

ASSIGNMENT – 3.1

CH.SANHITH REDDY

2303A52251

B-43

Question 1:

Zero-Shot Prompting (Palindrome Number Program)

Write a zero-shot prompt (without providing any examples) to generate a

Python function that checks whether a given number is a palindrome.

Task:

- Record the AI-generated code.
- Test the code with multiple inputs.
- Identify any logical errors or missing edge-case handling.

PROMPT:

checking whether given number it is palindrome or not

CODE:

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows files 1.4-AI-ASS.py and AI-ASS.py.
- Code Editor:** A central window titled "Q. AI Assisted Code" displays Python code for a palindrome checker. The code includes a docstring asking for a zero-shot prompt to generate a function that checks if a number is a palindrome. It defines a function `is_palindrome(n)` that handles negative numbers, converts the number to a string, and compares it with its reverse. It also includes a test case section with various numbers and their expected outcomes.
- Right Panel:** "RECENT SESSIONS" shows three completed tasks related to code optimization.
- Bottom Status Bar:** Shows file path (AI-Assisted Code), line count (Ln 1, Col 22), spaces (Spaces 4), encoding (UTF-8), and date/time (13:37 12-01-2026).

The screenshot shows an AI Assisted Code IDE interface. The left sidebar displays a project structure with files: AI-AssPy (the current file), 14-AI-ASS.py, and 14-AI-ASS.py. The main editor area contains the following Python code:

```
def is_palindrome(n):
    # Handle negative numbers - they are not palindromes
    if n < 0:
        return False

    # Convert number to string and compare with its reverse
    num_str = str(n)
    return num_str == num_str[::-1]

# Test cases
if __name__ == "__main__":
    test_cases = [
        123,           # True - palindrome
        -1234,         # False - negative number
        0,             # True - single digit
        101,           # False - not a palindrome
        1001,          # True - palindrome
        12321,         # True - palindrome
        123,           # False - not a palindrome
        9,             # True - single digit
        100,           # False - not a palindrome
    ]

    for num in test_cases:
        result = is_palindrome(num)
        print(f"Is {num} a palindrome? {result}")
```

The bottom terminal window shows the execution results:

```
PS C:\Users\adapala\vanishi.krish\Desktop\AI Assisted Code> & "C:/ikors/adapala vanishi krish/AppData/local/Programs/Python/Python313/python.exe" "c:/users/adapala\vanishi krish\Desktop\AI Assisted Code\14-AI-ASS.py"
Is 123 a palindrome? True
Is -1234 a palindrome? False
Is 0 a palindrome? True
Is 101 a palindrome? False
Is 1001 a palindrome? True
Is 12321 a palindrome? True
Is 123 a palindrome? False
Is 9 a palindrome? True
Is 100 a palindrome? False
PS C:\Users\adapala\vanishi.krish\Desktop\AI Assisted Code>
```

Question 2:

One-Shot Prompting (Factorial Calculation)

Write a one-shot prompt by providing one input-output example and ask

the AI to generate a Python function to compute the factorial of a given number.

Example: Input: 5 →

Output: 120

Task:

- Compare the generated code with a zero-shot solution.
 - Examine improvements in clarity and correctness. **PROMPT:** Factorial Calculation giving some instructions input data

CODE:

The screenshot shows the AI Assisted Code extension integrated into the Visual Studio Code interface. The main area displays Python code for calculating factorials, with AI-generated functions and comments. A sidebar on the right lists recent sessions, and a floating panel on the right provides AI assistance.

```
File Edit Selection View Go Run Terminal Help < > AI Assisted Code

EXPLORER AI ASSISTED CODE 1.4-AI-ASS.py 1.4-AI-ASS.py
AI-ASS.py 1.4-AI-ASS.py
AI-ASS.py

given number.
Example:
Input: 5 -> Output: 120
task:
🔗 Add Context... Keep undo 🗑️ Auto

1 def factorial(n):
2     """
3         Calculate the factorial of a number.
4         One-shot generated function based on the example: 5 -> 120
5     """
6
7     if n < 0:
8         raise ValueError("Factorial is not defined for negative numbers")
9     if n == 0 or n == 1:
10        return 1
11    result = 1
12    for i in range(2, n + 1):
13        result *= i
14    return result
15
16
17 # ZERO-SHOT SOLUTION (no example)
18
19 # "Generate a Python function to compute the Factorial"
20
21 def factorial_zero_shot(n):
22     """
23         Zero-shot generated function (without example guidance)
24     """
25     if n < 0:
26         raise ValueError("Factorial is not defined for negative numbers")
27     if n == 0:
28         return 1
29     return n * factorial_zero_shot(n - 1)
30
31
32 # TEST & COMPARISON
33
34 if __name__ == "__main__":
35     test_value = 5
36
37     print("One-Shot Result: [factorial(test_value)]") # Output: 120
38     print("Zero-Shot Result: [factorial_zero_shot(test_value)]") # Output: 120
39
40     print("\n--- Comparison ---")

Ask about your code
AI responses may be inaccurate. See Agent. Initiations to onboard AI onto your codebase.

File x AI-ASS.py Explore and understand...
Search Search
27°C Mostly sunny
Un 28, Col 1 Spaces: 4 UTF-8 Python 3.13.7 8 Go Live ENG IN 1347 12/10/2024
```

