

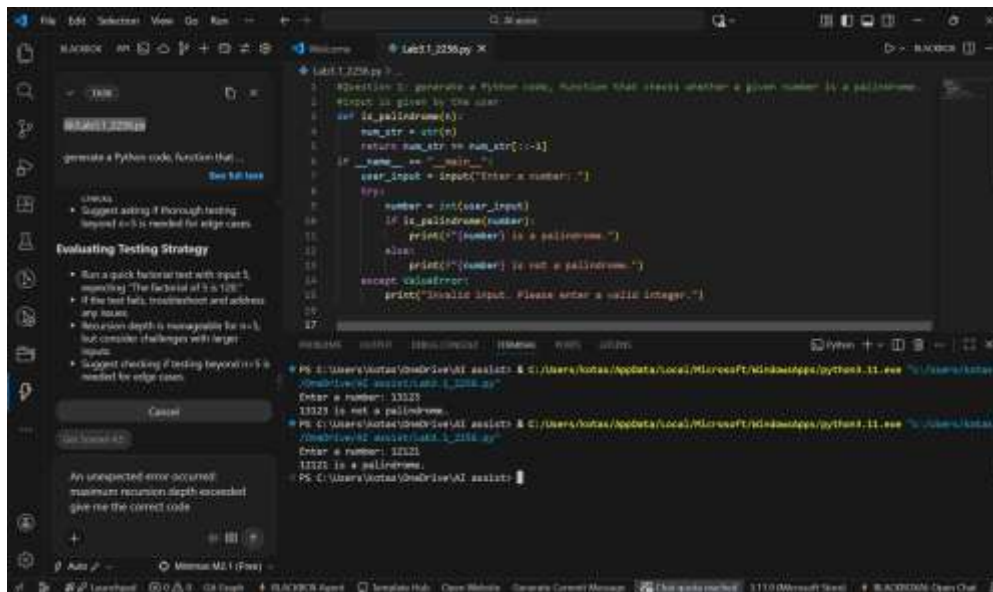
Lab Assignment- 3.1

AI Assisted Coding

A Harshita – 2303A52211

Question1: Zero-Shot Prompting (Palindrome Number Program)

Prompt: generate a Python code, function that checks whether a given number is a palindrome. And input is given by the user

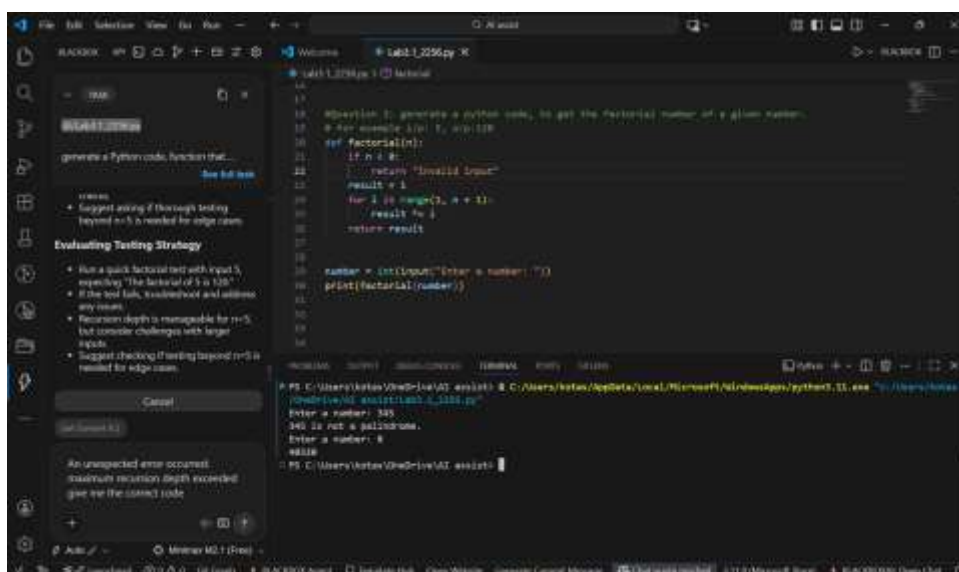


```
1. Question 1: generate a Python code, function that checks whether a given number is a palindrome.
2. Input is given by the user
3. def is_palindrome(n):
4.     num_str = str(n)
5.     return num_str == num_str[::-1]
6. if __name__ == "__main__":
7.     user_input = input("Enter a number: ")
8.     try:
9.         number = int(user_input)
10.        if is_palindrome(number):
11.            print(f"{number} is a palindrome.")
12.        else:
13.            print(f"{number} is not a palindrome.")
14.    except ValueError:
15.        print("Invalid input. Please enter a valid integer.")
16.
17.
18.
19.
20.
21.
22.
23.
24.
25.
26.
27.
28.
29.
30.
31.
32.
33.
34.
35.
36.
37.
38.
39.
40.
41.
42.
43.
44.
45.
46.
47.
48.
49.
50.
51.
52.
53.
54.
55.
56.
57.
58.
59.
60.
61.
62.
63.
64.
65.
66.
67.
68.
69.
70.
71.
72.
73.
74.
75.
76.
77.
78.
79.
80.
81.
82.
83.
84.
85.
86.
87.
88.
89.
90.
91.
92.
93.
94.
95.
96.
97.
98.
99.
100.
```

