

ASSIGNMENT - 3.4

2303A51060

Batch-10

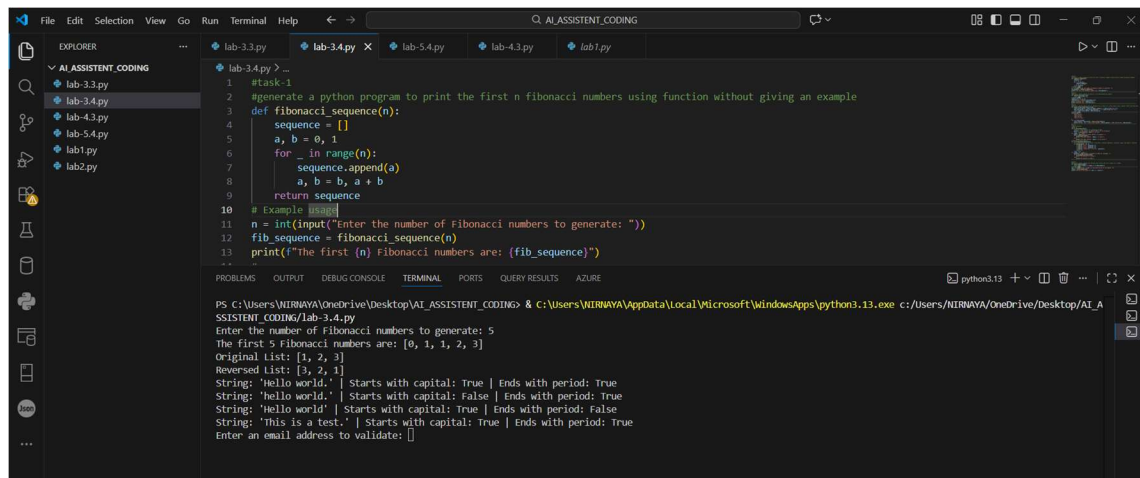
Task-1

Prompt: generate a Python program to print the first n Fibonacci numbers using a function without giving an example

Code:

```
def fibonacci_sequence(n):  
    sequence = []  
    a, b = 0, 1  
    for _ in range(n):  
        sequence.append(a)  
        a, b = b, a + b  
    return sequence  
  
# Example usage  
n = int(input("Enter the number of Fibonacci numbers to generate: "))  
fib_sequence = fibonacci_sequence(n)  
print(f"The first {n} Fibonacci numbers are: {fib_sequence}")
```

Output :



Code Analysis :

- The function `fibonacci_sequence(n)` generates Fibonacci numbers iteratively.
- Variables `a` and `b` store the previous two Fibonacci values.
- A `for` loop runs `n` times to generate required numbers.
- Each generated number is stored in a list for easy return.
- Function-based approach improves reusability and clarity.

Task-2

Prompt: **generate a Python program to reverse a list and provide one example**

Code:

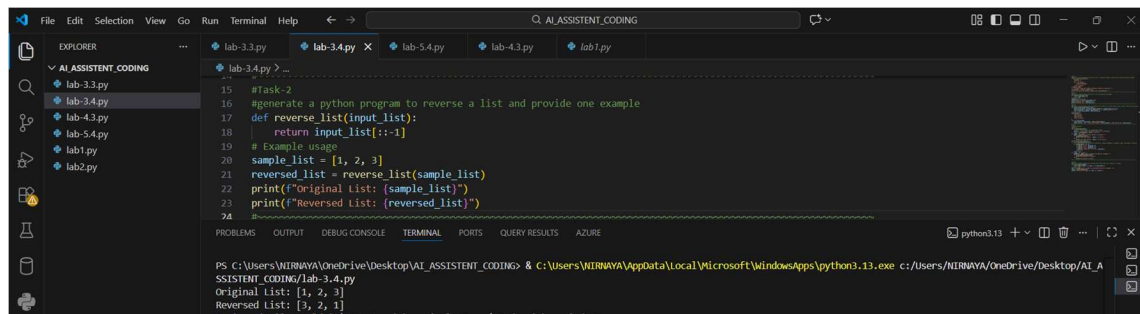
```

def reverse_list(input_list):
    return input_list[::-1]

# Example usage
sample_list = [1, 2, 3]
reversed_list = reverse_list(sample_list)
print(f"Original List: {sample_list}")
print(f"Reversed List: {reversed_list}")

```

Output :



Code Analysis :

- The function `reverse_list()` accepts a list as input.
- Python slicing `[::-1]` is used for efficient reversal.
- No additional loop or memory-intensive operations are required.
- Original list remains unchanged, ensuring data safety.
- Function allows reuse for any list input.

