ASSIGNMENT - 13

Name : Aravind Reddy

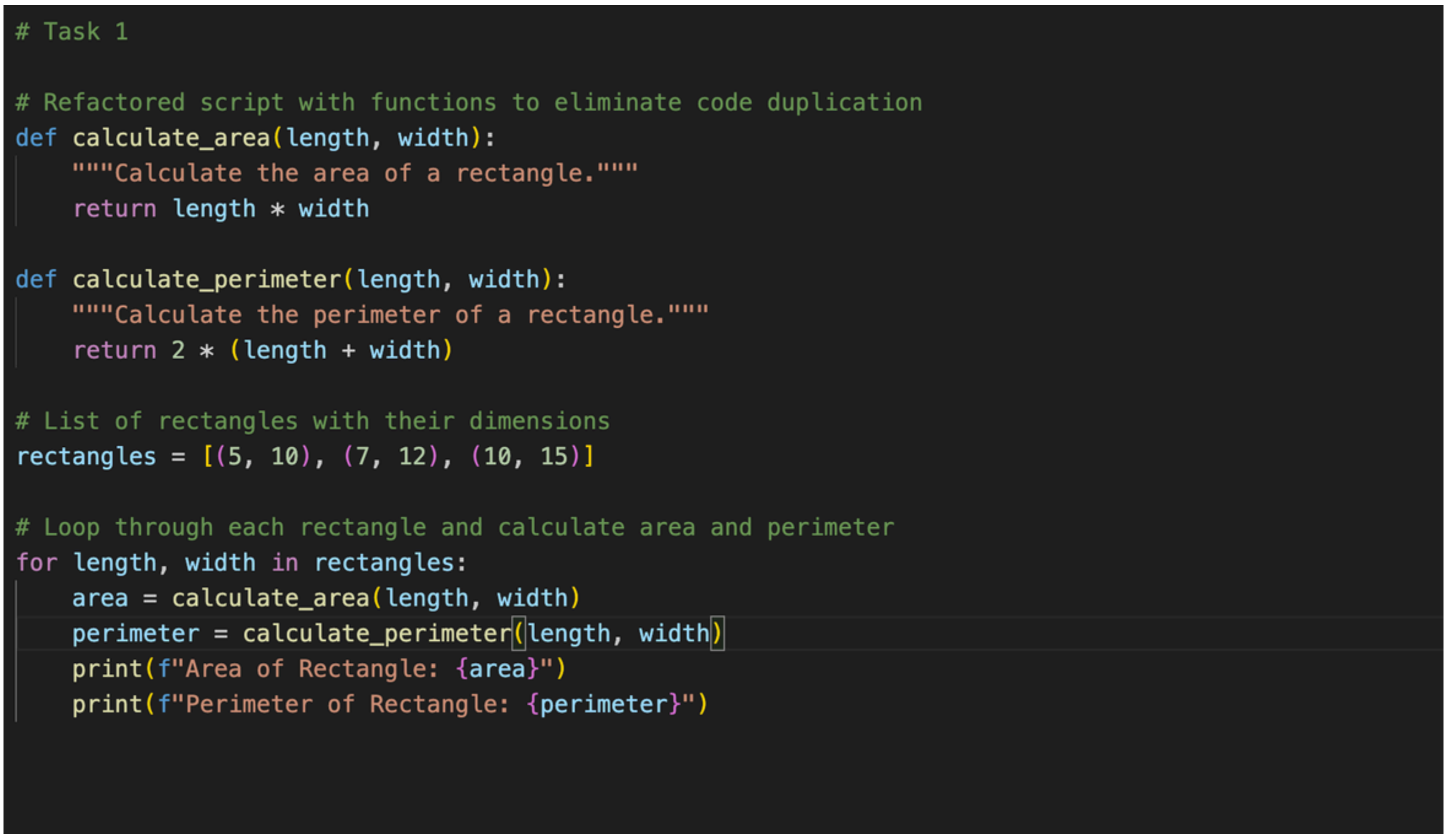
Hall Ticket No : 2303a51027

Batch No : 01

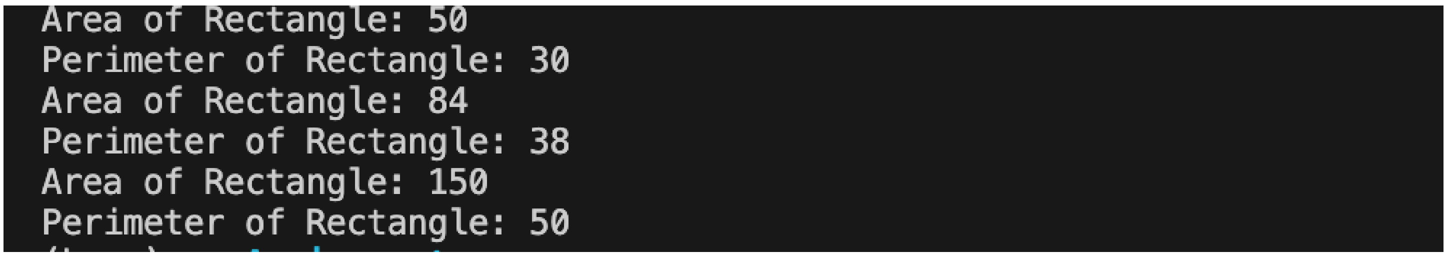
Course Title : AI Assisted Coding

Instructor's Name : Mr. S Naresh Kumar

Problem - 1:

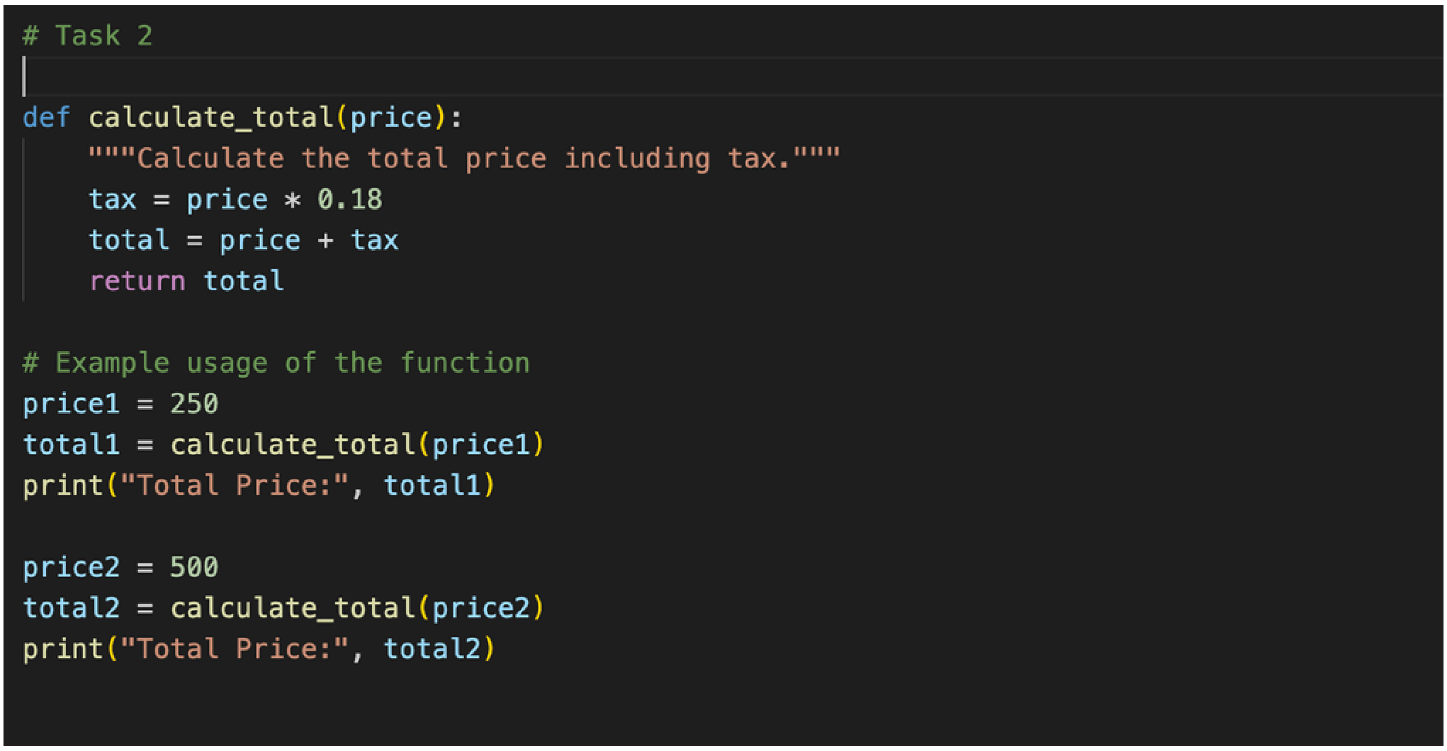


```python
# Task 1

# Refactored script with functions to eliminate code duplication
def calculate_area(length, width):
    """Calculate the area of a rectangle."""
    return length * width

def calculate_perimeter(length, width):
    """Calculate the perimeter of a rectangle."""
    return 2 * (length + width)

# List of rectangles with their dimensions
rectangles = [(5, 10), (7, 12), (10, 15)]

# Loop through each rectangle and calculate area and perimeter
for length, width in rectangles:
    area = calculate_area(length, width)
    perimeter = calculate_perimeter(length, width)
    print(f"Area of Rectangle: {area}")
    print(f"Perimeter of Rectangle: {perimeter}")
```

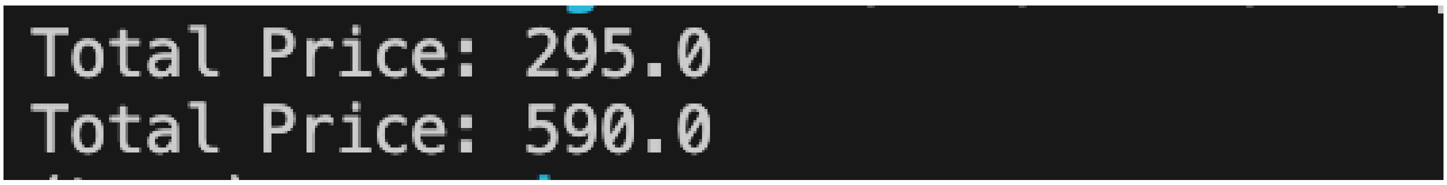


```
Area of Rectangle: 50
Perimeter of Rectangle: 30
Area of Rectangle: 84
Perimeter of Rectangle: 38
Area of Rectangle: 150
Perimeter of Rectangle: 50
```

Problem - 2:

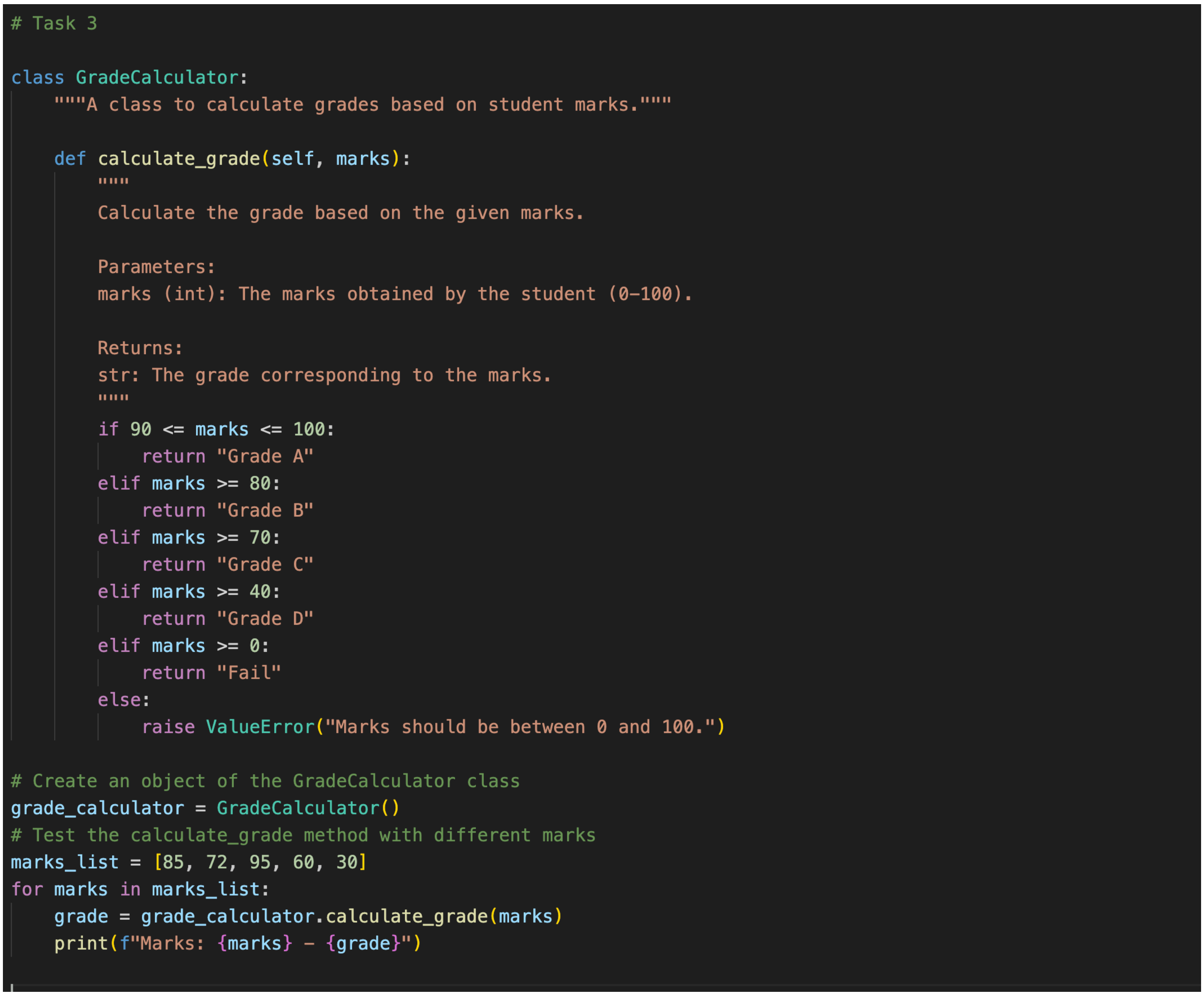


```python
# Task 2

def calculate_total(price):
    """Calculate the total price including tax."""
    tax = price * 0.18
    total = price + tax
    return total

# Example usage of the function
price1 = 250
total1 = calculate_total(price1)
print("Total Price:", total1)

price2 = 500
total2 = calculate_total(price2)
print("Total Price:", total2)
```

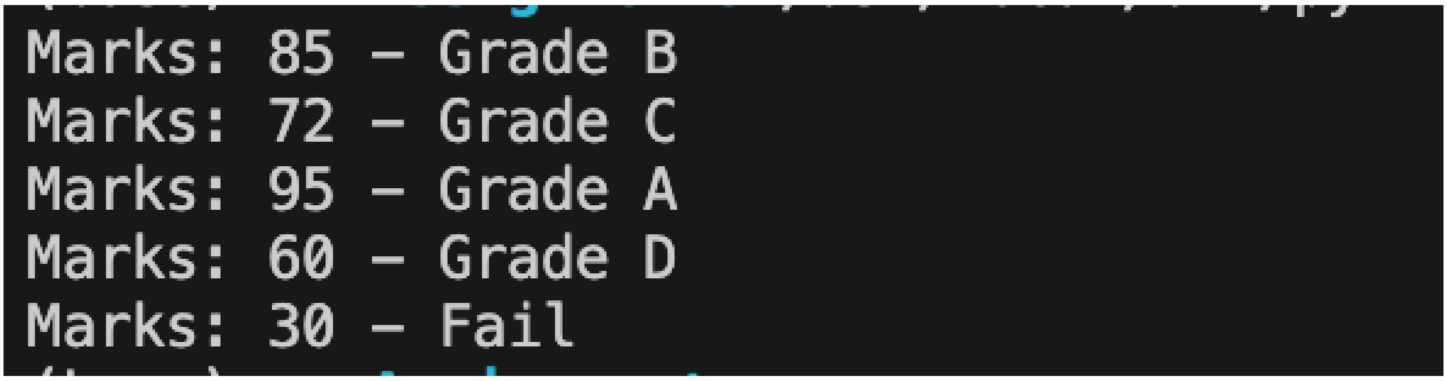


```
Total Price: 295.0
Total Price: 590.0
```

Problem - 3:

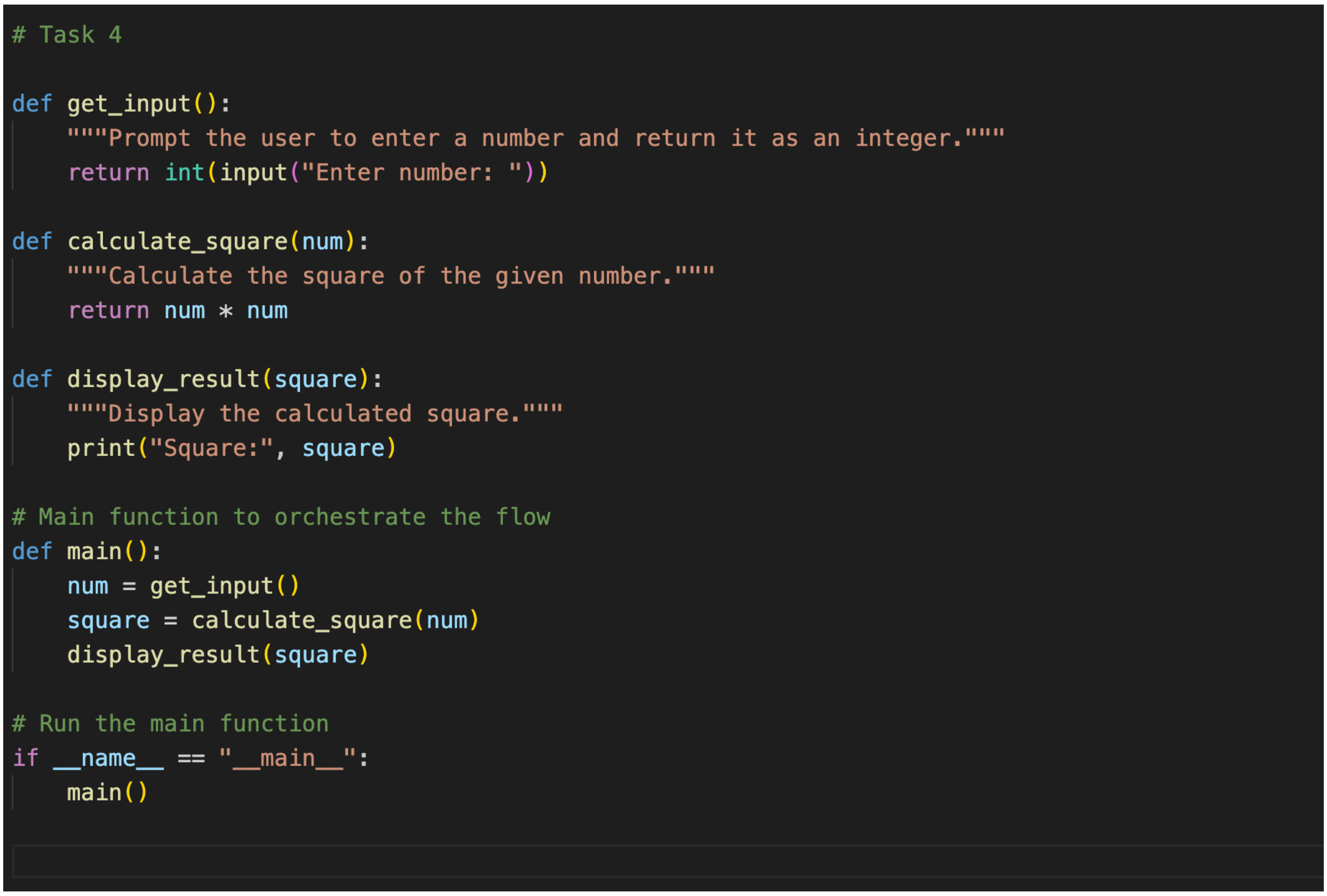


```python
# Task 3

class GradeCalculator:
    """A class to calculate grades based on student marks."""

    def calculate_grade(self, marks):
        """
        Calculate the grade based on the given marks.

        Parameters:
        marks (int): The marks obtained by the student (0-100).

        Returns:
        str: The grade corresponding to the marks.
        """
        if 90 <= marks <= 100:
            return "Grade A"
        elif marks >= 80:
            return "Grade B"
        elif marks >= 70:
            return "Grade C"
        elif marks >= 40:
            return "Grade D"
        elif marks >= 0:
            return "Fail"
        else:
            raise ValueError("Marks should be between 0 and 100.")

# Create an object of the GradeCalculator class
grade_calculator = GradeCalculator()
# Test the calculate_grade method with different marks
marks_list = [85, 72, 95, 60, 30]
for marks in marks_list:
    grade = grade_calculator.calculate_grade(marks)
    print(f"Marks: {marks} - {grade}")
```

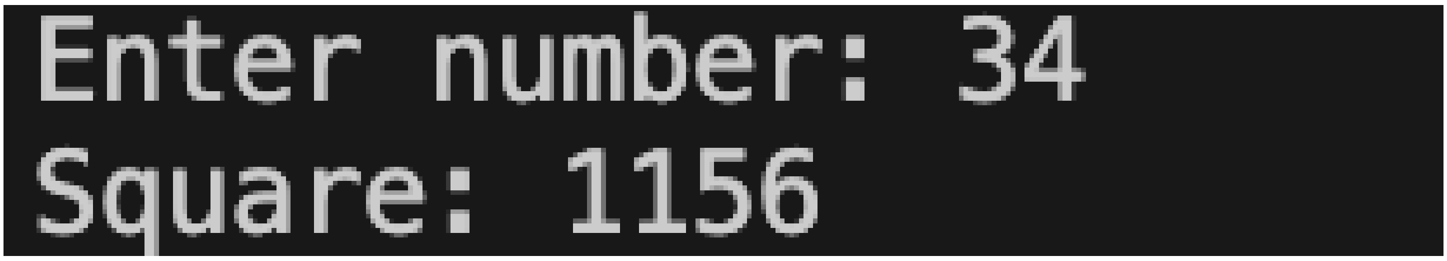


```
Marks: 85 - Grade B
Marks: 72 - Grade C
Marks: 95 - Grade A
Marks: 60 - Grade D
Marks: 30 - Fail
```

Problem - 4:

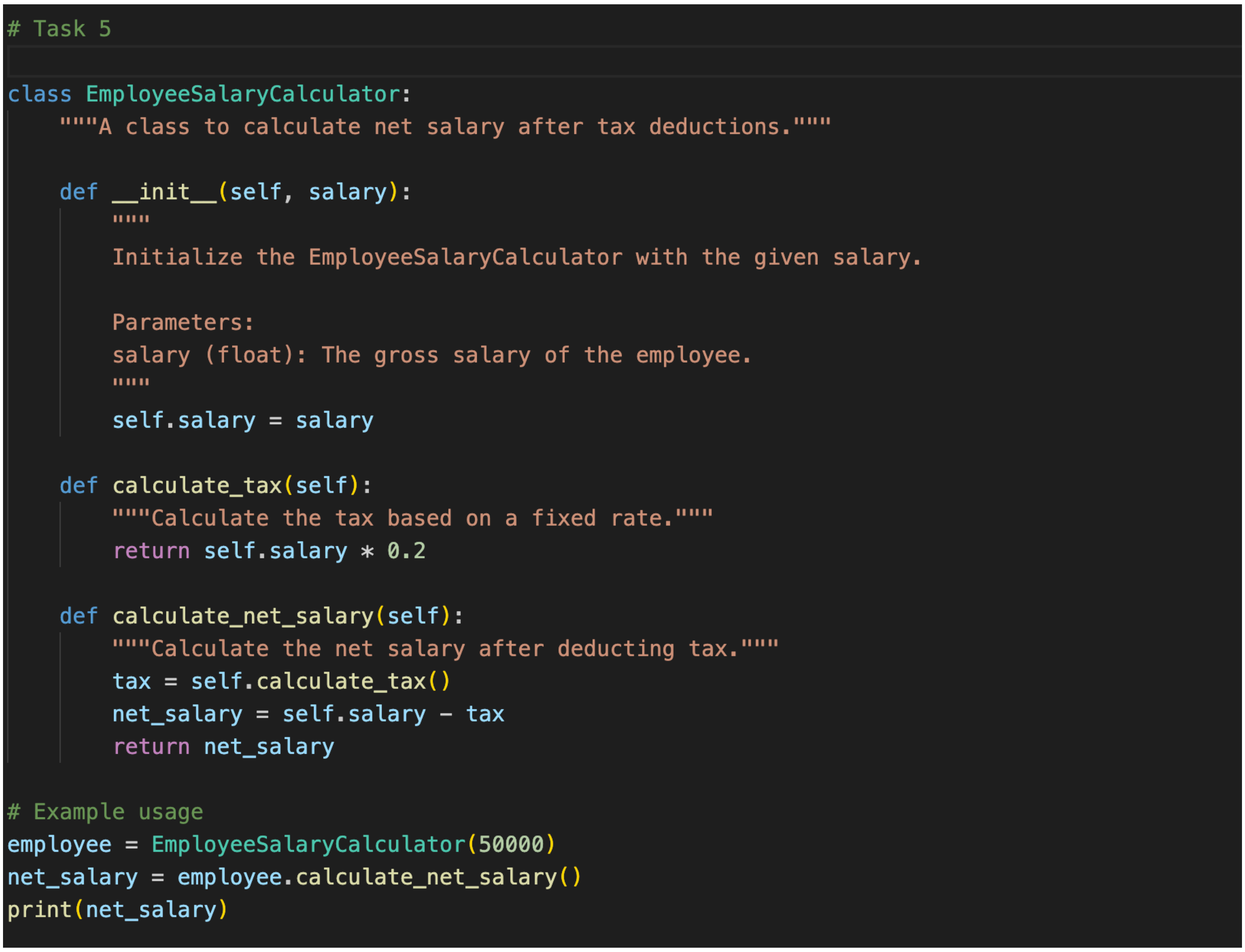


```python
# Task 4

def get_input():
    """Prompt the user to enter a number and return it as an integer."""
    return int(input("Enter number: "))

def calculate_square(num):
    """Calculate the square of the given number."""
    return num * num

def display_result(square):
    """Display the calculated square."""
    print("Square:", square)

# Main function to orchestrate the flow
def main():
    num = get_input()
    square = calculate_square(num)
    display_result(square)

# Run the main function
if __name__ == "__main__":
    main()
```

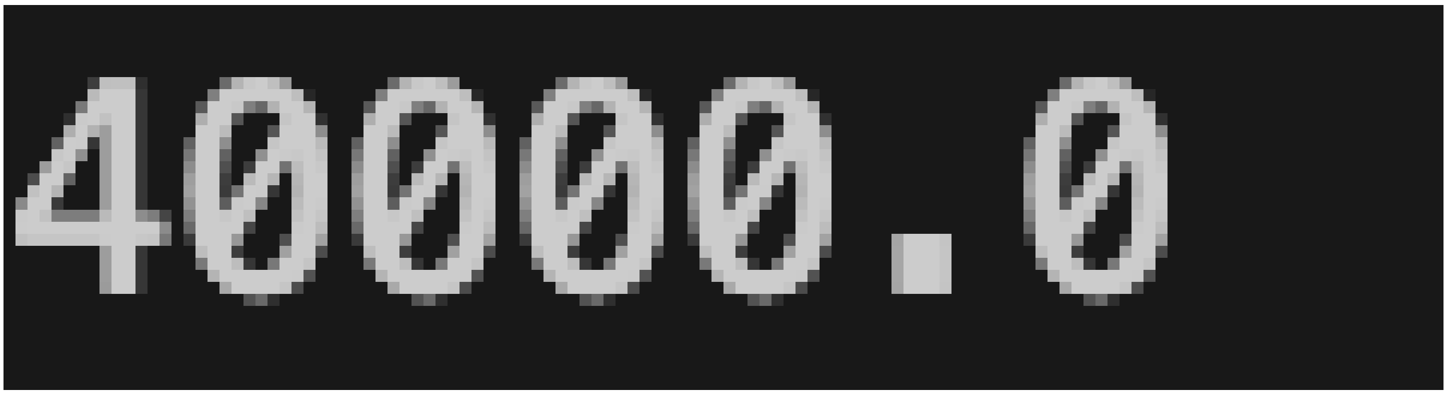


```
Enter number: 34
Square: 1156
```

Problem - 5:

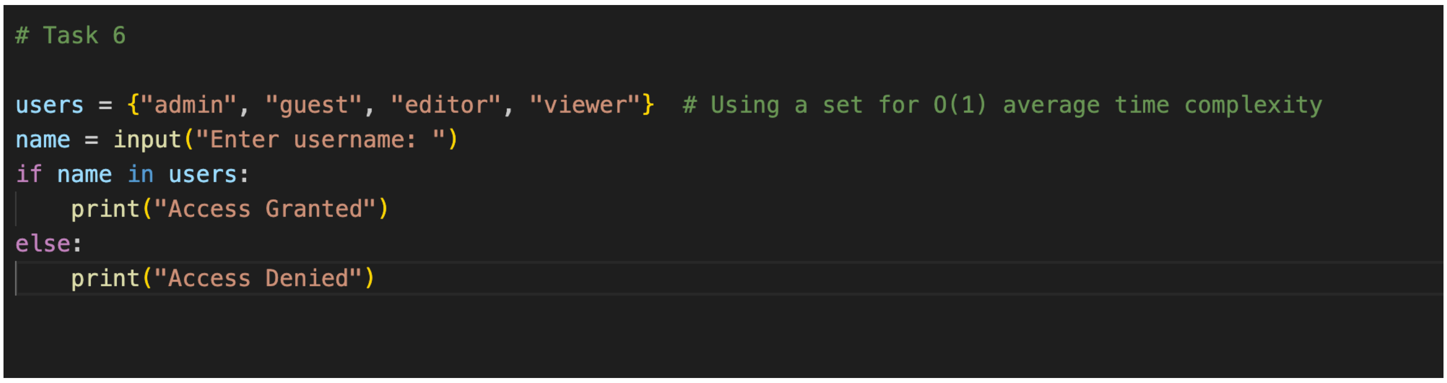


```python
# Task 5

class EmployeeSalaryCalculator:
    """A class to calculate net salary after tax deductions."""

    def __init__(self, salary):
        """
        Initialize the EmployeeSalaryCalculator with the given salary.

        Parameters:
        salary (float): The gross salary of the employee.
        """
        self.salary = salary

    def calculate_tax(self):
        """Calculate the tax based on a fixed rate."""
        return self.salary * 0.2

    def calculate_net_salary(self):
        """Calculate the net salary after deducting tax."""
        tax = self.calculate_tax()
        net_salary = self.salary - tax
        return net_salary

# Example usage
employee = EmployeeSalaryCalculator(50000)
net_salary = employee.calculate_net_salary()
print(net_salary)
```

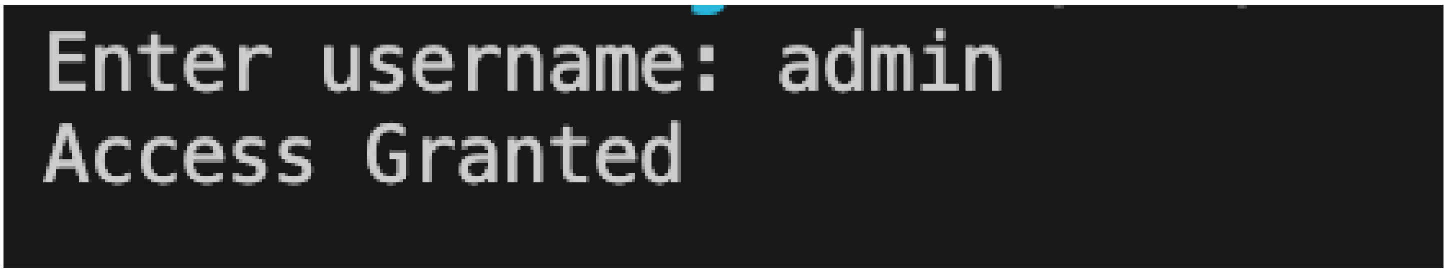


```
40000.0
```

Problem - 6:

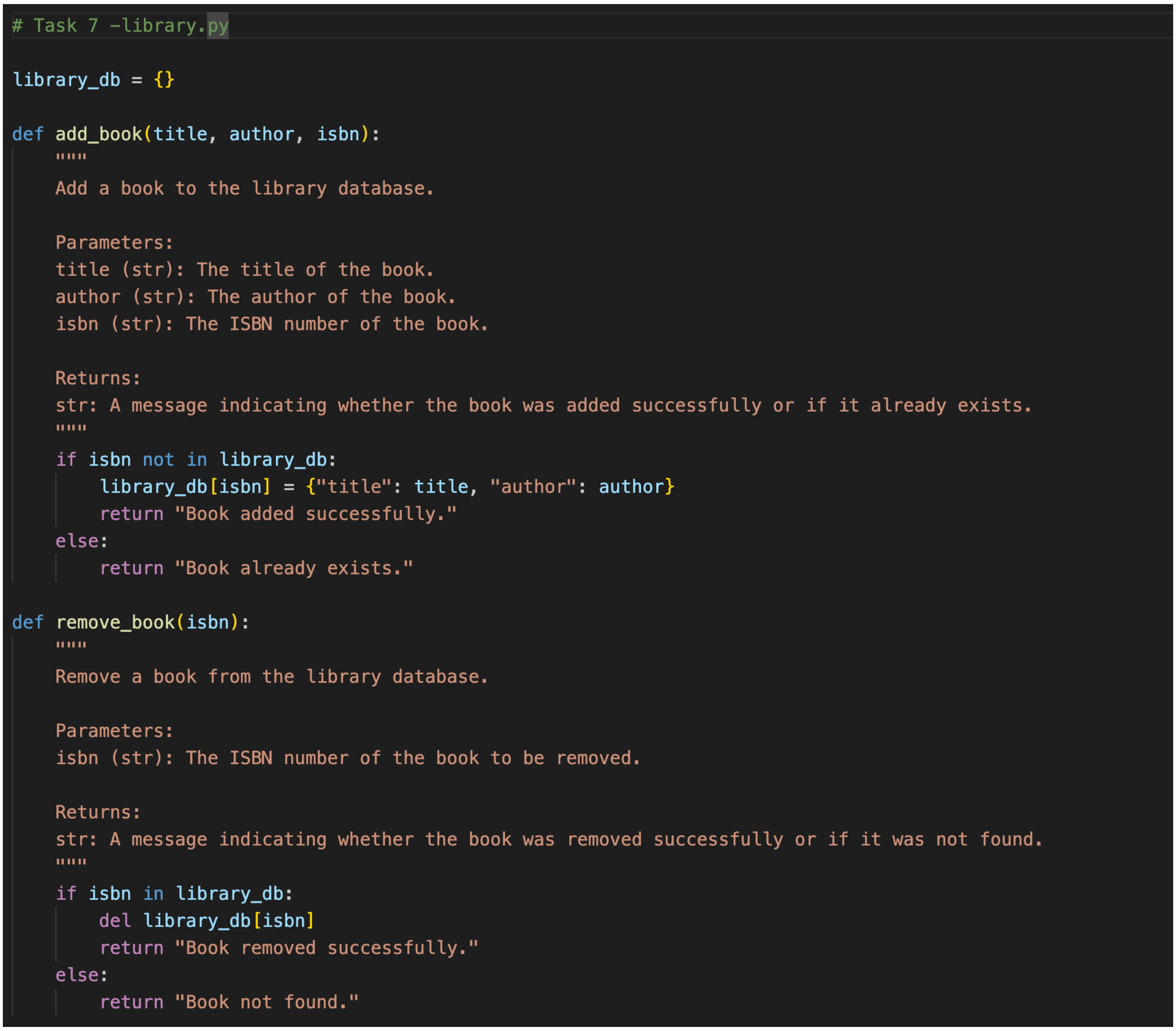


```python
# Task 6

users = {"admin", "guest", "editor", "viewer"}  # Using a set for O(1) average time complexity
name = input("Enter username: ")
if name in users:
    print("Access Granted")
else:
    print("Access Denied")
```

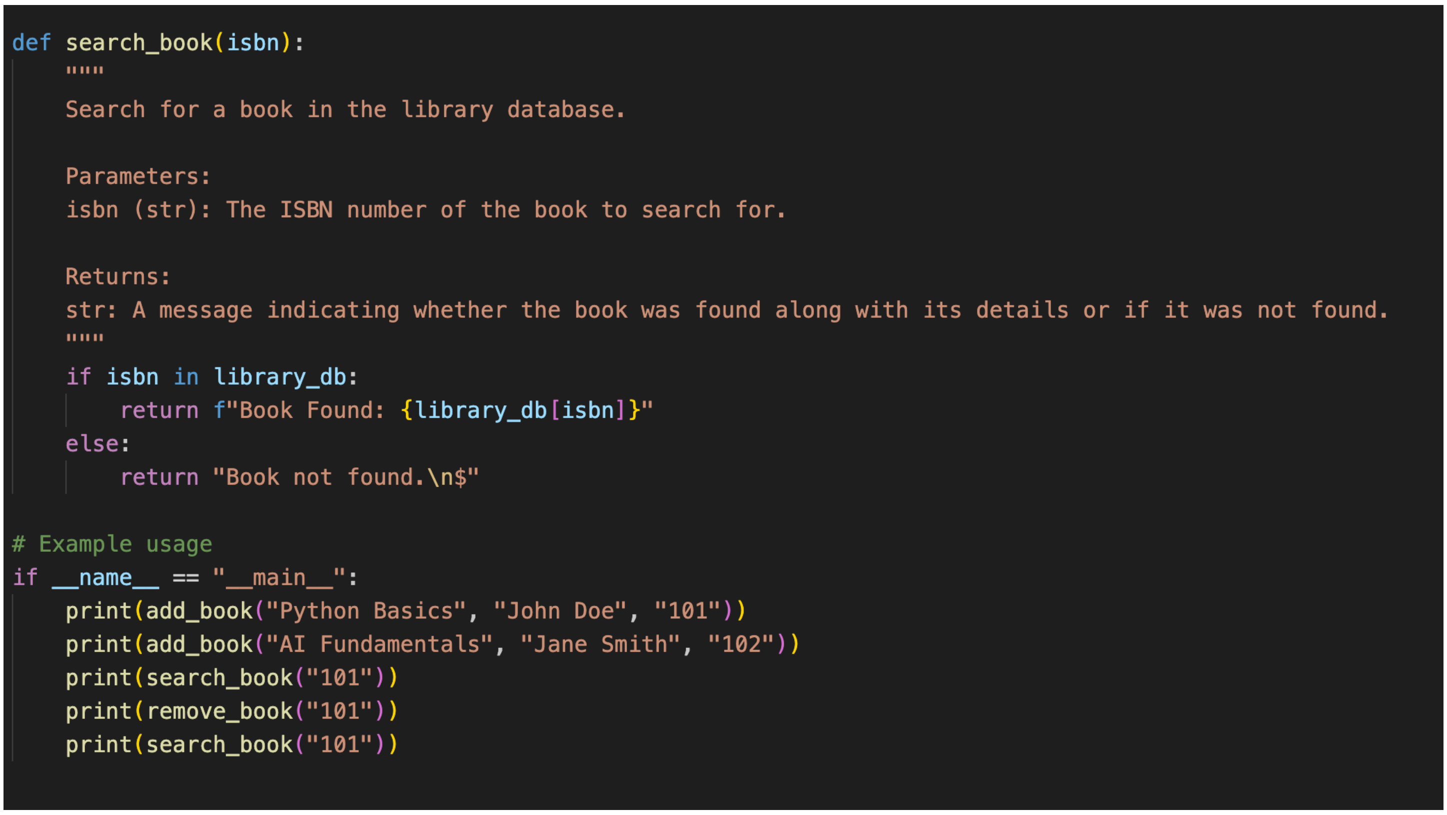


```
Enter username: admin
Access Granted
```

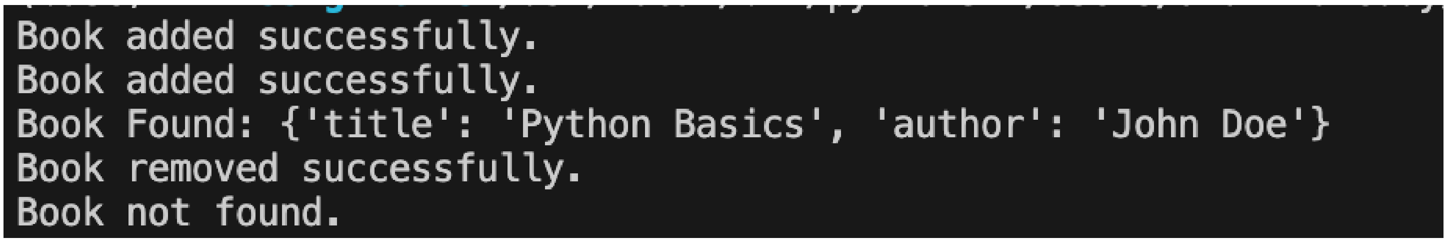
Problem - 7:



```python
# Task 7 -library.py

library_db = {}

def add_book(title, author, isbn):
    """
    Add a book to the library database.

    Parameters:
    title (str): The title of the book.
    author (str): The author of the book.
    isbn (str): The ISBN number of the book.

    Returns:
    str: A message indicating whether the book was added successfully or if it already exists.
    """
    if isbn not in library_db:
        library_db[isbn] = {"title": title, "author": author}
        return "Book added successfully."
    else:
        return "Book already exists."

def remove_book(isbn):
    """
    Remove a book from the library database.

    Parameters:
    isbn (str): The ISBN number of the book to be removed.

    Returns:
    str: A message indicating whether the book was removed successfully or if it was not found.
    """
    if isbn in library_db:
        del library_db[isbn]
        return "Book removed successfully."
    else:
        return "Book not found."
```

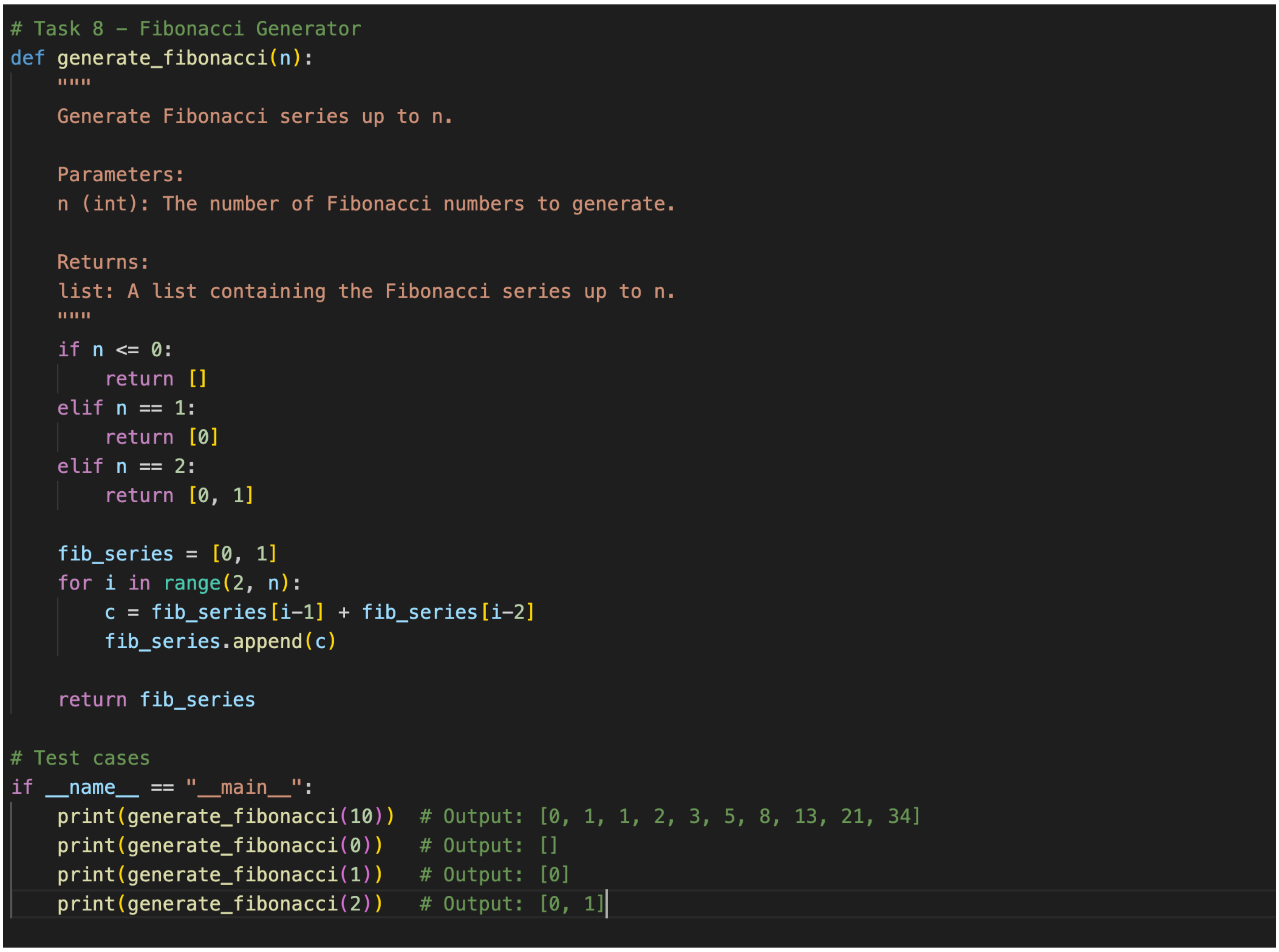


```python
def search_book(isbn):
    """
    Search for a book in the library database.

    Parameters:
    isbn (str): The ISBN number of the book to search for.

    Returns:
    str: A message indicating whether the book was found along with its details or if it was not found.
    """
    if isbn in library_db:
        return f"Book Found: {library_db[isbn]}"
    else:
        return "Book not found.\n$"

# Example usage
if __name__ == "__main__":
    print(add_book("Python Basics", "John Doe", "101"))
    print(add_book("AI Fundamentals", "Jane Smith", "102"))
    print(search_book("101"))
    print(remove_book("101"))
    print(search_book("101"))
```

