

Assignment-6.3

H.NO:2303A54060

Batch:47_B

1. Scenario

You are developing a simple student information management module.

Task

- Use an AI tool (GitHub Copilot / Cursor AI / Gemini) to complete a Student class.
- The class should include attributes such as name, roll number, and branch.
- Add a method `display_details()` to print student information.
- Execute the code and verify the output.
- Analyze the code generated by the AI tool for correctness and clarity.

Code & Output:

The screenshot shows a code editor with a Python file named `Lab6.py`. The code defines a `Student` class with an `__init__` method to initialize attributes `name`, `roll_number`, and `branch`. It also includes a `display_details` method to print these attributes. Two student objects, `student1` and `student2`, are created and their details are printed. The terminal below shows the execution of the script and the resulting output.

```
Lab6.py > ...
6
7
8 class Student:
9     def __init__(self, name, roll_number, branch):
10         # Initializing attributes
11         self.name = name
12         self.roll_number = roll_number
13         self.branch = branch
14
15     def display_details(self):
16         # Method to display student information
17         print("Student Details:")
18         print(f"Name : {self.name}")
19         print(f"Roll Number: {self.roll_number}")
20         print(f"Branch : {self.branch}")
21
22
23 # Example usage:
24 student1 = Student("Hasitha", "101", "Computer Science")
25 student2 = Student("Ravi", "102", "Mechanical Engineering")
26
27 # Display details of students
28 student1.display_details()
29 print() # Just for spacing
30 student2.display_details()
31
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Asus\OneDrive\Desktop\AIAC> & C:/Users/Asus/AppData/Local/Programs/Python/Python313/python.exe c:/Users/Asus/OneDrive/Desktop/AIAC/Lab6.py
Name      : Ravi
Roll Number: 102
Branch    : Mechanical Engineering
PS C:\Users\Asus\OneDrive\Desktop\AIAC>
```

Python Python Python Python

Prompt: You are in charge of developing a simple student information management module.create a student class while making sure the class includes attributes such as name,roll number and branch.Include a method display_details() to print student information.create all this using python

Code-Explanation: the student class consists of attributes such as Name,Roll number,Branch.Multiple objects such as student1 and student2 are created and their details are accessed outside the class using display_details() method.

2. Scenario

You are writing a utility function to display multiples of a given number.

Task

- **Prompt the AI tool to generate a function that prints the first 10 multiples of a given number using a loop.**
 - **Analyze the generated loop logic.**
 - **Ask the AI to generate the same functionality using another controlled looping structure (e.g., while instead of for).**

code & Output:

```
33 #> .
34
35 # Case 1: Using a for loop
36 def print_multiples_for(num):
37     print(f"First 10 multiples of {num} using for loop:")
38     for i in range(1, 11): # Loop from 1 to 10
39         print(f"{num} x {i} = {num * i}")
40
41
42 # Case 2: Using a while loop
43 def print_multiples_while(num):
44     print(f"\nFirst 10 multiples of {num} using while loop:")
45     i = 1
46     while i <= 10: # Loop until i reaches 10
47         print(f"{num} x {i} = {num * i}")
48         i += 1
49
50
51 # Example usage:
52 print_multiples_for(5)
53 print_multiples_while(7)
54
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Asus\Desktop\AIAC & C:/Users/Asus/AppData/Local/Programs/Python/Python313/python.exe c:/Users/Asus/Desktop/AIAC/Lab6.py
First 10 multiples of 5 using for loop:
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
```

Python Python Python Python

Prompt: you are writing a utility function to display multiples of a given number.make sure to print the first 10 multiples of a given number using a loop.print 2 cases where in one case you use a for loop and in one case you use a while loop to execute it **Code Explanation:** Here,For loop version: Iterates directly over numbers 1-10 using range(1, 11).Where as While loop version Starts with i = 1 and increments until i <= 10..Both prints the same values.

3. Scenario

You are building a basic classification system based on age.