Task-3

Prompt : generate a Python program with 2-3 examples of how to check if a string starts with a capital letter and ends with a period using a function.

Code :

```

def check_string_format(input_string):

    starts_with_capital = input_string[0].isupper() if input_string else False

    ends_with_period = input_string.endswith('.') if input_string else False

    return starts_with_capital, ends_with_period

# Example usage

test_strings = [

    "Hello world.",

    "hello world.",

    "Hello world",

    "This is a test."

]

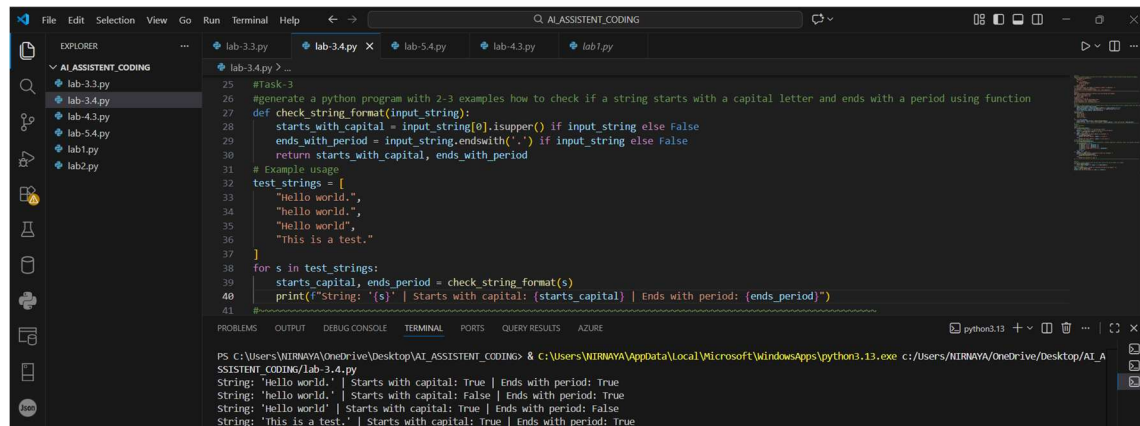
for s in test_strings:

    starts_capital, ends_period = check_string_format(s)

```

```
print(f"String: '{s}' | Starts with capital: {starts_capital} | Ends with period: {ends_period}")
```

Output :



The screenshot shows a VS Code editor with a file explorer on the left containing several Python files (lab-3.3.py to lab-3.7.py). The main editor window displays a Python script named lab-3.4.py. The script defines a function `check_string_format` that takes an input string and returns two Boolean values: `starts_with_capital` (checked using `input_string[0].isupper()`) and `ends_with_period` (checked using `input_string.endswith('.')`). It then uses `format()` to print these results for a list of test strings. The terminal at the bottom shows the execution of the script, displaying the output for each test string.

```
25 #task-3
26 #generate a python program with 2-3 examples how to check if a string starts with a capital letter and ends with a period using function
27 def check_string_format(input_string):
28     starts_with_capital = input_string[0].isupper() if input_string else False
29     ends_with_period = input_string.endswith('.') if input_string else False
30     return starts_with_capital, ends_with_period
31 # Example usage
32 test_strings = [
33     "Hello world.",
34     "hello world.",
35     "Hello world",
36     "This is a test."
37 ]
38 for s in test_strings:
39     starts_capital, ends_period = check_string_format(s)
40     print(f"String: '{s}' | Starts with capital: {starts_capital} | Ends with period: {ends_period}")
41 #
```

```
PS C:\Users\NIRNAYA\OneDrive\Desktop\AI_ASSISTENT_CODING> & C:\Users\NIRNAYA\AppData\Local\Microsoft\WindowsApps\python3.13.exe c:/Users/NIRNAYA/OneDrive/Desktop/AI_A
String: 'Hello world.' | Starts with capital: True | Ends with period: True
String: 'hello world.' | Starts with capital: False | Ends with period: True
String: 'Hello world' | Starts with capital: True | Ends with period: False
String: 'This is a test.' | Starts with capital: True | Ends with period: True
```

Code Analysis :

- The function checks both starting and ending conditions of a string.
- `isupper()` verifies whether the first character is capitalized.
- `endswith('.')` confirms proper sentence termination.
- Handles empty strings safely using conditional checks.
- Returns multiple Boolean values for detailed validation.

