

ASSIGNMENT -3.1

BATCH-30

ROLL-NO:2303A510G7

NAME-SRAVANI

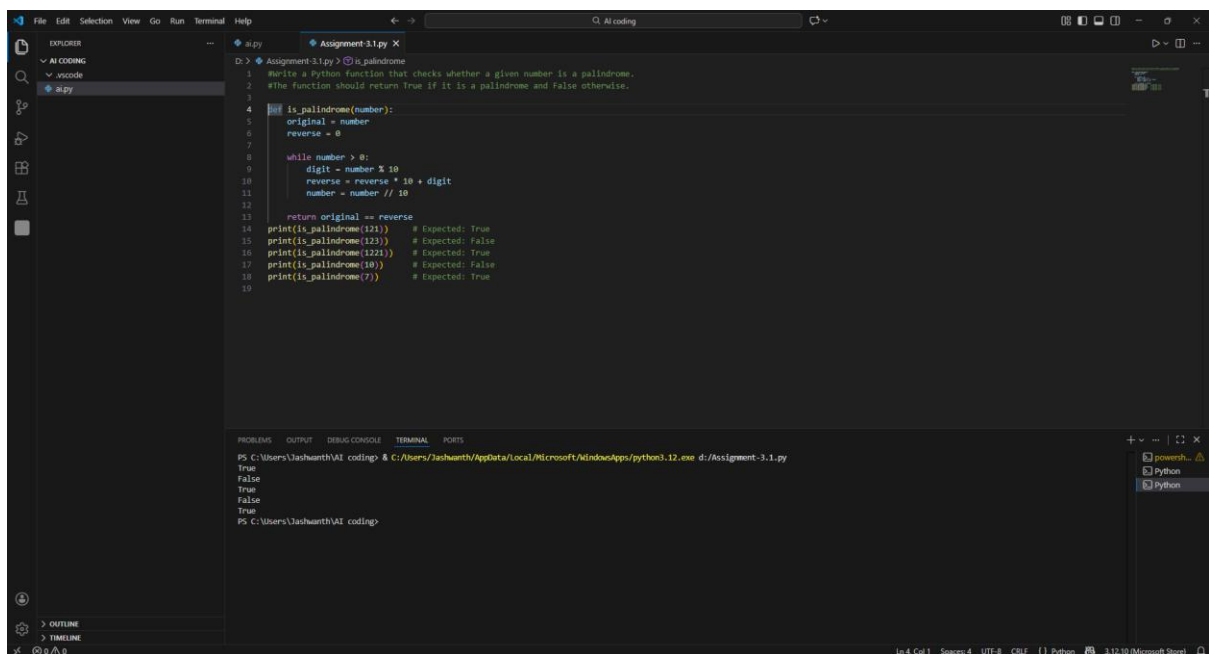
TASK-1:

ZERO-SHOT PROMPTING (PALINDROME NUMBER PROGRAM)

PROMPT:

Write a Python function that checks whether a given number is a palindrome. The function should return True if it is a palindrome and False otherwise.

CODE:



```
1 #Write a Python function that checks whether a given number is a palindrome.
2 #The function should return True if it is a palindrome and False otherwise.
3
4 def is_palindrome(number):
5     original = number
6     reverse = 0
7
8     while number > 0:
9         digit = number % 10
10        reverse = reverse * 10 + digit
11        number = number // 10
12
13    return original == reverse
14
15    print(is_palindrome(121))    # Expected: True
16    print(is_palindrome(123))    # Expected: False
17    print(is_palindrome(1221))   # Expected: True
18    print(is_palindrome(10))     # Expected: False
19    print(is_palindrome(7))      # Expected: True
20
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Jashwanth\AI coding> & C:\Users\Jashwanth\AppData\Local\Microsoft\WindowsApps\python.12.exe d:/Assignment-3.1.py
True
False
True
False
True
PS C:\Users\Jashwanth\AI coding>
```

OBSERVATION: -

The model is given only the explanation of the question -Any example or detailed explanation is not given -Answer is accurate but not specific with negative and non-integers values

TASK-2:

ONE-SHOT PROMPTING (FACTORIAL CALCULATION)

PROMPT:

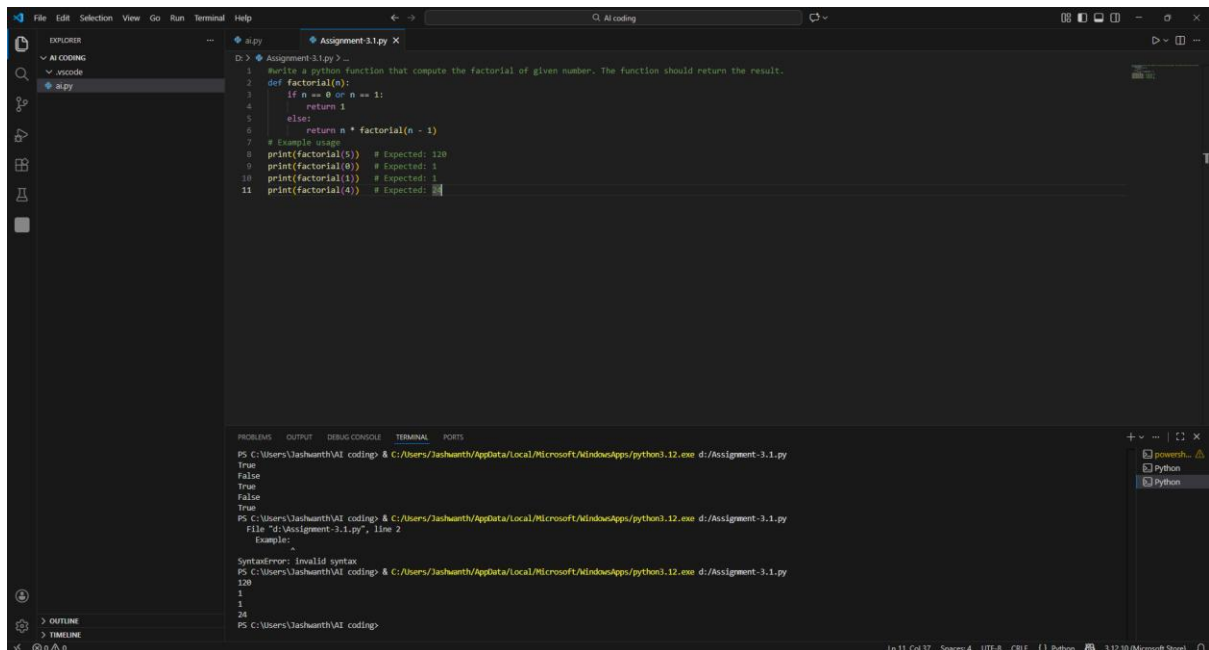
write a python function that compute the factorial of given number. The function should return the result.

Example:

Input:5

Output:120

CODE:



```
D:\> cd "D:\Assignment-3.1.py" & python3.12.exe d:/Assignment-3.1.py
1 #Write a python function that compute the factorial of given number. The function should return the result.
2 def factorial(n):
3     if n == 0 or n == 1:
4         return 1
5     else:
6         return n * factorial(n - 1)
7 # Example usage
8 print(factorial(5)) # Expected: 120
9 print(factorial(0)) # Expected: 1
10 print(factorial(1)) # Expected: 1
11 print(factorial(4)) # Expected: 24

PS C:\Users\Jashwanth\AI coding> & C:\Users\Jashwanth\AppData\Local\Microsoft\WindowsApps\python3.12.exe d:/Assignment-3.1.py
True
False
True
False
True
PS C:\Users\Jashwanth\AI coding> & C:\Users\Jashwanth\AppData\Local\Microsoft\WindowsApps\python3.12.exe d:/Assignment-3.1.py
File "d:/Assignment-3.1.py", line 2
Example:
SyntaxError: invalid syntax
PS C:\Users\Jashwanth\AI coding> & C:\Users\Jashwanth\AppData\Local\Microsoft\WindowsApps\python3.12.exe d:/Assignment-3.1.py
120
1
24
PS C:\Users\Jashwanth\AI coding>
```

OBSERVATION:

Clear understanding of the output better choice of logic-stack overflow, recursion complexity Correct handling of base case Improve code simplicity

TASK-3:

FEW-SHOT PROMPTING (ARMSTRONG NUMBER CHECK)

Prompt:

Example 1:

Input: 153

Output: Armstrong Number

Example 2:

Input: 370

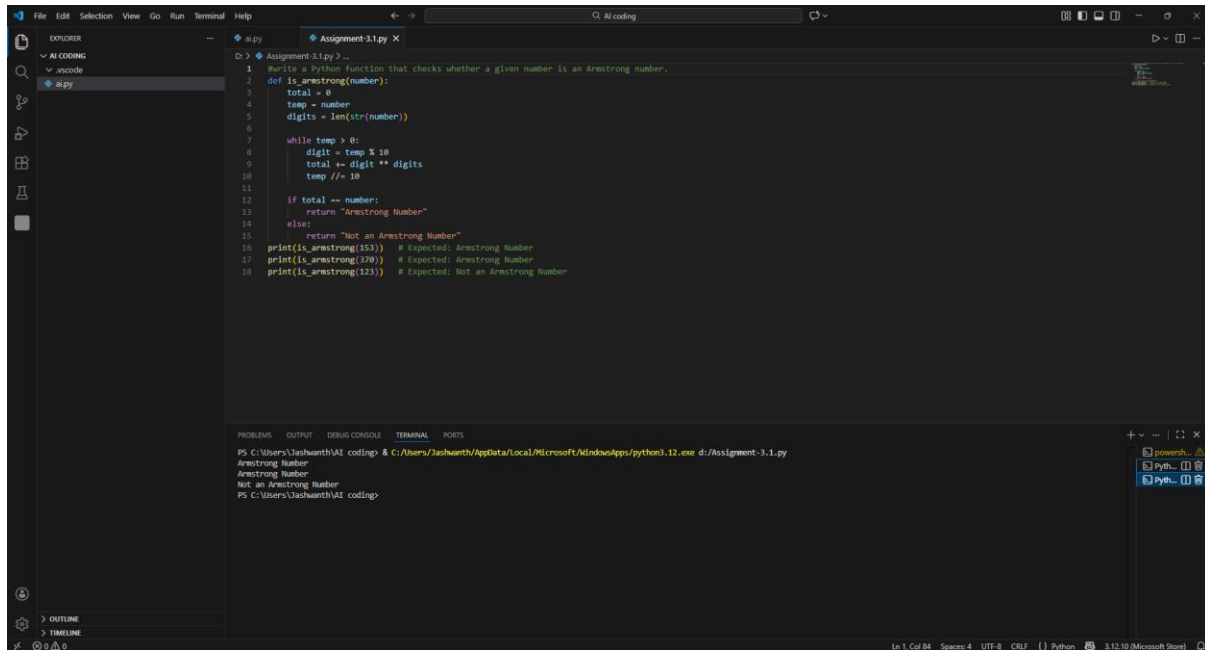
Output: Armstrong Number

Example 3:

Input: 123

Output: Not an Armstrong Number Now write a Python function that checks whether a given number is an Armstrong number. The function should return an appropriate result.

CODE:



```
D:\> cd Assignment-3.1.py > ...
1 #Write a Python function that checks whether a given number is an Armstrong number.
2 def is_armstrong(number):
3     total = 0
4     temp = number
5     digits = len(str(number))
6
7     while temp > 0:
8         digit = temp % 10
9         total += digit ** digits
10        temp //= 10
11
12    if total == number:
13        return "Armstrong Number"
14    else:
15        return "Not an Armstrong Number"
16
17    print(is_armstrong(153)) # Expected: Armstrong Number
18    print(is_armstrong(370)) # Expected: Armstrong Number
19    print(is_armstrong(123)) # Expected: Not an Armstrong Number
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Jashwanth\AI coding> & C:\Users\Jashwanth\AppData\Local\Microsoft\WindowsApps\python3.12.exe d:/Assignment-3.1.py

Armstrong Number

Armstrong Number

Not an Armstrong Number

PS C:\Users\Jashwanth\AI coding>

OBSERVATION:

Clear output formatting. structured way Correct logic selection Easy understanding of code Exact Appropriate answer Optimized and customized solution

TASK-4:

CONTEXT-MANAGED PROMPTING (OPTIMIZED NUMBER CLASSIFICATION)

PROMPT:

You are writing a Python program for number classification.

Requirements: -

Accept only integer input - Handle invalid and negative inputs properly - Classify the number as Prime, Composite, or Neither - Optimize the logic for efficiency (avoid unnecessary checks) - Return clear and user-friendly messages - Write clean and readable Python code Generate the program accordingly.

CODE:

The screenshot shows a Visual Studio Code editor window with a Python file named 'Assignment3.1.py'. The code defines a function 'classify_number(n)' that checks if a number is prime, composite, or neither. It includes comments and prints the results for numbers 2, 7, 10, 1, and 0. The terminal at the bottom shows the execution of the script, displaying the output: 'Armstrong Number', 'Not an Armstrong Number', 'Prime', 'Composite', and 'Neither Prime nor Composite'.

