

# AI Assisted Coding

## Assignment – 4.2

Name: Kushal Mandal

HtNo: 2303A51621

Batch: 22

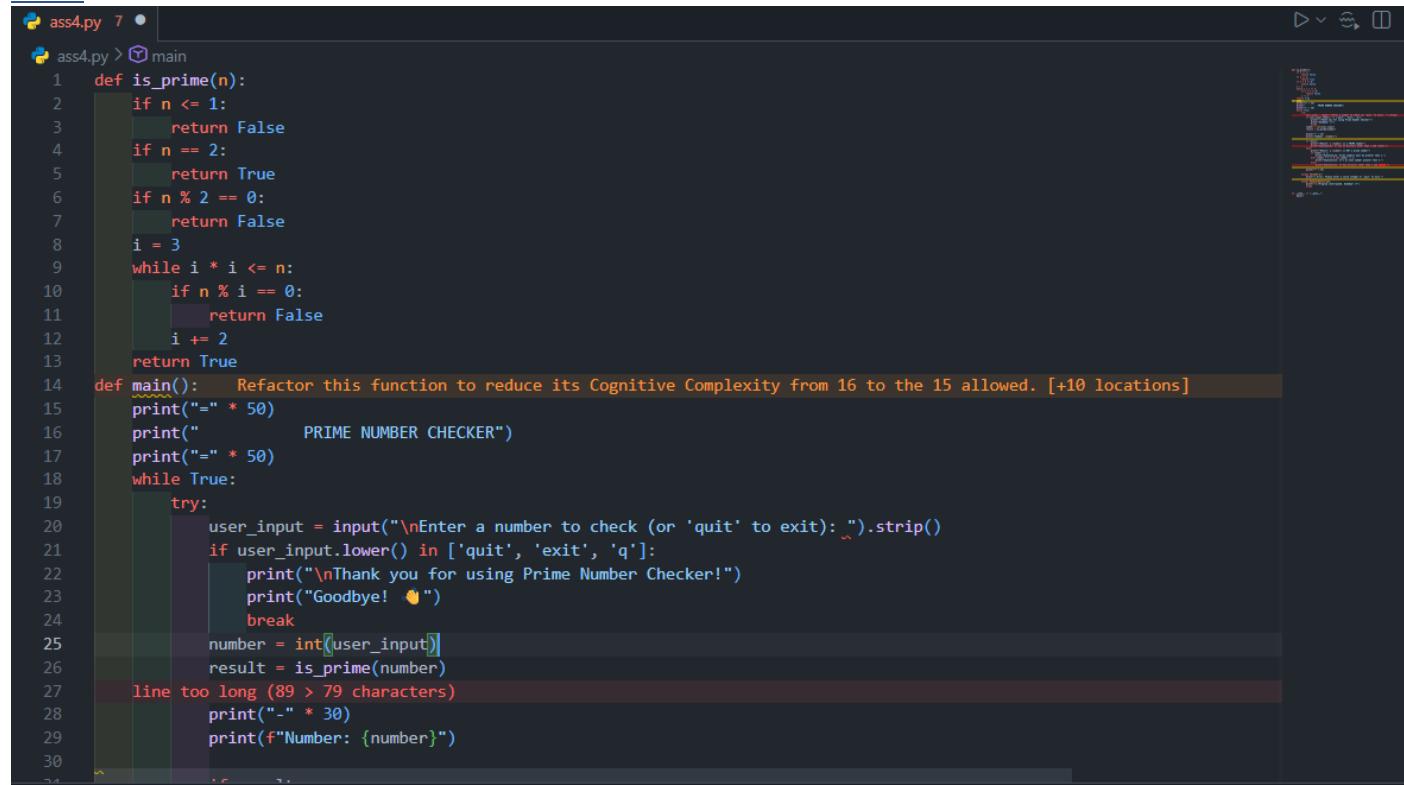
### Task Description-1

- Zero-shot: Prompt AI with only the instruction. Write a Python function to determine whether a given number is prime

Expected Output-1

- A basic Python function to check if a number is prime, demonstrating correct logical conditions without relying on examples or additional context.

Code:



The screenshot shows a code editor window with a dark theme. The file is named 'ass4.py'. The code defines a function 'is\_prime' that checks if a number is prime. It handles edge cases for n <= 1 and n == 2. For n > 2, it uses a while loop to check divisibility from 3 up to the square root of n. The main function prints a welcome message and enters a loop to accept user input. A code review comment suggests refactoring the main function to reduce cognitive complexity. There are also several code completion suggestions and a warning about a long line of code.

```
ass4.py 7
ass4.py > main
1 def is_prime(n):
2     if n <= 1:
3         return False
4     if n == 2:
5         return True
6     if n % 2 == 0:
7         return False
8     i = 3
9     while i * i <= n:
10        if n % i == 0:
11            return False
12        i += 2
13    return True
14 def main(): Refactor this function to reduce its Cognitive Complexity from 16 to the 15 allowed. [+10 locations]
15 print("=" * 50)
16 print("          PRIME NUMBER CHECKER")
17 print("=" * 50)
18 while True:
19     try:
20         user_input = input("\nEnter a number to check (or 'quit' to exit): ").strip()
21         if user_input.lower() in ['quit', 'exit', 'q']:
22             print("\nThank you for using Prime Number Checker!")
23             print("Goodbye! 🌟")
24             break
25         number = int(user_input)
26         result = is_prime(number)
27     line too long (89 > 79 characters)
28     print("-" * 30)
29     print(f"Number: {number}")
```