Task

- Ask the AI tool to generate nested if-elif-else conditional statements to classify age groups (e.g., child, teenager, adult, senior).
- Analyze the generated conditions and logic.
- Ask the AI to generate the same classification using alternative conditional structures (e.g., simplified conditions or dictionary-based logic).

code & output:

```
Lab6.py
56
57 #3.
58 def classify_age_nested(age):
59     """Return age group using nested if/elif/else."""
60     if age < 0:
61         return "invalid age"
62     else:
63         if age <= 12:
64             return "child"
65         elif age <= 19:
66             return "teenager"
67         else:
68             if age <= 64:
69                 return "adult"
70             else:
71                 return "senior"
72
73 # Examples
74 for a in [5, 16, 30, 70, -1]:
75     print(a, "+", classify_age_nested(a))
76
77 #dictionary-based logic
78 def classify_by_ranges(age):
79     ranges = {
80         (0, 12): "child",
81         (13, 19): "teenager",
82         (20, 64): "adult",
83         (65, float("inf")): "senior",
84     }
85     if age < 0:
86         return "invalid age"
87     for (lo, hi), label in ranges.items():
88         if lo <= age <= hi:
89             return label
90
91
92
93
94
95
96
97
98
99
99
```

```
Lab5.py
1 AGE GROUP CLASSIFICATION USING CONDITIONAL STATEMENTS
2
3 def classify_age_nested(age):
4     """Return age group using nested if/elif/else."""
5     if age < 0:
6         return "invalid age"
7     else:
8         if age <= 12:
9             return "child"
10        elif age <= 19:
11            return "teenager"
12        else:
13            if age <= 64:
14                return "adult"
15            else:
16                return "senior"
17
18 examples
19 a in [5, 16, 30, 70, -1]:
20 print(a, "+", classify_age_nested(a))
21
22
23
24
25
26
27
28
29
29
```

CHAT

← AGE GROUP CLASSIFICATION USING CONDITIONAL STATEMENTS

return age group using nested if/elif/else

if age < 0:
 return "invalid age"
else:
 if age <= 12:
 return "child"
 elif age <= 19:
 return "teenager"
 else:
 if age <= 64:
 return "adult"
 else:
 return "senior"

examples

a in [5, 16, 30, 70, -1]:
print(a, "+", classify_age_nested(a))

2) Dictionary-based alternatives 🌟

a) Range keys lookup

```
def classify_by_ranges(age):
    ranges = {
        (0, 12): "child",
        (13, 19): "teenager",
        (20, 64): "adult",
        (65, float("inf")): "senior"
    }
    if age < 0:
        return "invalid age"
    for (lo, hi), label in ranges.items():
        if lo <= age <= hi:
            return label
```

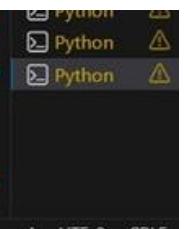
Describe what to build next

Agent Auto ⚡

```

● PS C:\Users\Asus\OneDrive\Desktop\AIAC> & C:/Users/Asus/AppData/Local/Programs/Python/Python313/python.exe c:/Users/Asus/OneDrive/Desktop/AIAC/Lab6.py
5 → child
16 → teenager
30 → adult
70 → senior
-1 → invalid age
○ PS C:\Users\Asus\OneDrive\Desktop\AIAC>

```



Prompt:

imagine you are building a classification system based on age.use nested if-elif-else control statements to classify age groups(child,teenager,adult,senior).Also provide the same classification using alternative conditional structures such as dictionary-based logic.

code-Explanation

Nested if-else are simple and fast.you can use them when a problem demands fixed rules whereas dictionary based approach is more flexible and easier to change but it is slightly slower because it iterates and evaluates functions

4. Scenario

You need to calculate the sum of the first n natural numbers.

