**Week1**

3) i) Write a program to calculate compound interest when principal, rate and number of periods are given.

**Program:**

def calculate\_compound\_interest(principal, rate, time):

"""

Calculate the compound interest.

:param principal: The initial amount of money (float or int).

:param rate: The annual interest rate (float or int).

:param time: The number of periods the money is invested for (float or int).

:return: The compound interest (float).

"""

# Convert rate from percentage to decimal

rate\_decimal = rate / 100

# Calculate the compound interest

amount = principal \* (1 + rate\_decimal) \*\* time

compound\_interest = amount - principal

return compound\_interest

# Input values

principal = float(input("Enter the principal amount: "))

rate = float(input("Enter the annual interest rate (in percentage): "))

time = float(input("Enter the number of periods: "))

# Calculate compound interest

interest = calculate\_compound\_interest(principal, rate, time)

# Output the result

print(f"The compound interest is: {interest:.2f}")

**Output:**

Enter the principal amount: 10000

Enter the annual interest rate (in percentage): 2

Enter the number of periods: 12

The compound interest is: 2682.42

3)ii) Given coordinates (x1, y1), (x2, y2) find the distance between two points

**Program:**

import math

def calculate\_distance(x1, y1, x2, y2):

"""

Calculate the distance between two points (x1, y1) and (x2, y2).

:param x1: x-coordinate of the first point (float or int).

:param y1: y-coordinate of the first point (float or int).

:param x2: x-coordinate of the second point (float or int).

:param y2: y-coordinate of the second point (float or int).

:return: The distance between the two points (float).

"""

distance = math.sqrt((x2 - x1)\*\*2 + (y2 - y1)\*\*2)

return distance

# Input coordinates

x1 = float(input("Enter x-coordinate of the first point: "))

y1 = float(input("Enter y-coordinate of the first point: "))

x2 = float(input("Enter x-coordinate of the second point: "))

y2 = float(input("Enter y-coordinate of the second point: "))

# Calculate distance

distance = calculate\_distance(x1, y1, x2, y2)

# Output the result

print(f"The distance between the points ({x1}, {y1}) and ({x2}, {y2}) is: {distance:.2f}")

**Output:**

Enter x-coordinate of the first point: 122

Enter y-coordinate of the first point: 12

Enter x-coordinate of the second point: 168

Enter y-coordinate of the second point: 16

The distance between the points (122.0, 12.0) and (168.0, 16.0) is: 46.17

4. Read name, address, email and phone number of a person through keyboard and print the details.

**Program:**

def get\_person\_details():

"""

Function to read a person's details from the keyboard.

"""

name = input("Enter your name: ")

address = input("Enter your address: ")

email = input("Enter your email: ")

phone = input("Enter your phone number: ")

return name, address, email, phone

def print\_person\_details(name, address, email, phone):

"""

Function to print a person's details.

"""

print("\n--- Personal Details ---")

print(f"Name: {name}")

print(f"Address: {address}")

print(f"Email: {email}")

print(f"Phone Number: {phone}")

# Get details from the user

name, address, email, phone = get\_person\_details()

# Print the details

print\_person\_details(name, address, email, phone)

**Output:**

Enter your name: Akshaya

Enter your address: Hyderabad,TN

Enter your email: akshaya@gmail.com

Enter your phone number: 9963368279

--- Personal Details ---

Name: Akshaya

Address: Hyderabad,TN

Email: akshaya@gmail.com

Phone Number: 9963368279

**Week2**

1. **Print the below triangle using for loop. 5**

**4 4**

**3 3 3**

**2 2 2 2**

**1 1 1 1 1**

**Program:**

# Number of rows

rows = 6

# Outer loop to handle the number of rows

for i in range(rows, 0, -1):

# Inner loop to print the repeated number in each row

for j in range(rows - i + 1):

print(i, end=" ")

# Move to the next line after each row

print()

**Output:**

6

5 5

4 4 4

3 3 3 3

2 2 2 2 2

1 1 1 1 1 1

**2. Write a program to check whether the given input is digit or lowercase character or uppercase character or a special character (use 'if-else-if' ladder)**

**Program:**

# Take input from the user

char = input("Enter a single character: ")

# Check if the input is a single character

if len(char) != 1:

print("Please enter only one character!")

else:

# Check if the character is a digit

if char.isdigit():

print(f"'{char}' is a digit.")

# Check if the character is a lowercase letter

elif char.islower():

print(f"'{char}' is a lowercase character.")