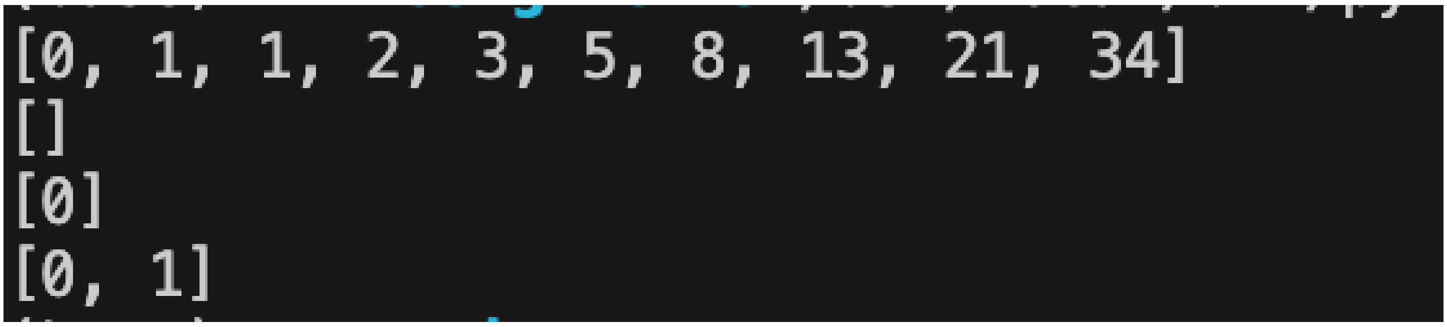


```
Book added successfully.
Book added successfully.
Book Found: {'title': 'Python Basics', 'author': 'John Doe'}
Book removed successfully.
Book not found.
```

Problem - 8:

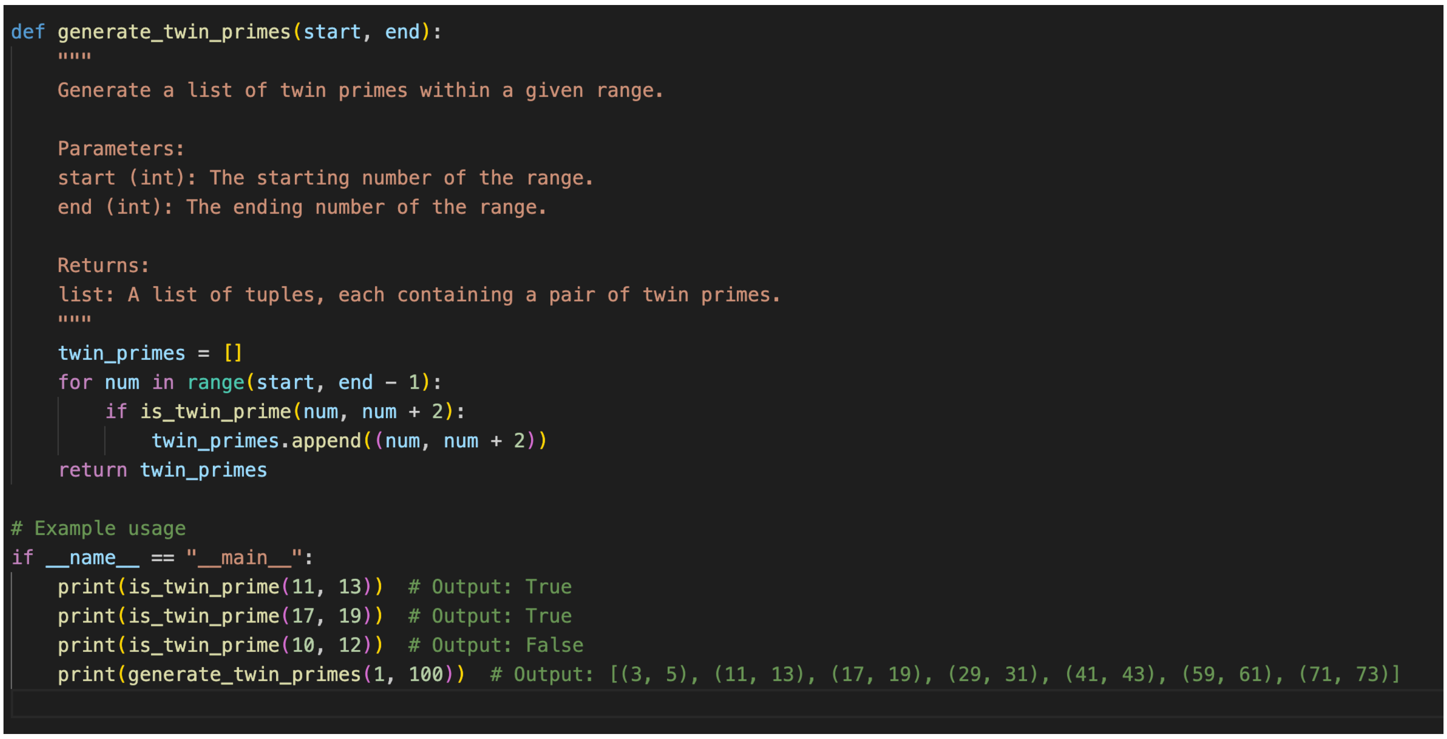


```python
# Task 8 - Fibonacci Generator
def generate_fibonacci(n):
    """
    Generate Fibonacci series up to n.

    Parameters:
    n (int): The number of Fibonacci numbers to generate.

    Returns:
    list: A list containing the Fibonacci series up to n.
    """
    if n <= 0:
        return []
    elif n == 1:
        return [0]
    elif n == 2:
        return [0, 1]

    fib_series = [0, 1]
    for i in range(2, n):
        c = fib_series[i-1] + fib_series[i-2]
        fib_series.append(c)

    return fib_series

# Test cases
if __name__ == "__main__":
    print(generate_fibonacci(10))  # Output: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
    print(generate_fibonacci(0))   # Output: []
    print(generate_fibonacci(1))   # Output: [0]
    print(generate_fibonacci(2))   # Output: [0, 1]
```

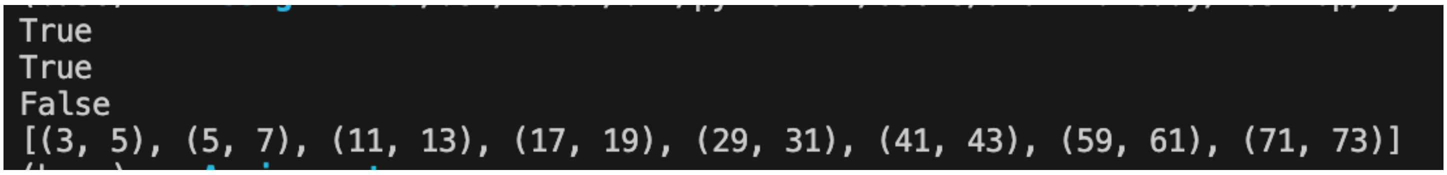


```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
[]
[0]
[0, 1]
```

Problem - 9:

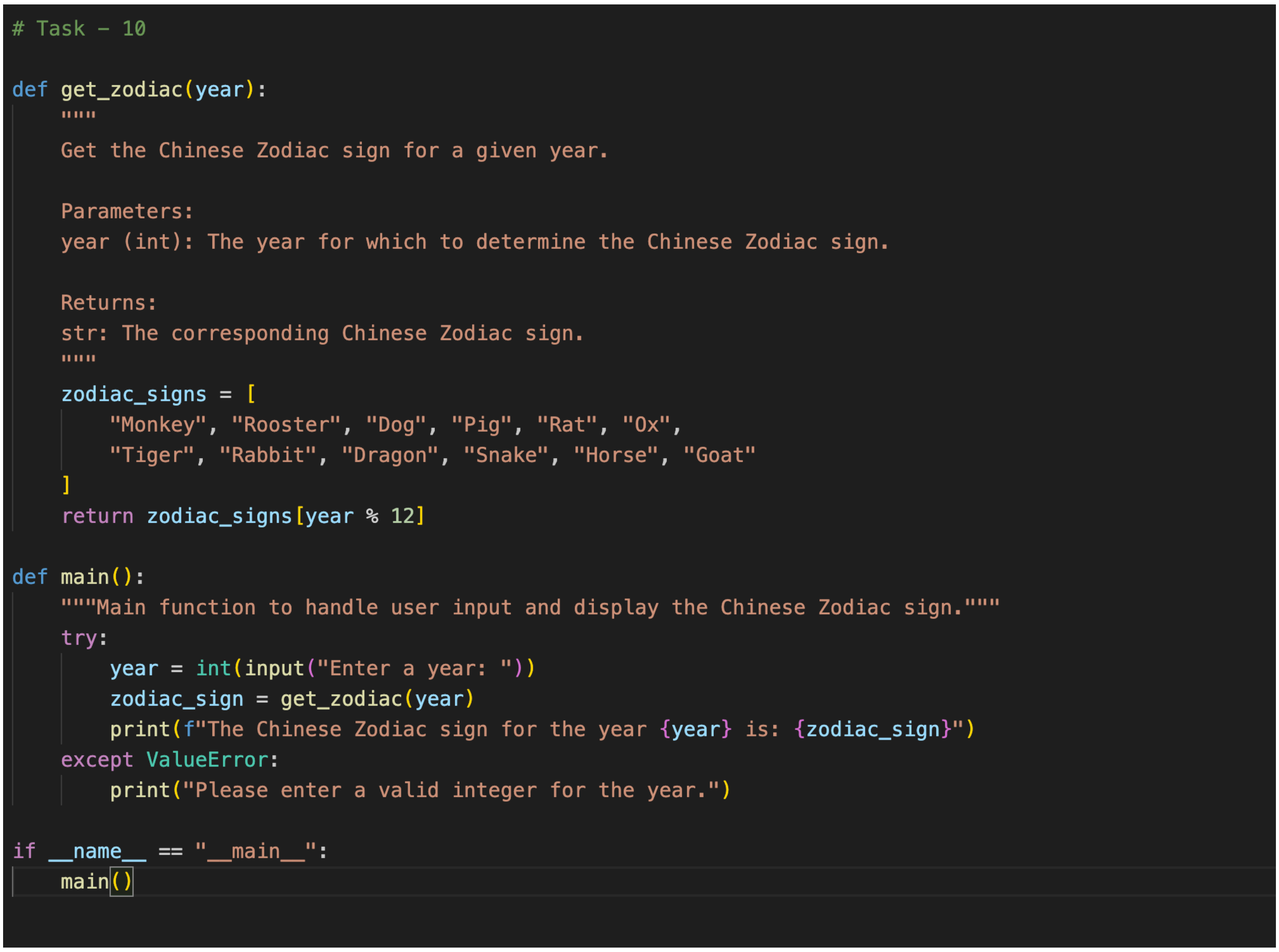


```python
def generate_twin_primes(start, end):
    """
    Generate a list of twin primes within a given range.

    Parameters:
    start (int): The starting number of the range.
    end (int): The ending number of the range.

    Returns:
    list: A list of tuples, each containing a pair of twin primes.
    """
    twin_primes = []
    for num in range(start, end - 1):
        if is_twin_prime(num, num + 2):
            twin_primes.append((num, num + 2))
    return twin_primes

# Example usage
if __name__ == "__main__":
    print(is_twin_prime(11, 13))  # Output: True
    print(is_twin_prime(17, 19))  # Output: True
    print(is_twin_prime(10, 12))  # Output: False
    print(generate_twin_primes(1, 100))  # Output: [(3, 5), (11, 13), (17, 19), (29, 31), (41, 43), (59, 61), (71, 73)]
```

