**St Stephen College**

**University of Delhi**

**Course :- Bsc. (prog) Computer Science**

**Semester :- 3rd**

**Year :- 2025-2026**

**Paper Name :- Object Oriented Programming Using Python**

**Unique Paper Code :- 2342011104**

**Practical File**

**Roll Number :- 24080582015**

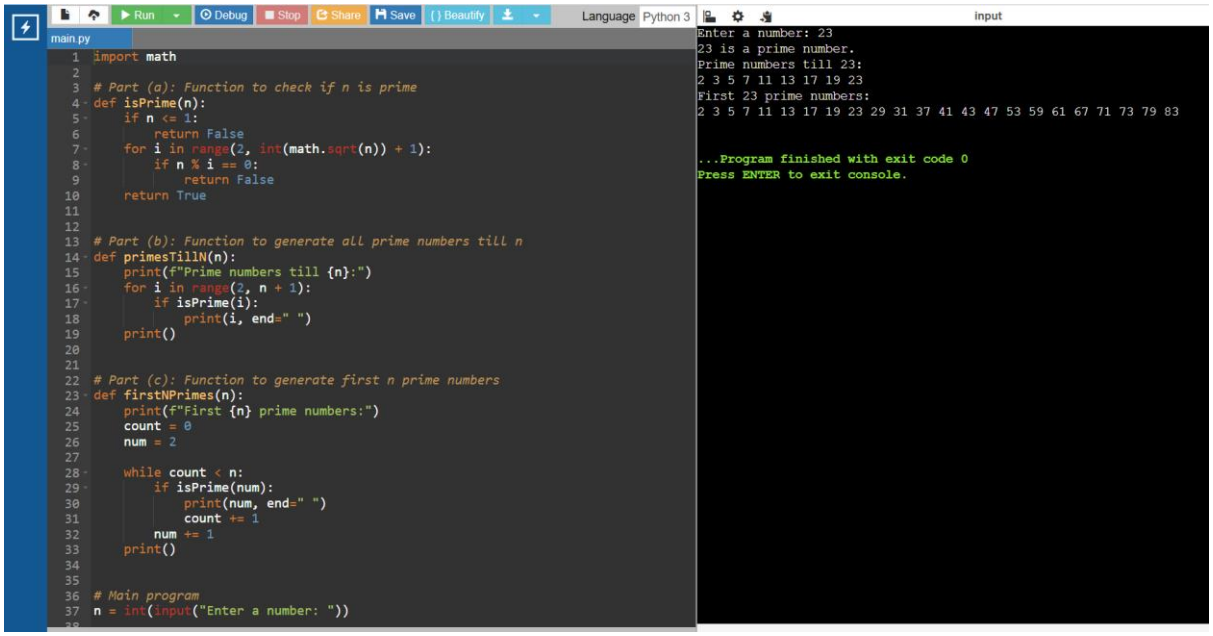
**Name :- Vishal Kumar Jha**

## Practical 1 :- Write a program to find the roots of a quadratic equation.

```python
import math

a = float(input("Enter value of a: "))
b = float(input("Enter value of b: "))
c = float(input("Enter value of c: "))

# Calculate discriminant
d = b*b - 4*a*c

if d > 0:
    root1 = (-b + math.sqrt(d)) / (2*a)
    root2 = (-b - math.sqrt(d)) / (2*a)
    print("Roots are real and different:")
    print("Root 1 =", root1)
    print("Root 2 =", root2)

elif d == 0:
    root = -b / (2*a)
    print("Roots are real and equal:")
    print("Root =", root)

else:
    real = -b / (2*a)
    imag = math.sqrt(-d) / (2*a)
    print("Roots are complex:")
    print(f"Root 1 = {real} + {imag}i")
    print(f"Root 2 = {real} - {imag}i")
```

```
Enter value of a: 1
Enter value of b: 6
Enter value of c: 2
Roots are real and different:
Root 1 = -0.3542486889354093
Root 2 = -5.645751311064591


...Program finished with exit code 0
Press ENTER to exit console.
```

## Practical 2 :- Write a program to accept a number 'n' and a. Check if 'n' is prime b. Generate all prime numbers till 'n' c. Generate first 'n' prime numbers This program may be done using functions.

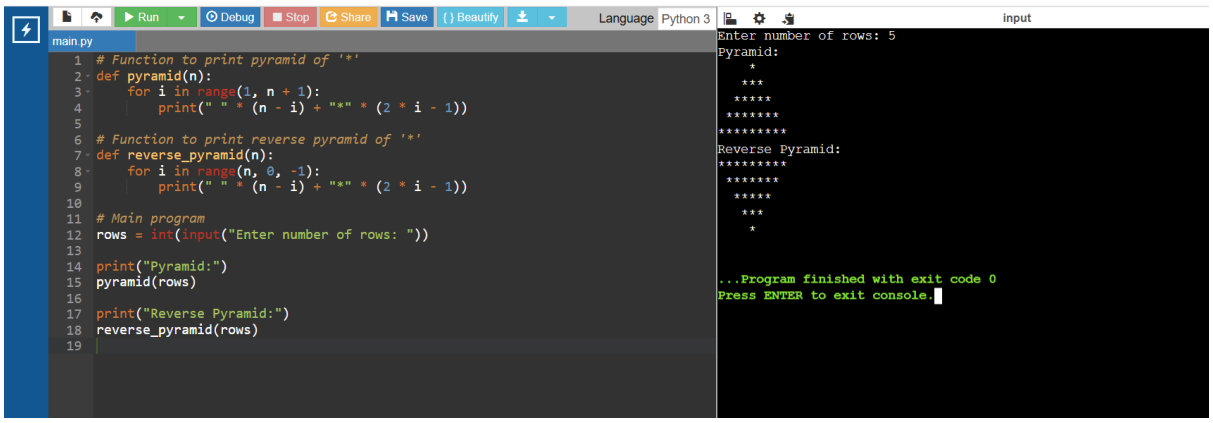


```python
import math

# Part (a): Function to check if n is prime
def isPrime(n):
    if n <= 1:
        return False
    for i in range(2, int(math.sqrt(n)) + 1):
        if n % i == 0:
            return False
    return True


# Part (b): Function to generate all prime numbers till n
def primesTillN(n):
    print(f"Prime numbers till {n}:")
    for i in range(2, n + 1):
        if isPrime(i):
            print(i, end=" ")
    print()


# Part (c): Function to generate first n prime numbers
def firstNPrimes(n):
    print(f"First {n} prime numbers:")
    count = 0
    num = 2

    while count < n:
        if isPrime(num):
            print(num, end=" ")
            count += 1
        num += 1
    print()


# Main program
n = int(input("Enter a number: "))
```