# Check if the character is an uppercase letter

elif char.isupper():

print(f"'{char}' is an uppercase character.")

# If none of the above, it is a special character

else:

print(f"'{char}' is a special character.")

**Output:**

Enter a single character: d

'd' is a lowercase character.

Enter a single character: as

Please enter only one character!

Enter a single character: Q

'Q' is an uppercase character.

Enter a single character: 2

'2' is a digit.

**3. Python Program to Print the Fibonacci sequence using while loop**

**Program:**

**# Function to print the Fibonacci sequence**

**def print\_fibonacci(n):**

**# Initialize the first two Fibonacci numbers**

**a, b = 0, 1**

**count = 0**

**# Check if the number of terms is valid**

**if n <= 0:**

**print("Please enter a positive integer.")**

**elif n == 1:**

**print(f"Fibonacci sequence up to {n} term:")**

**print(a)**

**else:**

**print(f"Fibonacci sequence up to {n} terms:")**

**while count < n:**

**print(a, end=" ")**

**# Update values**

**a, b = b, a + b**

**count += 1**

**# Input: Number of terms**

**n = int(input("Enter the number of terms: "))**

**# Print the Fibonacci sequence**

**print\_fibonacci(n)**

**Output:**

**Enter the number of terms: 10**

**Fibonacci sequence up to 10 terms:**

**0 1 1 2 3 5 8 13 21 34**

**4. Python program to print all prime numbers in a given interval (use break**

**Program:**

**# Function to check if a number is prime**

**def is\_prime(num):**

**if num <= 1:**

**return False**

**for i in range(2, int(num\*\*0.5) + 1):**

**if num % i == 0:**

**return False**

**return True**

**# Input: Interval from the user**

**start = int(input("Enter the start of the interval: "))**

**end = int(input("Enter the end of the interval: "))**

**print(f"Prime numbers between {start} and {end} are:")**

**# Loop through the interval and print prime numbers**

**for num in range(start, end + 1):**

**if is\_prime(num):**

**print(num, end=" ")**

**Output:**

**Enter the start of the interval: 90**

**Enter the end of the interval: 100**

**Prime numbers between 90 and 100 are:**

**97**

**Week3**

1. **i) Write a program to convert a list and tuple into arrays.**

**Program:**

import array

# Convert a list to an array

my\_list = [1, 2, 3, 4, 5]

array\_from\_list = array.array('i', my\_list) # 'i' denotes integer type

# Convert a tuple to an array

my\_tuple = (6, 7, 8, 9, 10)

array\_from\_tuple = array.array('i', my\_tuple) # 'i' denotes integer type

# Print the arrays

print("Array from list:", array\_from\_list)

print("Array from tuple:", array\_from\_tuple)

**Output:**

Array from list: array('i', [1, 2, 3, 4, 5])

Array from tuple: array('i', [6, 7, 8, 9, 10])

1. **ii) Write a program to find common values between two arrays.**

**Program:**

# Define two arrays (lists)

array1 = [1, 2, 3, 4, 5]

array2 = [4, 5, 6, 7, 8]

# Convert arrays to sets and find intersection

common\_values = set(array1).intersection(set(array2))

# Convert the result back to a list (optional)

common\_values = list(common\_values)

# Print the common values

print("Common values between the two arrays:", common\_values)

**Output:**

Common values between the two arrays: [4, 5]

**2. Write a function called gcd that takes parameters a and b and returns their greatest common divisor.**

**Program:**

def gcd(a, b):

"""

Compute the Greatest Common Divisor (GCD) of two numbers using the Euclidean Algorithm.

:param a: First number (integer).

:param b: Second number (integer).

:return: GCD of a and b.

"""

while b != 0:

a, b = b, a % b # Update a to b and b to the remainder of a divided by b

return a

# Example usage

num1 = int(input("Enter the first number: "))

num2 = int(input("Enter the second number: "))

result = gcd(num1, num2)

print(f"The GCD of {num1} and {num2} is: {result}")

**Output:**

Enter the first number: 6

Enter the second number: 12

The GCD of 6 and 12 is: 6

**3. Write a function called palindrome that takes a string argument and returns True if it is a palindrome and False otherwise. Remember that you can use the built-in function len to check the length of a string**.

