2303A1739

Gangula Navyasri

Task 1: Zero-shot Prompt – Fibonacci Series Generator

Task Description #1

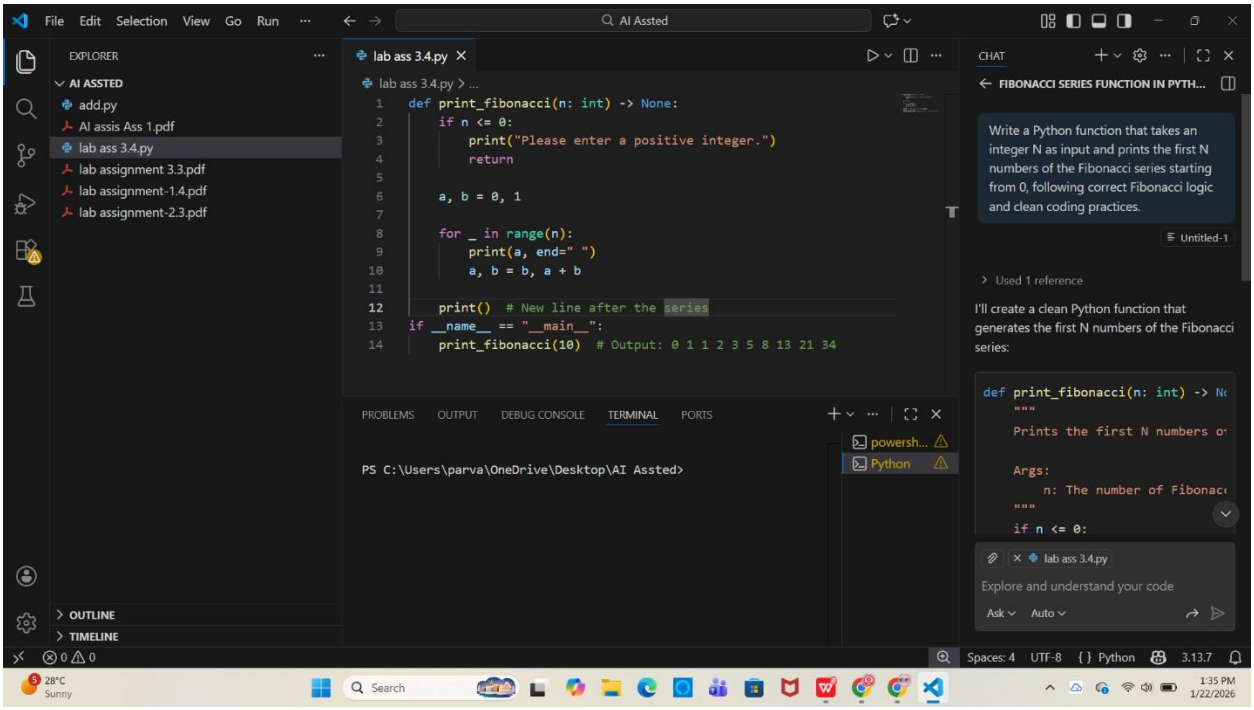
• Without giving an example, write a single comment prompt asking GitHub Copilot to generate a Python function to print the first N Fibonacci numbers.

Expected Output #1

• A complete Python function generated by Copilot without any example provided.

• Correct output for sample input N = 7 → 0 1 1 2 3 5 8

• Observation on how Copilot understood the instruction with zero context.



Task 2: One-shot Prompt – List Reversal Function

Task Description #2

• Write a comment prompt to reverse a list and provide one

example below the comment to guide Copilot.

