AI ASSISTED CODING

ASSIGNMENT – 3

NAME : K GANESH

ROLL NO : 2303A52356

```
# This is formatted as code
```

PROMPT :Zero-Shot Prompting (Palindrome Number Check)

CODE:

```
def is_palindrome_number(n):
    """
    Checks if a given integer is a palindrome number.

    A palindrome number reads the same forwards and backward.
    For example, 121 is a palindrome, but 123 is not.
    This function handles both positive and negative integers, but typically
    palindromes refer to non-negative numbers in a mathematical context.

    Args:
        n: An integer.

    Returns:
        True if n is a palindrome number, False otherwise.
    """
    # Negative numbers are generally not considered palindromes (e.g., -121
is not 121-)
    if n < 0:
        return False

    # Single digit numbers are palindromes
    if 0 <= n < 10:
        return True

    # Convert the number to a string to easily compare its digits
    str_n = str(n)

    # Compare the string with its reverse
    return str_n == str_n[::-1]
```

OUTPUT:

```
print("--- Palindrome Number Check (User Input) ---")
try:
    user_input = int(input("Enter an integer to check if it's a palindrome:
"))
    if is_palindrome_number(user_input):
        print(f"The number {user_input} is a palindrome number.")
    else:
        print(f"The number {user_input} is NOT a palindrome number.")
except ValueError:
    print("Invalid input. Please enter an integer.")

print("\n--- Test Cases ---")
test_numbers = [121, 123, 0, 5, 10, -121, 1221, 98789, 1001, 12345]
for num in test_numbers:
    if is_palindrome_number(num):
        print(f"The number {num} is a palindrome number.")
    else:
        print(f"The number {num} is NOT a palindrome number.")
```

```
--- Palindrome Number Check (User Input) ---
Enter an integer to check if it's a palindrome: 343
The number 343 is a palindrome number.

--- Test Cases ---
The number 121 is a palindrome number.
The number 123 is NOT a palindrome number.
The number 0 is a palindrome number.
The number 5 is a palindrome number.
The number 10 is NOT a palindrome number.
The number -121 is NOT a palindrome number.
The number 1221 is a palindrome number.
The number 98789 is a palindrome number.
The number 1001 is a palindrome number.
The number 12345 is NOT a palindrome number.
```

EXPLANATION :

Zero-shot prompting gives only the task description, so the AI must generate the palindrome program without examples.

The program checks whether a number reads the same forward and backward.

It prints whether the given number is a palindrome or not based on this comparison.

PROMPT: One-Shot Prompting (Factorial Calculation)

CODE:

```python
def factorial(n):
    """
    Calculates the factorial of a non-negative integer.

    The factorial of a non-negative integer n, denoted by n!, is the product
    of all positive integers less than or equal to n. The factorial of 0 is
1.
    For example, 5! = 5 * 4 * 3 * 2 * 1 = 120.

    Args:
        n: A non-negative integer.

    Returns:
        The factorial of n if n is non-negative, otherwise an error message.
    """
    if not isinstance(n, int) or n < 0:
        raise ValueError("Factorial is defined only for non-negative
integers.")

    if n == 0:
        return 1
    else:
        result = 1
        for i in range(1, n + 1):
            result *= i
        return result
```

OUTPUT:

```python
# --- Example Usage ---

print("--- Factorial Calculation (User Input) ---")
try:
    user_input = int(input("Enter a non-negative integer to calculate its
factorial: "))
    fact_result = factorial(user_input)
    print(f"The factorial of {user_input} is {fact_result}.")
except ValueError as e:
    print(f"Error: {e}")

print("\n--- Test Cases ---")
test_numbers = [0, 1, 5, 7, 10]
for num in test_numbers:
    try:
        fact_result = factorial(num)
        print(f"The factorial of {num} is {fact_result}.")
    except ValueError as e:
        print(f"Error calculating factorial for {num}: {e}")
```

```
--- Factorial Calculation (User Input) ---
Enter a non-negative integer to calculate its factorial: 5
The factorial of 5 is 120.

--- Test Cases ---
The factorial of 0 is 1.
The factorial of 1 is 1.
The factorial of 5 is 120.
The factorial of 7 is 5040.
The factorial of 10 is 3628800.
```

EXPLANATION:

The `factorial(n)` function calculates the factorial of a non-negative integer `n`.

