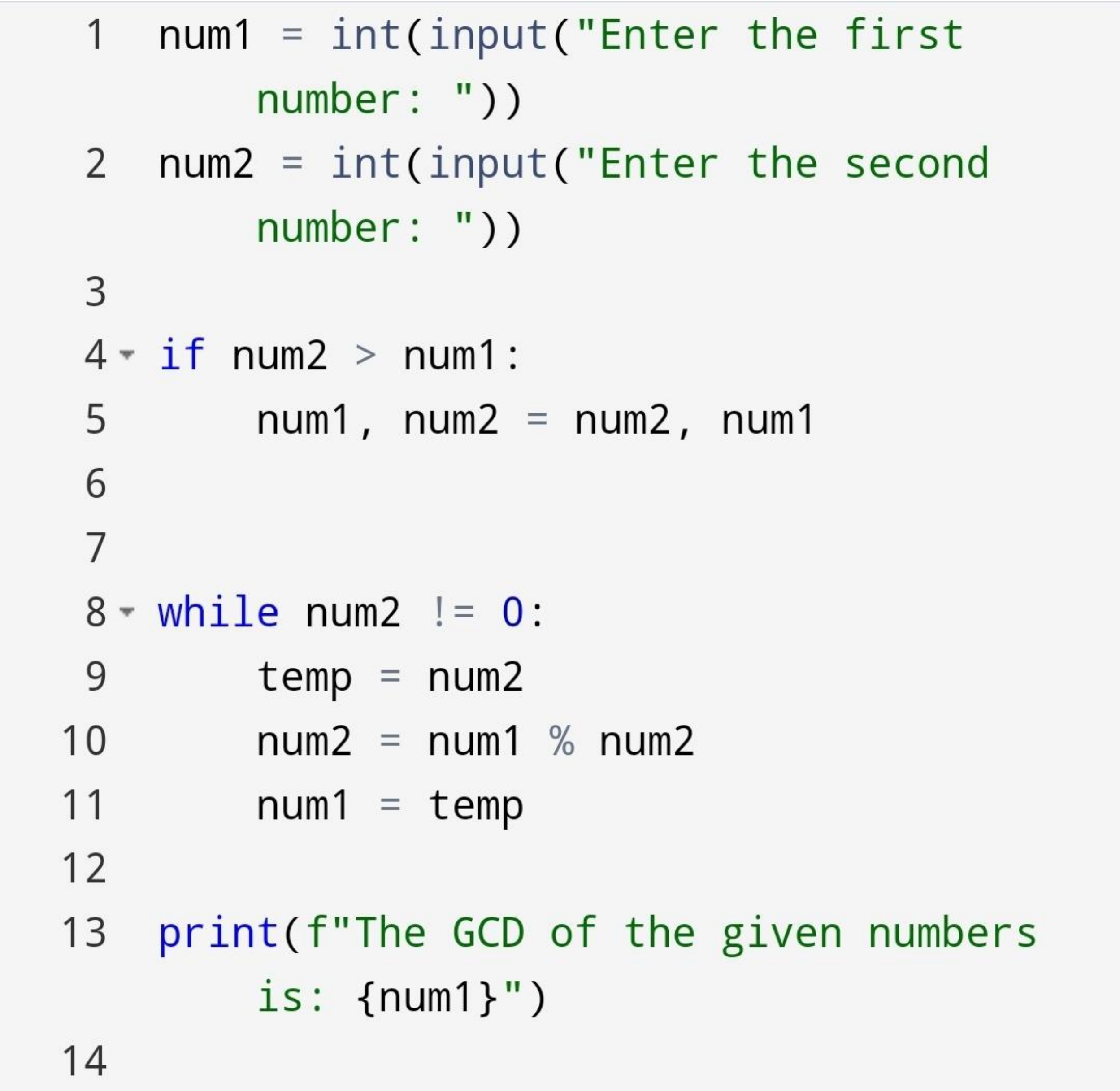
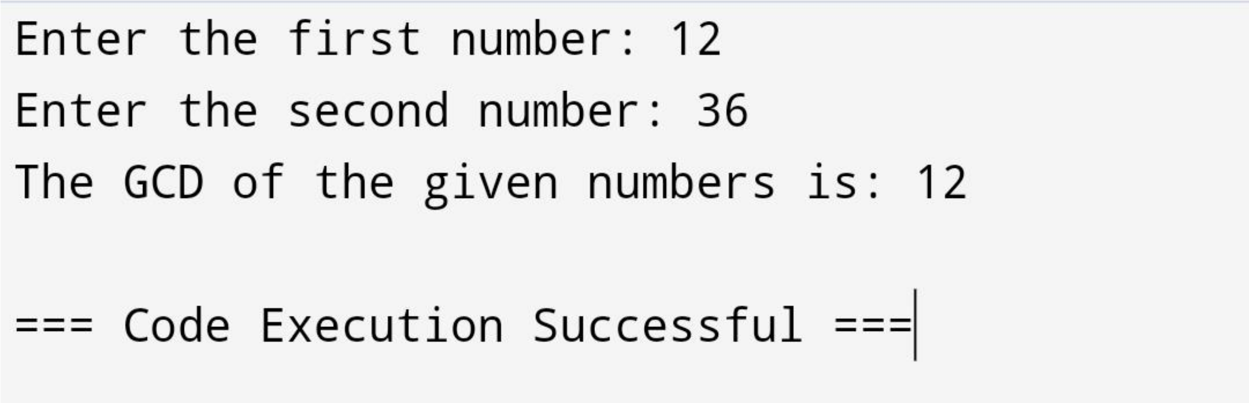
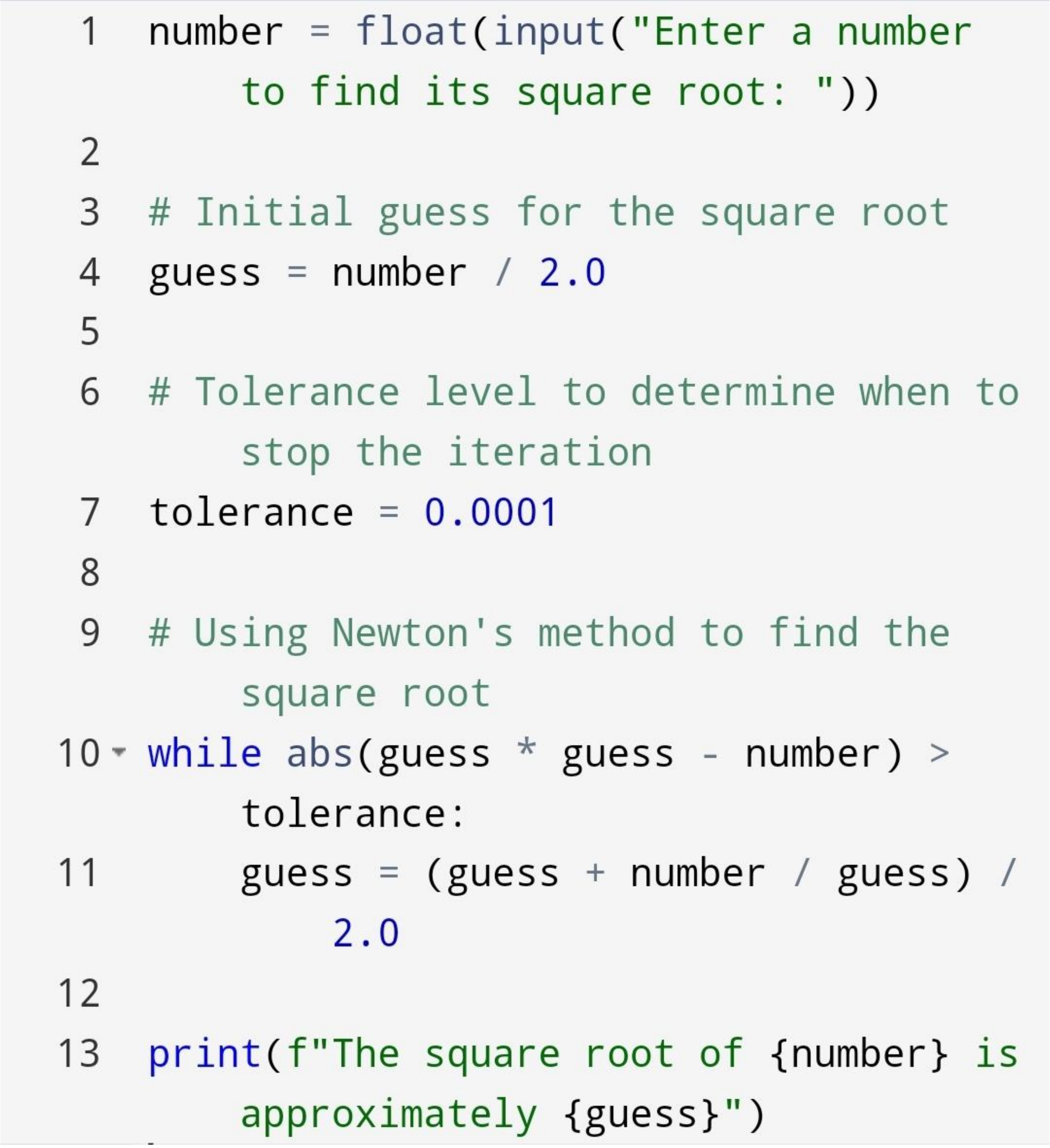


```python
num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the second number: "))

if num2 > num1:
    num1, num2 = num2, num1


while num2 != 0:
    temp = num2
    num2 = num1 % num2
    num1 = temp

print(f"The GCD of the given numbers is: {num1}")

```

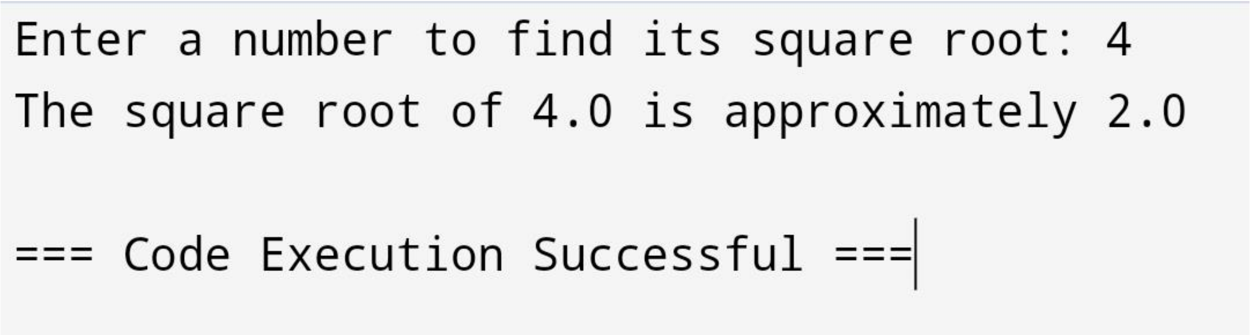


```
Enter the first number: 12
Enter the second number: 36
The GCD of the given numbers is: 12

=== Code Execution Successful ===
```