Task

- Use AI assistance to generate a `sum_to_n()` function using a `for` loop.
- Analyze the generated code.
- Ask the AI to suggest an alternative implementation using a `while` loop or a mathematical formula.

code & output:

```

Lab6.py x Lab5.py
Lab6.py > print_sum_natural
92
93 def print_sum_natural(n):
94     """Print the sum of the first n natural numbers (1..n)."""
95     try:
96         n = int(n)
97     except (TypeError, ValueError):
98         print("Invalid input: n must be an integer")
99         return
100    if n < 0:
101        print("Invalid input: n must be non-negative")
102        return
103
104    total = 0
105    for i in range(1, n + 1):
106        total += i
107    print(total)
108
109
110 if __name__ == "__main__":
111     try:
112         n = int(input("Enter n: "))
113     except EOFError:
114         print("No input received, exiting.")
115     else:
116         print_sum_natural(n)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Asus\OneDrive\Desktop\AIAC> & C:/Users/Asus/AppData/Local/Programs/Python/Python313/python.exe c:/users/asus/onedrive/desktop/aiac/lab6.py
File "c:/users/asus/onedrive/desktop/aiac/lab6.py", line 109, in <module>
    n=input()
ValueError: invalid literal for int() with base 10: '% C:/Users/Asus/AppData/Local/Programs/Python/Python313/python.exe c:/users/asus/onedrive/desktop/aiac/lab6.py'
PS C:\Users\Asus\OneDrive\Desktop\AIAC> & C:/Users/Asus/AppData/Local/Programs/Python/Python313/python.exe c:/users/asus/onedrive/desktop/aiac/lab6.py
Enter n: 10
55
○ PS C:\Users\Asus\OneDrive\Desktop\AIAC>

```



Prompt:

you are building a function to print the sum of n natural numbers.use a for loop to get the output