**Program:**

def palindrome(s):

rev = ''.join(reversed(s)) # Reverse the string and join it back

return s == rev # Compare the original string with the reversed string

# Example usage:

print(palindrome("racecar")) # True

print(palindrome("hello")) # False

**Output:**

True

False

**Week4**

**1. Write a function called is\_sorted that takes a list as a parameter and returns True if the list is sorted in ascending order and False otherwise.**

**Program:**

**def is\_sorted(lst):**

**# Check if each element is less than or equal to the next one**

**return all(lst[i] <= lst[i + 1] for i in range(len(lst) - 1))**

**# Example usage:**

**print(is\_sorted([1, 2, 3, 4, 5])) # True**

**print(is\_sorted([1, 3, 2, 4, 5])) # False**

**print(is\_sorted([5, 4, 3, 2, 1])) # False**

**print(is\_sorted([1])) # True (single element is considered sorted)**

**Output:**

True

False

False

**True**

**2. Write a function called has\_duplicates that takes a list and returns True if there is any element that appears more than once. It should not modify the original list.**

**Program:**

def has\_duplicates(lst):

# Use a set to track seen elements

seen = set()

for item in lst:

if item in seen: # If the item is already in the set, it's a duplicate

return True

seen.add(item) # Add the item to the set

return False # No duplicates found

# Example usage:

#print(has\_duplicates([1, 2, 3, 4, 5])) # False

#print(has\_duplicates([1, 2, 3, 4, 2])) # True

#print(has\_duplicates([])) # False (empty list has no duplicates)

#print(has\_duplicates([1])) # False (single element has no duplicates)

**Output:**

False

True

False

False

1. **i). Write a function called remove\_duplicates that takes a list and returns a new list with only the unique elements from the original. Hint: they don’t have to be in the same order.**

**Program:**

def remove\_duplicates(lst):

# Use a set to remove duplicates while preserving order

seen = set()

unique\_list = []

for item in lst:

if item not in seen: # If the item hasn't been seen before

unique\_list.append(item) # Add it to the unique list

seen.add(item) # Mark it as seen

return unique\_list

# Example usage:

print(remove\_duplicates([1, 2, 3, 4, 2, 3, 5])) # [1, 2, 3, 4, 5]

print(remove\_duplicates([1, 1, 1, 1])) # [1]

print(remove\_duplicates([])) # [] (empty list)

**Output:**

[1, 2, 3, 4, 5]

[1]

[]

**2)ii). The wordlist I provided, words.txt, doesn’t contain single letter words. So you might want to add “I”, “a”, and the empty string.**

**Program:**

with open("word.txt","w") as file:

file.write("I ")

file.write("a")

**Output: In word.txt** single letter words will be added.

1. **iii). Write a python code to read dictionary values from the user. Construct a function to invert its content. i.e., keys should be values and values should be keys.**

**Program:**

n = int(input("Enter the no of entries you want to enter : "))

dict = {}

for i in range(n):

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

value = input("Enter the value : ")

dict[key]=value

print(dict)

newdict={}

for pair in dict.items():

newdict[pair[1]]=pair[0]

print(newdict)

**Output:**

Enter the no of entries you want to enter : 1

Enter the key : 12

Enter the value : 1

{12: '1'}

{'1': 12}

1. **i) Add a comma between the characters. If the given word is 'Apple', it should become 'A,p,p,l,e'**

**Program:**

txt = '"a" "p" "p" "l" "e"'

commas\_added = ', '.join(txt.split())

print(commas\_added)

import re

test\_str = input("Enter a word:")

res = re.findall(r'\w+|\S', test\_str)

print("Result : " + str(res))

test\_str = input("Enter a comma-separated list: ")

res = [s.strip() for s in test\_str.split(',')]

print("Parsed List:", res)

**Output:**

"a", "p", "p", "l", "e"

Enter a word:python

Result : ['python']

Enter a comma-separated list: python,java,c,c++,Nodejs

Parsed List: ['python', 'java', 'c', 'c++', 'Nodejs']

**3)ii) Remove the given word in all the places in a string?**

**Program:**

print("Enter the String: ")

text = input()

print("Enter a Word to Delete: ")

word = input()

text = text.replace(word, "")

print()

print(text)

**Output:**

Enter the String:

python programming language

Enter a Word to Delete:

e

python programming language

**3) iii) Write a function that takes a sentence as an input parameter and replaces the first letter of every word with the corresponding upper case letter and the rest of the letters in the word by corresponding letters in lower case without using a built-in function?**