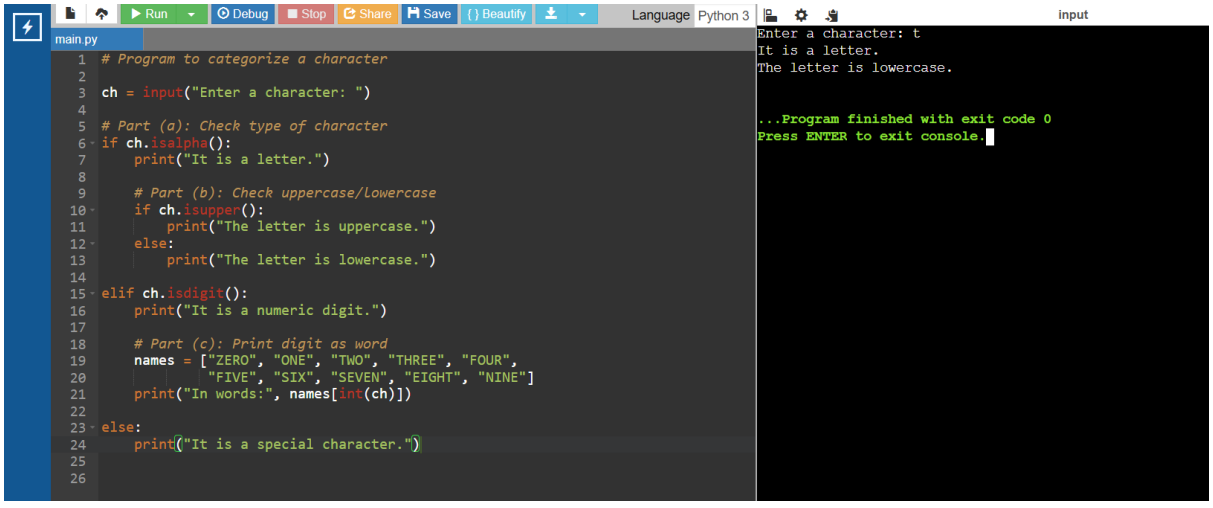
```
Enter a number: 23
23 is a prime number.
Prime numbers till 23:
2 3 5 7 11 13 17 19 23
First 23 prime numbers:
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83


...Program finished with exit code 0
Press ENTER to exit console.
```

```python

# Main program
n = int(input("Enter a number: "))

# Part (a)
if isPrime(n):
    print(n, "is a prime number.")
else:
    print(n, "is not a prime number.")

# Part (b)
primesTillN(n)

# Part (c)
firstNPrimes(n)
```