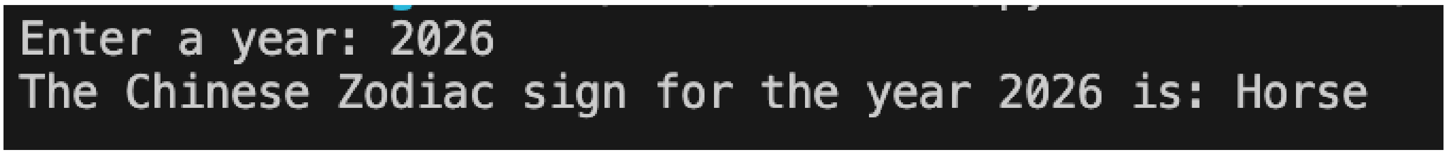


```
True
True
False
[(3, 5), (5, 7), (11, 13), (17, 19), (29, 31), (41, 43), (59, 61), (71, 73)]
```

Problem - 10:

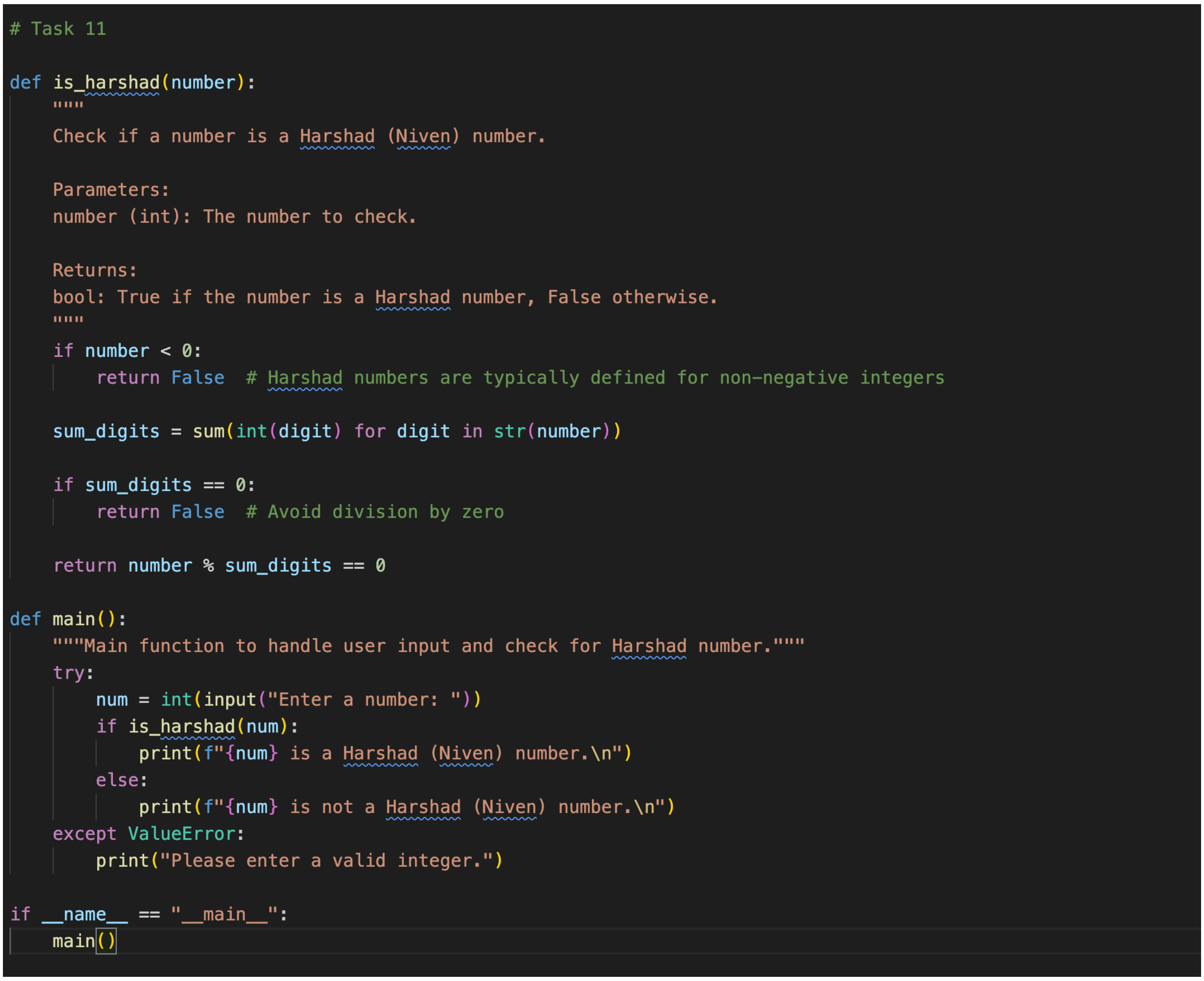


```python
# Task - 10

def get_zodiac(year):
    """
    Get the Chinese Zodiac sign for a given year.

    Parameters:
    year (int): The year for which to determine the Chinese Zodiac sign.

    Returns:
    str: The corresponding Chinese Zodiac sign.
    """
    zodiac_signs = [
        "Monkey", "Rooster", "Dog", "Pig", "Rat", "Ox",
        "Tiger", "Rabbit", "Dragon", "Snake", "Horse", "Goat"
    ]
    return zodiac_signs[year % 12]

def main():
    """Main function to handle user input and display the Chinese Zodiac sign."""
    try:
        year = int(input("Enter a year: "))
        zodiac_sign = get_zodiac(year)
        print(f"The Chinese Zodiac sign for the year {year} is: {zodiac_sign}")
    except ValueError:
        print("Please enter a valid integer for the year.")

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

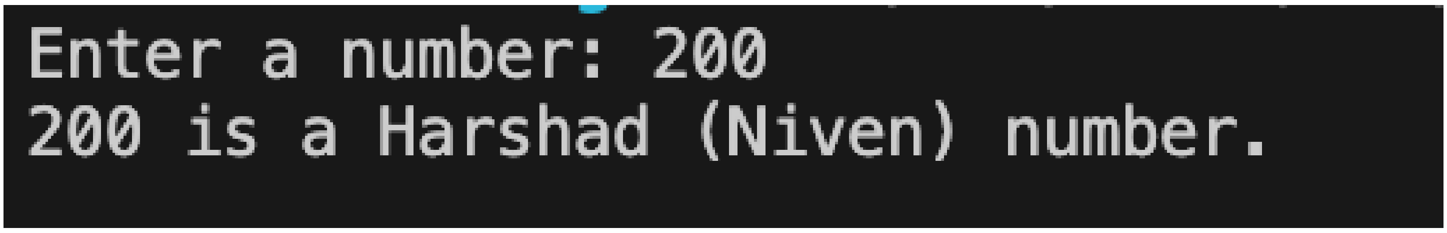


```
Enter a year: 2026
The Chinese Zodiac sign for the year 2026 is: Horse
```

Problem - 11:

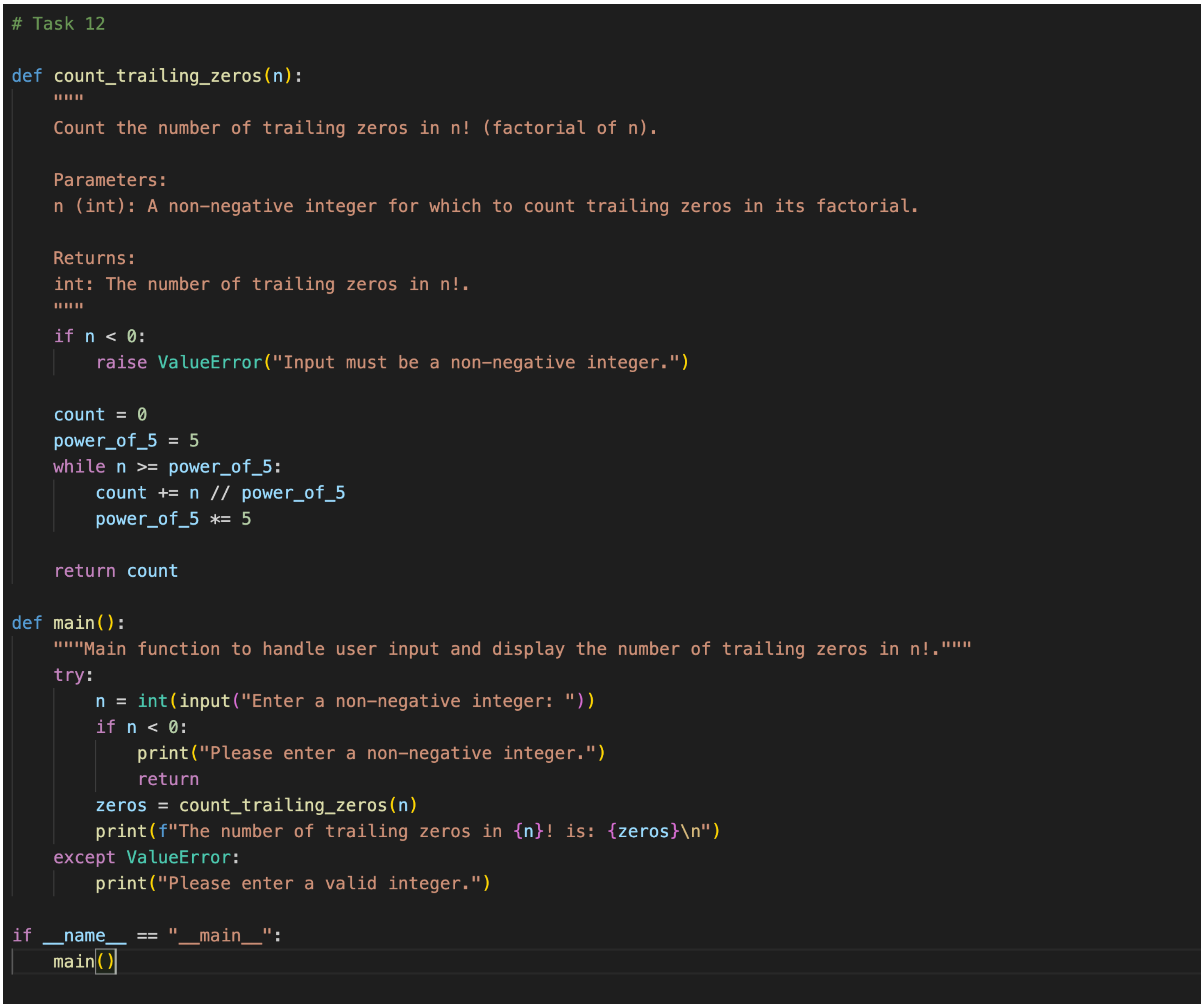


```python
# Task 11

def is_harshad(number):
    """
    Check if a number is a Harshad (Niven) number.