**Program:**

**txt=input("Enter a sentence:")**

**print(txt.title())**

**Output:**

**Enter a sentence:Python workspaces refer to the environment in which you develop and run your Python code**

**Python Workspaces Refer To The Environment In Which You Develop And Run Your Python Code**

**4. Writes a recursive function that generates all binary strings of n-bit length**

**Program:**

def printTheArray(arr, n):

for i in range(n):

print(arr[i], end=" ")

print()

# Function to generate all binary strings

def generateAllBinaryStrings(n, arr, i):

if i == n:

printTheArray(arr, n)

return

arr[i] = 0

generateAllBinaryStrings(n, arr, i + 1)

arr[i] = 1

generateAllBinaryStrings(n, arr, i + 1)

# Driver Code

if \_\_name\_\_ == "\_\_main\_\_":

n = 4

arr = [None] \* n

generateAllBinaryStrings(n, arr, 0)

**Output:**

**0 0 0 0**

**0 0 0 1**

**0 0 1 0**

**0 0 1 1**

**0 1 0 0**

**0 1 0 1**

**0 1 1 0**

**0 1 1 1**

**1 0 0 0**

**1 0 0 1**

**1 0 1 0**

**1 0 1 1**

**1 1 0 0**

**1 1 0 1**

**1 1 1 0**

**1 1 1 1**

**Week5**

**1. i) Write a python program that defines a matrix and prints**

**Program:**

row=int(input("Enter No of Rows for 1st Matrix:"))

column=int(input("Enter No of column for 1nd Matrix:"))

row1=int(input("Enter No of Rows for 2st Matrix:"))

column1=int(input("Enter No of column for 2nd Matrix:"))

X = [[int(input(("Enter value for X[",i,"][",j,"]:")))

for j in range(column)] for i in range(row)]

Y = [[int(input(("Enter value for Y[",i,"][",j,"]:")))

for j in range(column1)] for i in range(row1)]

print("1st Matrix X:",X)

print("2st Matrix Y:",Y)

**Output:**

Enter No of Rows for 1st Matrix:2

Enter No of column for 1nd Matrix:2

Enter No of Rows for 2st Matrix:2

Enter No of column for 2nd Matrix:2

('Enter value for X[', 0, '][', 0, ']:')12

('Enter value for X[', 0, '][', 1, ']:')13

('Enter value for X[', 1, '][', 0, ']:')14

('Enter value for X[', 1, '][', 1, ']:')15

('Enter value for Y[', 0, '][', 0, ']:')16

('Enter value for Y[', 0, '][', 1, ']:')17

('Enter value for Y[', 1, '][', 0, ']:')18

('Enter value for Y[', 1, '][', 1, ']:')19

1st Matrix X: [[12, 13], [14, 15]]

2st Matrix Y: [[16, 17], [18, 19]]

**1) ii) Write a python program to perform addition of two square matrices**

**Program:**

import numpy as np

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

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

result = np.array(X) + np.array(Y)

**print(result)**

**Output:**

**[[10 10 10]**

**[10 10 10]**

**[10 10 10]]**

**1)iii) Write a python program to perform multiplication of two square matrices**

**Program:**

import numpy as np

# Define two square matrices

A = np.array([[1, 2, 3],

[4, 5, 6],

[7, 8, 9]])

B = np.array([[9, 8, 7],

[6, 5, 4],

[3, 2, 1]])

# Perform matrix multiplication

result = np.dot(A, B)

# Print the result

print("Matrix A:")

print(A)

print("\nMatrix B:")

print(B)

print("\nMultiplication Result (A x B):")

print(result)

**Output:**

**Matrix A:**

**[[1 2 3]**

**[4 5 6]**

**[7 8 9]]**

**Matrix B:**

**[[9 8 7]**

**[6 5 4]**

**[3 2 1]]**

**Multiplication Result (A x B):**

**[[ 30 24 18]**

**[ 84 69 54]**

**[138 114 90]]**

**2.How do you make a module? Give an example of construction of a module using different geometrical shapes and operations on them as its functions.**

A **module** is simply a Python file (.py) that contains code — such as functions or classes — you want to reuse.