## Practical 3 :- Write a program to create a pyramid of the character '*' and a reverse pyramid.

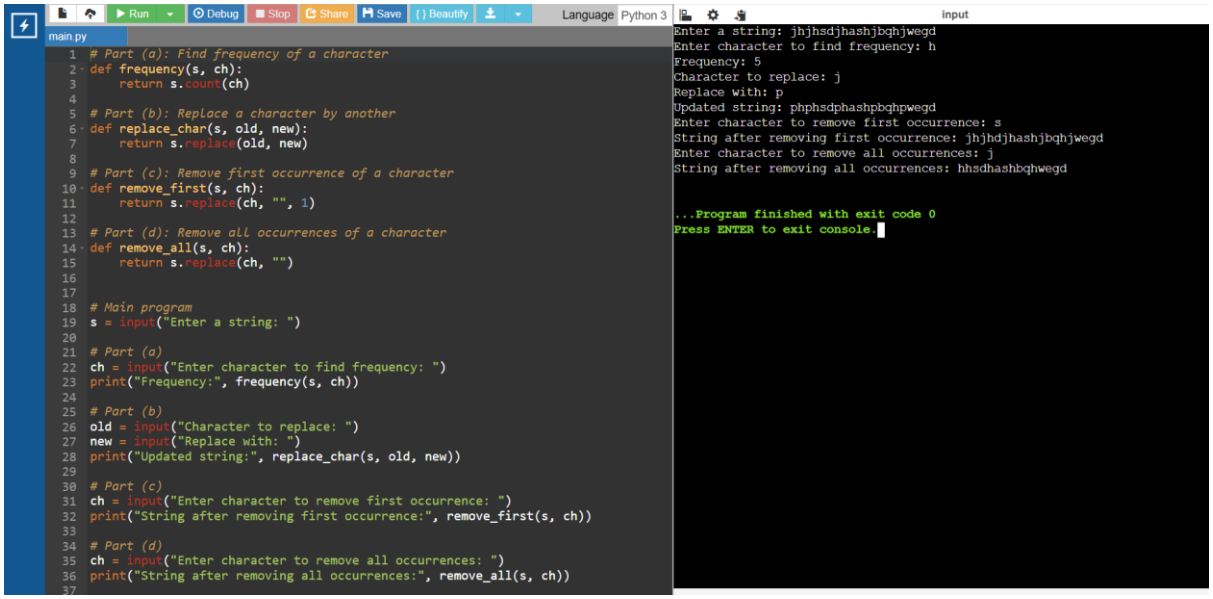



## Practical 4 :- Write a program that accepts a character and performs the following: a. print whether the character is a letter or numeric digit or a special character. b. if the character is a letter, print whether the letter is uppercase or lowercase c. if the character is a numeric digit, prints its name in text (e.g., if input is 9, output is NINE)




## Practical 5 :- Write a program to perform the following operations on a string: a. Find the frequency of a character in a string. b. Replace a character by another character in a string.

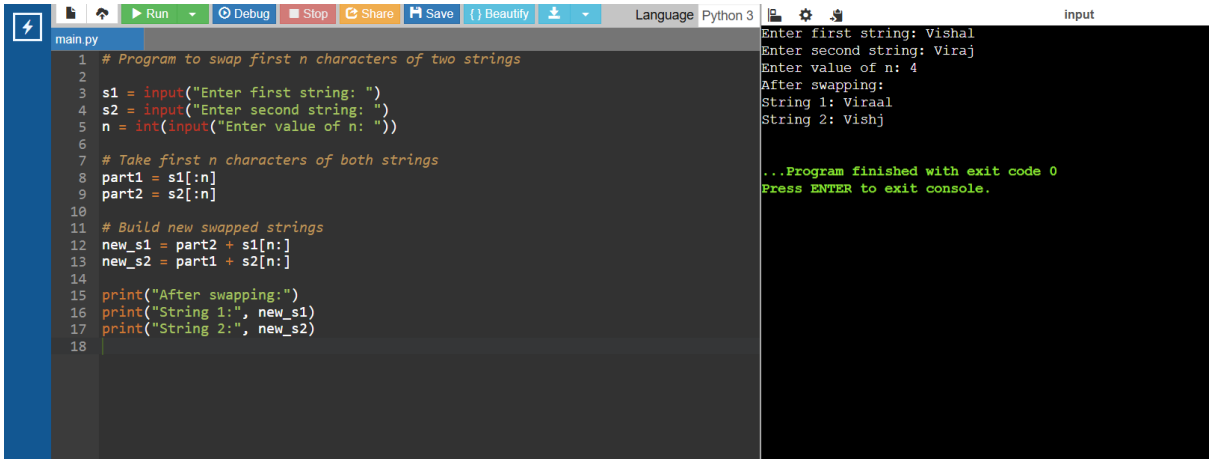
## c. Remove the first occurrence of a character from a string. d. Remove all occurrences of a character from a string.



```
# Part (a): Find frequency of a character
def frequency(s, ch):
    return s.count(ch)

# Part (b): Replace a character by another
def replace_char(s, old, new):
    return s.replace(old, new)

# Part (c): Remove first occurrence of a character
def remove_first(s, ch):
    return s.replace(ch, "", 1)

# Part (d): Remove all occurrences of a character
def remove_all(s, ch):
    return s.replace(ch, "")


# Main program
s = input("Enter a string: ")

# Part (a)
ch = input("Enter character to find frequency: ")
print("Frequency:", frequency(s, ch))

# Part (b)
old = input("Character to replace: ")
new = input("Replace with: ")
print("Updated string:", replace_char(s, old, new))

# Part (c)
ch = input("Enter character to remove first occurrence: ")
print("String after removing first occurrence:", remove_first(s, ch))

# Part (d)
ch = input("Enter character to remove all occurrences: ")
print("String after removing all occurrences:", remove_all(s, ch))
```

```
Enter a string: jhjhsdjhashjbqhjwegd
Enter character to find frequency: h
Frequency: 5
Character to replace: j
Replace with: p
Updated string: phphsdphashpbqhpwegd
Enter character to remove first occurrence: s
String after removing first occurrence: jhjhdjhashjbqhjwegd
Enter character to remove all occurrences: j
String after removing all occurrences: hhsdhashbqhwegd


...Program finished with exit code 0
Press ENTER to exit console.
```