    Parameters:
    number (int): The number to check.

    Returns:
    bool: True if the number is a Harshad number, False otherwise.
    """
    if number < 0:
        return False  # Harshad numbers are typically defined for non-negative integers

    sum_digits = sum(int(digit) for digit in str(number))

    if sum_digits == 0:
        return False  # Avoid division by zero

    return number % sum_digits == 0

def main():
    """Main function to handle user input and check for Harshad number."""
    try:
        num = int(input("Enter a number: "))
        if is_harshad(num):
            print(f"{num} is a Harshad (Niven) number.\n")
        else:
            print(f"{num} is not a Harshad (Niven) number.\n")
    except ValueError:
        print("Please enter a valid integer.")

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

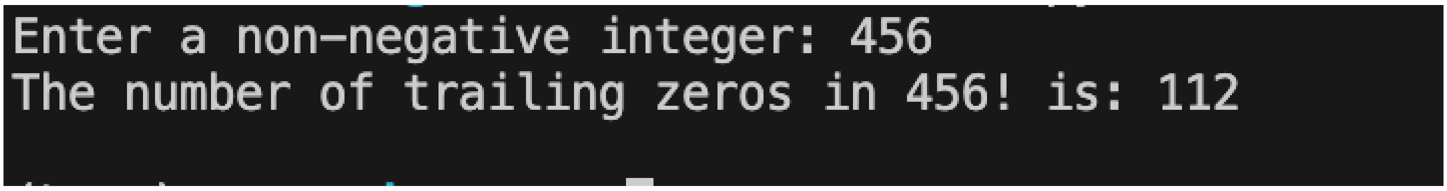


```
Enter a number: 200
200 is a Harshad (Niven) number.
```

Problem - 12:

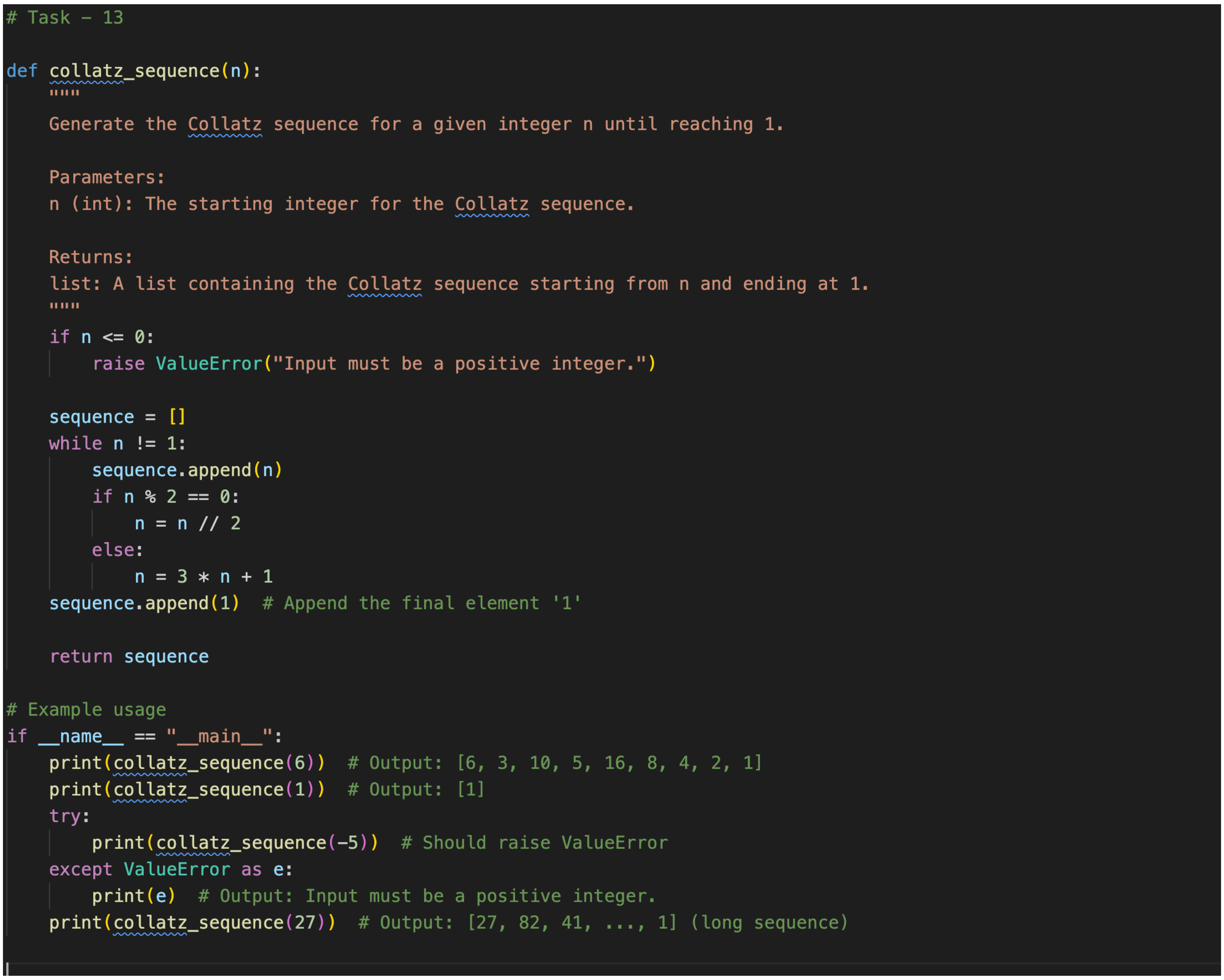


```python
# Task 12

def count_trailing_zeros(n):
    """
    Count the number of trailing zeros in n! (factorial of n).

    Parameters:
    n (int): A non-negative integer for which to count trailing zeros in its factorial.

    Returns:
    int: The number of trailing zeros in n!.
    """
    if n < 0:
        raise ValueError("Input must be a non-negative integer.")

    count = 0
    power_of_5 = 5
    while n >= power_of_5:
        count += n // power_of_5
        power_of_5 *= 5

    return count

def main():
    """Main function to handle user input and display the number of trailing zeros in n!."""
    try:
        n = int(input("Enter a non-negative integer: "))
        if n < 0:
            print("Please enter a non-negative integer.")
            return
        zeros = count_trailing_zeros(n)
        print(f"The number of trailing zeros in {n}! is: {zeros}\n")
    except ValueError:
        print("Please enter a valid integer.")

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

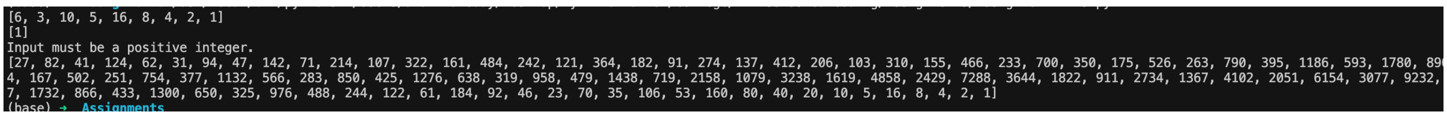


```
Enter a non-negative integer: 456
The number of trailing zeros in 456! is: 112
```

Problem - 13:

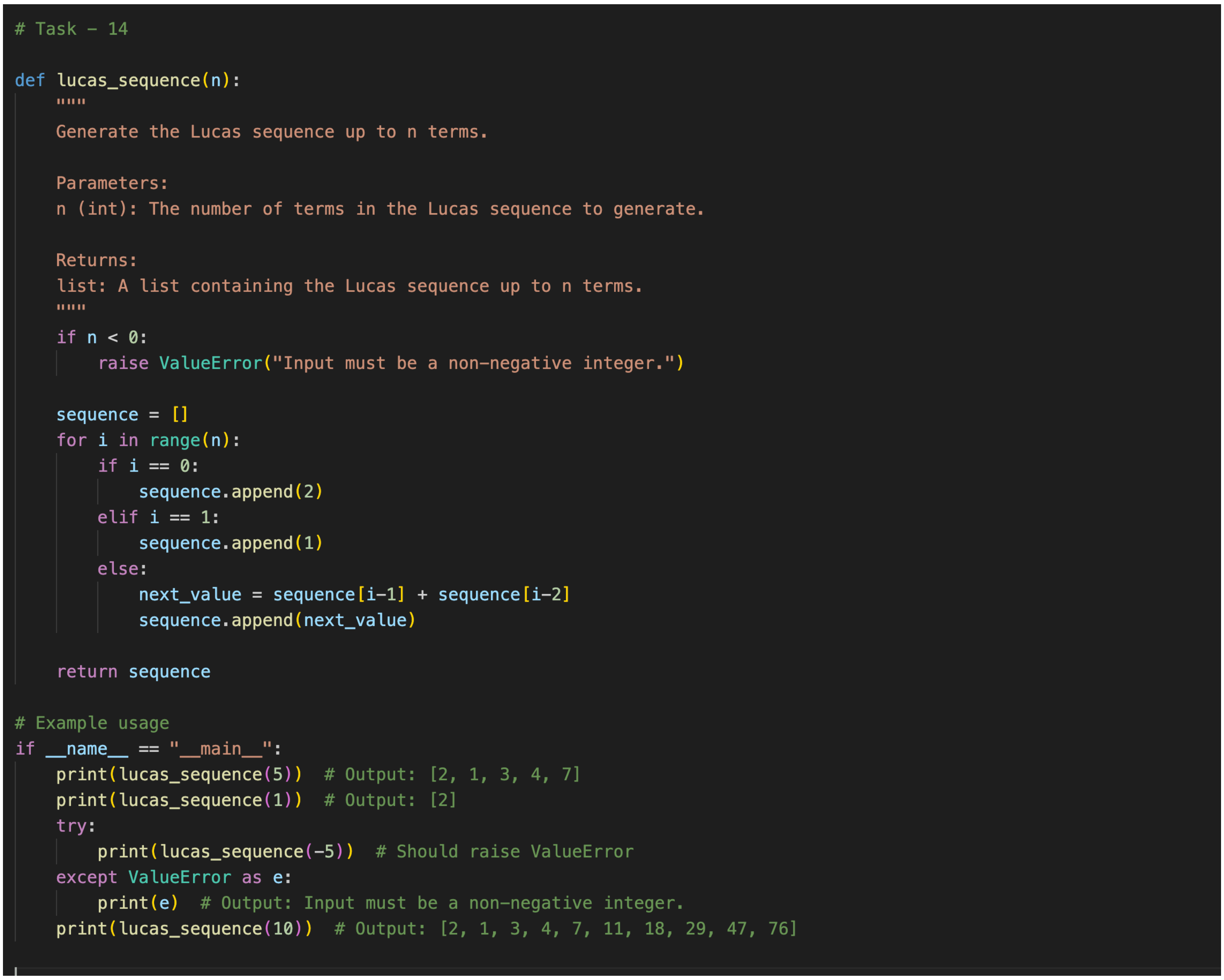


```python
# Task - 13

def collatz_sequence(n):
    """
    Generate the Collatz sequence for a given integer n until reaching 1.