**1)Create a file named geometry\_module.py**

**Program:**

#geometry\_module.py

**import math**

**# Area of a rectangle**

**def area\_rectangle(length, width):**

**return length \* width**

**# Area of a circle**

**def area\_circle(radius):**

**return math.pi \* radius \*\* 2**

**# Perimeter of a triangle**

**def perimeter\_triangle(a, b, c):**

**return a + b + c**

**# Volume of a cylinder**

**def volume\_cylinder(radius, height):**

**return math.pi \* radius \*\* 2 \* height**

**2)** Create another file in the same folder named main\_script.py

**# main\_script.py**

**import geometry\_module as gm**

**print("Area of rectangle:", gm.area\_rectangle(5, 3))**

**print("Area of circle:", gm.area\_circle(7))**

**print("Perimeter of triangle:", gm.perimeter\_triangle(3, 4, 5))**

**print("Volume of cylinder:", gm.volume\_cylinder(3, 10))**

**3)** **Run the script:**

python main\_script.py

**Output:**

Area of rectangle: 15

Area of circle: 153.93804002589985

Perimeter of triangle: 12

Volume of cylinder: 282.7433388230814

**3. Use the structure of exception handling all general purpose exceptions**

**Program:**

**try:**

**even\_numbers = [2,4,6,8]**

**print(even\_numbers[5])**

**except ZeroDivisionError:**

**print("Denominator cannot be 0.")**

**except IndexError:**

**print("Index Out of Bound.")**

**Output:**

**Index Out of Bound.**

**Week6**

**1. a. Write a function called draw\_rectangle that takes a Canvas and a Rectangle as arguments and draws a representation of the Rectangle on the Canvas.**

**b. Add an attribute named color to your Rectangle objects and modify draw\_rectangle so that it uses the color attribute as the fill color.**

**c. Write a function called draw\_point that takes a Canvas and a Point as arguments and draws a representation of the Point on the Canvas.**

**d. Define a new class called Circle with appropriate attributes and instantiate a few Circle objects. Write a function called draw\_circle that draws circles on the canvas**

**Program:**

import tkinter as tk

root = tk.Tk()

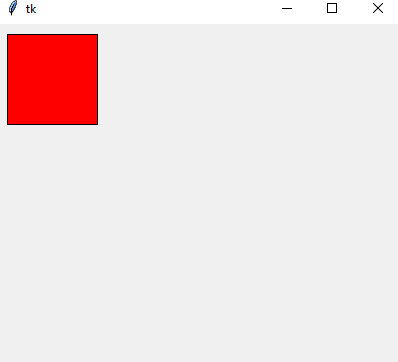
canvas = tk.Canvas(root, width=400, height=400)

canvas.pack()

canvas.create\_rectangle(10, 10, 100, 100, fill='red')

root.mainloop()

**Output:**



**2. Write a Python program to demonstrate the usage of Method Resolution Order (MRO) in multiple levels of Inheritances.**

**Program:**

class ClassA:

def m(self):

print("In ClassA")

class ClassB(ClassA):

def m(self):

print("In ClassB")

class ClassC(ClassA):

def m(self):

print("In ClassC")

class ClassD(ClassB, ClassC):

def m(self):

print("In ClassD")

# Create an object of ClassD

obj = ClassD()

# Call the method from the object

obj.m()

# Check the MRO (Method Resolution Order) of ClassD

print("\nMRO of ClassD:", ClassD.\_\_mro\_\_)

**Output:**

**In ClassD**

**MRO of ClassD: (<class '\_\_main\_\_.ClassD'>, <class '\_\_main\_\_.ClassB'>, <class '\_\_main\_\_.ClassC'>, <class '\_\_main\_\_.ClassA'>, <class 'object'>)**

**3. Write a python code to read a phone number and email-id from the user and validate it for correctness.**

**Program:**

import re

# Improved regex for validating email addresses

regex = re.compile(r'^[A-Za-z0-9]+([.-\_][A-Za-z0-9]+)\*@[A-Za-z0-9-]+\.[A-Za-z]{2,}$')

def isValid(email):

# Match the email with the regex

if re.fullmatch(regex, email):

print("Valid email")

else:

print("Invalid email")

# Test cases

isValid("name.surname@gmail.com")

isValid("anonymous123@yahoo.co.uk")

isValid("anonymous123@...uk") # Invalid email

isValid("...@domain.us") # Invalid email

**Output:**

**Valid email**

**Invalid email**

**Invalid email**

**Invalid email**

import re

# Function to validate phone numbers

def validate\_phone\_number(regex, phone\_number):

match = re.search(regex, phone\_number)

if match:

return True

return False

# Define the regex pattern for phone number validation

pattern = re.compile(r"(\+\d{1,3})?\s?\(?\d{1,4}\)?[\s.-]?\d{3}[\s.-]?\d{4}")