The screenshot shows the AI Assisted Code extension integrated into the Visual Studio Code interface. The top navigation bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a back/forward button. The left sidebar has sections for Explorer, AI ASSISTED CODE, and RECENT SESSIONS. The main editor area displays Python code for calculating factorials, with AI-generated code completion shown in green. The bottom status bar shows file paths, file statistics like line count, and system information.

```
File Edit Selection View Go Run Terminal Help < > AI Assisted Code

EXPLORER ... AI-Assist.py 1d-AI-Assist.py
AI ASSISTED CODE AI-Assist.py > factorial
AI-Assist.py

6     if n < 0:
7         raise ValueError("Factorial is not defined for negative numbers")
8     if n == 0 or n == 1:
9         result = 1
10    else:
11        result = 1
12        for i in range(2, n + 1):
13            result *= i
14        return result
15
16 # -----
17 # ZERO-SHOT SOLUTION (no example)
18 #
19 # "Generate a Python function to compute the factorial"
20
21 def factorial_zero_shot(n):
22     """Zero-shot generated function (without example guidance)"""
23     if n < 0:
24         raise ValueError("Factorial is not defined for negative numbers")
25     if n == 0:
26         return 1
27
28 # -----
29 # Comparison
30 # One-Shot Recursive approach (clearer, no recursion limit)
31 # Zero-Shot Recursive approach (elegant but uses stack)
32 PS C:\Users\adipala.vamshi.krish\Desktop\AI Assisted Code>
33 PS C:\Users\adipala.vamshi.krish\Desktop\AI Assisted Code>
```

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

Write a few-shot prompt by providing multiple input-output examples to guide the AI in generating a Python function to check whether a given number is an Armstrong number.

Examples:

- Input: 153 → Output: Armstrong Number
 - Input: 370 → Output: Armstrong Number
 - Input: 123 → Output: Not an Armstrong Number

Task:

- Analyze how multiple examples influence code structure and accuracy.
 - Test the function with boundary values and invalid inputs.

PROMPT: Armstrong Number Check

- Input: 153 → Output: Armstrong Number
 - Input: 370 → Output: Armstrong Number
 - Input: 123 → Output: Not an Armstrong Number

CODE:

The screenshot shows the Microsoft Visual Studio Code interface with the following details:

- File Explorer:** Shows a tree view with "AI ASSISTED CODE" expanded, containing "1.4-AI-ASS.py" and "AI-ASS.py".
- Code Editor:** Displays Python code for checking if a number is Armstrong. The code includes a docstring explaining that an Armstrong number is equal to the sum of its digits each raised to the power of the number of digits.
- Output Panel:** Shows examples of the function's behavior:
 - Input: 153 → Output: Armstrong Number
 - Input: 370 → Output: Armstrong Number
 - Input: 123 → Output: Not an Armstrong Number
- Bottom Status Bar:** Shows file path (AI-ASS.py), line 1, column 6, spaces 4, UTF-8 encoding, Python language, and file size 3.13 MB.
- Right Sidebar:** Features a "RECENT SESSIONS" list with three entries: "Code optimization and...", "Code optimization for...", and "Code optimization for...". It also includes a "Ask about your code" button and a "Generate Agent" button.

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows files like AI-AssPy, 1-AI-AssPy, and 1.4-AI-AssPy.
- Code Editor:** Displays a Python script for Armstrong number checking. The code includes comments explaining the logic and handling of invalid input.
- Terminal:** Shows the command PS C:\Users\adapale\... and the execution of the script, resulting in test results for various inputs.
- Right Panel:** Features an "Ask about your code" section with options for "Explore and understand", "Generate Agent", "Instructions to onboard", and "AI onto your codebase".
- Bottom Status Bar:** Provides information like file path (C:\Users\adapale\...), line count (Ln 41), column count (Col 50), spaces (Spaces: 4), encoding (UTF-8), language (Python), and date/time (12-01-2026).

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

Design a context-managed prompt with clear instructions and constraints to generate an optimized Python program that classifies a number as prime, composite, or neither.