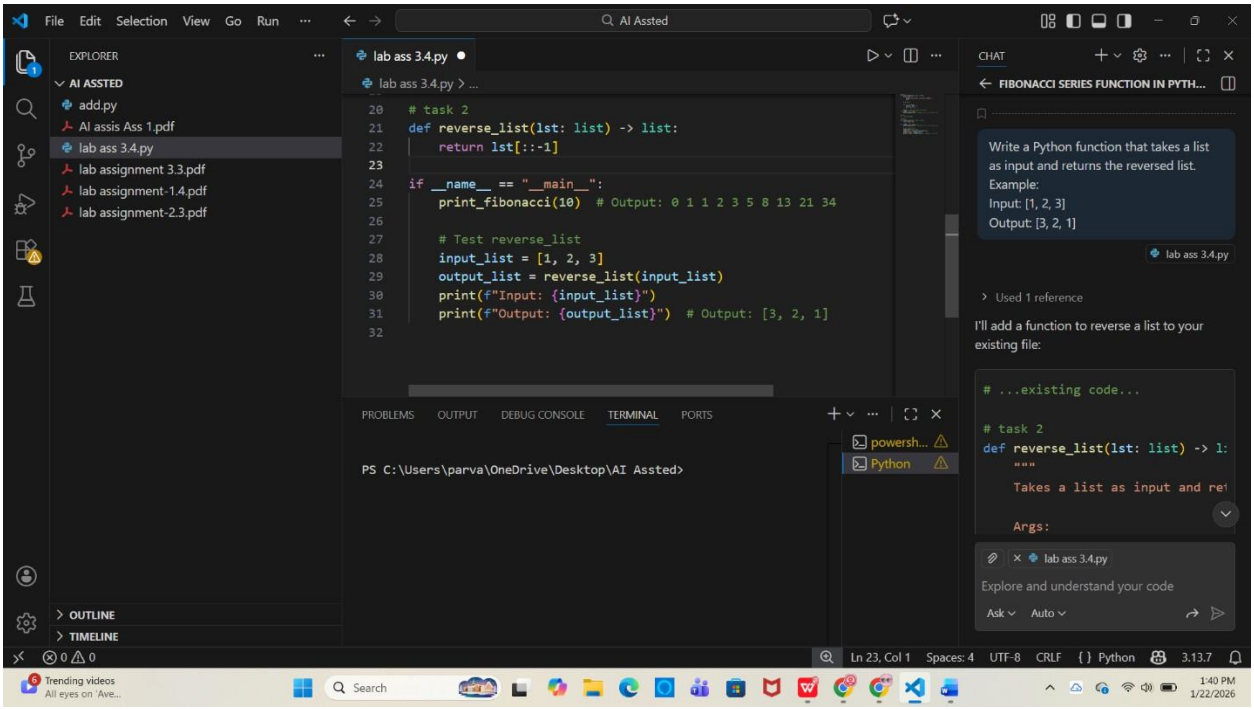
Expected Output #2

• Copilot-generated function to reverse a list using slicing or loop.

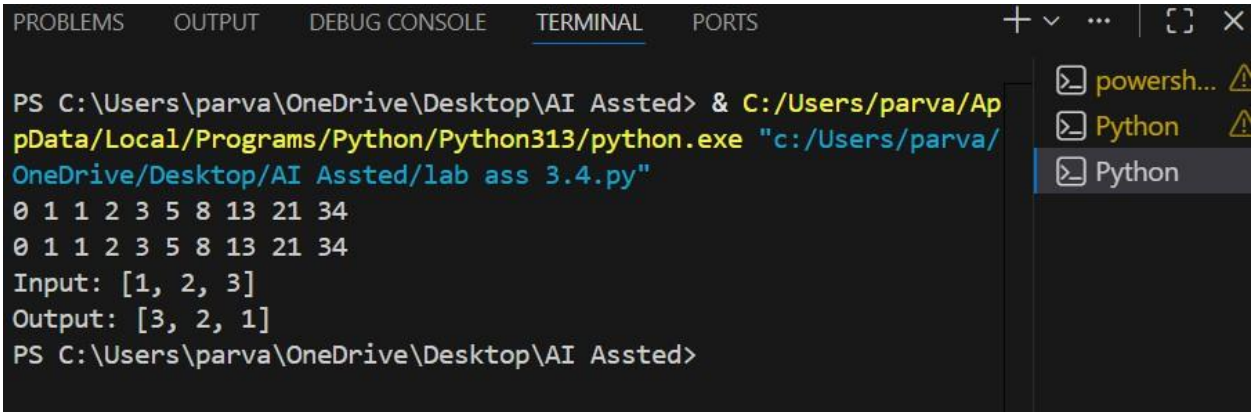
• Output: [3, 2, 1] for input [1, 2, 3]