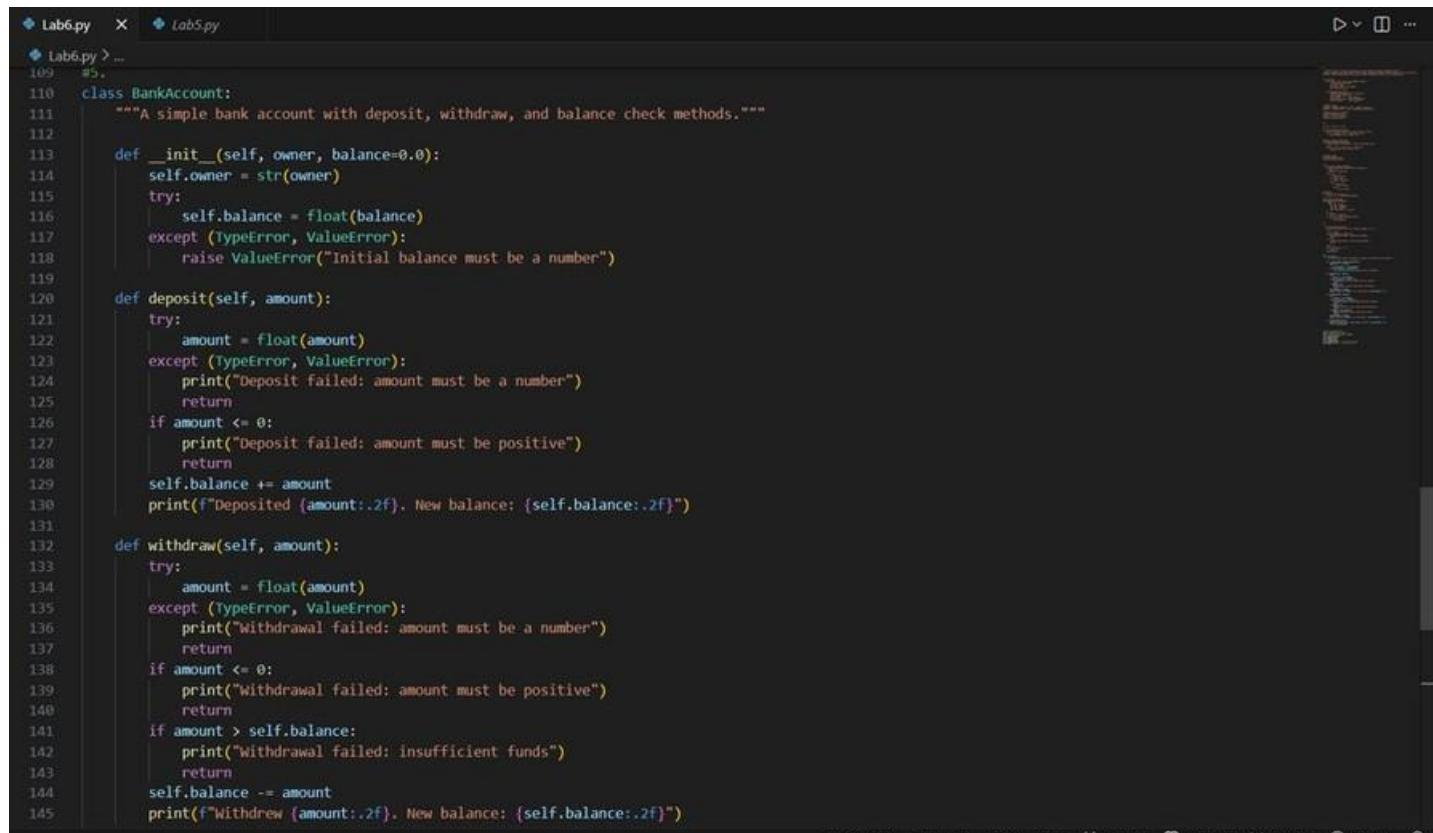
5. Scenario

You are designing a basic banking application.

Task

- Use AI tools to generate a Bank Account class with methods such as deposit(), withdraw(), and check_balance().
- Analyze the AI-generated class structure and logic.
- Add meaningful comments and explain the working of the code.

code & output:



```
Lab6.py  X  Lab5.py
Lab6.py > ...
109      #5.
110  class BankAccount:
111      """A simple bank account with deposit, withdraw, and balance check methods."""
112
113      def __init__(self, owner, balance=0.0):
114          self.owner = str(owner)
115          try:
116              self.balance = float(balance)
117          except (TypeError, ValueError):
118              raise ValueError("Initial balance must be a number")
119
120      def deposit(self, amount):
121          try:
122              amount = float(amount)
123          except (TypeError, ValueError):
124              print("Deposit failed: amount must be a number")
125              return
126          if amount <= 0:
127              print("Deposit failed: amount must be positive")
128              return
129          self.balance += amount
130          print(f"Deposited {amount:.2f}. New balance: {self.balance:.2f}")
131
132      def withdraw(self, amount):
133          try:
134              amount = float(amount)
135          except (TypeError, ValueError):
136              print("Withdrawal failed: amount must be a number")
137              return
138          if amount <= 0:
139              print("Withdrawal failed: amount must be positive")
140              return
141          if amount > self.balance:
142              print("Withdrawal failed: insufficient funds")
143              return
144          self.balance -= amount
145          print(f"Withdraw {amount:.2f}. New balance: {self.balance:.2f}")
```

```
Lab6.py  X  Lab5.py
Lab6.py > ...
110  class BankAccount:
111      def withdraw(self, amount):
112          if amount <= 0:
113              print("Withdrawal failed: amount must be positive")
114              return
115          if amount > self.balance:
116              print("Withdrawal failed: insufficient funds")
117              return
118          self.balance -= amount
119          print(f"Withdrew {amount:.2f}. New balance: {self.balance:.2f}")
120
121  def check_balance(self):
122      print(f"Account owner: {self.owner}, balance: {self.balance:.2f}")
123      return self.balance
124
125
126
127  # Demo for BankAccount
128  print('\nBankAccount demo:')
129  acct = BankAccount("Alice", 100.0)
130  acct.check_balance()
131  acct.deposit(50)
132  acct.withdraw(30)
133  acct.withdraw(200) # insufficient funds
134  acct.deposit("abc") # invalid deposit
```

```
PS C:\Users\Asus\OneDrive\Desktop\AIAC> & C:/Users/Asus/AppData/Local/Programs/Python/Python313/python.exe
BankAccount demo:
Account owner: Alice, balance: 100.00
Deposited 50.00. New balance: 150.00
Withdrew 30.00. New balance: 120.00
Withdrawal failed: insufficient funds
Deposit failed: amount must be a number
PS C:\Users\Asus\OneDrive\Desktop\AIAC> [ ]
```

Prompt:

you are designing a banking application.create a Bank Account class with methods such as deposit(),withdraw(),and check_balance().