

ASSIGNMENT – 3.1

A.VAMSHI KRISHNA

2303A52249

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 Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Toolbar:** Includes icons for file operations like Open, Save, and Run.
- Search Bar:** "Q AI Assisted Code".
- Left Sidebar:** EXPLORER (1.4-AI-ASS.py), RECENT SESSIONS (multiple sessions listed).
- Code Editor:** A large text area containing Python code for a palindrome checker. The code includes comments explaining the logic and test cases.
- Right Sidebar:** "Ask about your code" section with instructions and AI status.
- Bottom Status Bar:** Shows file path (AI-ASS.py), line count (Ln 1, Col 22), spaces (Spaces 4), encoding (UTF-8), file type (Python), file size (3.137 kB), and timestamp (12-01-2026). It also shows system information like weather (27°C, mostly sunny) and system status (ENG IN, 13:45).

```
1 def is_palindrome(n):
2     # Handle negative numbers - they are not palindromes
3     if n < 0:
4         return False
5
6     # Convert number to string and compare with its reverse
7     num_str = str(n)
8     return num_str == num_str[::-1]
9
10
11 # Test cases
12 if __name__ == "__main__":
13     test_cases = [
14         121,      # True - palindrome
15         -121,     # False - negative number
16         0,        # True - single digit
17         10,       # False - not a palindrome
18         1001,     # True - palindrome
19         12321,    # True - palindrome
20         123,      # False - not a palindrome
21         9,        # True - single digit
22         100,      # False - not a palindrome
23     ]
24
25     for num in test_cases:
26         result = is_palindrome(num)
27         print(f"Is {num} a palindrome? {result}")
```

The screenshot shows the AI Assisted Code interface. On the left, the Explorer sidebar lists files: 1-AI-ASS.py, 1.4-AI-ASS.py, and AI-ASS.py. The main editor area contains the following Python code:

```

1 #!/usr/bin/python
2
3 def is_palindrome(n):
4     # Handle negative numbers - they are not palindromes
5     if n < 0:
6         return False
7
8     # Convert number to string and compare with its reverse
9     num_str = str(n)
10    return num_str == num_str[::-1]
11
12 # Test cases
13 if __name__ == "__main__":
14     test_cases = [
15         (-121, "# False - negative number"),
16         (0, "# True - single digit"),
17         (10, "# False - not a palindrome"),
18         (1001, "# True - palindrome"),
19         (12321, "# True - palindrome"),
20         (123, "# False - not a palindrome"),
21         (9, "# True - single digit"),
22         (100, "# False - not a palindrome")
23     ]
24
25     for num in test_cases:
26         result = is_palindrome(num)
27         print(f"Is {num} a palindrome? {result}")

```

The terminal below shows the execution of the script:

```

PS C:\Users\adapala.vamsi.krish\Desktop\AI Assisted Code> & "C:/Users/adapala.vamsi.krish/AppData/Local/Programs/Python/Python311/python.exe" "c:/Users/adapala.vamsi.krish/Desktop/AI Assisted Code>
12321 is a palindrome? True
0 is a palindrome? True
1001 is a palindrome? True
100 is a palindrome? False
10 is a palindrome? False
-121 is a palindrome? False
9 is a palindrome? True
123 is a palindrome? False

```

The status bar at the bottom indicates it's 12:01:2026.

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 interface. On the left, the Explorer sidebar lists files: 1-AI-ASS.py, 1.4-AI-ASS.py, and AI-ASS.py. The main editor area contains the following Python code:

```

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

```

The terminal below shows the execution of the script:

```

PS C:\Users\adapala.vamsi.krish\Desktop\AI Assisted Code> & "C:/Users/adapala.vamsi.krish/AppData/Local/Programs/Python/Python311/python.exe" "c:/Users/adapala.vamsi.krish/Desktop/AI Assisted Code>
120
120
---- Comparison ----

```

The status bar at the bottom indicates it's 12:01:2026.

```

File Edit Selection View Go Run Terminal Help < > AI Assisted Code
EXPLORER AI ASSISTED CODE AI-ASSPy 14-AI-ASSPy AI-ASS.py
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

AI-ASSPy X 14-AI-ASSPy
AI-ASSPy ...
AI-ASSPy
def factorial(n):
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers")
    if n == 0 or n == 1:
        return 1
    result = 1
    for i in range(2, n + 1):
        result *= i
    return result

# -----
# ZERO-SHOT SOLUTION (no example)
# -----
# "Generate a Python function to compute the factorial"
# -----
def factorial_zero_shot(n):
    """Zero-shot generated function (without example guidance)"""
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers")
    if n == 0:
        return 1

POSITION OUTPUT DEBUG CONSOLE TERMINAL PORTS
Is publishing(False)
PS C:\Users\adapala_vamsi_krish\Desktop\AI Assisted Code & TCI\Users\adapala_vamsi_krish\AppData\Local\Programs\Python\Python311\python.exe" "C:\Users\adapala_vamsi_krish\Desktop\One-Shot Result: 120
Zero-Shot Result: 120
Comparison
One-Shot: Recursive approach (clearer, no recursion limit)
Zero-Shot: Recursive approach (elegant but uses stack)
One-Shot Result: 120
Zero-Shot Result: 120
PS C:\Users\adapala_vamsi_krish\Desktop\AI Assisted Code>

```

Ask about your code
AI response may be inaccurate.
Generate Agent
Instructions to onboard
AI onto your codebase.