1. **Input Validation:** It first checks if the input `n` is a non-negative integer. If not, it raises a `ValueError` because factorial is only defined for such numbers.
1. **Base Case (n=0):** The factorial of 0 is defined as 1, so if `n` is 0, the function immediately returns 1.
2. **Iterative Calculation:** For `n > 0`, it initializes a `result` to 1. Then, it iterates from 1 up to `n` (inclusive), multiplying `result` by each number in the loop. This computes the product of all positive integers up to `n`.
3. **Return Value:** The final `result` is the factorial of `n`.

PROMPT: Few-Shot Prompting (Armstrong Number Check)

CODE:

```
def is_armstrong_number(n):
    """
    Checks if a given positive integer is an Armstrong number.

    An Armstrong number is a number that is the sum of its own digits
    each raised to the power of the number of digits.
    For example, 153 = 1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153.

    Args:
        n: A positive integer.

    Returns:
        True if n is an Armstrong number, False otherwise.
    """
    if not isinstance(n, int) or n < 0:
        return False # Armstrong numbers are typically defined for positive
integers

    # Convert number to string to get its length and access digits
    str_n = str(n)
    num_digits = len(str_n)
```

```
    sum_of_powers = 0
    for digit_char in str_n:
        digit = int(digit_char)
        sum_of_powers += (digit ** num_digits)

    return sum_of_powers == n
```

OUTPUT:

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

print("--- Armstrong Number Check (User Input) ---")
try:
    user_input = int(input("Enter an integer to check if it's an Armstrong
number: "))
    if is_armstrong_number(user_input):
        print(f"The number {user_input} is an Armstrong number.")
    else:
        print(f"The number {user_input} is NOT an Armstrong number.")
except ValueError:
    print("Invalid input. Please enter an integer.")
for num in armstrong_candidates:
    if is_armstrong_number(num):
        print(f"The number {num} is an Armstrong number.")
    else:
        print(f"The number {num} is NOT an Armstrong number.")
```

```
--- Armstrong Number Check (User Input) ---
Enter an integer to check if it's an Armstrong number: 153
The number 153 is an Armstrong number.
The number 0 is an Armstrong number.
The number 1 is an Armstrong number.
The number 153 is an Armstrong number.
The number 370 is an Armstrong number.
The number 371 is an Armstrong number.
The number 407 is an Armstrong number.
The number 1634 is an Armstrong number.
The number 8208 is an Armstrong number.
The number 9474 is an Armstrong number.
The number 123 is NOT an Armstrong number.
The number 500 is NOT an Armstrong number.
The number 1000 is NOT an Armstrong number.
```

EXPLANATION:

An Armstrong number is a positive integer that is equal to the sum of its own digits, each raised to the power of the number of digits in the number.

1. **Input Validation:** The function first checks if the input `n` is a non-negative integer.

1. **Digit Count:** It converts the number to a string to easily determine the number of digits (`num_digits`).
2. **Sum of Powers:** It then iterates through each digit of the number, converts it back to an integer, and raises it to the `num_digits` power, adding the result to `sum_of_powers`.
3. **Comparison:** Finally, it compares `sum_of_powers` with the original number `n`. If they are equal, `n` is an Armstrong number.

PROMPT:Question 4: Context-Managed Prompting (Optimized Number Classification) **bold text**

CODE:

```
def check_prime_composite_or_neither(n):
    """
    Determines if a given integer is prime, composite, or neither.

    Args:
        n: An integer.