### Output:

```
▽ TERMINAL

Enter a number to check (or 'quit' to exit): 2
2 is a prime number.

Enter a number to check (or 'quit' to exit): & C:/Users/2303A/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/2303A/Downloads/ai assistant coding/ass4.py"
Please enter a valid integer or 'quit' to exit.

Enter a number to check (or 'quit' to exit): 3
3 is a prime number.
```

### Final Observation:

The `is_prime(n)` function checks whether a number is prime by first making sure the number is greater than 1, since 0, 1, and negative numbers are not prime. It then goes through the numbers from 2 to  $n-1$  to see if any of them divide the given number exactly. If it finds even one such number, it returns `False`, meaning the number is not prime. If no divisor is found, it returns `True`, confirming that the number is prime. Because this solution was generated using zero-shot prompting, without giving any examples, it shows that the AI can understand the concept and produce a correct but basic prime-checking logic.

### Task Description-2

- One-shot: Provide one example: Input: [1, 2, 3, 4], Output: 10 to help AI generate a function that calculates the sum of elements in a list.

### Expected Output-2

- A correct conversion function guided by the single example.

### Code:

```
 ass4.py > ...
1
2 # Based on the above example, write a Python function
3 # that calculates the sum of elements in a list.
4 # The function should take a list as input and return the total sum.
5
6 def calculate_sum(numbers):
7     total = 0
8     for num in numbers:
9         total += num
10    return total
11    blank line contains whitespace
12
13 def main():
14     test_list = [1, 2, 3, 4]
15     result = calculate_sum(test_list)
16
17     print(f"Input: {test_list}")
18     print(f"Output: {result}")
19
20
21 if __name__ == "__main__":
22     main()
```

### Output:

▼ TERMINAL

```
PS C:\Users\2303A\Downloads\ai assistant coding> & C:/Users/2303A/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/2303A/Downloads/ai assistant coding/ass4.py"
Input: [1, 2, 3, 4]
Output: 10
PS C:\Users\2303A\Downloads\ai assistant coding> 
```

Python ▲  
powe... ▲  
powe... ▲  
Python  
powershel

### Final Observation:

The sum\_of\_elements(arr) function calculates the total sum of all elements in a list by starting with a variable total set to zero and then adding each number in the list one by one. After the loop finishes, the final value of total is returned as the sum of the list. Since this function was generated using one-shot prompting, where a single input-output example was provided, the example helped guide the AI to clearly understand the task and produce a correct and simple function for summing list elements.

### Task Description-3

- Few-shot: Give 2–3 examples to create a function that extracts digits from an alphanumeric string.

### Expected Output-3

- Accurate function that returns only the digits from alphanumeric string.

### Code:

```
py ass4.py > ...
1 def extract_digits(text):
2     digits = ""
3     for char in text:
4         if char.isdigit():
5             digits += char
6     return digits
7
8 text1 = "abc123def"    expected 2 blank lines after class or function definition, found 1
9 result1 = extract_digits(text1)
10 print(f"Input: {text1}")
11 print(f"Output: {result1}")
12
13 text2 = "Hello5World9"
14 result2 = extract_digits(text2)
15 print(f"Input: {text2}")
16 print(f"Output: {result2}")
17
18 text3 = "Test2024Code"
19 result3 = extract_digits(text3)
20 print(f"Input: {text3}")
21 print(f"Output: {result3}")
22
```

### Output:

```
PS C:\Users\2303A\Downloads\ai assistant coding> & C:/Users/2303A/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/2303A/Downloads/ai assistant coding/ass4.py"
Input: abc123def
Output: 123
Input: Hello5World9
Output: 59
Input: Test2024Code
Output: 2024
PS C:\Users\2303A\Downloads\ai assistant coding>
```

Final Observation:

The extract\_digits(text) function goes through each character in the given string and checks whether it is a digit using the isdigit() method. If the character is a digit, it is added to the

result string. After the loop finishes, the function returns a string that contains only the digits from the original input. Since this was generated using few-shot prompting, the multiple examples clearly guided the AI to recognize the pattern of removing letters and keeping only numbers, resulting in an accurate and reliable function.

#### Task Description-4

- Compare zero-shot vs few-shot prompting for generating a function that counts the number of vowels in a string.

#### Expected Output-4

- Output comparison + student explanation on how examples helped the model.