Task-4

Prompt: **Email Validator**

Code:

```
import re
```

```
def is_valid_email(email):
```

```
    # Define a regex pattern for validating an Email
```

```
    pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
```

```
    return re.match(pattern, email) is not None
```

```
if __name__ == "__main__":
```

```
    email = input("Enter an email address to validate: ")
```

```
    if is_valid_email(email):
```

```

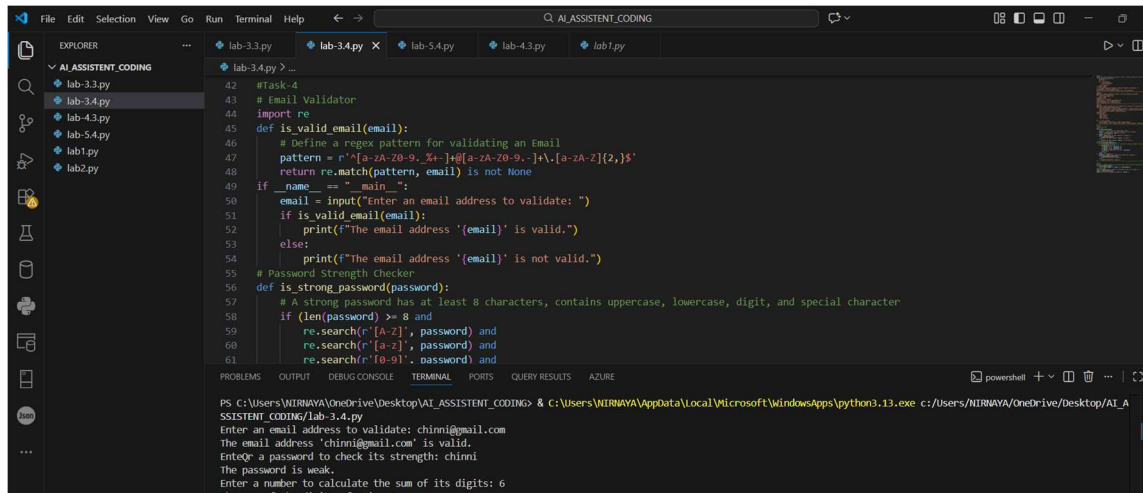
    print(f"The email address '{email}' is valid.")
else:
    print(f"The email address '{email}' is not valid.")

# Password Strength Checker
def is_strong_password(password):
    # A strong password has at least 8 characters, contains uppercase, lowercase, digit, and
    # special character
    if (len(password) >= 8 and
        re.search(r'[A-Z]', password) and
        re.search(r'[a-z]', password) and
        re.search(r'[0-9]', password) and
        re.search(r'[!@#$%^&*().,?":{}|<>]', password)):
        return True
    return False

if __name__ == "__main__":
    password = input("Enter a password to check its strength: ")
    if is_strong_password(password):
        print("The password is strong.")
    else:
        print("The password is weak.")

```

Output :



```
42 #Task-4
43 # Email Validator
44 import re
45 def is_valid_email(email):
46     # Define a regex pattern for validating an Email
47     pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
48     return re.match(pattern, email) is not None
49 if __name__ == "__main__":
50     email = input("Enter an email address to validate: ")
51     if is_valid_email(email):
52         print(f"The email address '{email}' is valid.")
53     else:
54         print(f"The email address '{email}' is not valid.")
55 # Password Strength Checker
56 def is_strong_password(password):
57     # A strong password has at least 8 characters, contains uppercase, lowercase, digit, and special character
58     if (len(password) >= 8 and
59         re.search(r'[A-Z]', password) and
60         re.search(r'[a-z]', password) and
61         re.search(r'[0-9]', password) and
```

PS C:\Users\NIRNAYA\OneDrive\Desktop\AI_ASSISTENT_CODING> & C:\Users\NIRNAYA\AppData\Local\Microsoft\WindowsApps\python3.13.exe c:\Users\NIRNAYA\OneDrive\Desktop\AI_ASSISTENT_CODING\lab-3.4.py

Enter an email address to validate: chinni@gmail.com

The email address 'chinni@gmail.com' is valid.