Vaibhav Gupta 0905CS221218

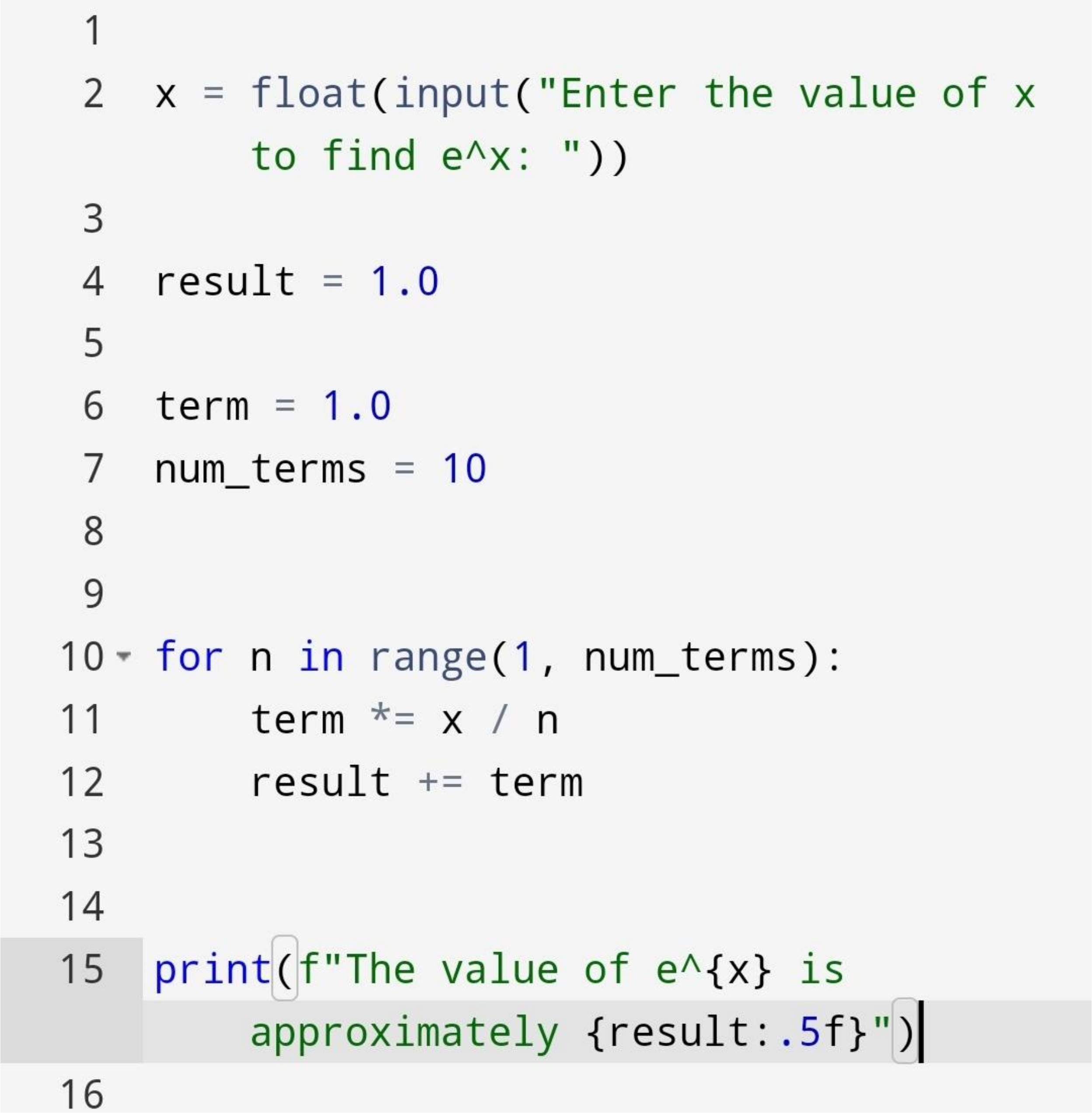


```python
number = float(input("Enter a number to find its square root: "))

# Initial guess for the square root
guess = number / 2.0

# Tolerance level to determine when to stop the iteration
tolerance = 0.0001

# Using Newton's method to find the square root
while abs(guess * guess - number) > tolerance:
    guess = (guess + number / guess) / 2.0

print(f"The square root of {number} is approximately {guess}")
```

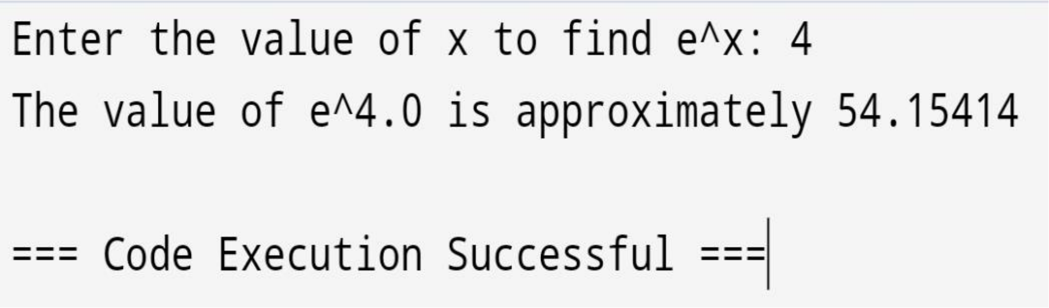


```
Enter a number to find its square root: 4
The square root of 4.0 is approximately 2.0

=== Code Execution Successful ===
```

Vaibhav Gupta 0905CS221218

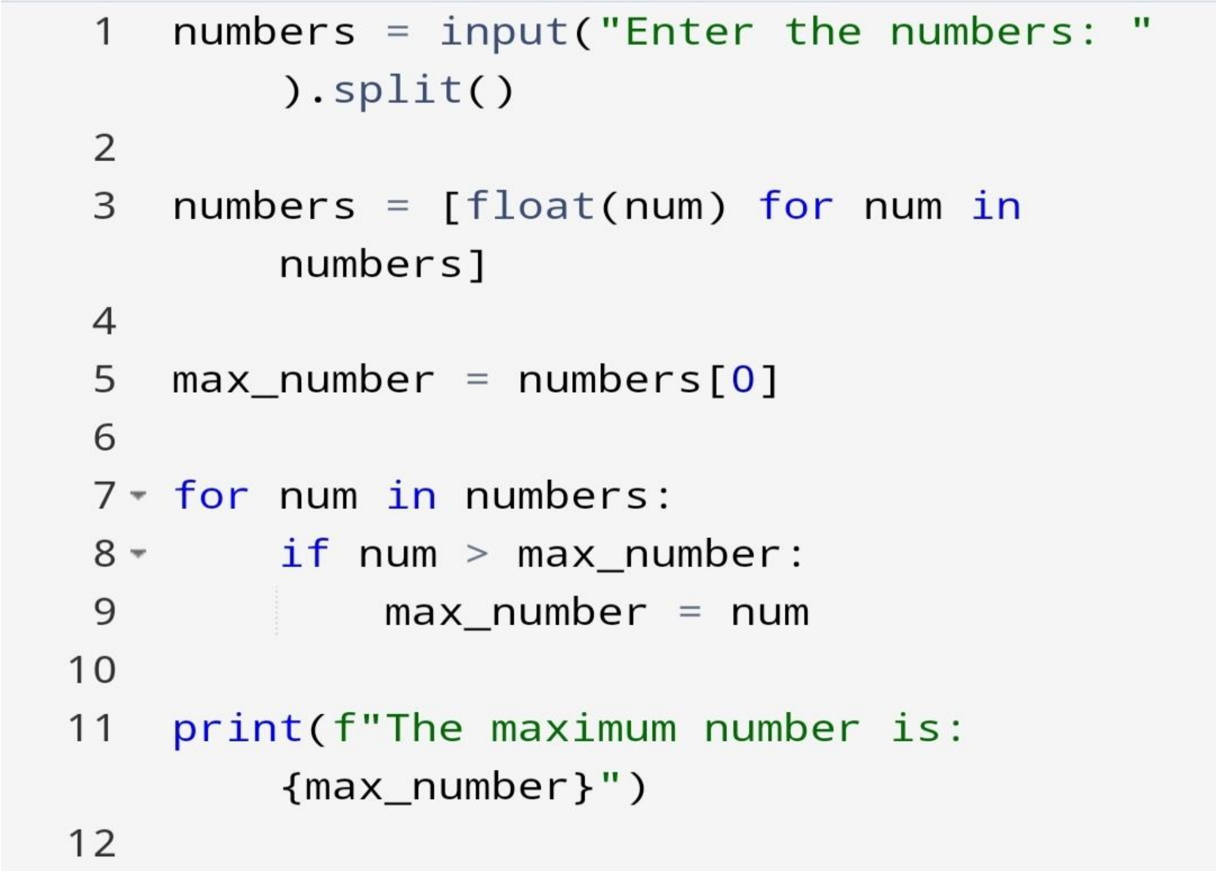


```python

x = float(input("Enter the value of x to find e^x: "))

result = 1.0

term = 1.0
num_terms = 10


for n in range(1, num_terms):
    term *= x / n
    result += term


print(f"The value of e^{x} is approximately {result:.5f}")

```

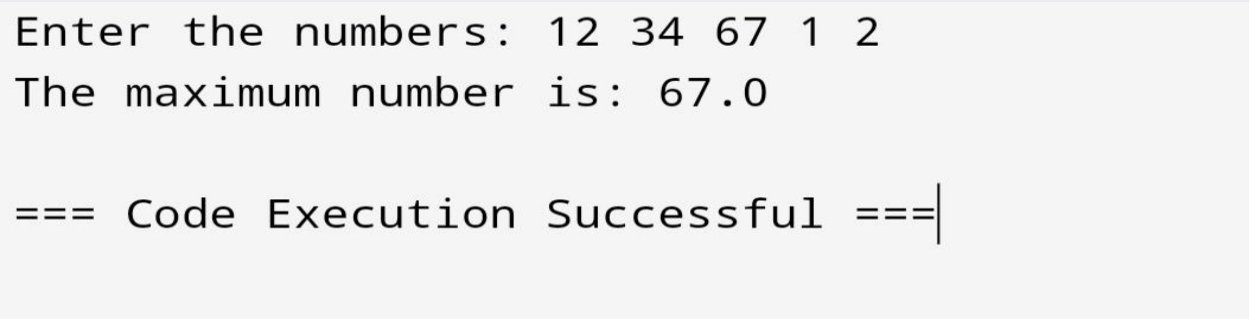


```
Enter the value of x to find e^x: 4
The value of e^4.0 is approximately 54.15414

=== Code Execution Successful ===
```

Vaibhav Gupta 0905CS221218

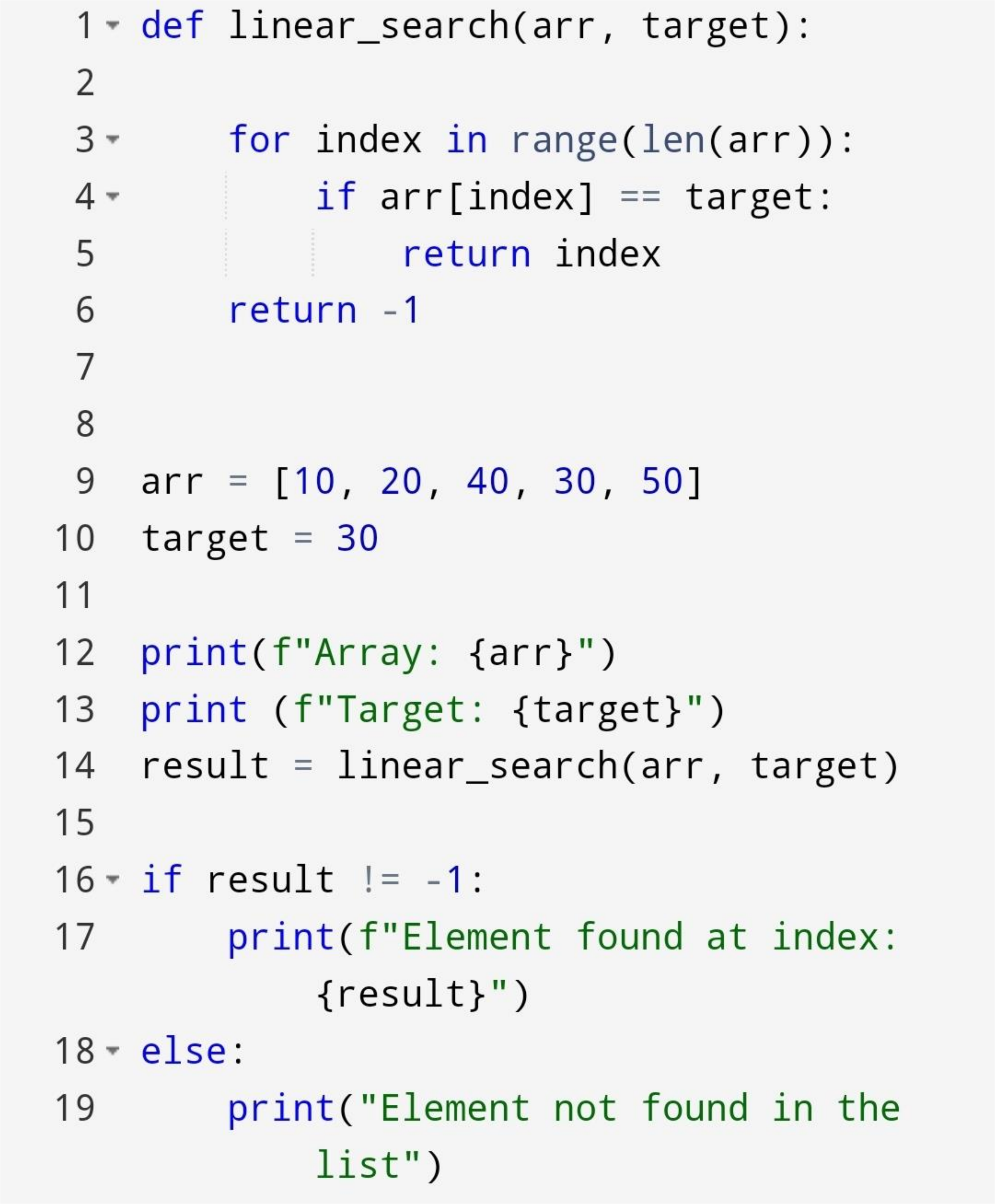


```python
numbers = input("Enter the numbers: ").split()

numbers = [float(num) for num in numbers]

max_number = numbers[0]

for num in numbers:
    if num > max_number:
        max_number = num

print(f"The maximum number is: {max_number}")