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:

```

File Edit Selection View Go Run Terminal Help < > AI Assisted Code
EXPLORER AI ASSISTED CODE AI-ASSPy 14-AI-ASSPy AI-ASS.py
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

AI-ASSPy X 14-AI-ASSPy
AI-ASSPy ...
AI-ASSPy
Examples:
• Input: 153 → Output: Armstrong Number
• Input: 370 → Output: Armstrong Number
• Input: 123 → Output: Not an Armstrong Number
Add Context...
def is_armstrong_number(num):
    """
    Check if a number is an Armstrong number.
    An Armstrong number is a number that is equal to the sum of its own digits each raised to the power of the number of digits.
    Examples:
    - 153 = 1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153 ✓
    - 370 = 3^3 + 7^3 + 0^3 = 27 + 343 + 0 = 370 ✓
    - 123 = 1^3 + 2^3 + 3^3 = 1 + 8 + 27 = 36 X
    """
    if not isinstance(num, int) or num < 0:
        return "Invalid Input"
    digits = str(num)
    num_digits = len(digits)
    sum_of_powers = sum(int(digit)**num_digits for digit in digits)
    return "Armstrong Number" if sum_of_powers == num else "Not an Armstrong Number"

# Test cases
test_cases = [
    (153, "Armstrong Number"),
    (370, "Armstrong Number"),
    (123, "Not an Armstrong Number"),
    (9474, "Armstrong Number"),
    (1634, "Armstrong Number"),
    (186, "Not an Armstrong Number"),
    (9, "Armstrong Number"),
    (0, "Armstrong Number"),
    (-153, "Invalid Input"),
    ("abc", "Invalid Input"),
]
for num, expected in test_cases:
    print(f"Testing {num}... {expected}")

print("Armstrong Number Checker - Test Results")
print("-" * 50)
for num, expected in test_cases:
    print(f"Input: {num}, Expected: {expected}, Actual: {is_armstrong_number(num)}")

POSITION OUTPUT DEBUG CONSOLE TERMINAL PORTS
Is publishing(False)
PS C:\Users\adapala_vamsi_krish\Desktop\AI Assisted Code & TCI\Users\adapala_vamsi_krish\AppData\Local\Programs\Python\Python311\python.exe" "C:\Users\adapala_vamsi_krish\Desktop\One-Shot Result: 120
Zero-Shot Result: 120
Comparison
One-Shot: Recursive approach (clearer, no recursion limit)
Zero-Shot: Recursive approach (elegant but uses stack)
One-Shot Result: 120
Zero-Shot Result: 120
PS C:\Users\adapala_vamsi_krish\Desktop\AI Assisted Code>

```

Ask about your code
AI response may be inaccurate.
Generate Agent
Instructions to onboard
AI onto your codebase.

The screenshot shows the Visual Studio Code interface with the AI-Assistant extension integrated. The top status bar displays "AI Assisted Code". The left sidebar has sections for "EXPLORER", "File", "Edit", "Selection", "View", "Go", "Run", "Terminal", "Help", and "RECENT SESSIONS". The main editor area shows Python code for checking Armstrong numbers, with syntax highlighting and code completion suggestions. The bottom status bar shows file paths, line numbers, and other development details.

```
AI-Assistant > AI-Assistant Code > AI-Assistant.py
```

```
AI-Assistant.py
```

```
1 def is_armstrong_number(num):
```

```
2     pass
```

```
3     # Check if a number is an Armstrong number.
```

```
4     # An Armstrong number is a number that is equal to the sum of
```

```
5     # its own digits each raised to the power of the number of digits.
```

```
6
```

```
7     Examples:
```

```
8     - 153 = 13 + 53 + 33 = 1 + 125 + 27 = 153 ✓
```

```
9     - 370 = 33 + 73 + 03 = 27 + 343 = 0 = 370 ✓
```

```
10    - 123 = 13 + 23 + 33 = 1 + 8 + 27 = 36 ✗
```

```
11
```

```
12    if not isinstance(num, int) or num < 0:
```

```
13        return "Invalid Input"
```

```
14
```

```
15    digits = str(num)
```

```
16    num_digits = len(digits)
```

```
17    sum_of_powers = sum(int(digit)**num_digits for digit in digits)
```

```
18
```

```
19    return "Armstrong Number" if sum_of_powers == num else "Not an Armstrong Number"
```

```
20
```

```
21
```

```
22 # Test cases
```

```
23
```

```
24
```

```
PROBLEMS OUTPUT TERMINAL PORTS
```

```
PS C:\Users\adapala.vamshi.krish\Desktop\AI Assisted Code & "C:/Users/adapala.vamshi.krish/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/adapala.vamshi.krish/Desktop/AI Assisted Code/AI-Assistant.py"
```

```
Armstrong Number Checker - Test Results
```

```
-----
```

```
✓ Input: 153 - Armstrong Number
```

```
✓ Input: 370 - Armstrong Number
```

```
✓ Input: 9474 - Armstrong Number
```

```
✓ Input: 9474 - Armstrong Number
```

```
✓ Input: 1634 - Armstrong Number
```

```
✓ Input: 1634 - Armstrong Number
```

```
✓ Input: 1 - Armstrong Number
```

```
✓ Input: 0 - Armstrong Number
```

```
✓ Input: -123 - Armstrong Number
```

```
✓ Input: abc - Invalid Input
```

```
PS C:\Users\adapala.vamshi.krish\Desktop\AI Assisted Code>
```