Enter a password to check its strength: chinni

The password is weak.

Enter a number to calculate the sum of its digits: 6

The sum of the digits of 6 is 6.

Code Analysis :

- Regular expressions (`re`) are used for pattern matching.
- Email validation ensures correct structure using a defined regex.
- Password checker verifies length, case, digits, and special characters.
- Separate functions improve modularity and readability.
- Enhances security by validating user credentials effectively.

Task 5

Prompt: **generate a Python program with a function that returns the sum of the digits of a number**

Code:

```
def sum_of_digits(number):
    return sum(int(digit) for digit in str(abs(number)))

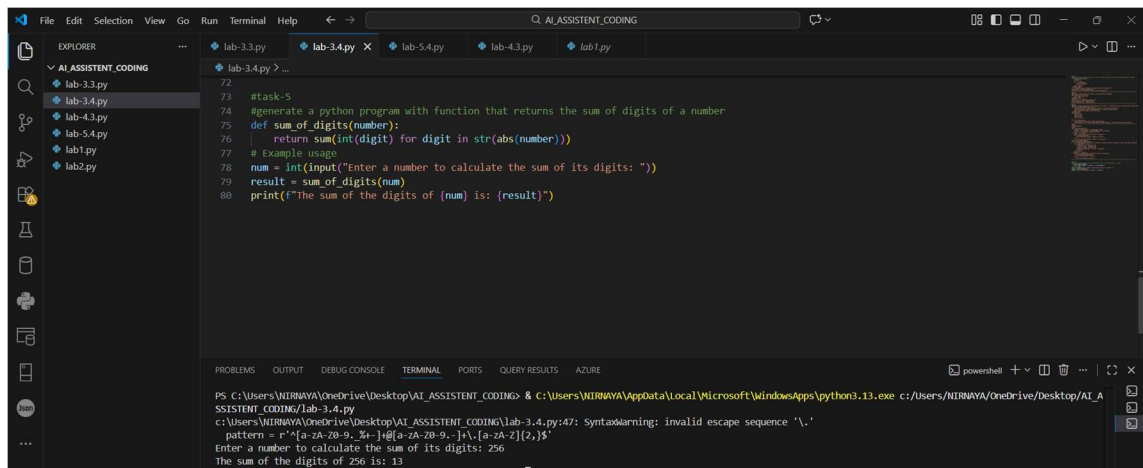
# Example usage

num = int(input("Enter a number to calculate the sum of its digits: "))

result = sum_of_digits(num)

print(f"The sum of the digits of {num} is: {result}")
```

Output :



The screenshot shows a Visual Studio Code editor window with the file explorer on the left displaying a project named 'AI_ASSISTENT_CODING'. The main editor area shows a Python file 'lab-3.4.py' with the following code:

```
72
73 #task-5
74 #generate a python program with function that returns the sum of digits of a number
75 def sum_of_digits(number):
76     return sum(int(digit) for digit in str(abs(number)))
77 # Example usage
78 num = int(input("Enter a number to calculate the sum of its digits: "))
79 result = sum_of_digits(num)
80 print(f"The sum of the digits of {num} is: {result}")
```

Below the editor, the terminal window shows the command prompt output:

```
PS C:\Users\NIRNAYA\OneDrive\Desktop\AI_ASSISTENT_CODING> & C:\Users\NIRNAYA\AppData\Local\Microsoft\WindowsApps\python3.13.exe c:/Users/NIRNAYA/OneDrive/Desktop/AI_A
SSISTENT_CODING/lab-3.4.py
c:\Users\NIRNAYA\OneDrive\Desktop\AI_ASSISTENT_CODING\lab-3.4.py:47: SyntaxWarning: invalid escape sequence '\.'
  pattern = r"[0-2A-Z0-9_?> ]*[0-2A-Z0-9_?> ]A\.[0-2A-Z]{2,}$"
Enter a number to calculate the sum of its digits: 256
The sum of the digits of 256 is: 13
```

Code Analysis :

- The function converts the number into a string for easy digit access.
- `abs()` ensures correct handling of negative numbers.
- `int()` converts each character back to a digit.
- `sum()` efficiently adds all digits in one line.
- Function returns the result, supporting reuse in other programs.