read()
number_of_characters = len(file_data)
number_of_words = len(file_data.split())
number_of_lines = len(file_data.split("\n"))
print("No. of characters in given file:", number_of_characters)
print("No. of words in given file:", number_of_words)
print("No. of lines in given file:", number_of_lines)



# 2. Frequency of each character in file.
character_frequency = {}
for character in file_data:
        character_frequency[character] = character_frequency.get(character,
                                                                  0) + 1
print("Frequency of each character in file:", str(character_frequency))
```

```
# 3. Words in Reverse order.
reverse_words = file_data.split(" ")
reverse_words = " ".join(reversed(reverse_words))
print("Words of file in reverse order:", reverse_words)


# 4. Even lines in "file1" and odd lines in "file2".
file1 = open("file1.txt", "w")
file2 = open("file2.txt", "w")
file_data_write = file_data.split("\n")
for i in range(len(file_data_write)):
        if i % 2 == 1:
                file1.write(file_data_write[i] + "\n")
        else:
                file2.write(file_data_write[i] + "\n")


file.close()
```

Output:

No. of characters in given file: 108

No. of words in given file: 17

No. of lines in given file: 5

Frequency of each character in file:

{' ': 21, 'H': 2, 'e': 8, 'l': 7, 'o': 7, 'E': 1, 'v': 1, 'r': 3, 'y': 4, 'n': 11, '\n': 4, 'T': 1, 'h': 3, 'i': 7, 's': 5, 'P': 2, 't': 4, 'f': 2, 'a': 5, 'd': 2, 'g': 2, 'S': 2, 'u': 2, 'L': 1, "'": 1}

Words of file in reverse order:

learn let's So

fun and easy is Python Learning

Solution Handling file Python is This

Everyone Hello

Q10. WAP to define a class Point with coordinates x and y as attributes. Create relevant methods and print the objects. Also define a method distance to calculate the distance between any two point objects.

Solution:

```
import math



class Point:
        global x
        global y



        def __init__(self, x: int, y: int):
                self.x = x
                self.y = y



        def print_attributes(self):
                print("Point x =", self.x)
                print("Point y =", self.y)



        def distance(point_one: Point, point_two: Point):
                x_coordinate_difference = point_two.x - point_one.x
                y_coordinate_difference = point_two.y - point_one.y
```

```
        # Calculate difference between points using the formula -
                sqrt((x1 - x2)^2 + (y1 - y2)^2)
        result = math.sqrt(x_coordinate_difference ** 2 +
                                    y_coordinate_difference ** 2)
        return result



print("Enter values for 1st point")
x_coordinate_one = int(input("Enter the value of x coordinate: "))
y_coordinate_one = int(input("Enter the value of y coordinate: "))


point_initial = Point(x_coordinate_one, y_coordinate_one)


print("Enter values for 2nd point")
x_coordinate_two = int(input("Enter the value of x coordinate: "))
y_coordinate_two = int(input("Enter the value of y coordinate: "))


point_final = Point(x_coordinate_two, y_coordinate_two)


print("Your entered points are: ")
print("Point 1: ")
point_initial.print_attributes()
print("Point 2: ")
point_final.print_attributes()


print("Distance between the two points: ", distance(point_initial, point_final))
```

Output 1:

Enter values for 1st point

Enter the value of x coordinate: 3

Enter the value of y coordinate: 2

Enter values for 2nd point

Enter the value of x coordinate: 7

Enter the value of y coordinate: 5

Your entered points are:

Point 1:

Point x = 3

Point y = 2

Point 2:

Point x = 7

Point y = 5

Distance between the two points: 5.0

Output 2:

Enter values for 1st point

Enter the value of x coordinate: 0

Enter the value of y coordinate: 0

Enter values for 2nd point

Enter the value of x coordinate: 6

Enter the value of y coordinate: 8

Your entered points are:

Point 1:

Point x = 0

Point y = 0

Point 2:

Point x = 6

Point y = 8

Distance between the two points: 10.0

Q11. Write a function that prints a dictionary where the keys are numbers between 1 and 5 and the values are cubes of the keys.

Solution:

```
def print_dict_cube():
        cubes_dict = dict()
        for i in range(1, 6):
                cubes_dict[i] = i**3
        print(cubes_dict)



print("Cubes between 1 and 5")
print_dict_cube()
```

Output:

```
Cubes between 1 and 5
{1: 1, 2: 8, 3: 27, 4: 64, 5: 125}
```

Q12. Consider a tuple t1=(1, 2, 5, 7, 9, 2, 4, 6, 8, 10). WAP to perform following operations:

a. Print half the values of the tuple in one line and the other half in the next line.
b. Print another tuple whose values are even numbers in the given tuple.
c. Concatenate a tuple t2=(11,13,15) with t1.
d. Return maximum and minimum value from this tuple.

Solution:

```
t1 = (1, 2, 5, 7, 9, 2, 4, 6, 8, 10)



# other half in the next line.
tuple_length = len(t1) // 2
print("First Half")
print(t1[:tuple_length])
print("Another Half")
print(t1[tuple_length:])


# 2. Print even values
tuple_even = tuple(i for i in t1 if i % 2 == 0)
print("Even tuple values")
print(tuple_even[:])


# 3. Concatenate another tuple.
t2 = (11, 13, 15)
t1 = t1 + t2
```

```
print("Concatenated Tuple")

print(t1[:])


# 4. Return maximum and minimum value from this tuple.

min = min(t1)

max = max(t1)

print("Minimum value in tuple:", min)

print("Maximum value in tuple:", max)
```

Output:

```
First Half

(1, 2, 5, 7, 9)

Another Half

(2, 4, 6, 8, 10)

Even tuple values

(2, 2, 4, 6, 8, 10)

Concatenated Tuple

(1, 2, 5, 7, 9, 2, 4, 6, 8, 10, 11, 13, 15)

Minimum value in tuple: 1

Maximum value in tuple: 15
```

Q13. WAP to accept a name from a user. Raise and handle appropriate exception(s) if the text entered by the user contains digits and/or special characters.

Solution:

```
user_name = input("Enter Name: ")

try:

        for character in user_name:

                if character.isalpha() or character.isspace():

                        continue

                else:

                        raise Exception("Entered username contains Number or
                                                special Character")

        print("Username is valid and contains no number or special
                                                character.")

except:

print("Error:")

print("Entered String is invalid as it contains Number or Special
                                                Character")
```

Output 1:

Enter Name: Harsh Dubey

Username is valid and contains no number or special character.

Output 2:

Enter Name: Harsh Dubey 2023

Error:

Entered String contains Number or Special Character