```

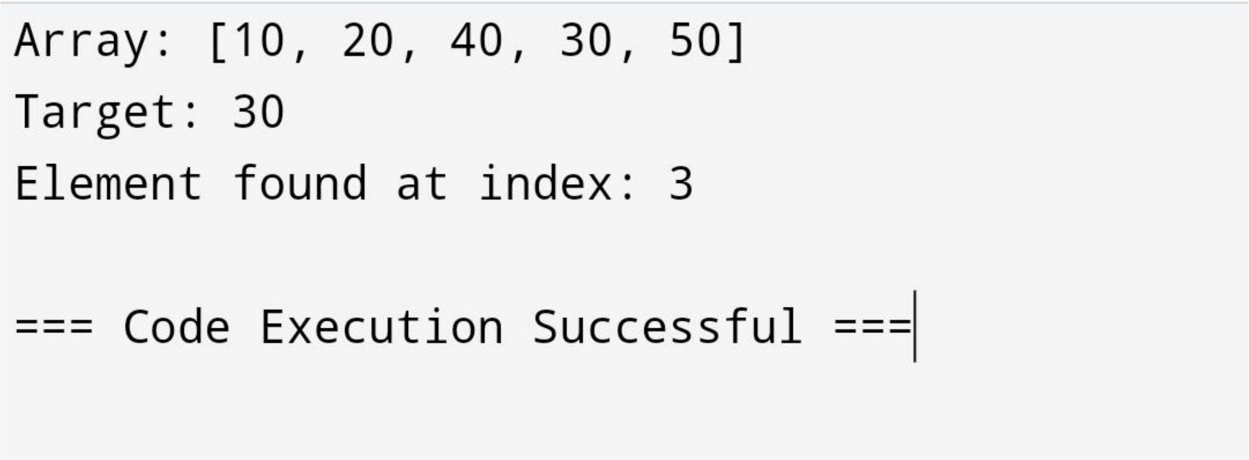


```
Enter the numbers: 12 34 67 1 2
The maximum number is: 67.0

=== Code Execution Successful ===
```

Vaibhav Gupta 0905CS221218

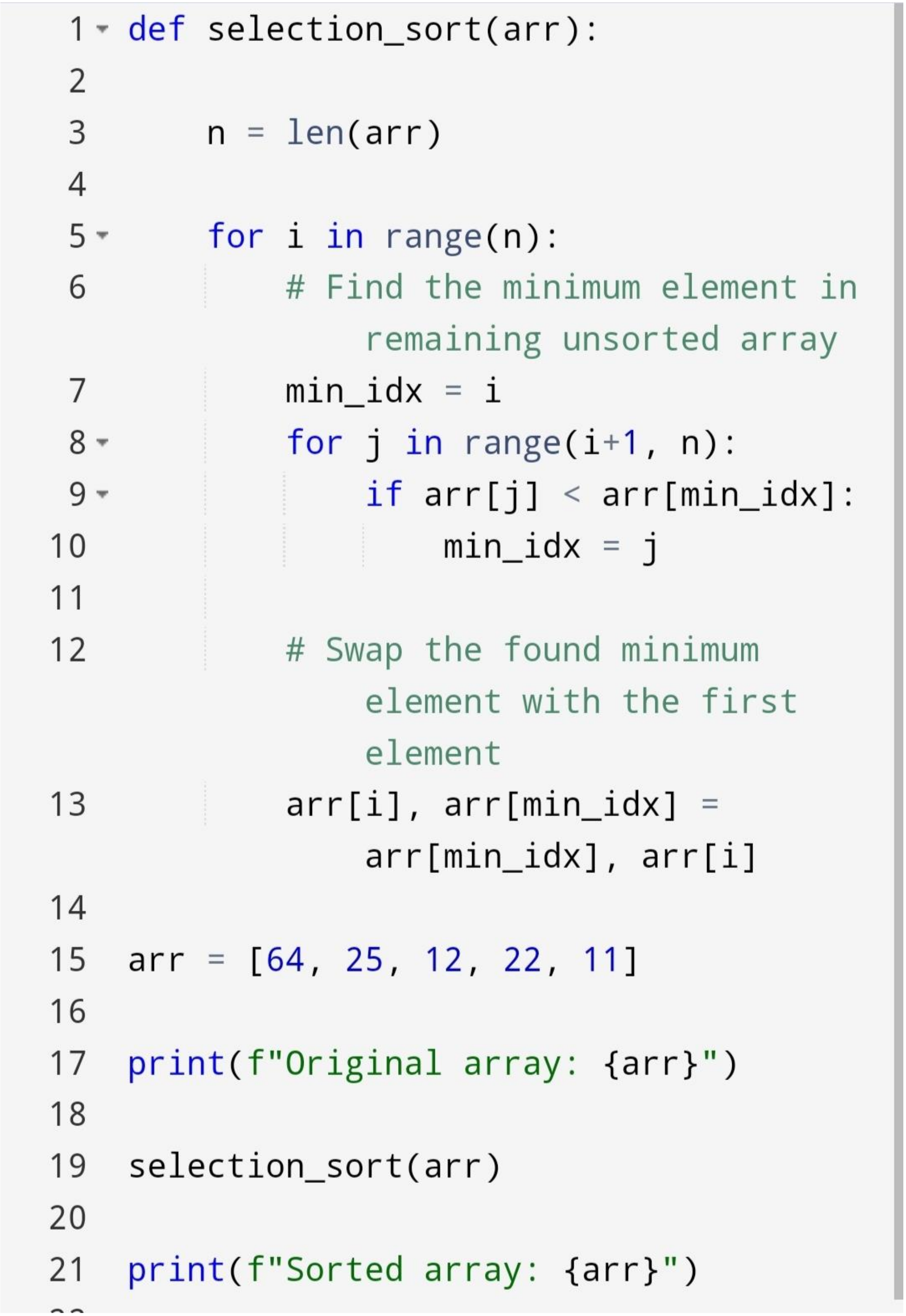


```python
def linear_search(arr, target):

    for index in range(len(arr)):
        if arr[index] == target:
            return index
    return -1


arr = [10, 20, 40, 30, 50]
target = 30

print(f"Array: {arr}")
print (f"Target: {target}")
result = linear_search(arr, target)

if result != -1:
    print(f"Element found at index: {result}")
else:
    print("Element not found in the list")
```

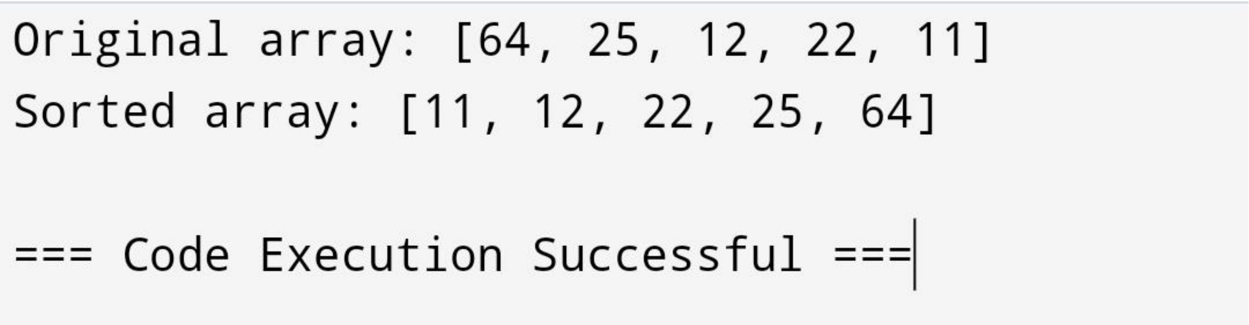


```
Array: [10, 20, 40, 30, 50]
Target: 30
Element found at index: 3

=== Code Execution Successful ===
```

