

# ASSIGNMENT – 3.1

B.MANUDEEP REDDY

2303A52380

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, RECENT SESSIONS (with items like "Code optimization and... Completed Local 3 days").
- Code Editor:** A large window containing Python code. The code defines a function `is\_palindrome` that checks if a number is a palindrome by comparing it with its reverse string. It includes test cases for various numbers.

```
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}")
```

- 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", date "12-01-2026", and time "13:45".

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface with the following details:

- File Explorer:** Shows a tree view with three items: "AI-ASS.py", "AI-ASS.pyw", and "AI-ASS.py".
- Editor:** Displays Python code for checking if a number is a palindrome. The code includes a main function that iterates through a list of test cases, printing the result of each call to `is_palindrome`.
- Terminal:** Shows the output of running the script with various test cases. The output indicates which numbers are palindromes and which are not.
- Bottom Status Bar:** Shows the current file path as "C:\Users\adapala\OneDrive\Desktop\AI Assisted Code\AI-ASS.pyw", the Python version as "Python 3.11.3", and the current line as "13".
- Right Sidebar:** Contains sections for "RECENT SESSIONS" (with entries for "Code optimization and...", "Code optimization for...", and "Code optimization for..."), "Ask about your code" (with a note about AI responses being inaccurate), and "Generate Agent" (with instructions to onboard AI onto your codebase).

## 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

- task:**

  - Compare the generated code with a zero-shot solution.
  - Examining improvements in clarity and correctness.

**PROMPT:** Factorial Calculation giving some instructions input data

CODE:

The screenshot shows the AI Assisted Code feature in Visual Studio Code. The left sidebar displays a tree view of files, with 'AI ASSISTED CODE' expanded to show '1.4-AI-ASS.py' and 'AI-ASS.py'. The main editor area contains Python code for calculating factorials. The right sidebar features a 'RECENT SESSIONS' list with three entries: 'Code optimization and...', 'Code optimization for ...', and 'Code optimization for ...'. Below this is a 'Keep Undo' button. A large central panel displays two versions of the factorial function: a 'One-shot generated function' and a 'Zero-shot solution'. At the bottom, a 'TEST & COMPARISON' section compares the results of both functions with a test value of 5. The status bar at the bottom right shows the file path as 'Ln 28, Col 1', space usage as 'Space: 4', and the current file as 'Python'. The bottom navigation bar includes icons for file operations like 'New', 'Open', 'Save', and 'Run'.

File Edit Selection View Go Run Terminal Help

AI Assisted Code

EXPLORER

AI ASSISTED CODE

1.4-AI-ASS.py

AI-ASS.py

AI-Assistant x 1.4-AI-ASS.py

AI-Assistant > factorial

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

is\_prime[100]: True

PS C:\Users\adipala.vamshi.krish\Desktop\AI Assisted Code> ll "C:/Users/adipala.vamshi.krish/AppData/Local/Programs/Python/Python313/python.exe" "C:/Users/adipala.vamshi.krish/Desktop/AI Assisted Code/AI-ASS.py"

One-Shot Result: 120

Zero-Shot Result: 120

Comparison

One-Shot: Recursive approach (cleaner, no recursion limit)

Zero-Shot: Recursive approach (elegant but uses stack)

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

AI SESSIONS

RECENT SESSIONS

Code optimization and... Completed Local 1 day

Code optimization for... Completed Local 1 day

Code optimization for... Completed Local 1 day

Show More

Ask about your code

AI responses are being generated

Generate Agent

Instructions to onboard AI onto your codebase.

AI-Assistant x AI-ASS.py

Explore and understand Ask Auto ▾ Go Live

Search

Unl. Col 25 Spaces: 4 UTF-0 CRLF

13.17 ENG IN

27°C Mostly sunny

hp Th chrome

11:47 12/01/2024

### 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 a code editor interface with the following details:

- File Explorer:** Shows two files: `AI-ASS.py` and `1.4-AI-ASS.py`. The `AI-ASS.py` file is open.
- Code Editor:** Displays Python code for checking if a number is Armstrong. The code includes examples and a test case section. A sidebar on the right shows AI-assisted suggestions like "Code optimization and..." and "Ask about your code".
- Terminal:** Shows the command `AI ASSISTED Code`.
- Bottom Bar:** Includes icons for file operations, search, and various system status indicators.

```
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"),
    (1064, "Armstrong Number"),
    (123, "Not an Armstrong Number"),
    (0, "Armstrong Number"),
    (9, "Armstrong Number"),
    (-153, "Invalid Input"),
    ("abc", "Invalid Input"),
]
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)}")
```

The screenshot shows a code editor interface with multiple tabs open. The active tab is 'AI-ASS.py' containing Python code for checking if a number is Armstrong. The code includes examples and a test cases section. The terminal below shows the execution of the script and its test cases. A sidebar on the right provides AI assistance and recent sessions.

```

1 def is_armstrong_number(num):
2     """
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 = 1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153 ✓
9             370 = 3^3 + 7^3 + 0^3 = 27 + 343 + 0 = 370 ✓
10            133 = 1^3 + 3^3 + 3^3 = 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

```

PROGRAM 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"

AI Assisted Code/AI-Ass.py - Test Results

-----

✓ Input: 153 = Armstrong Number  
✓ Input: 370 = Armstrong Number  
✓ Input: 133 = Not an Armstrong Number  
✓ Input: 9472 = Armstrong Number  
✓ Input: 9474 = Not an Armstrong Number  
✓ Input: 100 = Not an Armstrong Number  
✓ Input: 100 = Armstrong Number  
✓ Input: abc = Invalid Input  
✓ Input: -153 = Invalid Input  
✓ Input: abc = Invalid Input