- It generates correct logic but does not handle negative numbers explicitly.\

Question 2: One-Shot Prompting (Factorial Calculation)

Prompt: generate a python code, to get the factorial number of a given number. for example, i/p: 5, o/p:120



```
1. Question 1: generate a python code, to get the factorial number of a given number.
2. For example i/p: 5, o/p:120
3. def factorial(n):
4.     if n < 0:
5.         return "Invalid Input"
6.     result = 1
7.     for i in range(1, n + 1):
8.         result *= i
9.     return result
10.
11.
12.
13.
14.
15.
16.
17.
18.
19.
20.
21.
22.
23.
24.
25.
26.
27.
28.
29.
30.
31.
32.
33.
34.
35.
36.
37.
38.
39.
40.
41.
42.
43.
44.
45.
46.
47.
48.
49.
50.
51.
52.
53.
54.
55.
56.
57.
58.
59.
60.
61.
62.
63.
64.
65.
66.
67.
68.
69.
70.
71.
72.
73.
74.
75.
76.
77.
78.
79.
80.
81.
82.
83.
84.
85.
86.
87.
88.
89.
90.
91.
92.
93.
94.
95.
96.
97.
98.
99.
100.
```

- When compared to zero- short prompting this prompt improves correctness and handles edge cases like zero and negative values.

Question 3: Few-Shot Prompting (Armstrong Number Check)

Prompt: Generate a Python code to check whether a number is an Armstrong number.

Examples: i/p: 153, o/p: Armstrong Number

i/p: 370, o/p: Armstrong Number

i/p: 123, o/p: Not an Armstrong Number

```
28 #Question 3: Write a Python code to check whether a number is an Armstrong number.
29 #Example: i/p: 153, o/p: Armstrong Number
30 # i/p: 370, o/p: Armstrong Number
31 # i/p: 123, o/p: Not an Armstrong Number
32 number = int(input("Enter a number: "))
33 print(factorial(number))
34
35 def is_armstrong(num):
36     digits = str(num)
37     power = len(digits)
38     total = 0
39
40     for d in digits:
41         total += int(d) ** power
42
43     return total == num
44
45 number = int(input("Enter a number: "))
46 if is_armstrong(number):
47     print("Armstrong Number")
48 else:
49     print("Not an Armstrong Number")
```

- Few-shot prompting improves structural accuracy and produces organized logic.

