Lab-3.1

Course : AI Assisted Coding

Topic:- design and refine prompts using different prompting strategies to generate Python programs for basic computational problems.

Name : Mohammed Sabir

Hall Ticket No : 2303A51506

Batch : 22

Task-01

Final Optimal Prompt:

write a python program to check given number is a palindrom or not

---

Code Screenshot :

```python
import sys
def is_palindrome(n: int) -> bool:
    return str(n) == str(n)[::-1]
if __name__ == "__main__":
    try:
        n = int(input("Enter a number: "))
        if is_palindrome(n):
            print(f"{n} is a palindrome.")
        else:
            print(f"{n} is not a palindrome.")
    except ValueError:
        print("Invalid input. Please enter a valid integer.")
        sys.exit(1)
    except Exception as e:
        print(f"An error occurred: {e}")
        sys.exit(1)
```

Output Screenshot:

```
PS C:\Users\sakir\OneDrive\Desktop\Ai-Assistant> & C:\Python314\python.exe c:/Users/sakir/OneDrive/Desktop/Ai-Assistant/lab3.py
Enter a number: 123
123 is not a palindrome.
PS C:\Users\sakir\OneDrive\Desktop\Ai-Assistant>
>> ^C
PS C:\Users\sakir\OneDrive\Desktop\Ai-Assistant> & C:\Python314\python.exe c:/Users/sakir/OneDrive/Desktop/Ai-Assistant/lab3.py
Enter a number: 12121
12121 is a palindrome.
PS C:\Users\sakir\OneDrive\Desktop\Ai-Assistant>
```

Explanation/Justification/Observation (100 words / 5 – 6 sentence) :

The program checks whether a given number is a palindrome. The is_palindrome() function converts the integer into a string and compares it with its reverse using slicing ([::-1]). If both are equal, the function returns True, otherwise False. This approach is simple and efficient because it avoids complex mathematical operations. In the main block, user input is taken as an integer and validated using a try-except structure to handle invalid entries. Based on the function result, the program prints whether the number is a palindrome or not. Overall, it demonstrates string manipulation, conditional logic, and proper exception handling.

Task-02

Final Optimal Prompt:

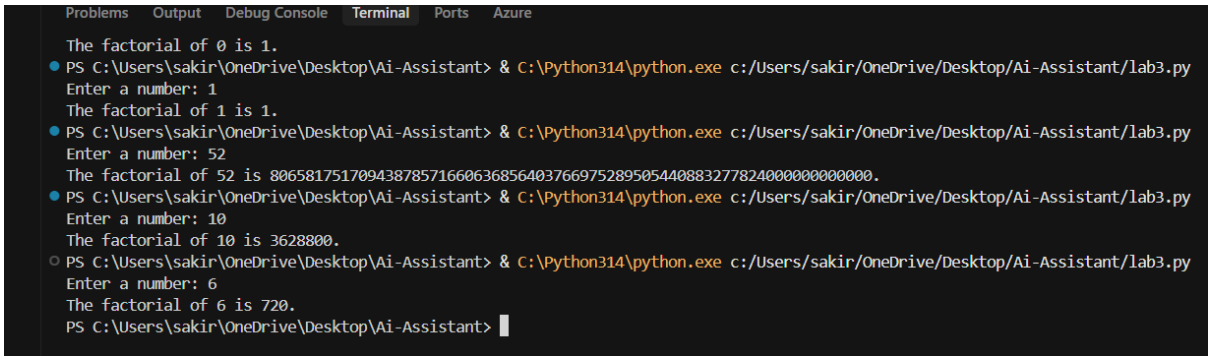
write a python program for factorial calculation

example: 5 -> 120

Code Screenshot :

```python
import sys
def factorial(n: int) -> int:
    if n < 0:
        return "Factorial is not defined for negative numbers."
    elif n == 0:
        return 1
    else:
        return n * factorial(n - 1)
if __name__ == "__main__":
    try:
        n = int(input("Enter a number: "))
        print(f"The factorial of {n} is {factorial(n)}.")
    except ValueError:
        print("Invalid input. Please enter a valid integer.")
        sys.exit(1)
    except Exception as e:
        print(f"An error occurred: {e}")
        sys.exit(1)
```

Output Screenshot:



Explanation/Justification/Observation (100 words / 5 – 6 sentence) :