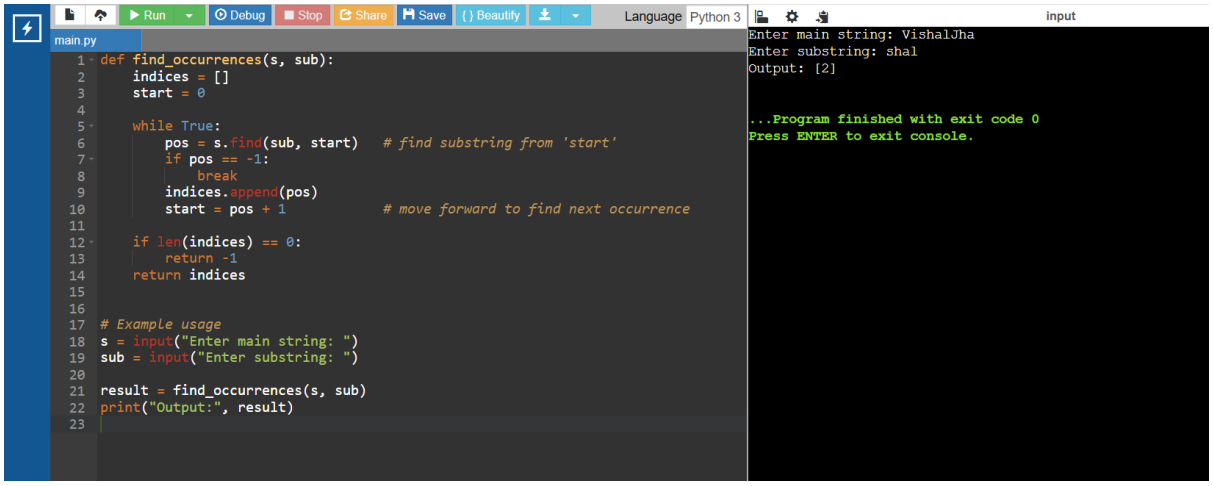
## Practical 6:- Write a program to swap the first n characters of two strings.



```
# Program to swap first n characters of two strings

s1 = input("Enter first string: ")
s2 = input("Enter second string: ")
n = int(input("Enter value of n: "))

# Take first n characters of both strings
part1 = s1[:n]
part2 = s2[:n]

# Build new swapped strings
new_s1 = part2 + s1[n:]
new_s2 = part1 + s2[n:]

print("After swapping:")
print("String 1:", new_s1)
print("String 2:", new_s2)
```

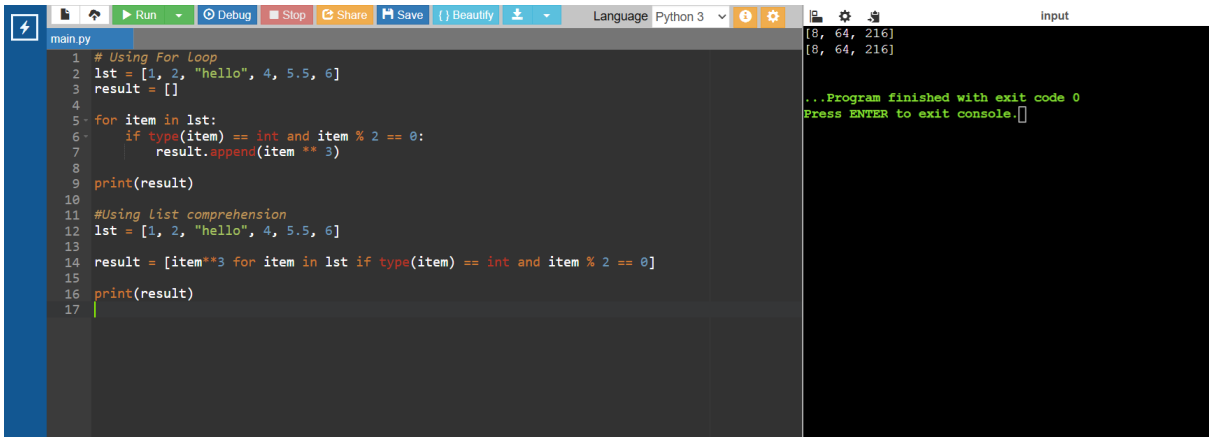
```
Enter first string: Vishal
Enter second string: Viraj
Enter value of n: 4
After swapping:
String 1: Viraal
String 2: Vishj


...Program finished with exit code 0
Press ENTER to exit console.
```

## Practical 7 :- 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.




```python
def find_occurrences(s, sub):
    indices = []
    start = 0

    while True:
        pos = s.find(sub, start)   # find substring from 'start'
        if pos == -1:
            break
        indices.append(pos)
        start = pos + 1            # move forward to find next occurrence

    if len(indices) == 0:
        return -1
    return indices


# Example usage
s = input("Enter main string: ")
sub = input("Enter substring: ")

result = find_occurrences(s, sub)
print("Output:", result)
```

## Practical 8 :- Write a program 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 the following: a. 'for' loop b. list comprehension