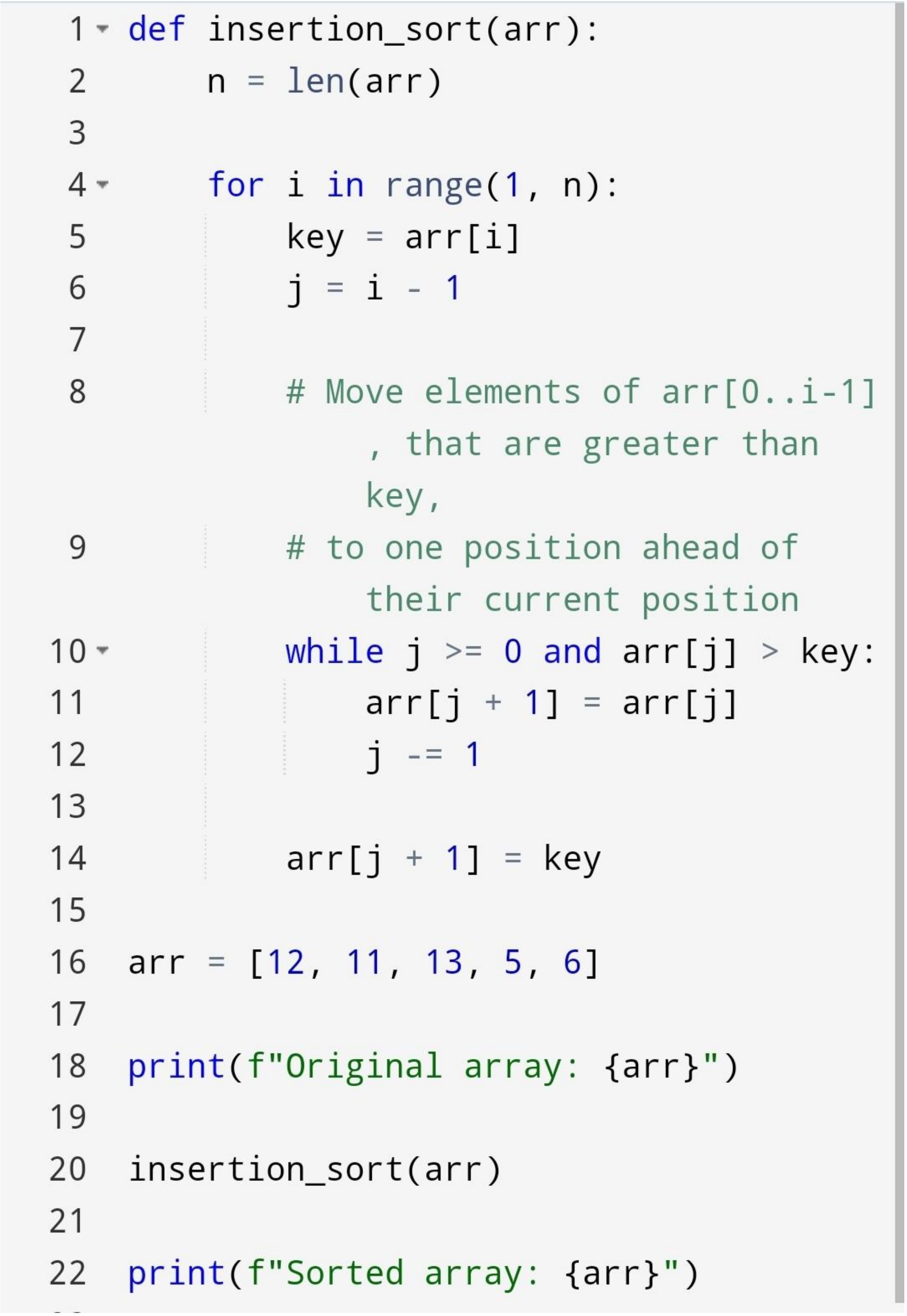
Vaibhav Gupta 0905CS221218



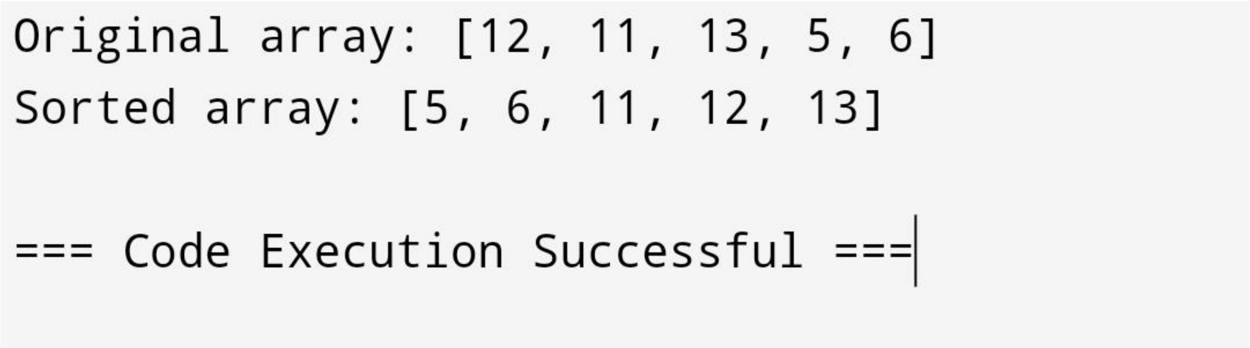
```python
def selection_sort(arr):

    n = len(arr)

    for i in range(n):
        # Find the minimum element in remaining unsorted array
        min_idx = i
        for j in range(i+1, n):
            if arr[j] < arr[min_idx]:
                min_idx = j

        # Swap the found minimum element with the first element
        arr[i], arr[min_idx] = arr[min_idx], arr[i]

arr = [64, 25, 12, 22, 11]

print(f"Original array: {arr}")

selection_sort(arr)

print(f"Sorted array: {arr}")
```



```
Original array: [64, 25, 12, 22, 11]
Sorted array: [11, 12, 22, 25, 64]

=== Code Execution Successful ===
```

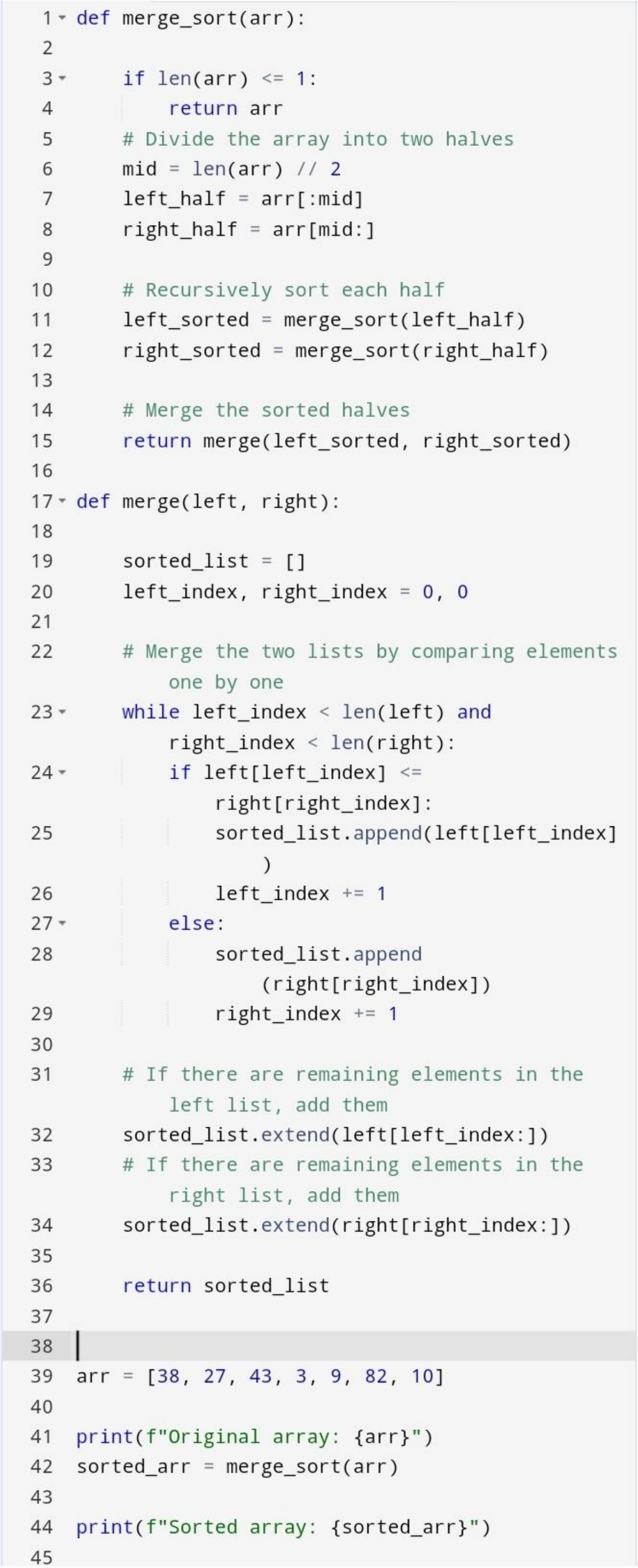


```python
def insertion_sort(arr):
    n = len(arr)

    for i in range(1, n):
        key = arr[i]
        j = i - 1

        # Move elements of arr[0..i-1], that are greater than key,
        # to one position ahead of their current position
        while j >= 0 and arr[j] > key:
            arr[j + 1] = arr[j]
            j -= 1

        arr[j + 1] = key

arr = [12, 11, 13, 5, 6]

print(f"Original array: {arr}")

insertion_sort(arr)

print(f"Sorted array: {arr}")
```

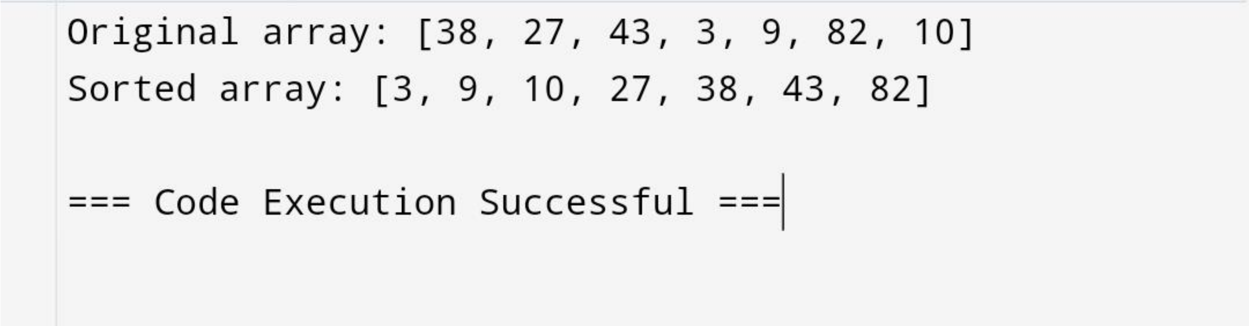


```
Original array: [12, 11, 13, 5, 6]
Sorted array: [5, 6, 11, 12, 13]

=== Code Execution Successful ===
```

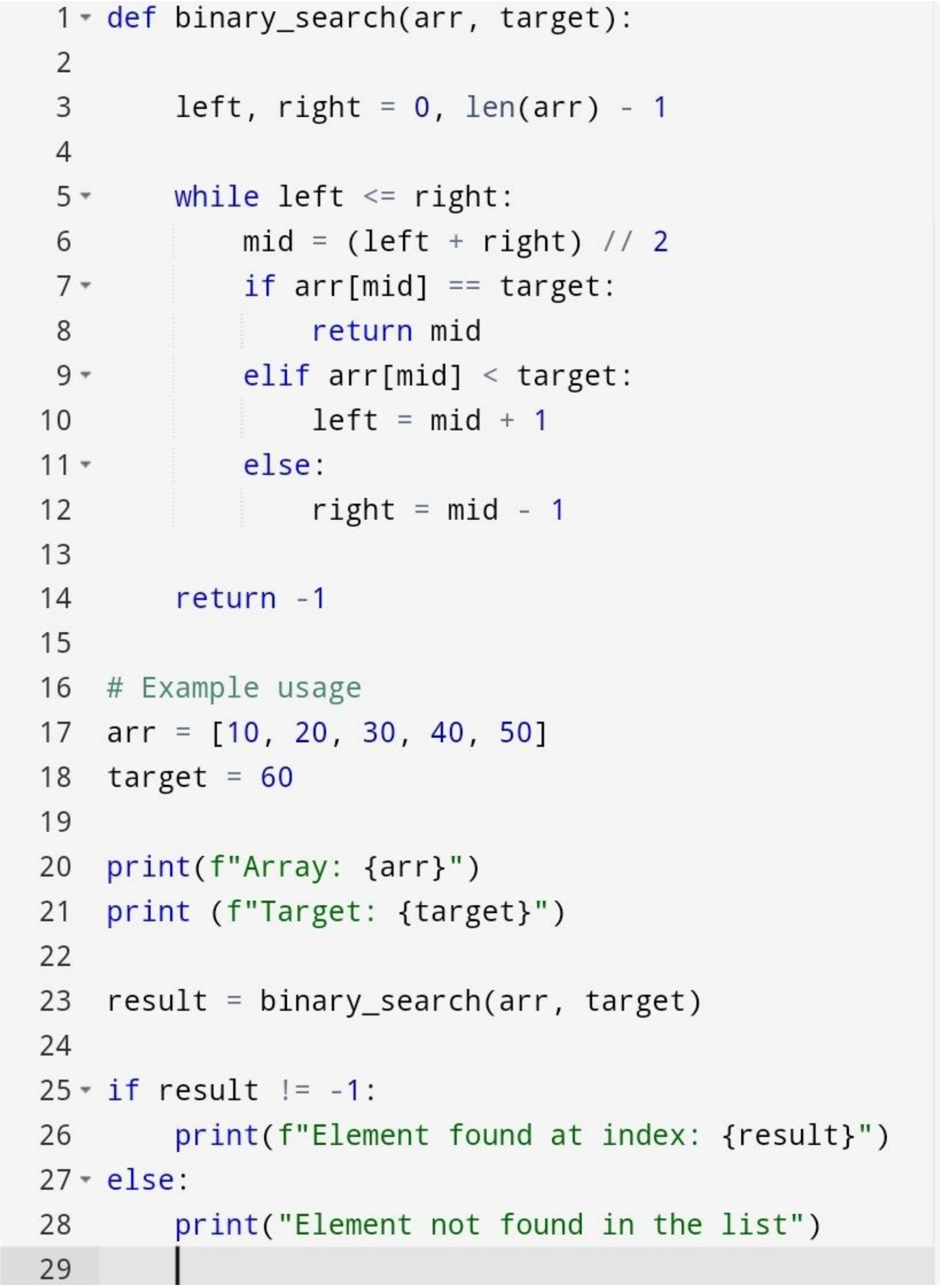
Vaibhav Gupta 0905CS221218



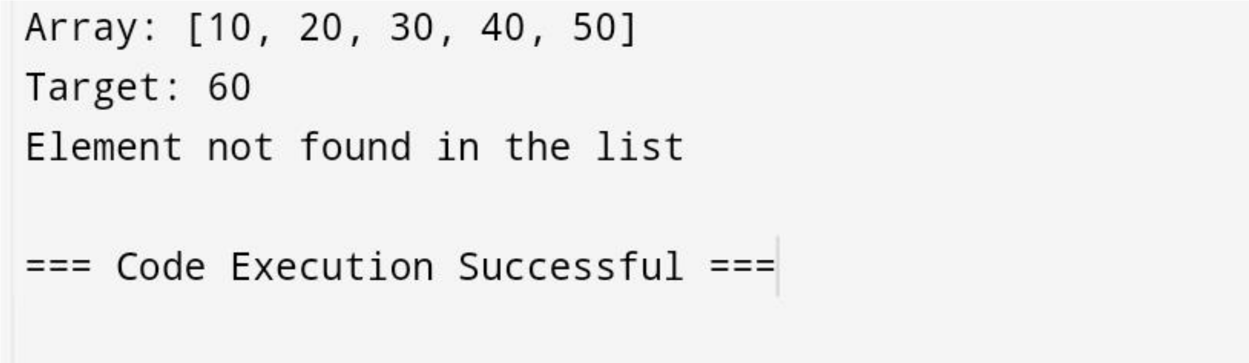
```python
def merge_sort(arr):

    if len(arr) <= 1:
        return arr
    # Divide the array into two halves
    mid = len(arr) // 2
    left_half = arr[:mid]
    right_half = arr[mid:]

    # Recursively sort each half
    left_sorted = merge_sort(left_half)
    right_sorted = merge_sort(right_half)

    # Merge the sorted halves
    return merge(left_sorted, right_sorted)

def merge(left, right):

    sorted_list = []
    left_index, right_index = 0, 0

    # Merge the two lists by comparing elements one by one
    while left_index < len(left) and right_index < len(right):
        if left[left_index] <= right[right_index]:
            sorted_list.append(left[left_index])
            left_index += 1
        else:
            sorted_list.append(right[right_index])
            right_index += 1

    # If there are remaining elements in the left list, add them
    sorted_list.extend(left[left_index:])
    # If there are remaining elements in the right list, add them
    sorted_list.extend(right[right_index:])

    return sorted_list


arr = [38, 27, 43, 3, 9, 82, 10]

print(f"Original array: {arr}")
sorted_arr = merge_sort(arr)

print(f"Sorted array: {sorted_arr}")