# Test cases for phone numbers

test\_phone\_numbers = [

"+1 (555) 123-4567", # Valid

"555-123-4567", # Valid

"555 123 4567", # Valid

"+44 (0) 20 1234 5678", # Valid

"02012345678", # Valid

"25", # Invalid (Too short)

"invalid phone number" # Invalid (Invalid format)

]

# Check each phone number and print whether it's valid or not

for number in test\_phone\_numbers:

print(f"{number}: {validate\_phone\_number(pattern, number)}")

**Output:**

+1 (555) 123-4567: True

555-123-4567: True

555 123 4567: True

+44 (0) 20 1234 5678: True

02012345678: True

25: False

invalid phone number: False

**Week7**

**1. Write a Python code to merge two given file contents into a third file.**

**Program:**

with open("file1.txt", "r") as fh1:

with open("file2.txt", "r") as fh2:

with open("mergefile.txt", "w") as fh3:

# Read all lines from both files and combine them into a list 'q'

q = fh1.readlines() + fh2.readlines()

# Write the combined lines into the output file 'mergefile.txt'

fh3.writelines(q)

1. **Write a Python code to open a given file and construct a function to check for given words present in it and display on found**.

**Program:**

# Open the file in read mode

with open('file1.txt') as file:

# Read the contents of the file

contents = file.read()

# Prompt the user to input a word to search for

search\_word = input("Enter a word you want to search in the file: ")

# Check if the word is present in the file contents

if search\_word in contents:

print('Word found')

else:

print('Word not found')

**Output:**

Enter a word you want to search in the file: Anvi

Word found

**3. Write a Python code to Read text from a text file, find the word with most number of occurrences**

**Program:**

**from collections import Counter**

**import string**

**def find\_most\_frequent\_word(file\_name):**

**# Open and read the file**

**with open(file\_name, 'r') as file:**

**content = file.read()**

**# Remove punctuation and convert text to lowercase for consistent comparison**

**content = content.translate(str.maketrans('', '', string.punctuation)).lower()**

**# Split the content into words**

**words = content.split()**

**# Use Counter to count occurrences of each word**

**word\_counts = Counter(words)**

**# Find the word with the highest frequency**

**most\_common\_word, most\_common\_count = word\_counts.most\_common(1)[0]**

**print(f"The word with the most occurrences is '{most\_common\_word}' with {most\_common\_count} occurrences.")**

**# Example usage:**

**file\_name = input("Enter the file name (e.g., 'file1.txt'): ")**

**find\_most\_frequent\_word(file\_name)**

**Output:**

Enter the file name (e.g., 'file1.txt'): file1.txt

The word with the most occurrences is 'this' with 4 occurrences.

1. **Write a function that reads a file file1 and displays the number of words, number of vowels, blank spaces, lower case letters and uppercase letters.**

**Program:**

**def count\_file\_details(file\_name):**

**# Open the file in read mode**

**try:**

**with open(file\_name, 'r') as file:**

**content = file.read()**

**# Initialize counters**

**word\_count = 0**

**vowel\_count = 0**

**space\_count = 0**

**lowercase\_count = 0**

**uppercase\_count = 0**

**vowels = "aeiouAEIOU"**

**# Iterate through each character in the content**

**for char in content:**

**# Check for vowels**

**if char in vowels:**

**vowel\_count += 1**

**# Check for blank spaces**

**if char == ' ':**

**space\_count += 1**

**# Check for lowercase letters**

**if char.islower():**

**lowercase\_count += 1**

**# Check for uppercase letters**

**if char.isupper():**

**uppercase\_count += 1**

**# Split the content into words and count them**

**words = content.split()**

**word\_count = len(words)**

**# Print the results**

**print(f"Number of words: {word\_count}")**

**print(f"Number of vowels: {vowel\_count}")**

**print(f"Number of blank spaces: {space\_count}")**

**print(f"Number of lowercase letters: {lowercase\_count}")**

**print(f"Number of uppercase letters: {uppercase\_count}")**

**except FileNotFoundError:**

**print(f"The file '{file\_name}' was not found. Please check the file name and try again.")**

**# Example usage:**

**file\_name = input("Enter the file name (e.g., 'file1.txt'): ")**

**count\_file\_details(file\_name)**

**Output:**

Sample Output: Number of Uppercase characters are: 10

Number of Lowercase characters are: 250

Number of blankspaces: 0

Number of Vowels characters are: 98

Number of Consonants characters are: 10