• Observation on how adding a single example improved Copilot's

Accuracy



1 and 2 tasks output



Task 3: Few-shot Prompt – String Pattern Matching

Task Description #3

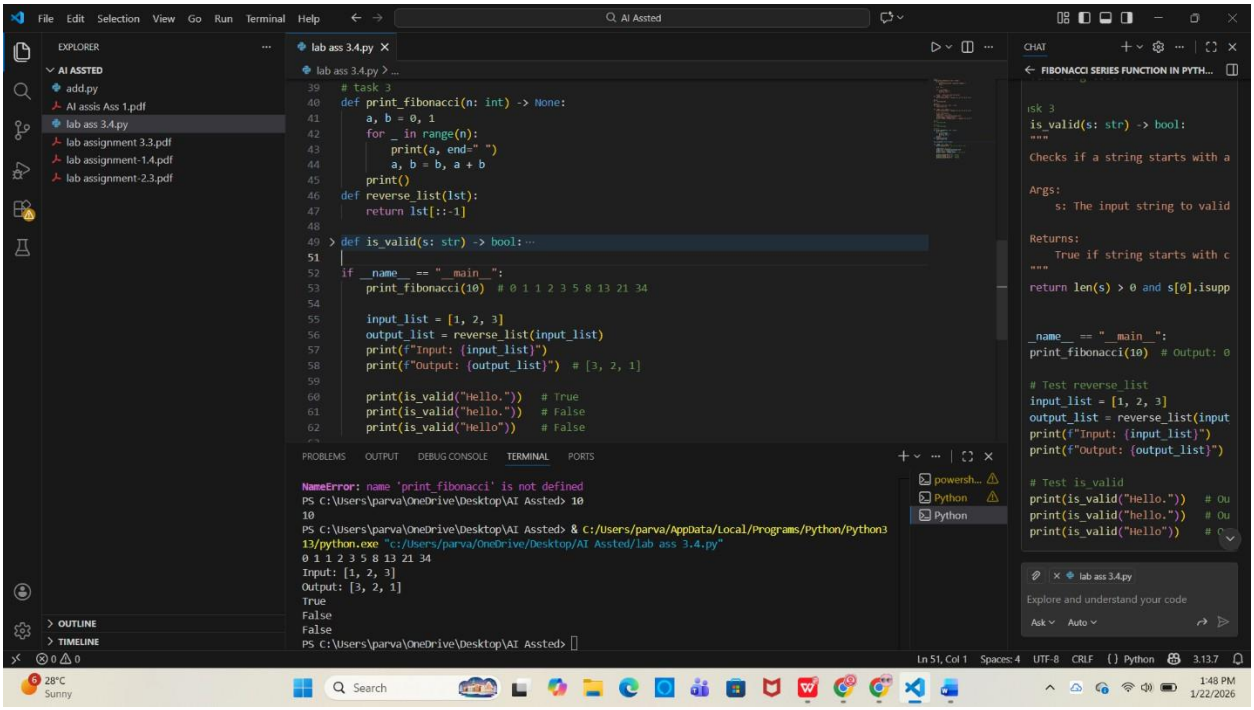
• Write a comment with 2–3 examples to help Copilot understand how to check if a string starts with a capital letter and ends with a period.

Expected Output #3

• A function is_valid() that checks the pattern.

• Output: True or False based on input.

• Students reflect on how multiple examples guide Copilot to generate more accurate code.



Task 4: Zero-shot vs Few-shot – Email Validator

Task Description #4

• First, prompt Copilot to write an email validation function using zero-shot (just the task in comment).

• Then, rewrite the prompt using few-shot examples.