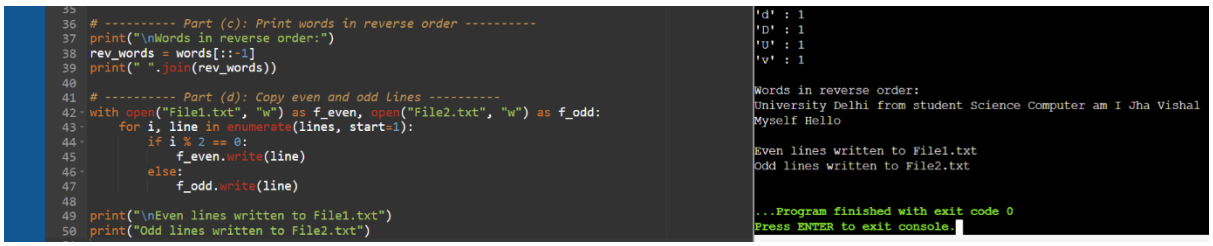


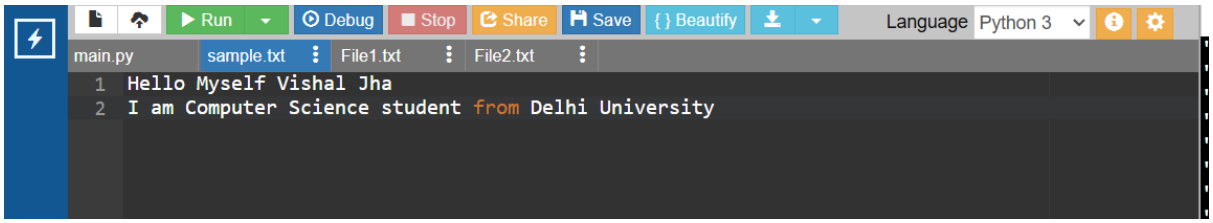

```python
# Using For Loop
lst = [1, 2, "hello", 4, 5.5, 6]
result = []

for item in lst:
    if type(item) == int and item % 2 == 0:
        result.append(item ** 3)

print(result)

#Using list comprehension
lst = [1, 2, "hello", 4, 5.5, 6]

result = [item**3 for item in lst if type(item) == int and item % 2 == 0]

print(result)
```

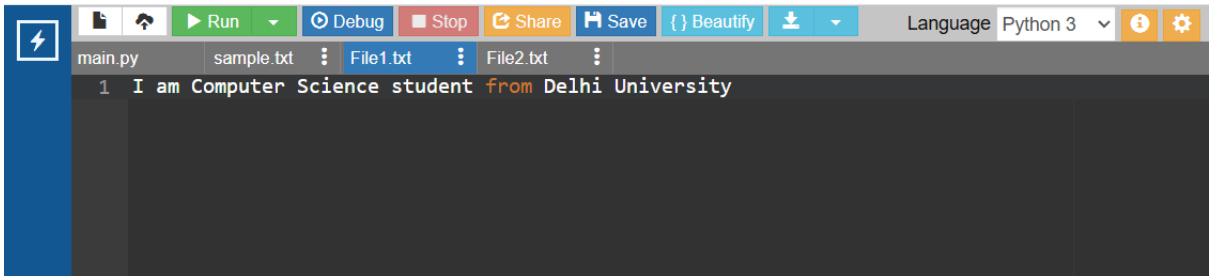
## Practical 9 :- Write a program to read a file and

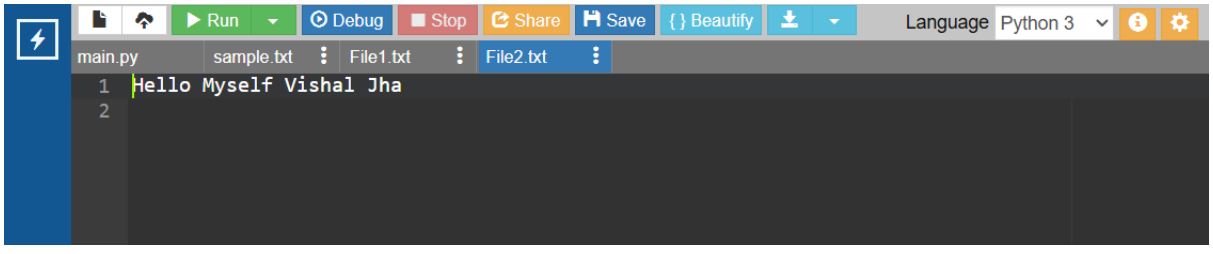
a. Print the total number of characters, words and lines in the file.
b. Calculate the frequency of each character in the file. Use a variable of dictionary type to maintain the count.
c. Print the words in reverse order.
d. Copy even lines of the file to a file named 'File1' and odd lines to another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


Run
Debug
Stop
Share
Save
{ } Beautify
Language
Python 3
main.py
sample.txt
File1.txt
File2.txt
filename = "sample.txt"
# ---------- Part (a): Counts ----------
with open(filename, "r") as f:
content = f.read()
# total characters
num_chars = len(content)
# total words
words = content.split()
num_words = len(words)
# total lines
with open(filename, "r") as f:
lines = f.readlines()
num_lines = len(lines)
print("Total Characters:", num_chars)
print("Total Words:", num_words)
print("Total Lines:", num_lines)
# ---------- Part (b): Character Frequency ----------
freq = {}
for ch in content:
if ch in freq:
freq[ch] += 1
else:
freq[ch] = 1
print("\nCharacter Frequencies:")
for ch, count in freq.items():
print(repr(ch), ":", count)
Total Characters: 75
Total Words: 12
Total Lines: 2
Character Frequencies:
'H' : 1
'e' : 8
'l' : 5
'o' : 3
' ' : 10
'M' : 1
'y' : 2
's' : 4
'f' : 2
'V' : 1
'i' : 5
'h' : 3
'a' : 3
'J' : 1
'\n' : 1
'I' : 1
'm' : 3
'C' : 1
'p' : 1
'u' : 2
't' : 4
'r' : 3
'S' : 1
'c' : 2
'n' : 3
'd' : 1
'D' : 1
'U' : 1
'v' : 1