```



```
Original array: [38, 27, 43, 3, 9, 82, 10]
Sorted array: [3, 9, 10, 27, 38, 43, 82]

=== Code Execution Successful ===
```

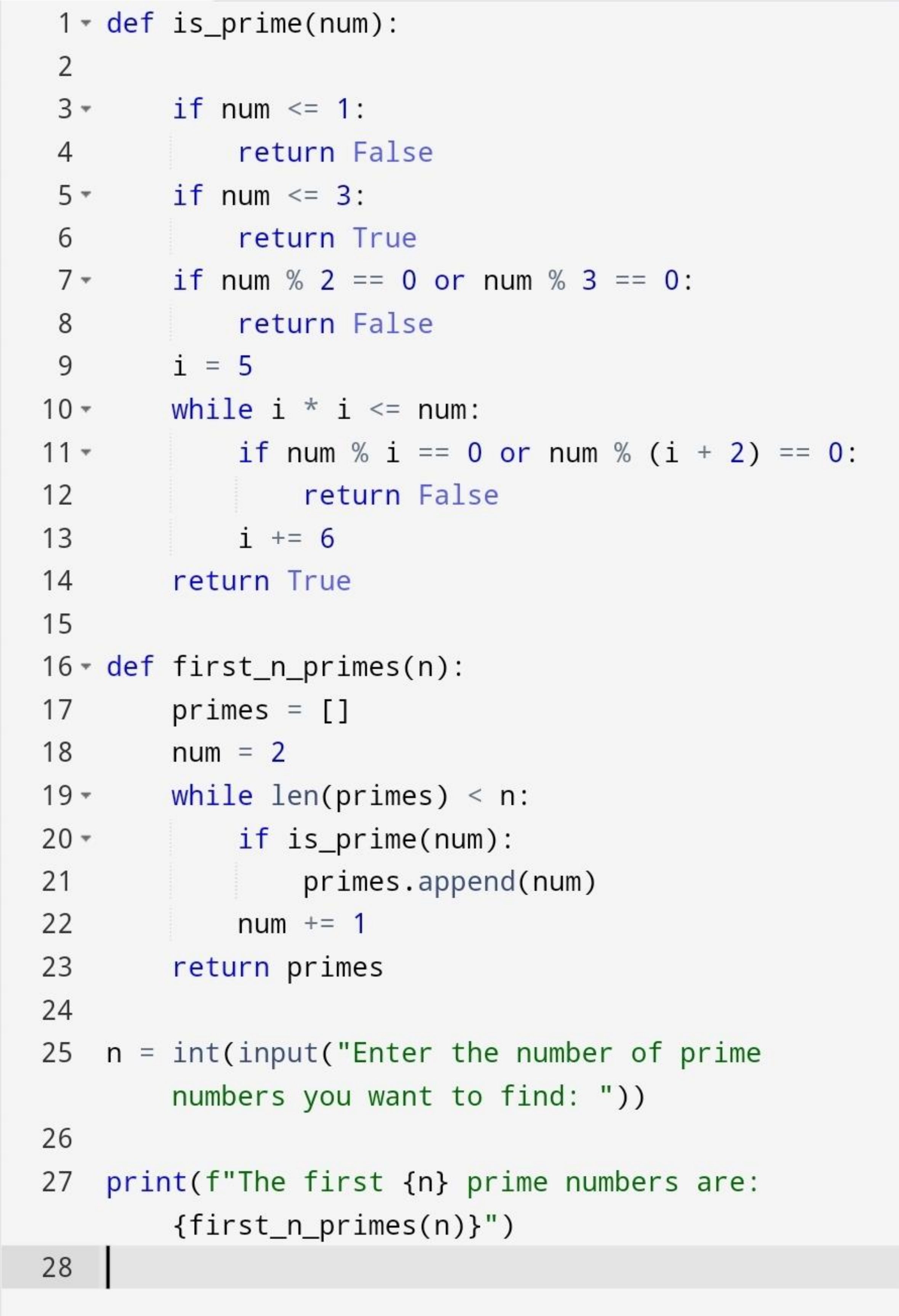


```python
def binary_search(arr, target):

    left, right = 0, len(arr) - 1

    while left <= right:
        mid = (left + right) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            left = mid + 1
        else:
            right = mid - 1

    return -1

# Example usage
arr = [10, 20, 30, 40, 50]
target = 60

print(f"Array: {arr}")
print (f"Target: {target}")

result = binary_search(arr, target)

if result != -1:
    print(f"Element found at index: {result}")
else:
    print("Element not found in the list")

```

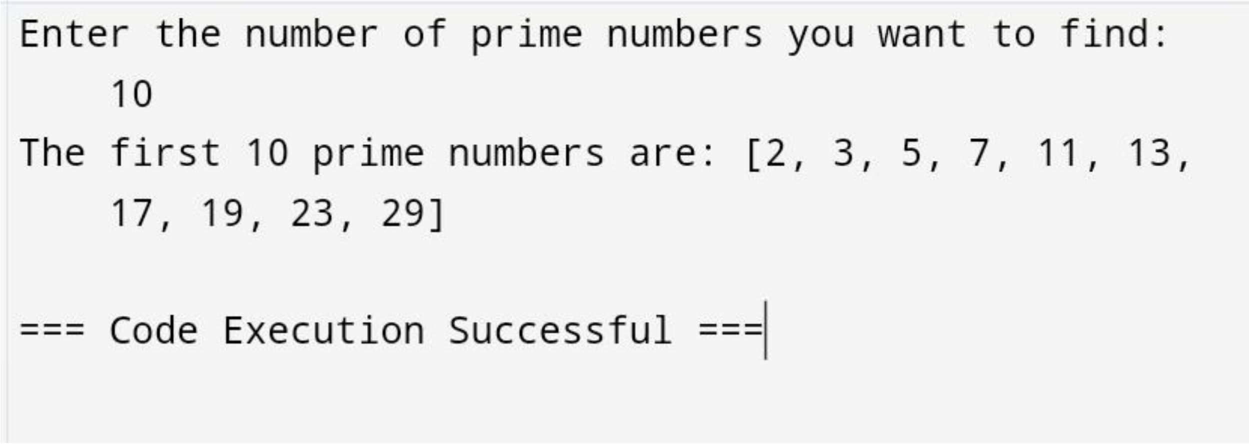


```
Array: [10, 20, 30, 40, 50]
Target: 60
Element not found in the list

=== Code Execution Successful ===
```

Vaibhav Gupta 0905CS221218

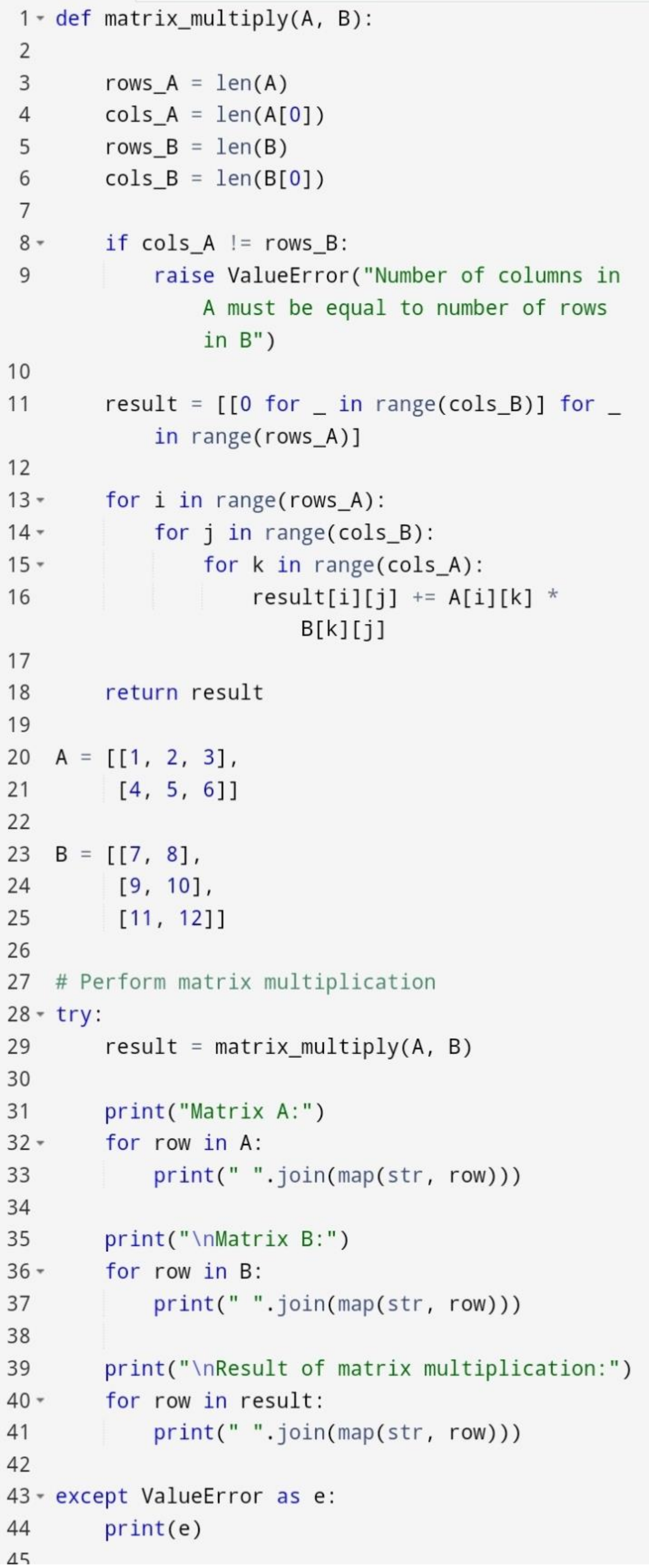


```python
def is_prime(num):

    if num <= 1:
        return False
    if num <= 3:
        return True
    if num % 2 == 0 or num % 3 == 0:
        return False
    i = 5
    while i * i <= num:
        if num % i == 0 or num % (i + 2) == 0:
            return False
        i += 6
    return True

def first_n_primes(n):
    primes = []
    num = 2
    while len(primes) < n:
        if is_prime(num):
            primes.append(num)
        num += 1
    return primes

n = int(input("Enter the number of prime numbers you want to find: "))

print(f"The first {n} prime numbers are: {first_n_primes(n)}")