Task:

- Ensure proper input validation.
- Optimize the logic for efficiency.
- Compare the output with earlier prompting strategies.

PROMPT: Optimized Number Classification CODE:

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows files like AI-AssPy, 1-AI-AssPy, and 1.4-AI-AssPy.
- Code Editor:** Displays a Python script for number classification. The code includes comments and handles edge cases like negative numbers and zero.
- Terminal:** Shows the command PS C:\Users\adapale\... and the execution of the script, resulting in test results for various inputs.
- Right Panel:** Features an "Ask about your code" section with options for "Explore and understand", "Generate Agent", "Instructions to onboard", and "AI onto your codebase".
- Bottom Status Bar:** Provides information like file path (C:\Users\adapale\...), line count (Ln 41), column count (Col 8), spaces (Spaces: 4), encoding (UTF-8), language (Python), and date/time (12-01-2026).

```

File Edit Selection View Go Run Terminal Help < > Q AI Assisted Code
EXPLORER AI ASSISTED CODE AI-ASS.py ...
AI-ASS.py classify_number
1 def classify_number(num):
2     """
3         Classifies a number as prime, composite, or neither.
4
5     Args:
6         num: Integer to classify
7
8     Returns:
9         str: Classification result
10    """
11    # Input validation
12    try:
13        num = int(num)
14    except (ValueError, TypeError):
15        return "Invalid input: Please enter a valid integer."
16
17    # Handle special cases
18    if num < 0:
19        return "Neither: Negative numbers are not classified."
20    if num == 1:
21        return "Neither: 0 and 1 are neither prime nor composite."
22
23    # Prime check
24    if num == 2:
25        return "Prime"
26    for i in range(2, num):
27        if num % i == 0:
28            return "Composite"
29
30    return "Prime"
31
32: Prime
33: Prime
34: Composite
35: Composite
36: Prime
37: Prime
38: Composite
39: Prime
40: Composite
41: Neither: 0 and 1 are neither prime nor composite.
42: Neither: 0 and 1 are neither prime nor composite.
43: Neither: Negative numbers are not classified.
44: Invalid input: Please enter a valid integer.
PS C:\Users\adapala vamshi krish\Desktop\AI Assisted Code>

```

RECENT SESSIONS

- Code optimization and... Completed Local 3 days
- Code optimization for ... Completed Local 3 days
- Code optimization for ... Completed Local 3 days

Show More

Ask about your code

AI responses may be inaccurate.

Generate Agent Instructions to onboard AI onto your codebase.

OUTLINE TIMELINE

27°C Mostly sunny

Python + v w ... x

Search

13:50 12-01 2026

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

Write a zero-shot prompt (without providing any examples) to generate a Python function that checks whether a given number is a perfect number.

Task:

- Record the AI-generated code.
- Test the program with multiple inputs.
- Identify any missing conditions or inefficiencies in the logic.

PROMPT: Perfect Number Check without providing any examples CODE:

```

File Edit Selection View Go Run Terminal Help < > Q AI Assisted Code
EXPLORER AI ASSISTED CODE AI-ASS.py ...
AI-ASS.py ...
Generate code
Add Context...
Keep Update
1 def is_perfect_number(n):
2     """
3         Check if a number is a perfect number.
4         A perfect number is a positive integer that equals the sum of its proper divisors
5         (divisors excluding the number itself).
6
7     if n <= 1:
8         return False
9
10    divisor_sum = 0
11    for i in range(1, n):
12        if n % i == 0:
13            divisor_sum += i
14
15    return divisor_sum == n
16
17
18 # Test cases
19 test_numbers = [6, 28, 496, 8128, 10, 1, 0, -6, 100]
20 print("Perfect Number Checker")
21 print("-" * 40)
22 for num in test_numbers:
23     result = is_perfect_number(num)
24     print(f"{num}: {result}")

```

RECENT SESSIONS

- Code optimization and... Completed Local 3 days
- Code optimization for ... Completed Local 3 days
- Code optimization for ... Completed Local 3 days

Show More

Ask about your code

AI responses may be inaccurate.

Generate Agent Instructions to onboard AI onto your codebase.