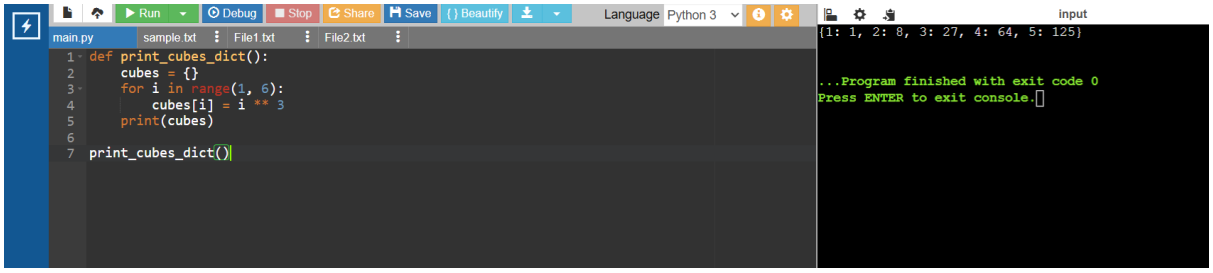

# ---------- Part (c): Print words in reverse order ----------
print("\nWords in reverse order:")
rev_words = words[::-1]
print(" ".join(rev_words))
# ---------- Part (d): Copy even and odd lines ----------
with open("File1.txt", "w") as f_even, open("File2.txt", "w") as f_odd:
for i, line in enumerate(lines, start=1):
if i % 2 == 0:
f_even.write(line)
else:
f_odd.write(line)
print("\nEven lines written to File1.txt")
print("Odd lines written to File2.txt")
'd' : 1
'D' : 1
'U' : 1
'v' : 1
Words in reverse order:
University Delhi from student Science Computer am I Jha Vishal
Myself Hello
Even lines written to File1.txt
Odd lines written to File2.txt
...Program finished with exit code 0
Press ENTER to exit console.


Run
Debug
Stop
Share
Save
{ } Beautify
Language
Python 3
main.py
sample.txt
File1.txt
File2.txt
Hello Myself Vishal Jha
I am Computer Science student from Delhi University

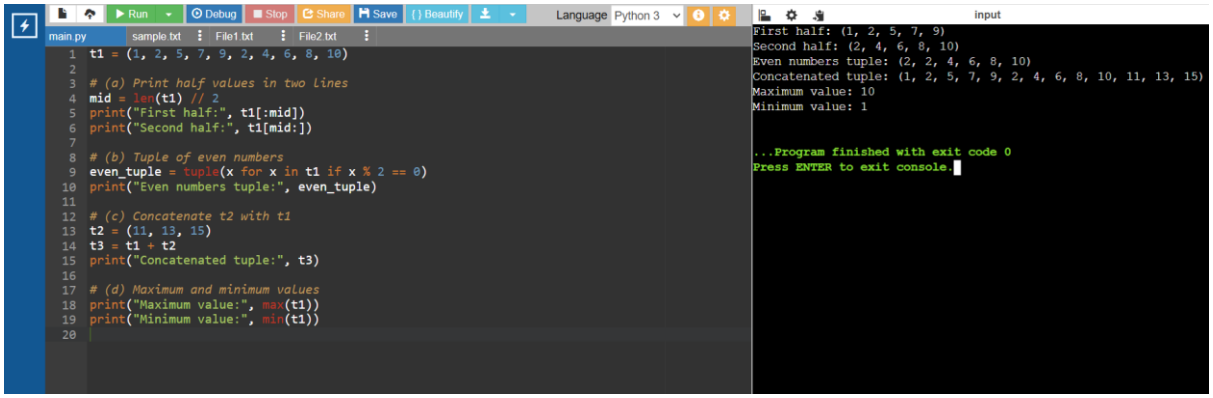

Run
Debug
Stop
Share
Save
{ } Beautify
Language
Python 3
main.py
sample.txt
File1.txt
File2.txt
I am Computer Science student from Delhi University


Run
Debug
Stop
Share
Save
{ } Beautify
Language
Python 3
main.py
sample.txt
File1.txt
File2.txt
Hello Myself Vishal Jha

**Practical 10 :- Write a function that prints a dictionary where the keys are numbers between 1 and 5 and the values are the cubes of the keys.**