PS C:\Users\adapala vamsi krish\Desktop\AI Assisted Code>

Ask about your 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

Explore and understand: Ask Auto Go Live

Ln 41 Col 50 Spaces: 4 UTF-8 Python 3.13.7 ENG IN 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 multiple tabs open. The active tab is '1.4-AI-ASS.py' containing Python code for classifying numbers as prime, composite, or neither. The code includes documentation, input validation, and prime/composite checks. The terminal below shows the execution of the script and its test cases. A sidebar on the right provides AI assistance and recent sessions.

```

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 == 0 or num == 1:
21                return "Neither: 0 and 1 are neither prime nor composite."
22
23            # Check for prime
24            if num == 2:
25                return "Prime"
26            if num % 2 == 0:
27                return "Composite"
28
29            # Optimized prime check: only check odd divisors up to sqrt(num)
30            i = 3
31            while i * i <= num:
32                if num % i == 0:
33                    return "Composite"
34                i += 2
35
36            return "Prime"
37
38
39 def main():
40     """Main function to test the classifier."""
41     test_cases = [2, 3, 4, 5, 17, 20, 97, 100, 1, 0, -5, "abc"]
42
43     print("Number Classification Results:\n")
44     for num in test_cases:

```

PROGRAM 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/1.4-AI-ASS.py"

AI Assisted Code/1.4-AI-ASS.py - Test Results

-----

Keep Undo ⌛

Auto ▾

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

All responses may be inaccurate.

Generate Agent Instructions to onboard AI onto your codebase.

Explore and understand: Ask Auto Go Live

Ln 2, Col 8 Spaces: 4 UTF-8 Python 3.13.7 ENG IN 12-01-2026

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

- File Explorer:** Shows two files: "AI-ASSpy" and "1.4-AI-ASS.py".
- Terminal:** Displays the output of running the code, showing classification results for various numbers (2, Prime, Composite, etc.).
- Code Editor:** Shows the generated Python code for classifying numbers.
- Right Panel:** Shows recent sessions for AI optimization tasks and an "Ask about your code" feature.
- Bottom:** Shows the taskbar with various application icons and system status.

```

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 == 0 or num == 1:
21        return "Neither: 0 and 1 are neither prime nor composite."
22
23    # Prime
24    if num == 2:
25        return "Prime"
26
27    # Composite
28    if num > 2 and num % 2 == 0:
29        return "Composite"
30
31    # Check for prime
32    for i in range(3, int(num**0.5) + 1):
33        if num % i == 0:
34            return "Composite"
35
36    return "Prime"
37
38
39 Number Classification Results:
40
41 2: Prime
42 3: Prime
43 4: Composite
44 5: Composite
45 7: Prime
46 10: Composite
47 97: Prime
48 106: Composite
49
50 1: Neither: 0 and 1 are neither prime nor composite.
51 2: Neither: Negative numbers are not classified.
52 3: Neither: Negative numbers are not classified.
53 4: Neither: Negative numbers are not classified.
54 5: Neither: Negative numbers are not classified.
55 7: Neither: Negative numbers are not classified.
56 10: Neither: Negative numbers are not classified.
57 97: Neither: Negative numbers are not classified.
58 abc: Invalid input: Please enter a valid integer.
59
60 P5 C:\Users\adapala.vamshi.krish\Desktop\AI Assisted 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 Visual Studio Code interface with the following details:

- File Explorer:** Shows two files: "1.4-AI-ASS.py" and "AI-ASS.py".
- Code Editor:** Shows the generated Python code for checking if a number is perfect.
- Right Panel:** Shows recent sessions for AI optimization tasks and an "Ask about your code" feature.
- Bottom:** Shows the taskbar with various application icons and system status.

```

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
8     if n < 1:
9         return False
10
11     divisor_sum = 0
12     for i in range(1, n):
13         if n % i == 0:
14             divisor_sum += i
15
16     return divisor_sum == n
17
18 # Test cases
19 test_numbers = [6, 28, 496, 8128, 10, 1, 0, -6, 100]
20
21 print("Perfect Number Checker")
22 print("= * =")
23 for num in test_numbers:
24     result = is_perfect_number(num)
25     print(f"{num}: {result}")

```

```

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 # Test cases
18 test_numbers = [6, 28, 496, 8128, 10, 1, 0, -6, 100]
19
20 print("Perfect Number Checker")
21 print("-" * 40)
22
23 for number in test_numbers:
24     result = is_perfect_number(number)
25     print(f"{number}: {result}")
26
27 print("-" * 40)
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

```

### 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:

```

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: '{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, 3.5, "abc", -10, 1]
29
30     print("Testing Even/Odd Classification:")
31     for test_input:
32         result = classify_number(test_input)
33         print(result)
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

```

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

- File Path:** 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\1.4-AI-ASS.py"
- Code Content:**

```
def classify_number(user_input):
    """
    Classify a number as even or odd with proper input validation.

    Examples:
    - Input: 8 -> Output: Even
    - Input: 15 -> Output: Odd
    - Input: 0 -> Output: Even
    - Input: -4 -> Output: Even
    - Input: -7 -> Output: Odd

    try:
        # Validate and convert input to integer
        number = int(user_input)
    except ValueError:
        print("Error: 'abc' is not a valid integer. Please enter a whole number.")

    if number % 2 == 0:
        return f"Input: {number} + Output: Even"
    else:
        return f"Input: {number} + Output: Odd"
```
- Terminal Output:**

```
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: 2 -> Output: Even
Input: 1 -> Output: Odd
```
- Right Panel:** Shows recent sessions and an "Ask about your code" feature.
- Bottom Bar:** Includes a weather widget (27°C), system icons, and status information (Ln 33, Col 22, Spaces: 4, UTF-8, Python, ENG IN, 13:51, 12-01-2024).