    Returns:
        A string indicating 'Prime', 'Composite', or 'Neither'.
    """
    if n < 2:
        return "Neither (Numbers less than 2 are neither prime nor
composite)"
    elif n == 2:
        return "Prime"
    elif n % 2 == 0:
        return "Composite (Even numbers greater than 2 are composite)"
    else:
        # Check for divisibility from 3 up to the square root of n
        # We only need to check odd numbers as even numbers have already been
handled.
        i = 3
        while i * i <= n:
            if n % i == 0:
                return "Composite"
            i += 2
        return "Prime"
```

OUTPUT:

```
user_input = int(input("Enter an integer: "))
user_result = check_prime_composite_or_neither(user_input)
print(f"The number {user_input} is: {user_result}")
```

```
Enter an integer: 5
The number 5 is: Prime
```

EXPLANATION: Numbers less than or equal to 1 are classified as neither prime nor composite.

For numbers greater than 1, we check divisibility starting from 2.

The loop runs only up to $\sqrt{n}$ to reduce unnecessary checks and improve efficiency.

If any divisor is found, the number is composite.

If no divisors are found, the number is prime.

PROMPT :Question 5: Zero-Shot Prompting (Perfect Number Check)

CODE:

```
def is_perfect_number(n):
    """
    Checks if a given positive integer is a perfect number.

    A perfect number is a positive integer that is equal to the sum of its
    proper positive divisors (divisors excluding the number itself).
    For example, 6 is a perfect number because 1 + 2 + 3 = 6.

    Args:
        n: A positive integer.

    Returns:
        True if n is a perfect number, False otherwise.
    """
    if n <= 0:
        return False

    sum_of_divisors = 1 # Start with 1 because 1 is a divisor for all
positive integers

    # Check for divisors from 2 up to sqrt(n)
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            sum_of_divisors += i
            if i * i != n: # Avoid adding the same divisor twice for perfect
squares
                sum_of_divisors += (n // i)

    return sum_of_divisors == n and n != 1 # 1 is not considered a perfect
number
```

OUTPUT:

```
print("--- Perfect Number Check (User Input) ---")
try:
    user_input = int(input("Enter an integer to check if it's perfect: "))
    if is_perfect_number(user_input):
        print(f"The number {user_input} is a perfect number.")
    else:
        print(f"The number {user_input} is NOT a perfect number.")
except ValueError:
    print("Invalid input. Please enter an integer.")
```

```
--- Perfect Number Check (User Input) ---
Enter an integer to check if it's perfect: 6
The number 6 is a perfect number.
```

Explanation of Perfect Numbers:

A perfect number is a positive integer that is equal to the sum of its positive divisors, excluding the number itself. For instance:

- **6** is a perfect number because its proper divisors are 1, 2, and 3, and 1 + 2 + 3 = 6.
- **28** is a perfect number because its proper divisors are 1, 2, 4, 7, and 14, and 1 + 2 + 4 + 7 + 14 = 28.

PROMPT:Few-Shot Prompting (Even or Odd Classification with Validation) **bold text**

CODE:

```
def check_even_or_odd(n):
    """
    Determines if a given integer is even or odd.

    Args:
        n: An integer.

    Returns:
        A string indicating 'Even' or 'Odd'.
    """
    if n % 2 == 0:
        return "Even"
    else:
        return "Odd"
```

OUTPUT:

```
print("--- Even or Odd Check (User Input) ---")
try:
    user_input = int(input("Enter an integer: "))
    result = check_even_or_odd(user_input)
```

```
    print(f"The number {user_input} is {result}.")
except ValueError:
    print("Invalid input. Please enter a valid integer.")
```

```
--- Even or Odd Check (User Input) ---
Enter an integer: 34
The number 34 is Even.
```

Testing the program with negative numbers and non-integer inputs.

```
print("--- Even or Odd Check (User Input) ---")
try:
    user_input = int(input("Enter an integer: "))
    result = check_even_or_odd(user_input)
    print(f"The number {user_input} is {result}.")
except ValueError:
    print("Invalid input. Please enter a valid integer.")
```

```
--- Even or Odd Check (User Input) ---
Enter an integer: -10
The number -10 is Even.
```

EXPLANATION :

The program first validates the user input using try-except to ensure it is an integer.

It checks whether the number is divisible by 2 to determine if it is even or odd.

Based on the result, it prints an appropriate and clear message.