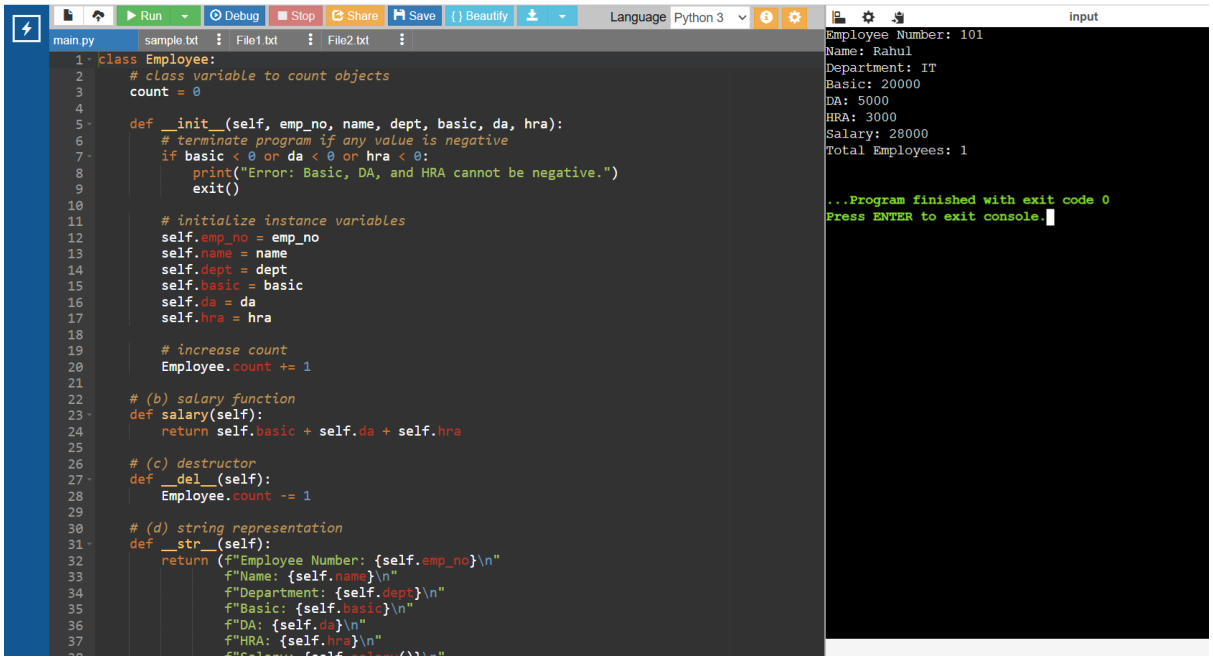

**Practical 11 :- Consider a tuple t1=(1, 2, 5, 7, 9, 2, 4, 6, 8, 10). Write a program to perform following operations: a) Print half the values of the tuple in one line and the other half in the next line. b) Print another tuple whose values are even numbers in the given tuple. c) Concatenate a tuple t2=(11, 13, 15) with t1. d) Return maximum and minimum value from this tuple.**




**Practical 12 :- Define a class Employee that stores information about employees in the company. The class should contain the following:**

(i) **data members - count (to keep a record of all the objects being created for this class) and for every employee: $\text{employee number, Name, Dept, Basic, DA}$ and $\text{HRA}$.**

(ii) **function members: a. $\text{__init__}$ method to initialize and/or update the members. Add statements to ensure that the program is terminated if any of $\text{Basic, DA}$ and $\text{HRA}$ is set to a negative value. b. $\text{function salary}$, that returns salary as the sum of $\text{Basic, DA}$ and $\text{HRA}$. c. $\text{__del__}$ function to decrease the number of objects created for the class. d. $\text{__str__}$ function to display the details of an employee along with the salary of an employee in a proper format.**



```
class Employee:
    # class variable to count objects
    count = 0

    def __init__(self, emp_no, name, dept, basic, da, hra):
        # terminate program if any value is negative
        if basic < 0 or da < 0 or hra < 0:
            print("Error: Basic, DA, and HRA cannot be negative.")
            exit()

        # initialize instance variables
        self.emp_no = emp_no
        self.name = name
        self.dept = dept
        self.basic = basic
        self.da = da
        self.hra = hra

        # increase count
        Employee.count += 1

    # (b) salary function
    def salary(self):
        return self.basic + self.da + self.hra

    # (c) destructor
    def __del__(self):
        Employee.count -= 1

    # (d) string representation
    def __str__(self):
        return (f"Employee Number: {self.emp_no}\n"
                f"Name: {self.name}\n"
                f"Department: {self.dept}\n"
                f"Basic: {self.basic}\n"
                f"DA: {self.da}\n"
                f"HRA: {self.hra}\n"
```

```
Employee Number: 101
Name: Rahul
Department: IT
Basic: 20000
DA: 5000
HRA: 3000
Salary: 28000
Total Employees: 1


...Program finished with exit code 0
Press ENTER to exit console.
```

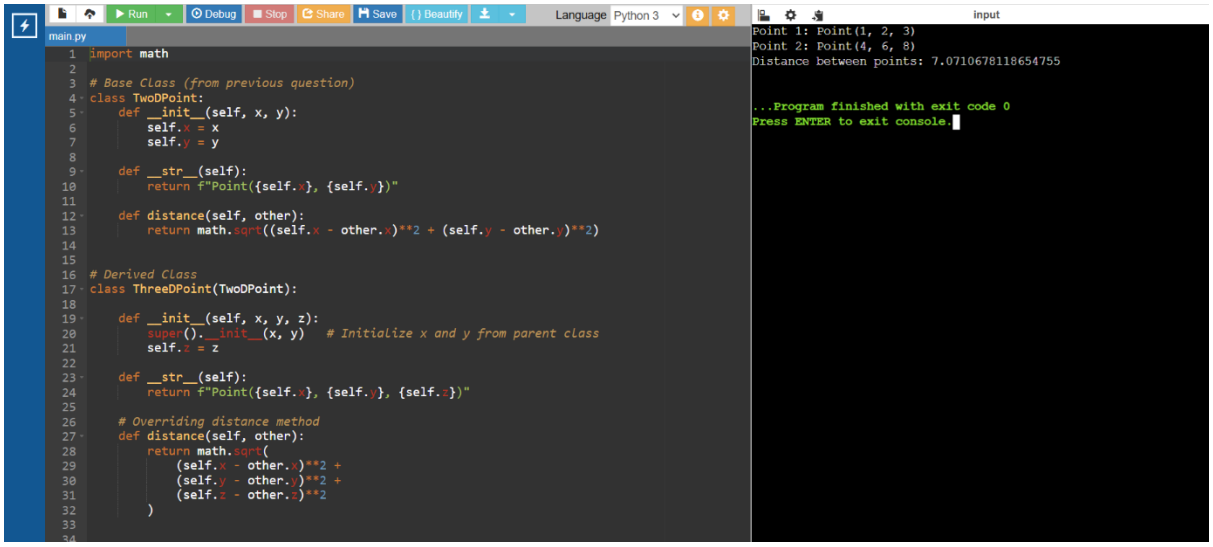
```
    # (d) string representation
    def __str__(self):
        return (f"Employee Number: {self.emp_no}\n"
                f"Name: {self.name}\n"
                f"Department: {self.dept}\n"
                f"Basic: {self.basic}\n"
                f"DA: {self.da}\n"
                f"HRA: {self.hra}\n"
                f"Salary: {self.salary()}\n"
                f"Total Employees: {Employee.count}")

# ---------------- Example Usage ----------------

e1 = Employee(101, "Rahul", "IT", 20000, 5000, 3000)
print(e1)
```