```
D:\> Assignment3.1.py
1 # Writing a python program for number classification.
2 def classify_number(n):
3     if not isinstance(n, int):
4         return "Invalid input"
5
6     if n <= 1:
7         return "Neither Prime nor Composite"
8
9     for i in range(2, int(n ** 0.5) + 1):
10         if n % i == 0:
11             return "Composite"
12
13     return "Prime"
14
15 print(classify_number(2))
16 print(classify_number(7))
17 print(classify_number(10))
18 print(classify_number(1))
19 print(classify_number(0))
```

Terminal Output:

```
PS C:\Users\Jashwanth\AI coding> & C:\Users\Jashwanth\AppData\Local\Microsoft\WindowsApps\python3.12.exe d:/Assignment-3.1.py
Armstrong Number
Not an Armstrong Number
Prime
Composite
Neither Prime nor Composite
PS C:\Users\Jashwanth\AI coding>
```

OBSERVATION:

The role is defined Constrains are clearly stated Efficiency and validation of the code but the inputs should be specified more clearly mentioned

TASK-5:

ZERO-SHOT PROMPTING (PERFECT NUMBER CHECK) VALIDATION)

PROMPT:

Write a Python function that checks whether a given number is a perfect number. The function should return an appropriate result.

CODE:

```
D:\> Assignment-3.1.py X
1 # Write a Python function that checks whether a given number is a perfect number.
2 def is_perfect(number):
3     total = 0
4
5     for i in range(1, number):
6         if number % i == 0:
7             total += i
8
9     if total == number:
10        return "Perfect Number"
11    else:
12        return "Not a Perfect Number"
13
14 print(is_perfect(6))    # 1 + 2 + 3 = 6
15 print(is_perfect(28))  # 1 + 2 + 4 + 7 + 14 =
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Jashwanth\AI coding> & C:\Users\Jashwanth\AppData\Local\Microsoft\WindowsApps\python3.12.exe d:/Assignment-3.1.py
Armstrong Number
Armstrong Number
Not an Armstrong Number
PS C:\Users\Jashwanth\AI coding> & C:\Users\Jashwanth\AppData\Local\Microsoft\WindowsApps\python3.12.exe d:/Assignment-3.1.py
Prime
Composite
Neither Prime nor Composite
Neither Prime nor Composite
PS C:\Users\Jashwanth\AI coding> & C:\Users\Jashwanth\AppData\Local\Microsoft\WindowsApps\python3.12.exe d:/Assignment-3.1.py
Perfect Number
Perfect Number
PS C:\Users\Jashwanth\AI coding>

OBSERVATION:

No input validation – if negative or float any. Inefficient for large input Did not specify input constraints No edge case handling seen

TASK-6:

FEW-SHOT PROMPTING (EVEN OR ODD CLASSIFICATION WITH VALIDATION)

PROMPT:

Example 1:

Input: 8

Output: Even

Example 2:

Input: 15

Output: Odd

Example 3:

Input: 0

Output: Even Now write a Python program that determines whether a given number is Even or Odd. The program should include proper input validation and return clear messages.

CODE:

The image shows a Visual Studio Code editor window with a Python file named `Assignment 3.1.py`. The code defines a function `check_even_odd` that takes a value and an integer `int` as arguments. It checks if the value is an instance of `int`; if not, it returns "Invalid input". If it is an integer, it checks if the value is even (using `value % 2 == 0`) and returns "Even" or "Odd" accordingly. The script then calls this function with inputs 8, 15, and 0, printing the results.

```
D:\> Assignment 3.1.py
1 Write a Python program that determines whether a given number is Even or Odd
2 def check_even_odd(value, int):
3     if not isinstance(value, int):
4         return "Invalid input"
5
6     if value % 2 == 0:
7         return "Even"
8     else:
9         return "Odd"
10 print(check_even_odd(8))
11 print(check_even_odd(15))
12 print(check_even_odd(0))
13
```

The terminal output shows the execution of the script, displaying the results for each input: Armstrong Number, Not an Armstrong Number, Prime, Composite, Neither Prime nor Composite, Neither Prime nor Composite, Perfect Number, Perfect Number, Even, Odd, and Even.

```
PS C:\Users\Jashwanth\AI coding> & C:/Users/Jashwanth/AppData/Local/Microsoft/WindowsApps/python3.12.exe d:/Assignment-3.1.py
Armstrong Number
Not an Armstrong Number
PS C:\Users\Jashwanth\AI coding> & C:/Users/Jashwanth/AppData/Local/Microsoft/WindowsApps/python3.12.exe d:/Assignment-3.1.py
Prime
Composite
Neither Prime nor Composite
Neither Prime nor Composite
PS C:\Users\Jashwanth\AI coding> & C:/Users/Jashwanth/AppData/Local/Microsoft/WindowsApps/python3.12.exe d:/Assignment-3.1.py
Perfect Number
Perfect Number
PS C:\Users\Jashwanth\AI coding> & C:/Users/Jashwanth/AppData/Local/Microsoft/WindowsApps/python3.12.exe d:/Assignment-3.1.py
Even
Odd
Even
PS C:\Users\Jashwanth\AI coding>
```

OBSERVATION:

Negative integer are handled correctly Program safely rejected non integer inputs
Improve input handling Clear and consistent output