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:

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:

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

- File Explorer:** Shows a folder named "AI ASSISTED CODE" containing files "1.4-AI-ASS.py" and "AI-ASS.py".
- Code Editor:** Displays Python code for checking if a number is perfect. The code includes a docstring, a function definition, a loop to calculate divisors, and test cases. A tooltip from the AI assistant suggests using `isinstance(num, int)` instead of `type(num) == int`.
- Output Panel:** Shows AI-generated instructions: "Explore and understand", "Ask", and "Auto".
- Bottom Status Bar:** Shows file path (ln 25, Col 30), spaces used (Spaces: 4), Python version (Python 3.13.7), and system status (ENG IN).

The screenshot shows the Visual Studio Code interface with the "AI Assisted Code" extension open. The left sidebar has a tree view with "EXPLORER" expanded, showing "AI ASSISTED CODE", "1.4-AI-ASS.py", and "AI-ASS.py". The main editor area displays Python code for checking if a number is perfect. The bottom terminal shows the code running and outputting results for various test numbers. A floating panel on the right says "Ask about your code" and "AI onto your codebase".

```
def is_perfect_number(n):
    """
    Checks if a number is a perfect number.
    A perfect number is a positive integer that equals the sum of its proper divisors
    (divisors excluding the number itself).
    """
    if n <= 1:
        return False

    divisor_sum = 0
    for i in range(1, n):
        if n % i == 0:
            divisor_sum += i

    return divisor_sum == n

# Test cases
test_numbers = [6, 28, 496, 8128, 10, 1, 0, -6, 100]
for num in test_numbers:
    print(f"Perfect Number Checker")
    print(f"{num}: {is_perfect_number(num)}")
```

PS C:\Users\adapala vanshi krish\Desktop\AI Assisted Code> & "C:/Users/adapala vanshi krish/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/adapala vanshi krish/Desktop\AI Assisted Code\1.4-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 vanshi krish\Desktop\AI Assisted Code>

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:

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

- File Explorer:** Shows a tree view with 'EXPLORER' expanded, containing 'AI ASSISTED CODE' and '1.4-AI-ASS.py'. A context menu is open over '1.4-AI-ASS.py' with options like 'Analyze how examples improve input handling and output'.
- Code Editor:** Displays the 'AI-ASS.py' file. The code defines a function to classify numbers as even or odd. It includes examples and a test case section. The code editor has a dark theme with syntax highlighting.
- Terminal:** Shows the command 'python -m venv env'.
- Status Bar:** Shows 'ln 33, Col 22' and other system information.
- Activity Bar:** Includes icons for 'OUTLINE', 'TIMELINE', and 'DASHBOARD'.
- Bottom Taskbar:** Shows various application icons.
- Right Sidebar:** Shows 'RECENT SESSIONS' with three entries: 'Code optimization and...', 'Code optimization for...', and 'Code optimization for...'. It also includes a 'Show More' button and a 'Ask about your code' button.

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

- File Explorer:** Shows two files: "AI-ASS.py" and "1.4-AI-ASS.py".
- Code Editor:** Displays Python code for classifying numbers as even or odd. The code includes examples and handles ValueError exceptions.
- Terminal:** Shows the output of running the script, displaying classification results for various inputs and an error message for non-integer input.
- Right Panel:** Features the "Ask about your code" feature, which provides AI-generated responses to common questions like "What is the purpose of this code?" and "How can I improve this code?". It also shows recent sessions and optimization status.