The program calculates the factorial of a given number using recursion. The factorial() function calls itself repeatedly until it reaches the base case. If the number is negative, it returns a message stating that factorial is not defined. If the number is zero, it returns 1, which is the base condition. Otherwise, it multiplies the number by the factorial of the previous number (n * factorial(n - 1)). The main block takes user input and handles errors using try-except statements. Overall, the program demonstrates recursion, conditional logic, base cases, and proper exception handling in Python

---

Task-03

Final Optimal Prompt:

write a python program to whether the number is Armstron Number or not

example: 153 -> True

example: 123 -> False

example: 1634 -> True

example: 1234 -> False

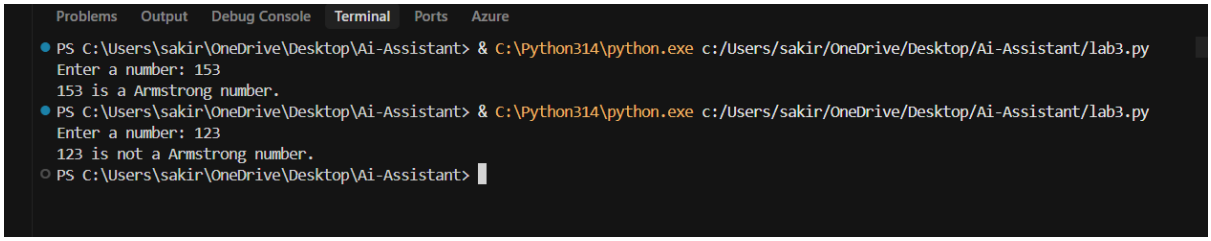
example: 1234 -> False

Code Screenshot :

```python
import sys
def is_armstrong(n: int) -> bool:
    return sum(int(digit) ** len(str(n)) for digit in str(n)) == n
if __name__ == "__main__":
    try:
        n = int(input("Enter a number: "))
        if is_armstrong(n):
            print(f"{n} is a Armstrong number.")
        else:
            print(f"{n} is not a Armstrong number.")
    except ValueError:
        print("Invalid input. Please enter a valid integer.")
        sys.exit(1)
    except Exception as e:
        print(f"An error occurred: {e}")
        sys.exit(1)

```

Output Screenshot:



Explanation/Justification/Observation (100 words / 5 – 6 sentence) :

The program checks whether a given number is an Armstrong number. The is_armstrong() function converts the number into a string to determine the number of digits. It then calculates the sum of each digit raised to the power of the total number of digits using a generator expression. If the computed sum equals the original number, the function returns True; otherwise, it returns False. The main block takes user input and handles errors using try-except statements to ensure valid integer input. Overall, the program demonstrates string conversion, mathematical operations, generator expressions, and proper exception handling.

Task-04

Final Optimal Prompt:

write a pyhton program to check the given number is a prime number, composite number, neither

handle errors when i gave negative input or string input

Code Screenshot :

```python
import sys
def is_prime(n: int) -> bool:
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True
if __name__ == "__main__":
    try:
        n = int(input("Enter a number: "))
        if is_prime(n):
            print(f"{n} is a prime number.")
        else:
            print(f"{n} is a composite number.")
    except ValueError:
        print("Invalid input. Please enter a valid integer.")
        sys.exit(1)
    except Exception as e:
        print(f"An error occurred: {e}")
        sys.exit(1)
```

Output Screenshot:

```
PS C:\Users\sakir\OneDrive\Desktop\Ai-Assistant> & C:\Python314\python.exe c:/Users/sakir/OneDrive/Desktop/Ai-Assistant/lab3.py
Enter a number: 6
6 is a compositenumber.
PS C:\Users\sakir\OneDrive\Desktop\Ai-Assistant> & C:\Python314\python.exe c:/Users/sakir/OneDrive/Desktop/Ai-Assistant/lab3.py
Enter a number: 4
4 is a composite number.
PS C:\Users\sakir\OneDrive\Desktop\Ai-Assistant> & C:\Python314\python.exe c:/Users/sakir/OneDrive/Desktop/Ai-Assistant/lab3.py
Enter a number: 11
11 is a prime number.
PS C:\Users\sakir\OneDrive\Desktop\Ai-Assistant>
```

Explanation/Justification/Observation (100 words / 5 – 6 sentence) :