OUTLINE TIMELINE

27°C Mostly sunny

Python + v w ... x

Search

13:50 12-01 2026

```

File Edit Selection View Go Run Terminal Help < > Q AI Assisted Code
EXPLORER ... AI-ASS.py 1-AI-ASS.py
AI-ASS.py ...
1 def is_perfect_number(n):
2     """
3         Check if a number is a perfect number.
4         A perfect number is a positive integer that equals the sum of its proper divisors
5             (divisors excluding the number itself).
6     """
7     if n <= 1:
8         return False
9
10    divisor_sum = 0
11    for i in range(1, n):
12        if n % i == 0:
13            divisor_sum += i
14
15    return divisor_sum == n
16
17
18 # Test cases
19 test_numbers = [6, 28, 496, 8128, 10, 1, 0, -6, 100]
20
21 print("Perfect Number Checker")
22 print("-" * 40)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\adapala vamsi krish\Desktop\AI Assisted Code> & "C:/Users/adapala vamsi krish/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/adapala vamsi krish/Desktop/AI Assisted Code/AI-ASS.py"
Perfect Number Checker
-----
6: True
28: True
496: True
8128: True
10: False
1: False
0: False
-6: False
100: False
PS C:\Users\adapala vamsi krish\Desktop\AI Assisted Code>

```

OUTLINE TIMELINE

27°C Mostly sunny

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

Write a few-shot prompt by providing multiple input-output examples to guide the AI in generating a Python program that determines whether a given number is even or odd, including proper input validation.

Examples:

- Input: 8 → Output: Even
- Input: 15 → Output: Odd • Input: 0 → Output: Even **Task:**
- Analyze how examples improve input handling and output clarity.
- Test the program with negative numbers and non-integer inputs.

PROMPT:

Even or Odd Classification with Validation

- Input: 8 → Output: Even
- Input: 15 → Output: Odd
- Input: 0 → Output: Even

CODE:

```

File Edit Selection View Go Run Terminal Help < > Q AI Assisted Code
EXPLORER ... AI-ASS.py 1-AI-ASS.py
AI-ASS.py ...
1 def classify_number(user_input):
2     """
3         Classify a number as even or odd with proper input validation.
4     """
5     Examples:
6     - Input: 8 -> Output: Even
7     - Input: 15 -> Output: Odd
8     - Input: 0 -> Output: Even
9     - Input: -4 -> Output: Even
10    - Input: 7 -> Output: Odd
11
12    try:
13        # Validate and convert input to integer
14        number = int(user_input)
15
16        # Classify as even or odd
17        if number % 2 == 0:
18            return f"Input: {number} -> Output: Even"
19        else:
20            return f"Input: {number} -> Output: Odd"
21
22    except ValueError:
23        return f"Error: '{user_input}' is not a valid integer. Please enter a whole number."
24
25
26 # Test cases
27 if __name__ == "__main__":
28     test_inputs = [8, 15, 0, -4, -7, 3.5, "abc", -10, 1]
29
30     print("Testing Even/Odd Classification:")
31     for test in test_inputs:
32         result = classify_number(test)
33         print(result)

```

OUTLINE TIMELINE

27°C Mostly sunny

File Edit Selection View Go Run Terminal Help ↻ Q AI Assisted Code

EXPLORER

AI ASSISTED CODE

1.4-AI-ASS.py

AI-ASS.py

AI-ASS.py ~

```
1 def classify_number(user_input):
2     """
3         Classify a number as even or odd with proper input validation.
4     """
5     Examples:
6     - Input: 8 → Output: Even
7     - Input: 15 → Output: Odd
8     - Input: 0 → Output: Even
9     - Input: -4 → Output: Even
10    - Input: -7 → Output: Odd
11
12    try:
13        # Validate and convert input to integer
14        number = int(user_input)
15
16        # Classify as even or odd
17        if number % 2 == 0:
18            return f"Input: {number} → Output: Even"
19        else:
20            return f"Input: {number} → Output: Odd"
21
22    except ValueError:
23        return "Error: abc is not a valid integer. Please enter a whole number."
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\adapala vamshi krish\Desktop\AI Assisted Code & "C:/Users/adapala vamshi krish/AppData/Local/Programs/Python/Python311/python.exe" "c:/Users/adapala vamshi krish/Desktop/AI Assisted Code/AI-ASS.py"

Testing Even/Odd Classification:

```
Input: 8 → Output: Even
Input: 15 → Output: Odd
Input: 0 → Output: Even
Input: -4 → Output: Even
Input: -7 → Output: Odd
Input: 3 → Output: Odd
Error: abc is not a valid integer. Please enter a whole number.
Input: -10 → Output: Even
Input: 1 → Output: Odd
PS C:\Users\adapala vamshi krish\Desktop\AI Assisted Code
```

OUTLINE TIMELINE

Search

AI ASSISTED CODE

RECENT SESSIONS

- Code optimization and... completed Local +3 days
- Code optimization for ... completed Local +3 days
- Code optimization for ... completed Local +3 days

Show More

Ask about your code

AI responses may be inaccurate.

Generate Agent

Instructions to onboard AI onto your codebase.

AI-ASS.py

Explore and understand AI-ASS.py

Ask Auto ▾ Go Live ▾

Ln 33, Col 22 Spaces: 4 UTF-8 Python 3.13.7 12-01-2026