Question 4: Context-Managed Prompting (Optimized Number Classification)

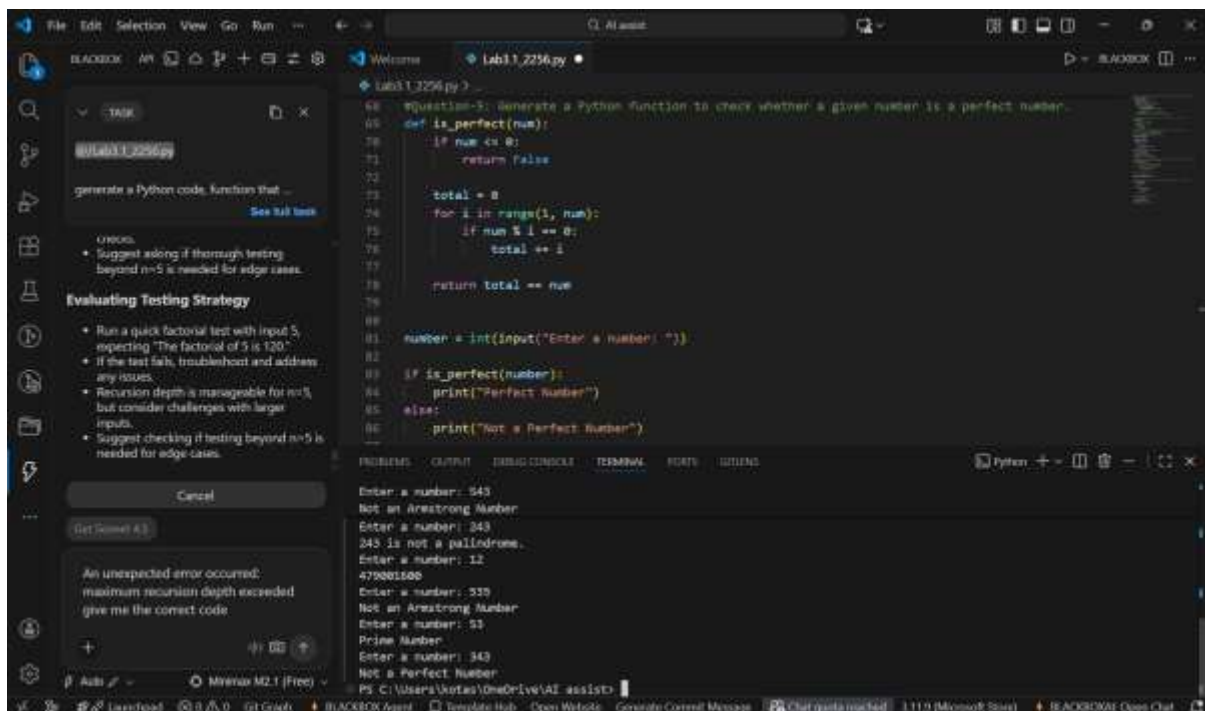
Prompt: Write an optimized Python program to classify a number as prime, composite, or neither. Ensure input validation and efficient logic.

```
35 #Question 4: Write an optimized Python program to classify a number as prime, composite, or neither.
36 #Ensure input validation and efficient logic.
37
38 def classify_number(num):
39     if num <= 1:
40         return "Neither Prime nor Composite"
41
42     for i in range(2, int(num ** 0.5) + 1):
43         if num % i == 0:
44             return "Composite Number"
45
46     return "Prime Number"
47
48 number = int(input("Enter a number: "))
49 print(classify_number(number))
```

- Context-managed prompts produce optimized and validation-aware solutions suitable for real-world applications.