## Practical 13 :- Write a program to define a class '2DPoint' 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.

```
import math

class TwoDPoint:

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

    # method to print object details
    def __str__(self):
        return f"Point({self.x}, {self.y})"

    # method to calculate distance from another point
    def distance(self, other):
        return math.sqrt((self.x - other.x)**2 + (self.y - other.y)**2)


# ---------- Example Usage ----------

p1 = TwoDPoint(3, 4)
p2 = TwoDPoint(7, 1)

print("Point 1:", p1)
print("Point 2:", p2)

print("Distance between points:", p1.distance(p2))
```

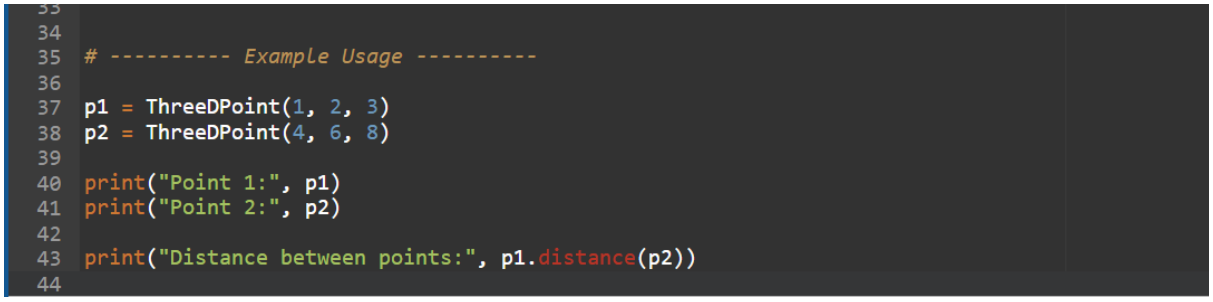
```
Point 1: Point(3, 4)
Point 2: Point(7, 1)
Distance between points: 5.0


...Program finished with exit code 0
Press ENTER to exit console.
```

## Practical 14 :- Inherit the above class (referring to '2DPoint' from Q13) to create a "3Dpoint" with additional attribute z. Override the method defined in "2DPoint" class, to calculate distance between two points of the "3Dpoint" class.



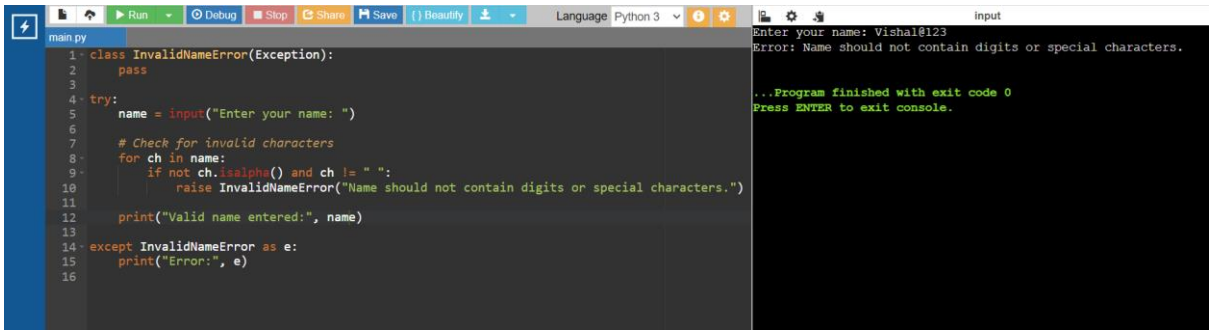
```python
import math

# Base Class (from previous question)
class TwoDPoint:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __str__(self):
        return f"Point({self.x}, {self.y})"

    def distance(self, other):
        return math.sqrt((self.x - other.x)**2 + (self.y - other.y)**2)


# Derived Class
class ThreeDPoint(TwoDPoint):

    def __init__(self, x, y, z):
        super().__init__(x, y)   # Initialize x and y from parent class
        self.z = z

    def __str__(self):
        return f"Point({self.x}, {self.y}, {self.z})"

    # Overriding distance method
    def distance(self, other):
        return math.sqrt(
            (self.x - other.x)**2 +
            (self.y - other.y)**2 +
            (self.z - other.z)**2
        )
```

```
Point 1: Point(1, 2, 3)
Point 2: Point(4, 6, 8)
Distance between points: 7.0710678118654755


...Program finished with exit code 0
Press ENTER to exit console.
```



```python


# ---------- Example Usage ----------

p1 = ThreeDPoint(1, 2, 3)
p2 = ThreeDPoint(4, 6, 8)

print("Point 1:", p1)
print("Point 2:", p2)

print("Distance between points:", p1.distance(p2))

```

## Practical 15 :- Write a program 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.



```python
class InvalidNameError(Exception):
    pass

try:
    name = input("Enter your name: ")

    # Check for invalid characters
    for ch in name:
        if not ch.isalpha() and ch != " ":
            raise InvalidNameError("Name should not contain digits or special characters.")

    print("Valid name entered:", name)

except InvalidNameError as e:
    print("Error:", e)
```

```
Enter your name: Vishal@123
Error: Name should not contain digits or special characters.


...Program finished with exit code 0
Press ENTER to exit console.
```