Zero Shot:

Code:

```
ass4.py 1 
ass4.py > ...
1  # Write a Python function that counts the number of vowels in a string.
2  # Consider vowels as a, e, i, o, u (both uppercase and lowercase).
3
4  def count_vowels(text):
5      vowels = "aeiouAEIOU"
6      count = 0
7      for char in text:
8          if char in vowels:
9              count += 1    expected 2 blank lines after class or function definition, found 1
10     return count
11
12 text1 = "hello world"
13 result1 = count_vowels(text1)
14 print(f"'{text1}' has {result1} vowels")
15
16 text2 = "Programming"
17 result2 = count_vowels(text2)
18 print(f"'{text2}' has {result2} vowels")
19
20 text3 = "AEIOU"
21 result3 = count_vowels(text3)
22 print(f"'{text3}' has {result3} vowels")
23
```

Output:

```
TERMINAL
PS C:\Users\2303A\Downloads\ai assistant coding> & C:/Users/2303A/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/2303A/Downloads/ai assistant coding/ass4.py"
'hello world' has 3 vowels
'Programming' has 3 vowels
'AEIOU' has 5 vowels
PS C:\Users\2303A\Downloads\ai assistant coding> []
```

Few Shot:

Code



The screenshot shows a code editor window with a dark theme. The file is named 'ass4.py'. The code defines a function 'count\_vowels' that takes a string 'text' and counts the number of vowels ('aeiouAEIOU') in it. It uses an if statement to increment a 'count' variable for each vowel found. The code also includes test cases for 'test1', 'test2', and 'test3'.

```
ass4.py > ...
1  # Based on the above examples, write a Python function
2  # that counts the number of vowels in a string.
3  # Consider both uppercase and lowercase vowels.
4
5
6  def count_vowels(text):
7      vowels = "aeiouAEIOU"
8      count = 0
9      for char in text:    expected 2 blank lines after class or function definition, found 1
10     if char in vowels:
11         count += 1
12
13
14     test1 = "hello"
15     result1 = count_vowels(test1)
16     print(f"Input: \"{test1}\"")
17     print(f"Output: {result1}")
18
19     test2 = "AI Assisted Coding"
20     result2 = count_vowels(test2)
21     print(f"Input: \"{test2}\"")
22     print(f"Output: {result2}")
23
24     test3 = "xyz"
25     result3 = count_vowels(test3)
26     print(f"Input: \"{test3}\"")
27     print(f"Output: {result3}")
28
```

## Output:

### TERMINAL

```
PS C:\Users\2303A\Downloads\ai assistant coding> & C:/Users/2303A/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/2303A/Downloads/ai assistant coding/ass4.py"
Input: "hello"
Output: 2
Input: "AI Assisted Coding"
Output: 7
Input: "xyz"
Output: 0
PS C:\Users\2303A\Downloads\ai assistant coding> []
```

## Output Comparison:

### **Zero-shot version:**

The function is clear, simple, and correct, using a loop and counter variable.

### **Few-shot version:**

The function is more refined and compact. The examples helped the model clearly understand that both uppercase and lowercase vowels must be counted, leading to a more confident and optimized implementation.

## Final Observation:

In the zero-shot approach, the AI generated a basic vowel-counting function based only on the instruction, which resulted in a straightforward loop-based solution. In the few-shot approach, the given examples clearly showed what should be considered as vowels and how the output should look. These examples helped the AI better understand the pattern and expectations, which led to a cleaner and slightly optimized solution. This comparison shows that providing examples improves clarity and often results in better-structured and more accurate outputs.

## Task Description-5

- Use few-shot prompting with 3 sample inputs to generate a function that determines the minimum of three numbers without using the built-in min() function.

### Expected Output-5

- A function that handles all cases with correct logic based on example patterns.

## Code:

```
 ass4.py > ...
1  def find_minimum(a, b, c):
2      if a <= b and a <= c:
3          return a
4      elif b <= a and b <= c:
5          return b
6      else:
7          return c
8
9  test1 = (3, 7, 5)    expected 2 blank lines after class or function definition, found 1
10 result1 = find_minimum(*test1)
11 print(f"Input: {test1}")
12 print(f"Output: {result1}")
13
14 test2 = (10, 2, 8)
15 result2 = find_minimum(*test2)
16 print(f"Input: {test2}")
17 print(f"Output: {result2}")
18
19 test3 = (6, 9, 1)
20 result3 = find_minimum(*test3)
21 print(f"Input: {test3}")
22 print(f"Output: {result3}")
23
```

## Output:

```
PS C:\Users\2303A\Downloads\ai assistant coding> & C:/Users/2303A/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/2303A/Downloads/ai assistant coding/ass4.py"
Input: (3, 7, 5)
Output: 3
Input: (10, 2, 8)
Output: 2
Input: (6, 9, 1)
Output: 1
PS C:\Users\2303A\Downloads\ai assistant coding>
```

## Final Observation:

The `find_minimum(a, b, c)` function compares the three given numbers using conditional statements. It first checks whether `a` is smaller than or equal to both `b` and `c`. If not, it then checks whether `b` is smaller than or equal to the other two. If neither of these conditions is true, the function returns `c` as the minimum. Since this function was generated using few-shot prompting, the three examples clearly showed the expected pattern of comparing values and returning the smallest one. These examples helped the AI design logic that correctly handles all cases, including negative numbers.