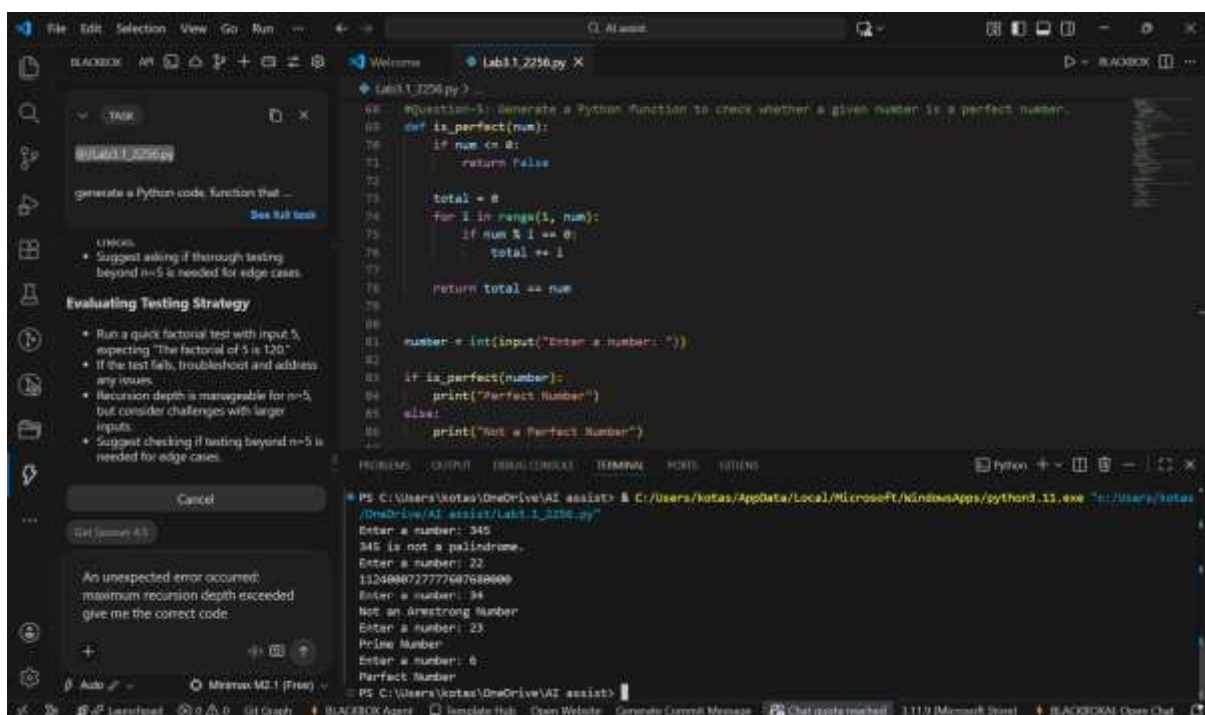
Question 5: Zero-Shot Prompting (Perfect Number Check)

Prompt: Generate a Python function to check whether a given number is a perfect number.



```
68 #Question-5: Generate a Python function to check whether a given number is a perfect number.
69 def is_perfect(num):
70     if num <= 0:
71         return False
72
73     total = 0
74     for i in range(1, num):
75         if num % i == 0:
76             total += i
77
78     return total == num
79
80
81 number = int(input("Enter a number: "))
82
83 if is_perfect(number):
84     print("Perfect Number")
85 else:
86     print("Not a Perfect Number")
```

Enter a number: 543
Not an Armstrong Number
Enter a number: 343
343 is not a palindrome.
Enter a number: 12
479001600
Enter a number: 339
Not an Armstrong Number
Enter a number: 53
Prime Number
Enter a number: 343
Not a Perfect Number



```
68 #Question-5: Generate a Python function to check whether a given number is a perfect number.
69 def is_perfect(num):
70     if num <= 0:
71         return False
72
73     total = 0
74     for i in range(1, num):
75         if num % i == 0:
76             total += i
77
78     return total == num
79
80
81 number = int(input("Enter a number: "))
82
83 if is_perfect(number):
84     print("Perfect Number")
85 else:
86     print("Not a Perfect Number")
```

Enter a number: 345
345 is not a palindrome.
Enter a number: 22
1120000727776768000
Enter a number: 34
Not an Armstrong Number
Enter a number: 23
Prime Number
Enter a number: 6
Perfect Number

- Zero-shot prompting works but is less optimized due to unnecessary full-range iteration.