```

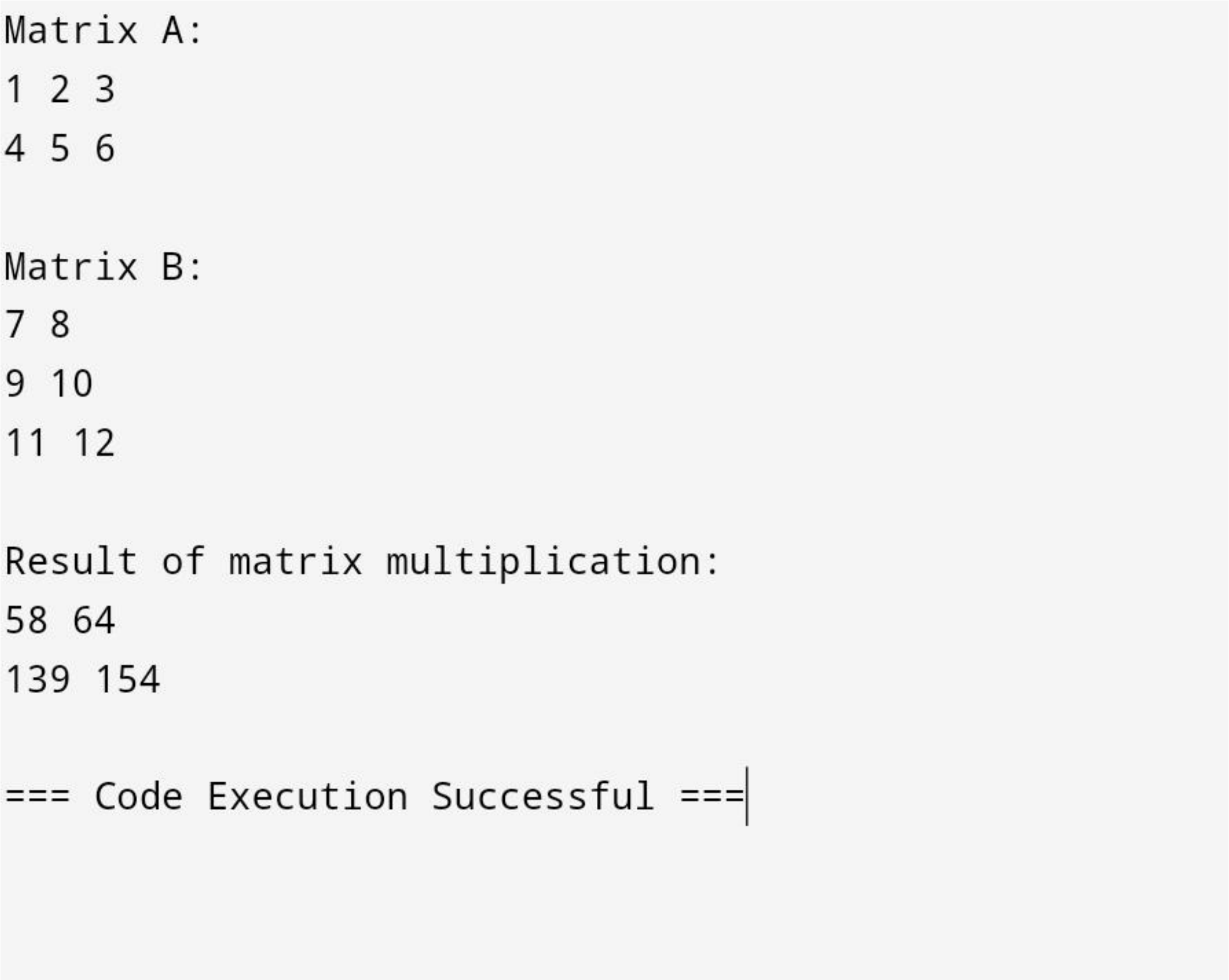


```
Enter the number of prime numbers you want to find:
    10
The first 10 prime numbers are: [2, 3, 5, 7, 11, 13,
    17, 19, 23, 29]

=== Code Execution Successful ===
```

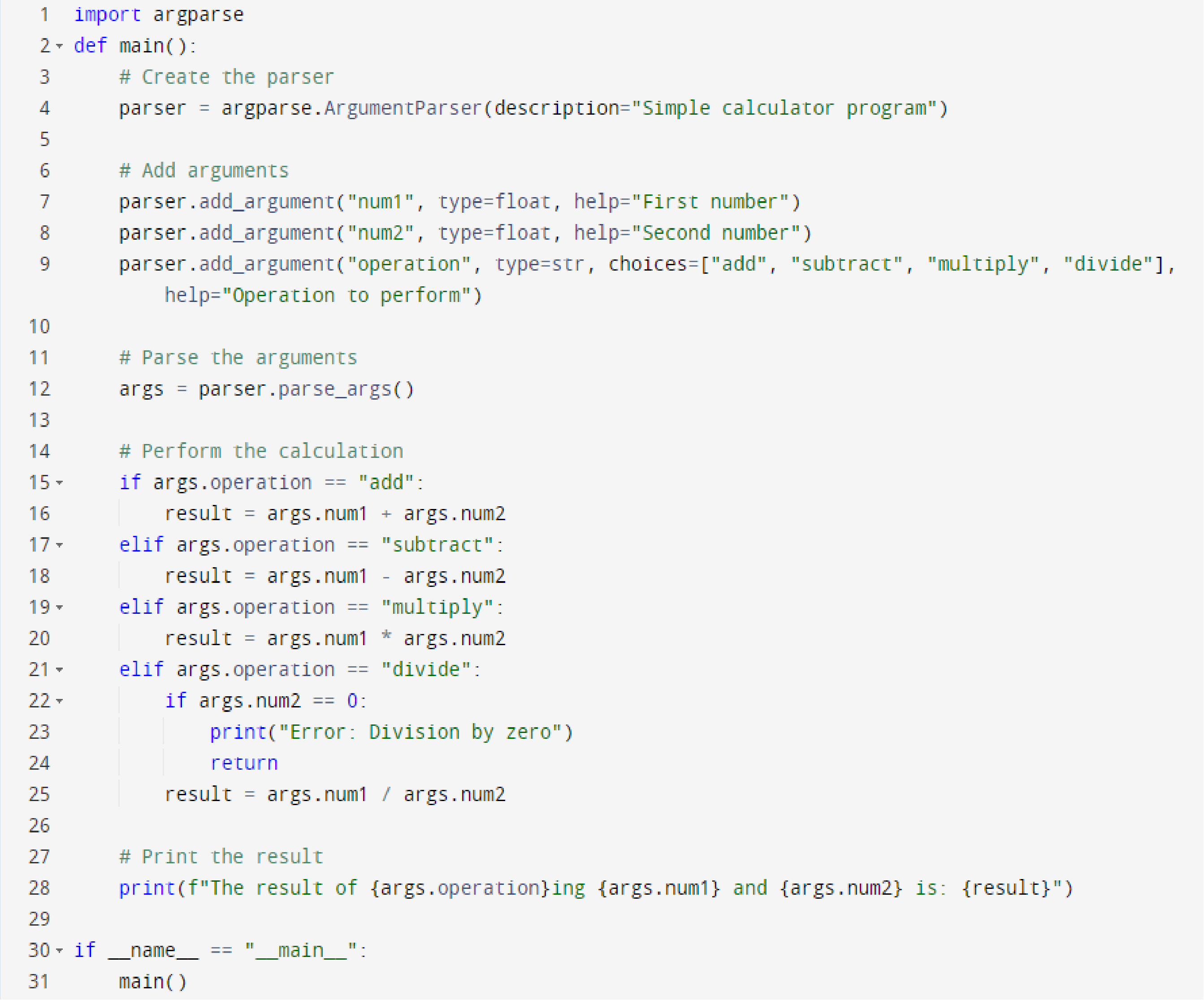
Vaibhav Gupta 0905CS221218



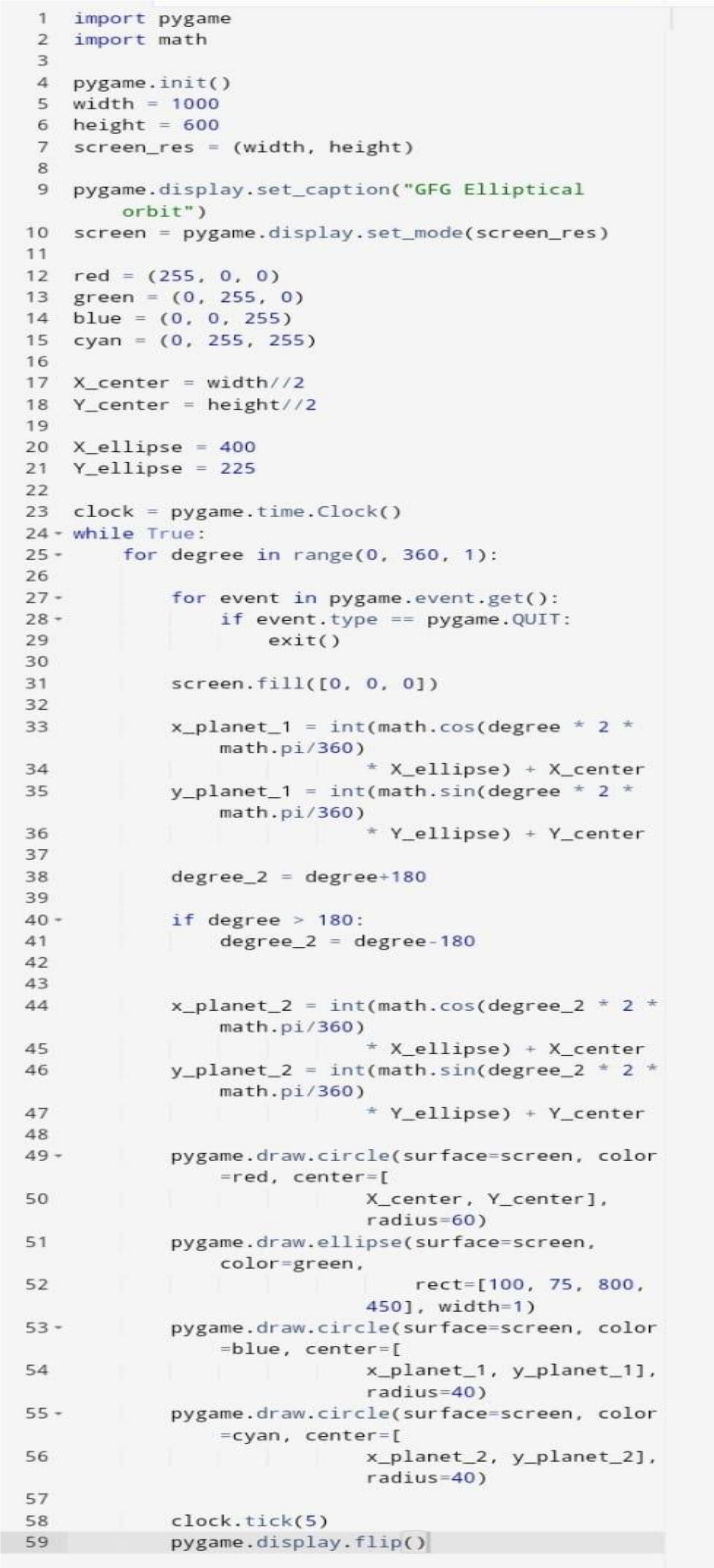
```python
def matrix_multiply(A, B):

    rows_A = len(A)
    cols_A = len(A[0])
    rows_B = len(B)
    cols_B = len(B[0])

    if cols_A != rows_B:
        raise ValueError("Number of columns in A must be equal to number of rows in B")

    result = [[0 for _ in range(cols_B)] for _ in range(rows_A)]

    for i in range(rows_A):
        for j in range(cols_B):
            for k in range(cols_A):
                result[i][j] += A[i][k] * B[k][j]

    return result

A = [[1, 2, 3],
     [4, 5, 6]]

B = [[7, 8],
     [9, 10],
     [11, 12]]

# Perform matrix multiplication
try:
    result = matrix_multiply(A, B)

    print("Matrix A:")
    for row in A:
        print(" ".join(map(str, row)))

    print("\nMatrix B:")
    for row in B:
        print(" ".join(map(str, row)))

    print("\nResult of matrix multiplication:")
    for row in result:
        print(" ".join(map(str, row)))

except ValueError as e:
    print(e)
```