    Parameters:
    n (int): The starting integer for the Collatz sequence.

    Returns:
    list: A list containing the Collatz sequence starting from n and ending at 1.
    """
    if n <= 0:
        raise ValueError("Input must be a positive integer.")

    sequence = []
    while n != 1:
        sequence.append(n)
        if n % 2 == 0:
            n = n // 2
        else:
            n = 3 * n + 1
    sequence.append(1)  # Append the final element '1'

    return sequence

# Example usage
if __name__ == "__main__":
    print(collatz_sequence(6))  # Output: [6, 3, 10, 5, 16, 8, 4, 2, 1]
    print(collatz_sequence(1))  # Output: [1]
    try:
        print(collatz_sequence(-5))  # Should raise ValueError
    except ValueError as e:
        print(e)  # Output: Input must be a positive integer.
    print(collatz_sequence(27))  # Output: [27, 82, 41, ..., 1] (long sequence)
```

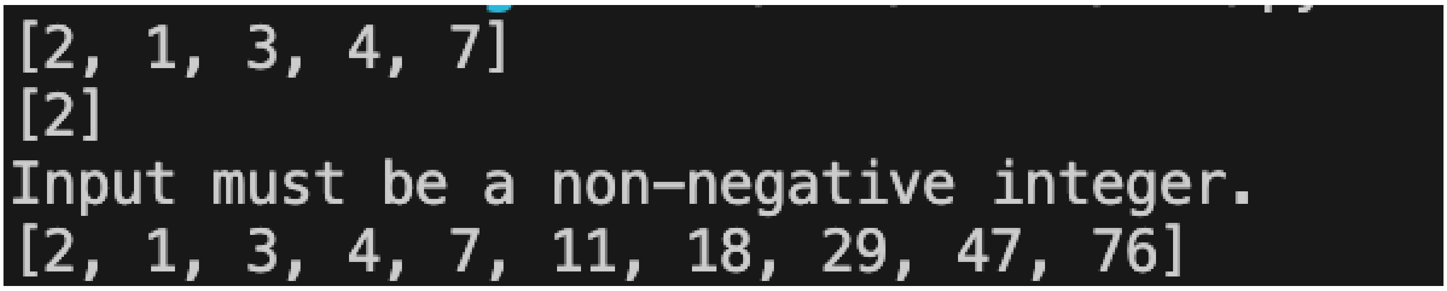


```
[6, 3, 10, 5, 16, 8, 4, 2, 1]
[1]
Input must be a positive integer.
[27, 82, 41, 124, 62, 31, 94, 47, 142, 71, 214, 107, 322, 161, 484, 242, 121, 364, 182, 91, 274, 137, 412, 206, 103, 310, 155, 466, 233, 700, 350, 175, 526, 263, 790, 395, 1186, 593, 1780, 89
4, 167, 502, 251, 754, 377, 1132, 566, 283, 850, 425, 1276, 638, 319, 958, 479, 1438, 719, 2158, 1079, 3238, 1619, 4858, 2429, 7288, 3644, 1822, 911, 2734, 1367, 4102, 2051, 6154, 3077, 9232,
7, 1732, 866, 433, 1300, 650, 325, 976, 488, 244, 122, 61, 184, 92, 46, 23, 70, 35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1]
(base) → Assignments
```

Problem - 14:

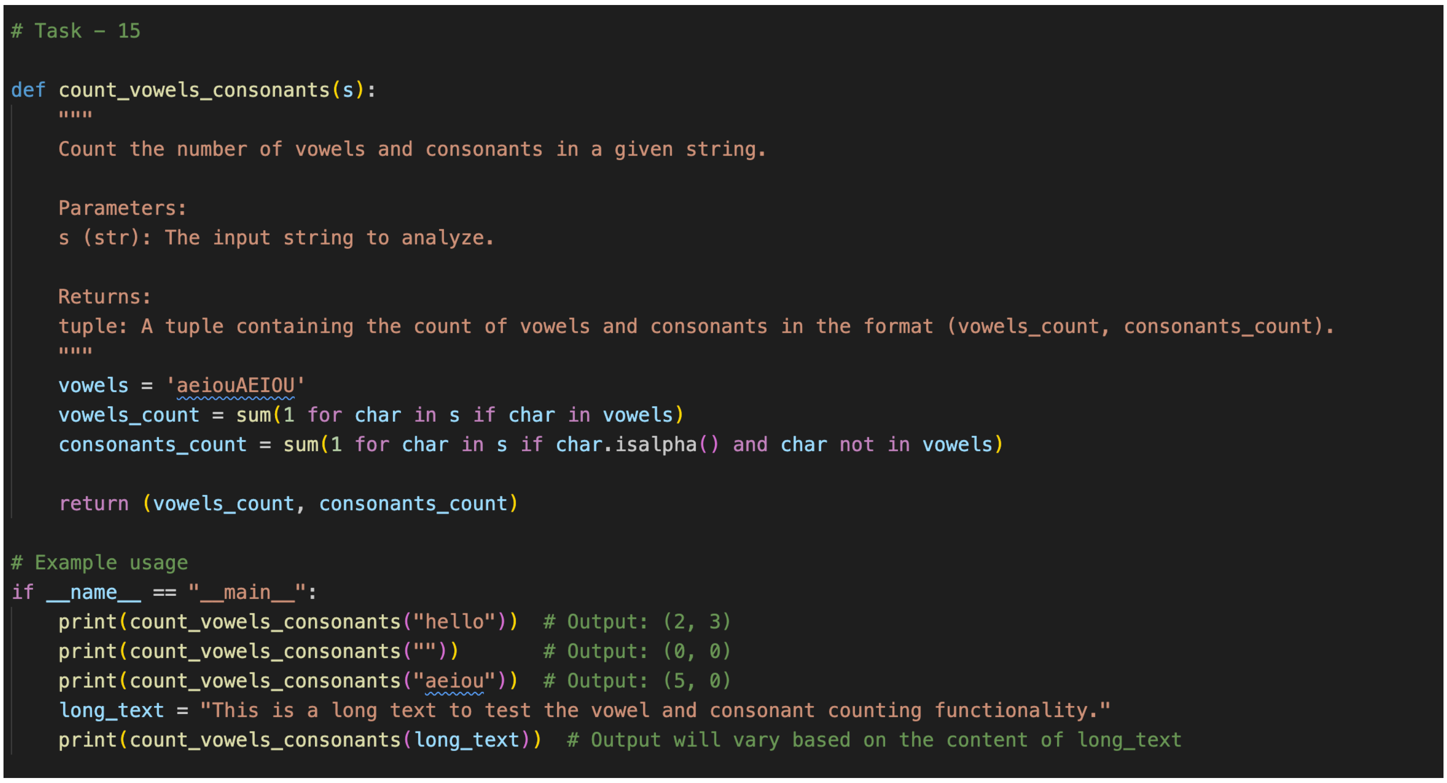


```python
# Task - 14

def lucas_sequence(n):
    """
    Generate the Lucas sequence up to n terms.

    Parameters:
    n (int): The number of terms in the Lucas sequence to generate.

    Returns:
    list: A list containing the Lucas sequence up to n terms.
    """
    if n < 0:
        raise ValueError("Input must be a non-negative integer.")

    sequence = []
    for i in range(n):
        if i == 0:
            sequence.append(2)
        elif i == 1:
            sequence.append(1)
        else:
            next_value = sequence[i-1] + sequence[i-2]
            sequence.append(next_value)

    return sequence

# Example usage
if __name__ == "__main__":
    print(lucas_sequence(5))  # Output: [2, 1, 3, 4, 7]
    print(lucas_sequence(1))  # Output: [2]
    try:
        print(lucas_sequence(-5))  # Should raise ValueError
    except ValueError as e:
        print(e)  # Output: Input must be a non-negative integer.
    print(lucas_sequence(10))  # Output: [2, 1, 3, 4, 7, 11, 18, 29, 47, 76]
```

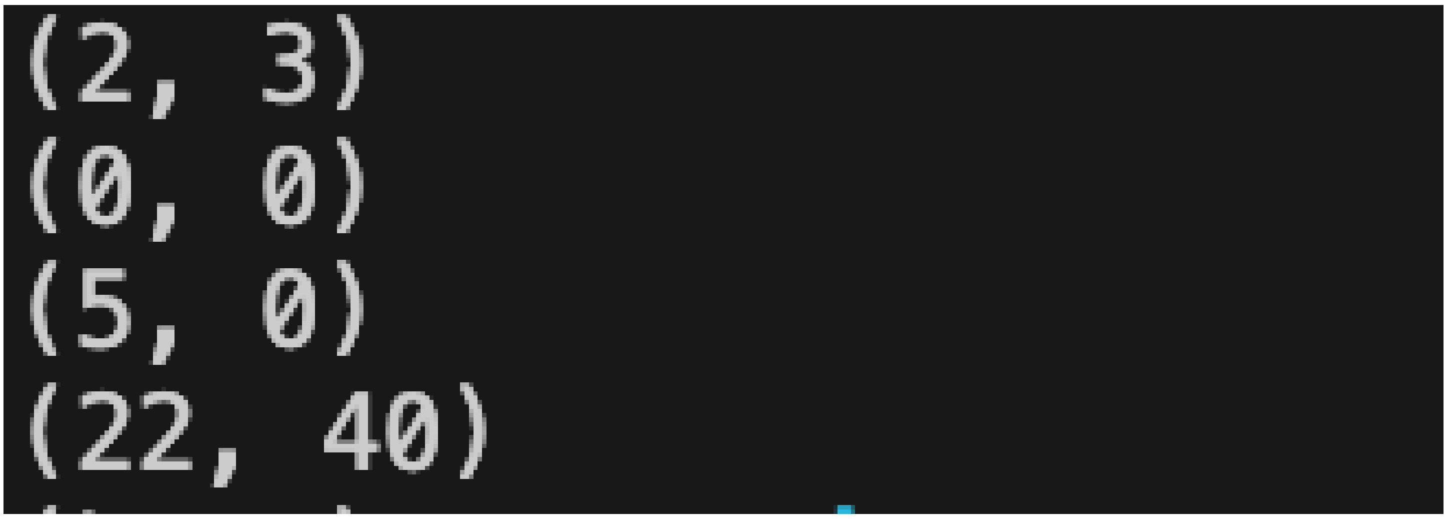


```
[2, 1, 3, 4, 7]
[2]
Input must be a non-negative integer.
[2, 1, 3, 4, 7, 11, 18, 29, 47, 76]
```

Problem - 15:



```python
# Task - 15

def count_vowels_consonants(s):
    """
    Count the number of vowels and consonants in a given string.

    Parameters:
    s (str): The input string to analyze.

    Returns:
    tuple: A tuple containing the count of vowels and consonants in the format (vowels_count, consonants_count).
    """
    vowels = 'aeiouAEIOU'
    vowels_count = sum(1 for char in s if char in vowels)
    consonants_count = sum(1 for char in s if char.isalpha() and char not in vowels)

    return (vowels_count, consonants_count)

# Example usage
if __name__ == "__main__":
    print(count_vowels_consonants("hello"))  # Output: (2, 3)
    print(count_vowels_consonants(""))       # Output: (0, 0)
    print(count_vowels_consonants("aeiou"))  # Output: (5, 0)
    long_text = "This is a long text to test the vowel and consonant counting functionality."
    print(count_vowels_consonants(long_text))  # Output will vary based on the content of long_text
```



```
(2, 3)
(0, 0)
(5, 0)
(22, 40)
```