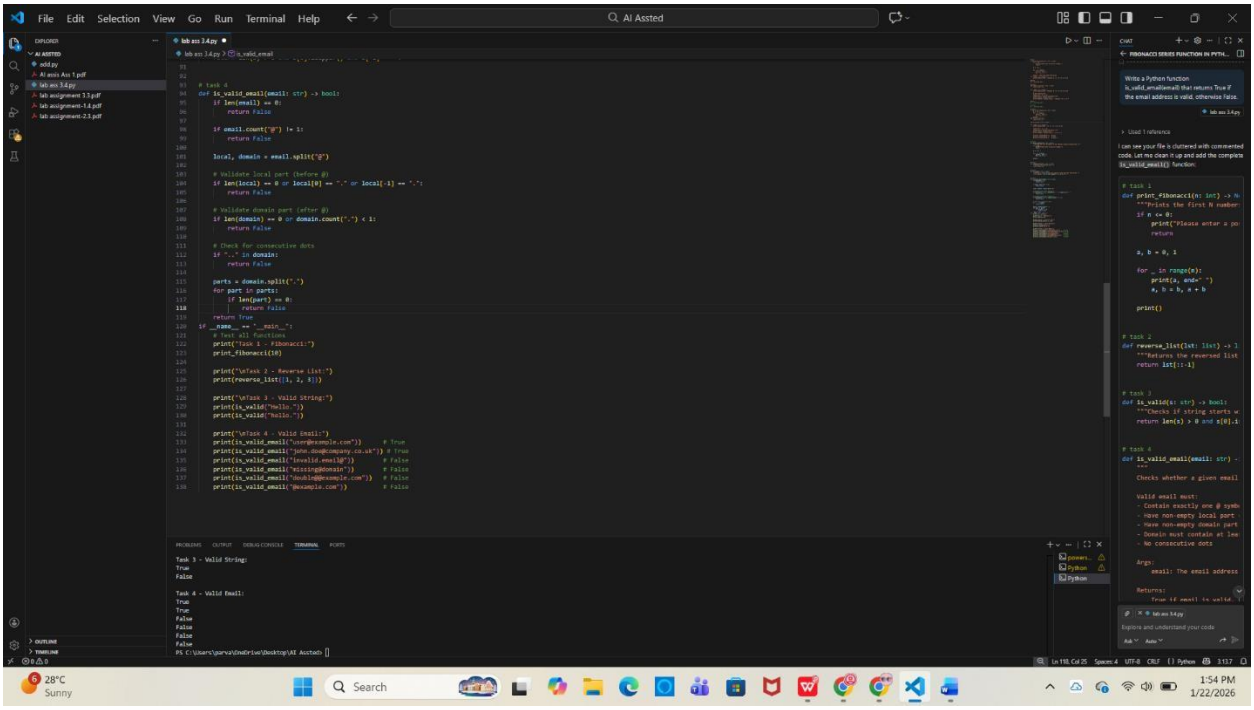
Expected Output #4

• Compare both outputs:

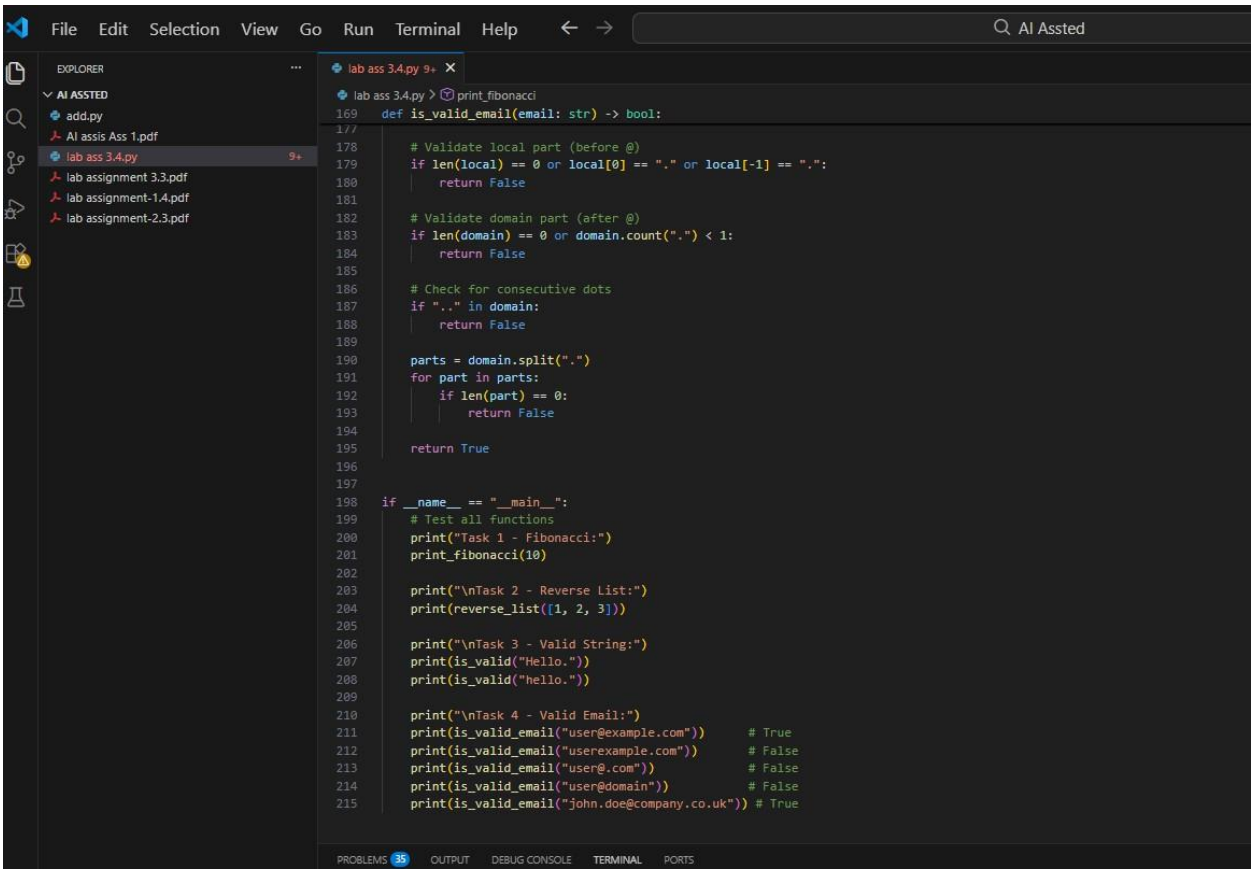
Zero-shot may result in basic or generic validation.

Few-shot gives detailed and specific logic (e.g., @ and domain checking).

• Submit both code versions and note how few-shot improves reliability.



```python
#task4.2
# task 1
def print_fibonacci(n: int) -> None:
    if n <= 0:
        print("Please enter a positive integer.")
        return

    a, b = 0, 1

    for _ in range(n):
        print(a, end=" ")
        a, b = b, a + b

    print()


# task 2
def reverse_list(lst: list) -> list:
    """Returns the reversed list."""
    return lst[::-1]


# task 3
def is_valid(s: str) -> bool:
    """Checks if string starts with capital letter and ends with period."""
    return len(s) > 0 and s[0].isupper() and s[-1] == "."


# task 4
def is_valid_email(email: str) -> bool:
    if len(email) == 0:
        return False

    if email.count("@") != 1:
        return False

    local, domain = email.split("@")
```



Task 5: Prompt Tuning – Summing Digits of a Number

Task Description #5

• Experiment with 2 different prompt styles to generate a function

that returns the sum of digits of a number.