The program checks whether a given number is prime or composite. The is_prime() function first verifies if the number is less than or equal to 1, returning False since such numbers are not prime. It then checks divisibility from 2 up to the square root of the number to improve efficiency. If any divisor is found, the function returns False; otherwise, it returns True. In the main block, the user enters an integer, and the program prints whether it is prime or composite. Exception handling ensures valid input. Overall, the program demonstrates optimized logic and proper error handling

---

Tack-05

Final Optimal Prompt:

write a python program to check the give number is perfect number or not

---

Code Screenshot :

```
import sys
def is_perfect(n: int) -> bool:
    return sum(i for i in range(1, n) if n % i == 0) == n
if __name__ == "__main__":
    try:
        n = int(input("Enter a number: "))
        if is_perfect(n):
            print(f"{n} is a perfect number.")
        else:
            print(f"{n} is not a perfect number.")
    except ValueError:
        print("Invalid input. Please enter a valid integer.")
        sys.exit(1)
    except Exception as e:
        print(f"An error occurred: {e}")
        sys.exit(1)
```

Output Screenshot:

```
PS C:\Users\sakir\OneDrive\Desktop\Ai-Assistant> & C:\Python314\python.exe c:/Users/sakir/OneDrive/Desktop/Ai-Assistant/lab3.py
Enter a number: 19
19 is not a perfect number.
PS C:\Users\sakir\OneDrive\Desktop\Ai-Assistant> & C:\Python314\python.exe c:/Users/sakir/OneDrive/Desktop/Ai-Assistant/lab3.py
Enter a number: 12
12 is not a perfect number.
PS C:\Users\sakir\OneDrive\Desktop\Ai-Assistant> & C:\Python314\python.exe c:/Users/sakir/OneDrive/Desktop/Ai-Assistant/lab3.py
Enter a number: 6
6 is a perfect number.
PS C:\Users\sakir\OneDrive\Desktop\Ai-Assistant>
```

Explanation/Justification/Observation (100 words / 5 – 6 sentence) :

The program checks whether a given number is a perfect number. The is_perfect() function calculates the sum of all proper divisors of the number (excluding the number itself) using a generator expression inside the sum() function. It iterates from 1 to n-1 and adds the divisors that evenly divide the number. If the total sum equals the original number, the function returns True, indicating it is a perfect number; otherwise, it returns False. The main block handles user input and exceptions properly. Overall, the program demonstrates looping, conditional logic, and efficient use of Python expressions

Tack-06

Final Optimal Prompt:

write a python program the user should give multipule inputs at a time and print the odd or even for each inputto check the given number is odd or even when i can give multipulw inputs

example1:

input:15 20 18

output:

15 is odd

20 is even

18 is even

example2:

input:1 20 18

output:

1 is odd

20 is even

18 is even

Code Screenshot :

```
142  import sys
143  def is_odd_even(n: int) -> str:
144      if n % 2 == 0:
145          return f"{n} is even"
146      else:
147          return f"{n} is odd"
148  if __name__ == "__main__":
149      try:
150          n = int(input("Enter a number: "))
151          print(is_odd_even(n))
152      except ValueError:
153          print("Invalid input. Please enter a valid integer.")
154          sys.exit(1)
155
```

Output Screenshot:

```
2 is even
PS C:\Users\sakir\OneDrive\Desktop\Ai-Assistant> & C:\Python314\python.exe c:/Users
Enter a number: 52
52 is even
PS C:\Users\sakir\OneDrive\Desktop\Ai-Assistant> & C:\Python314\python.exe c:/Users
Enter a number: 5 10
Invalid input. Please enter a valid integer.
PS C:\Users\sakir\OneDrive\Desktop\Ai-Assistant> & C:\Python314\python.exe c:/Users
Enter a number: 10
10 is even
PS C:\Users\sakir\OneDrive\Desktop\Ai-Assistant>
```

Explanation/Justification/Observation (100 words / 5 – 6 sentence) :

The program determines whether a given number is odd or even. The is_odd_even() function uses the modulus operator (%) to check divisibility by 2. If the remainder is zero, the number is even; otherwise, it is odd. The function returns a formatted string indicating the result. In the main block, the program takes integer input from the user and prints the

result returned by the function. Exception handling is implemented using a try-except block to manage invalid inputs such as non-integer values. Overall, the program demonstrates basic conditional logic and proper input validation in Python.