

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 :

```

1 #task-1
2 #generate a python program to print the first n fibonacci numbers using function without giving an example
3 def fibonacci_sequence(n):
4     sequence = []
5     a, b = 0, 1
6     for _ in range(n):
7         sequence.append(a)
8         a, b = b, a + b
9     return sequence
10
11 # Example usage
12 n = int(input("Enter the number of Fibonacci numbers to generate: "))
13 fib_sequence = fibonacci_sequence(n)
14 print(f"The first {n} Fibonacci numbers are: {fib_sequence}")

```

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

Enter the number of Fibonacci numbers to generate: 5

The First 5 Fibonacci numbers are: [0, 1, 1, 2, 3]

Original List: [1, 2, 3]

Reversed List: [3, 2, 1]

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

Enter an email address to validate: []

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 :

```

File Edit Selection View Go Run Terminal Help < > Q AIASSISTENT_CODING
EXPLORER ... lab-3.3.py lab-3.4.py x lab-3.4.py lab-4.3.py lab1.py
AI ASSISTENT CODING
lab-3.4.py > ...
15 #Task-2
16 #generate a python program to reverse a list and provide one example
17 def reverse_list(input_list):
18     return input_list[::-1]
19 # Example usage
20 sample_list = [1, 2, 3]
21 reversed_list = reverse_list(sample_list)
22 print("Original List: ", sample_list)
23 print("Reversed List: ", reversed_list)
24

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULTS AZURE
python3.13 + x ... x
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
Original List: [1, 2, 3]
Reversed List: [3, 2, 1]

```

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 code editor interface with the following details:

- File Explorer:** Shows files like lab-3.3.py, lab-3.4.py, lab-4.3.py, lab-5.4.py, lab1.py, and lab2.py.
- Code Editor:** The active file is lab-3.4.py, containing Python code for validating strings based on capitalization and punctuation.
- Terminal:** Shows the command `python3.13` being run, followed by the output of the program's execution.
- Output:** The terminal output shows four test strings: "Hello world.", "hello world.", "Hello world", and "This is a test.". The program correctly identifies the first two as starting with a capital letter and ending with a period, while the last two are identified as not meeting both criteria.

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 :

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

- File Explorer:** Shows files in the workspace, including `lab-3.3.py`, `lab-3.4.py` (the active file), `lab-4.3.py`, and `lab1.py`.
- Terminal:** The terminal window displays Python code for validating email addresses and checking password strength. It includes regex patterns for email validation and logic to print validation messages.
- Output:** Shows the command prompt and the path to the workspace.
- Terminal Output:** Displays the execution of the script and its output, including an email validation and a password strength check.

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

Output :

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

PS C:\Users\NIRNAYA\OneDrive\Desktop\AI_ASSISTANT_CODING & C:\Users\NIRNAYA\AppData\Local\Microsoft\WindowsApps\python3.13.exe c:/Users/NIRNAYA/OneDrive/Desktop/AI_ASSISTANT_CODING/lab-3.4.py
c:\Users\NIRNAYA\OneDrive\Desktop\AI_ASSISTANT_CODING\lab-3.4.py:47: SyntaxWarning: invalid escape sequence '\'.
 pattern = r'^[a-zA-Z0-9.-]+@[a-zA-Z0-9.-]+\.[a-zA-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.