Style 1: Generic task prompt

Style 2: Task + Input/Output example

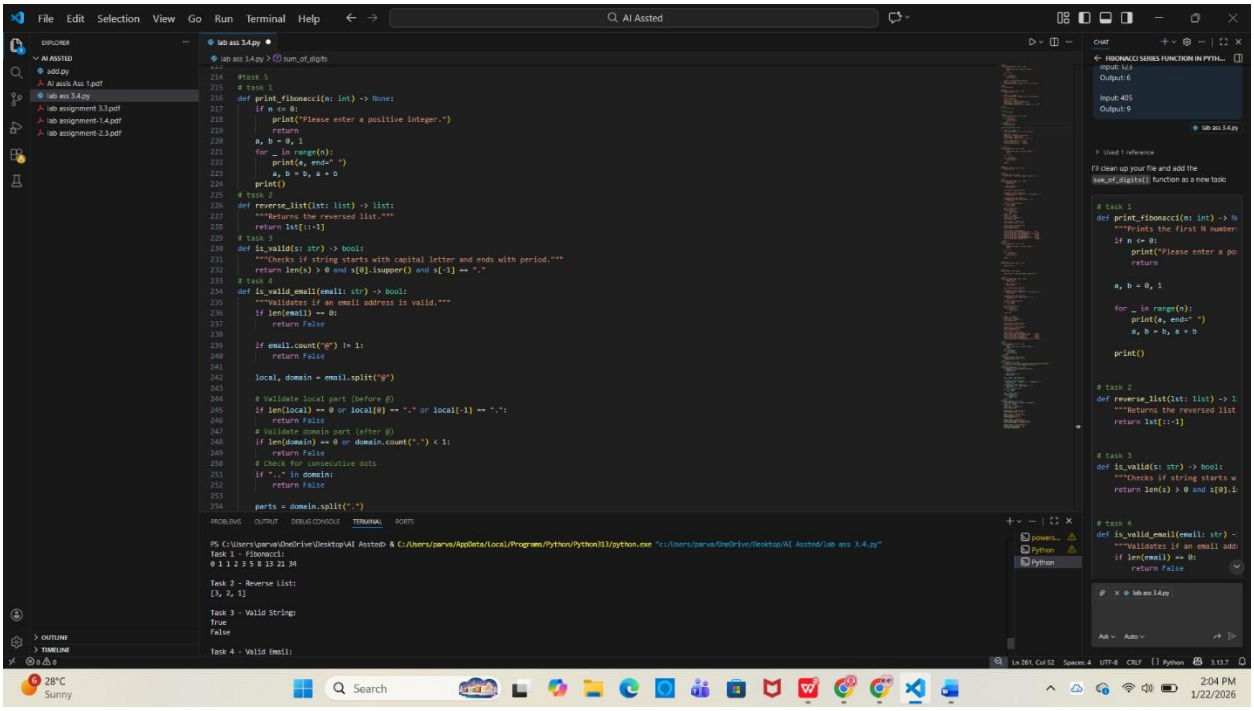
Expected Output #5

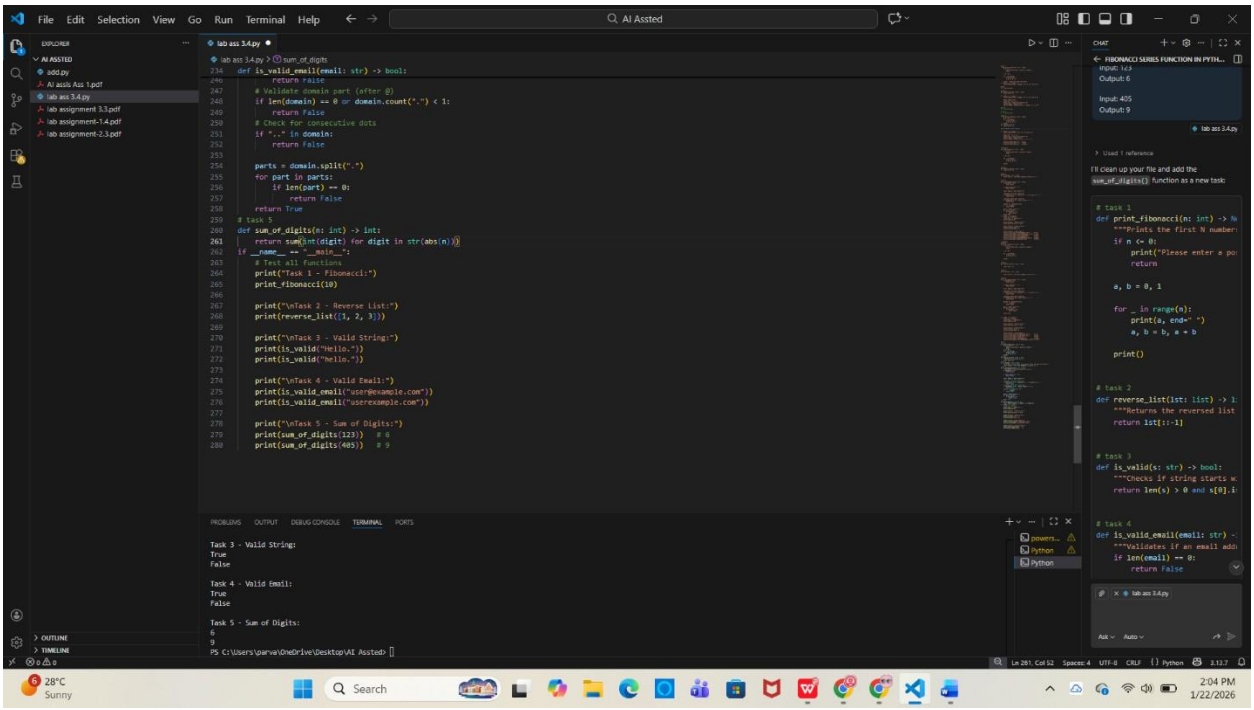
• Two versions of the sum_of_digits() function.