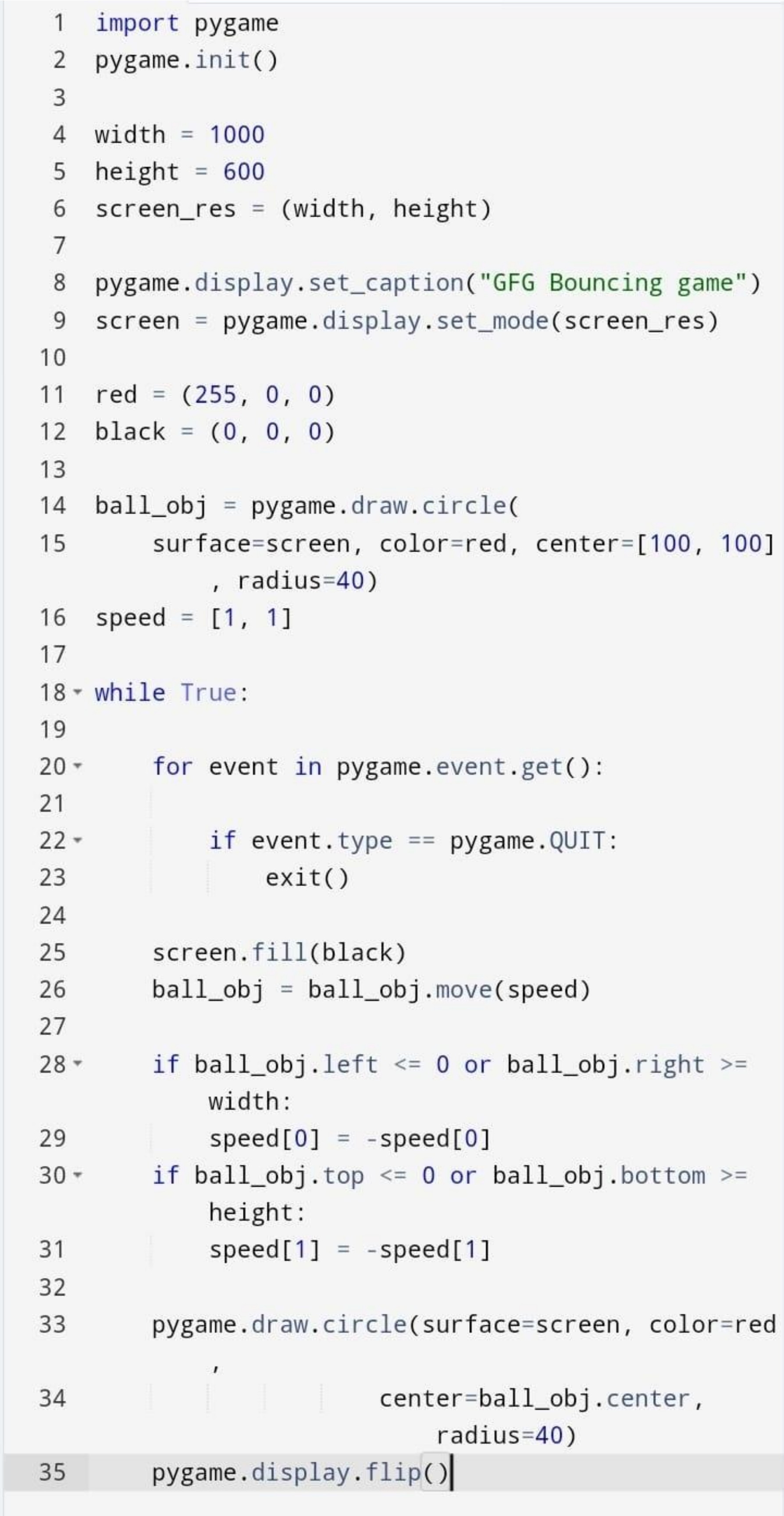
```
Matrix A:
1 2 3
4 5 6

Matrix B:
7 8
9 10
11 12

Result of matrix multiplication:
58 64
139 154

=== Code Execution Successful ===
```



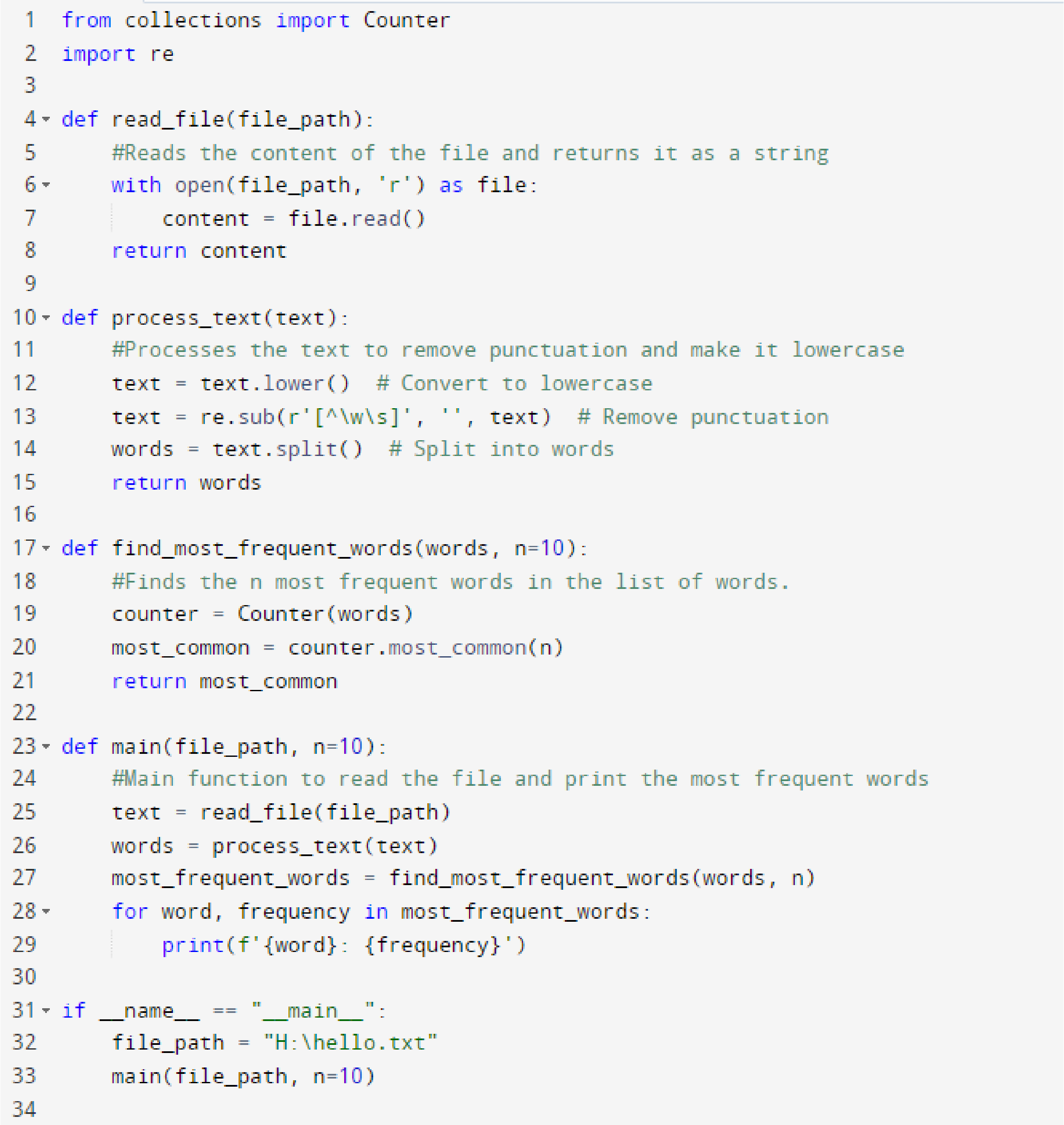
```python
import argparse
def main():
    # Create the parser
    parser = argparse.ArgumentParser(description="Simple calculator program")

    # Add arguments
    parser.add_argument("num1", type=float, help="First number")
    parser.add_argument("num2", type=float, help="Second number")
    parser.add_argument("operation", type=str, choices=["add", "subtract", "multiply", "divide"], help="Operation to perform")

    # Parse the arguments
    args = parser.parse_args()

    # Perform the calculation
    if args.operation == "add":
        result = args.num1 + args.num2
    elif args.operation == "subtract":
        result = args.num1 - args.num2
    elif args.operation == "multiply":
        result = args.num1 * args.num2
    elif args.operation == "divide":
        if args.num2 == 0:
            print("Error: Division by zero")
            return
        result = args.num1 / args.num2

    # Print the result
    print(f"The result of {args.operation}ing {args.num1} and {args.num2} is: {result}")

if __name__ == "__main__":
    main()
```



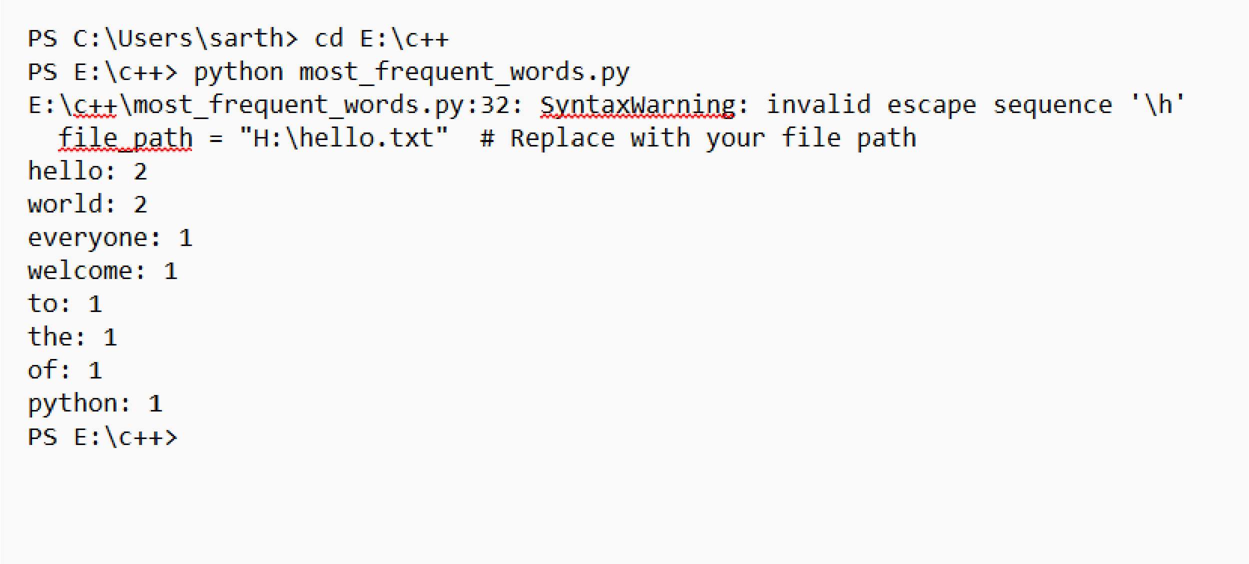
```
import pygame
import math

pygame.init()
width = 1000
height = 600
screen_res = (width, height)

pygame.display.set_caption("GFG Elliptical orbit")
screen = pygame.display.set_mode(screen_res)

red = (255, 0, 0)
green = (0, 255, 0)
blue = (0, 0, 255)
cyan = (0, 255, 255)

X_center = width//2
Y_center = height//2

X_ellipse = 400
Y_ellipse = 225

clock = pygame.time.Clock()
while True:
    for degree in range(0, 360, 1):

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                exit()

        screen.fill([0, 0, 0])

        x_planet_1 = int(math.cos(degree * 2 * math.pi/360)
                            * X_ellipse) + X_center
        y_planet_1 = int(math.sin(degree * 2 * math.pi/360)
                            * Y_ellipse) + Y_center

        degree_2 = degree+180

        if degree > 180:
            degree_2 = degree-180


        x_planet_2 = int(math.cos(degree_2 * 2 * math.pi/360)
                            * X_ellipse) + X_center
        y_planet_2 = int(math.sin(degree_2 * 2 * math.pi/360)
                            * Y_ellipse) + Y_center

        pygame.draw.circle(surface=screen, color=red, center=[
                            X_center, Y_center], radius=60)
        pygame.draw.ellipse(surface=screen, color=green,
                                rect=[100, 75, 800, 450], width=1)
        pygame.draw.circle(surface=screen, color=blue, center=[
                            x_planet_1, y_planet_1], radius=40)
        pygame.draw.circle(surface=screen, color=cyan, center=[
                            x_planet_2, y_planet_2], radius=40)

        clock.tick(5)
        pygame.display.flip()
```