Question 6: Few-Shot Prompting (Even or Odd Classification with Validation)

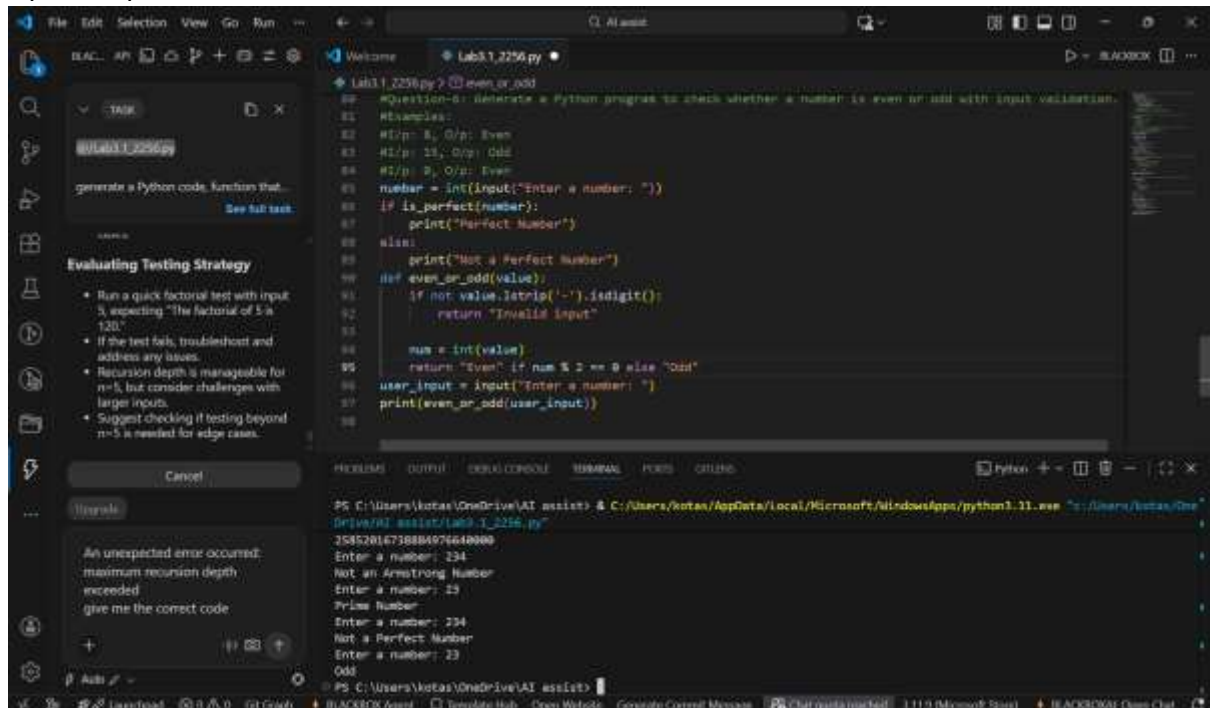
Prompt: Generate a Python program to check whether a number is even or odd with input validation.

Examples:

I/p: 8, O/p: Even

I/p: 15, O/p: Odd

I/p: 0, O/p: Even



```
#Question-6: Generate a Python program to check whether a number is even or odd with input validation.
#Examples:
#I/p: 8, O/p: Even
#I/p: 15, O/p: Odd
#I/p: 0, O/p: Even
number = int(input("Enter a number: "))
if is_perfect(number):
    print("Perfect Number")
else:
    print("Not a Perfect Number")
def even_or_odd(value):
    if not value.lstrip('-').isdigit():
        return "Invalid Input"
    num = int(value)
    return "Even" if num % 2 == 0 else "Odd"
user_input = input("Enter a number: ")
print(even_or_odd(user_input))
```

PS C:\Users\kotan\OneDrive\AI assist> C:\Users\kotan\AppData\Local\Microsoft\WindowsApps\python1.11.exe "C:\Users\kotan\OneDrive\AI assist\lab3_1_2256.py"

3585281673888497649000

Enter a number: 134

Not an Armstrong Number

Enter a number: 23

Prime Number

Enter a number: 234

Not a Perfect Number

Enter a number: 23

Odd

- Few-shot prompting significantly improves the quality of AI-generated code by providing clear input–output examples.
- The generated program handles input validation effectively, correctly classifies even and odd numbers, and manages negative and non-integer inputs more reliably compared to zero-shot prompting.