• Example Output: sum_of_digits(123) → 6

• Short analysis: which prompt produced cleaner or more

optimized code and why?


File Edit Selection View Go Run Terminal Help
AI Assted
lab ass 3.4.py
sum_of_digits
#task 5
# task 1
def print_fibonacci(n: int) -> None:
if n <= 0:
print("Please enter a positive integer.")
return
a, b = 0, 1
for _ in range(n):
print(a, end=" ")
a, b = b, a + b
print()
# task 2
def reverse_list(lst: list) -> list:
"""Returns the reversed list."""
return lst[::-1]
# task 3
def is_valid(s: str) -> bool:
"""Checks if string starts with capital letter and ends with period."""
return len(s) > 0 and s[0].isupper() and s[-1] == "."
# task 4
def is_valid_email(email: str) -> bool:
"""Validates if an email address is valid."""
if len(email) == 0:
return False
if email.count("@") != 1:
return False
local, domain = email.split("@")
# Validate local part (before @)
if len(local) == 0 or local[0] == "." or local[-1] == ".":
return False
# Validate domain part (after @)
if len(domain) == 0 or domain.count(".") < 1:
return False
# Check for consecutive dots
if ".." in domain:
return False
parts = domain.split(".")
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\parva\OneDrive\Desktop\AI Assted> & C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/parva/OneDrive/Desktop/AI Assted/lab ass 3.4.py"
Task 1 - Fibonacci:
0 1 1 2 3 5 8 13 21 34
Task 2 - Reverse List:
[3, 2, 1]
Task 3 - Valid String:
True
False
Task 4 - Valid Email:
CHAT
FIBONACCI SERIES FUNCTION IN PYTH...
Output: 6
Input: 405
Output: 9
lab ass 3.4.py
Used 1 reference
I'll clean up your file and add the sum_of_digits() function as a new task:
Ln 261, Col 52 Spaces: 4 UTF-8 CRLF Python 3.13.7
28°C Sunny
Search
2:04 PM 1/22/2026


File Edit Selection View Go Run Terminal Help
AI Assted
lab ass 3.4.py
sum_of_digits
def is_valid_email(email: str) -> bool:
# Validate domain part (after @)
if len(domain) == 0 or domain.count(".") < 1:
return False
# Check for consecutive dots
if ".." in domain:
return False
parts = domain.split(".")
for part in parts:
if len(part) == 0:
return False
return True
# task 5
def sum_of_digits(n: int) -> int:
return sum(int(digit) for digit in str(abs(n)))
if __name__ == "__main__":
# Test all functions
print("Task 1 - Fibonacci:")
print_fibonacci(10)
print("\nTask 2 - Reverse List:")
print(reverse_list([1, 2, 3]))
print("\nTask 3 - Valid String:")
print(is_valid("Hello."))
print(is_valid("hello."))
print("\nTask 4 - Valid Email:")
print(is_valid_email("user@example.com"))
print(is_valid_email("userexample.com"))
print("\nTask 5 - Sum of Digits:")
print(sum_of_digits(123)) # 6
print(sum_of_digits(405)) # 9
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Task 3 - Valid String:
True
False
Task 4 - Valid Email:
True
False
Task 5 - Sum of Digits:
6
9
PS C:\Users\parva\OneDrive\Desktop\AI Assted>
CHAT
FIBONACCI SERIES FUNCTION IN PYTH...
Output: 6
Input: 405
Output: 9
lab ass 3.4.py
Used 1 reference
I'll clean up your file and add the sum_of_digits() function as a new task:
Ln 261, Col 52 Spaces: 4 UTF-8 CRLF Python 3.13.7
28°C Sunny
Search
2:04 PM 1/22/2026