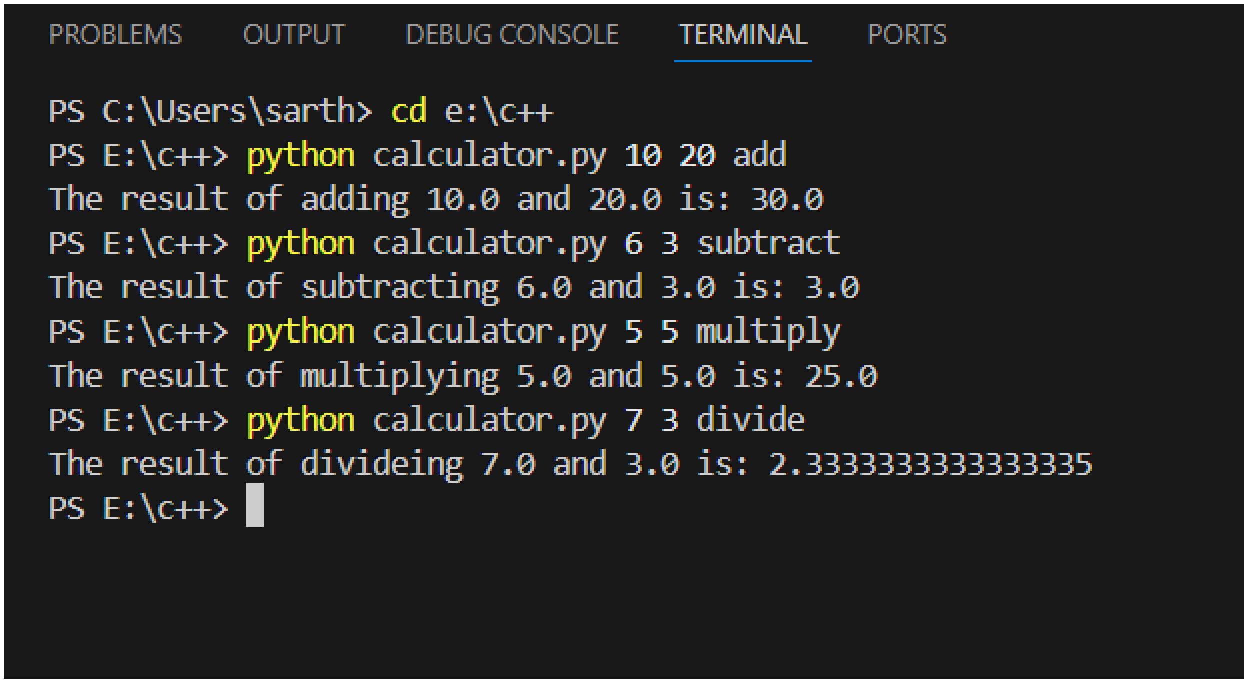
```python
import pygame
pygame.init()

width = 1000
height = 600
screen_res = (width, height)

pygame.display.set_caption("GFG Bouncing game")
screen = pygame.display.set_mode(screen_res)

red = (255, 0, 0)
black = (0, 0, 0)

ball_obj = pygame.draw.circle(
    surface=screen, color=red, center=[100, 100], radius=40)
speed = [1, 1]

while True:

    for event in pygame.event.get():

        if event.type == pygame.QUIT:
            exit()

    screen.fill(black)
    ball_obj = ball_obj.move(speed)

    if ball_obj.left <= 0 or ball_obj.right >= width:
        speed[0] = -speed[0]
    if ball_obj.top <= 0 or ball_obj.bottom >= height:
        speed[1] = -speed[1]

    pygame.draw.circle(surface=screen, color=red,
                       center=ball_obj.center, radius=40)
    pygame.display.flip()
```



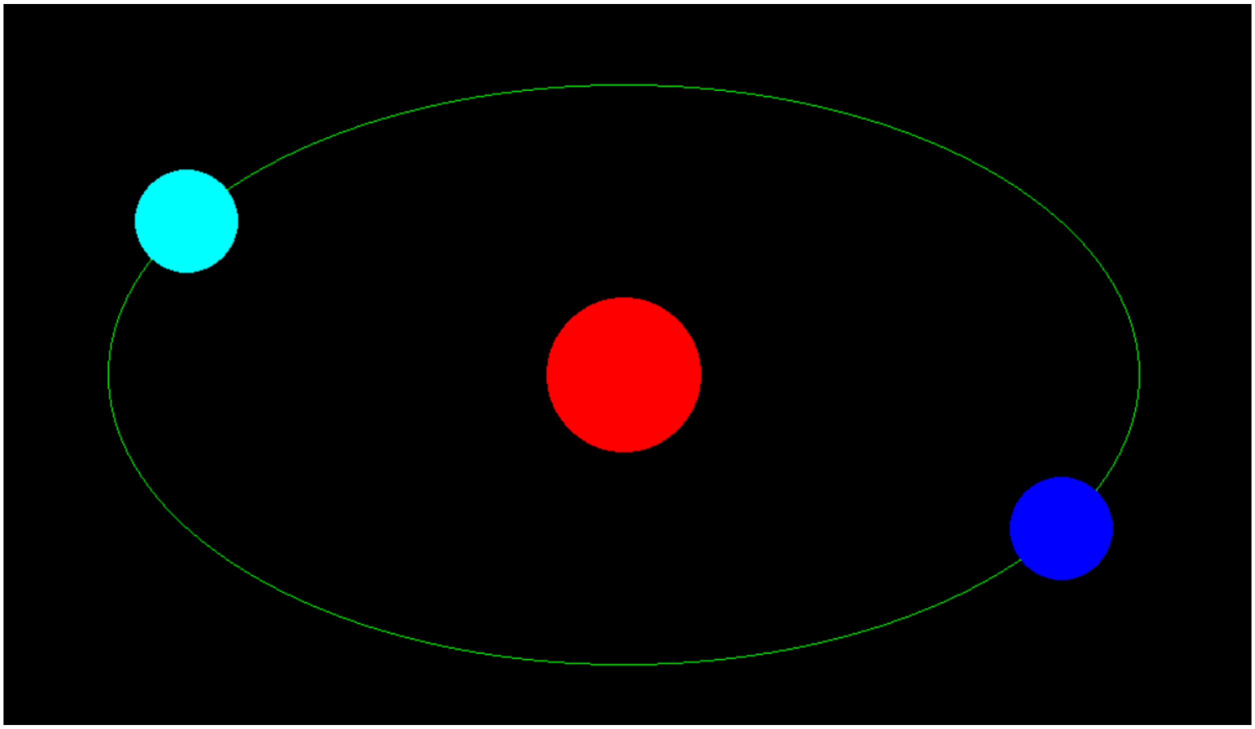
```python
from collections import Counter
import re

def read_file(file_path):
    #Reads the content of the file and returns it as a string
    with open(file_path, 'r') as file:
        content = file.read()
    return content

def process_text(text):
    #Processes the text to remove punctuation and make it lowercase
    text = text.lower()  # Convert to lowercase
    text = re.sub(r'[^\w\s]', '', text)  # Remove punctuation
    words = text.split()  # Split into words
    return words

def find_most_frequent_words(words, n=10):
    #Finds the n most frequent words in the list of words.
    counter = Counter(words)
    most_common = counter.most_common(n)
    return most_common

def main(file_path, n=10):
    #Main function to read the file and print the most frequent words
    text = read_file(file_path)
    words = process_text(text)
    most_frequent_words = find_most_frequent_words(words, n)
    for word, frequency in most_frequent_words:
        print(f'{word}: {frequency}')

if __name__ == "__main__":
    file_path = "H:\hello.txt"
    main(file_path, n=10)

```



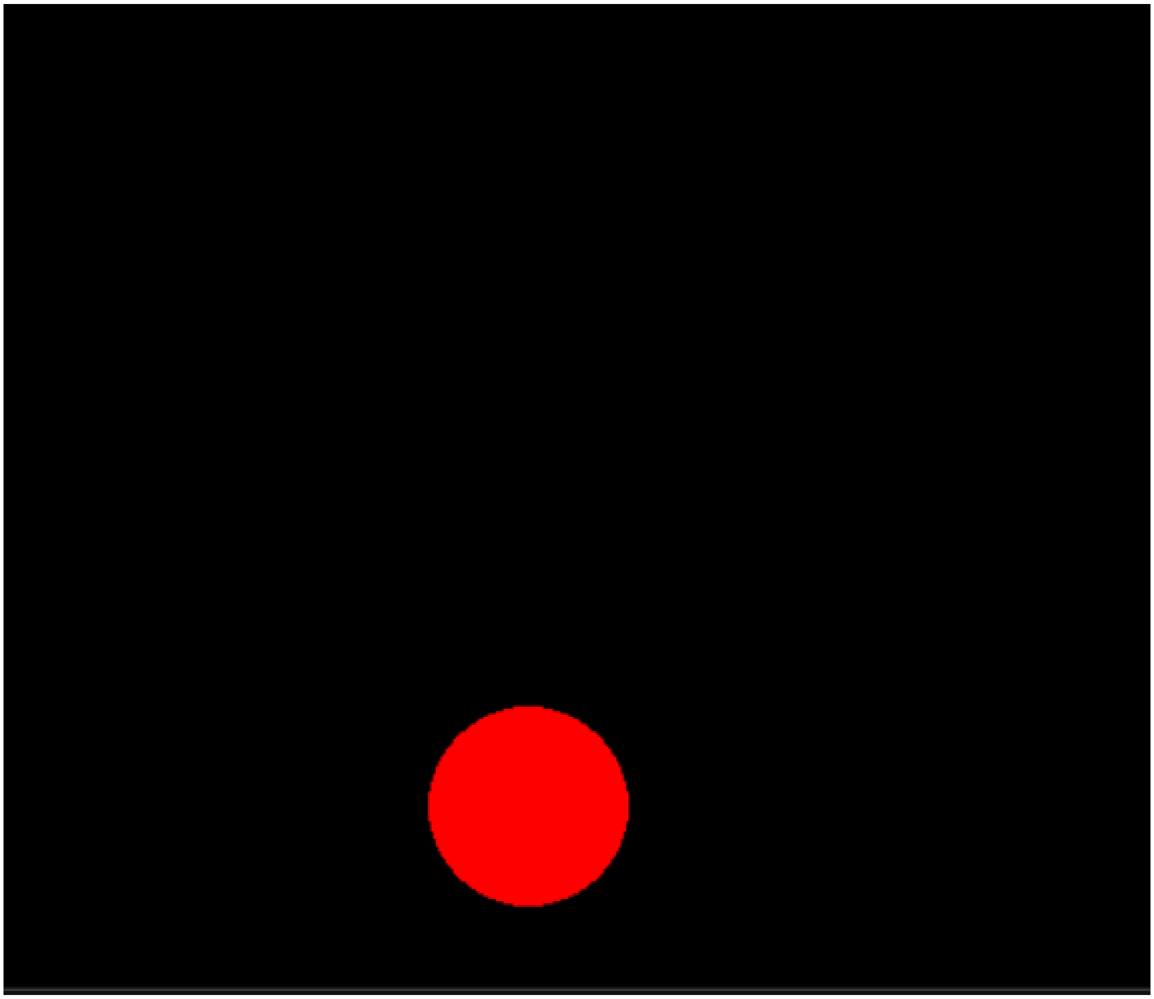
```
PS C:\Users\sarth> cd E:\c++
PS E:\c++> python most_frequent_words.py
E:\c++\most_frequent_words.py:32: SyntaxWarning: invalid escape sequence '\h'
  file_path = "H:\hello.txt"  # Replace with your file path
hello: 2
world: 2
everyone: 1
welcome: 1
to: 1
the: 1
of: 1
python: 1
PS E:\c++>
```



```
PS C:\Users\sarth> cd e:\c++
PS E:\c++> python calculator.py 10 20 add
The result of adding 10.0 and 20.0 is: 30.0
PS E:\c++> python calculator.py 6 3 subtract
The result of subtracting 6.0 and 3.0 is: 3.0
PS E:\c++> python calculator.py 5 5 multiply
The result of multiplying 5.0 and 5.0 is: 25.0
PS E:\c++> python calculator.py 7 3 divide
The result of divideing 7.0 and 3.0 is: 2.3333333333333335
PS E:\c++>
```



Vaibhav Gupta 0905CS221218



Vaibhav Gupta 0905CS221218