

Assignment-6.3

Hall ticket no:2303A51523

Type your text

Task Description #1: Classes (Student Class)

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.

Prompt:develop a simple student information management module using python

In this task, an AI tool was used to generate a Python `Student` class with attributes such as name, roll number, and branch. The class included a constructor to initialize values and a `display_details()` method to print student information. The generated code was clear, correct, and followed basic object-oriented programming principles.

The screenshot shows a code editor interface with a dark theme. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a search bar. The left sidebar shows file navigation. The main code area contains the following Python code:

```

C:\> Users > appam > AppData > Local > Packages > 5319275A.What'sAppDesktop_cvlg1gavmjgm > LocalState > sessions > ADE69961C7978A860A3B495981C384FAD2CC1095 > transfers > 2026-05 > assignment 6.3.py > Student
1 # Lab 6: AI-Based Code Completion - Classes, Loops, and Conditionals
2
3 print("/*")
4 print("TASK 1: STUDENT CLASS")
5 print("*/")
6
7 # Task 1: Student Class
8 class Student:
9     """A class to represent a student with basic information."""
10
11     def __init__(self, name, roll_number, branch):
12         """
13             Constructor to initialize student attributes.
14
15             Args:
16                 name (str): Student's name
17                 roll number (int): Student's roll number
18                 branch (str): Student's branch/department
19
20             ...
21
22             self.name = name
23             self.roll_number = roll_number
24             self.branch = branch

```

Below the code, there is a PROBLEMS tab with one error, an OUTPUT tab, and a DEBUG CONSOLE tab. The DEBUG CONSOLE tab shows the following output:

```

=====
TASK 1: STUDENT CLASS
=====

Student 1:
Name: Alice Johnson
Roll Number: 101
Branch: Computer Science

Student 2:
Name: Anil Yadav
Roll Number: 102
Branch: Electronics

```

The status bar at the bottom shows the terminal type as powershell, Python version as 3.14.3, and the date/time as 04-02-2026.

Task Description #2: Loops (Multiples of a Number)

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).

while instead of for).

Prompt:print first 10 multiples of a number

AI assistance was used to create a function that prints the first ten multiples of a given number using a `for` loop. The same logic was also generated using a `while`

loop. Both approaches produced correct results, with the `for` loop being simpler and more readable, while the `while` loop offered more control.

The screenshot shows a code editor window with a dark theme. The file is named `assignment 6.3.py`. The code contains two functions: `print_multiples_for` and `print_multiples_while`. Both functions print the first 10 multiples of a given number. The `for` loop version is simpler and cleaner, while the `while` loop version is more verbose. The output terminal shows the results for both loops, which are identical. A tooltip provides analysis of the loops: "FOR loop: Best for known iterations, cleaner syntax, more Pythonic". The system tray at the bottom shows the date as 04-02-2026.

```
C:\> Users > appam > AppData > Local > Packages > 5319275A.WhatAppDesktop_cvlgvanyjgm > LocalState > sessions > ADE69961C7978A860A3B495981C384FAD2CC1095 > transfers > 2026-05 > assignment 6.3.py > Student

File Edit Selection View Go Run Terminal Help ← → Search

assignment 6.3.py X
C:\> print(" - 70")
42
43 # Task 2: Loops - Print first 10 multiples of a number
44
45 # Approach 1: Using for loop
46 def print_multiples_for(number):
47     """Print first 10 multiples of a given number using for loop."""
48     print(f"\nFirst 10 multiples of {number} (using for loop):")
49     for i in range(1, 11):
50         multiple = number * i
51         print(f"\t{number} x {i} = {multiple}", end=" ")
52     print()
53
54 # Approach 2: Using while loop
55 def print_multiples_while(number):
56     """Print first 10 multiples of a given number using while loop."""
57     print(f"\nFirst 10 multiples of {number} (using while loop):")
58     i = 1
59     while i <= 10:
60         multiple = number * i
61         print(f"\t{number} x {i} = {multiple}", end=" ")
62         i += 1
63     print()

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

+ ... | x
powershell
Python Deb...
Ln 10, Col 5 Spaces: 4 UTF-8 CRLF Python 3.14.3 09:56
ENG IN 04-02-2026

TASK 2: LOOPS (MULTIPLES OF A NUMBER)
-----
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 5 x 6 = 30 5 x 7 = 35 5 x 8 = 40 5 x 9 = 45 5 x 10 = 50

First 10 multiples of 7 (using while loop):
7 x 1 = 7 7 x 2 = 14 7 x 3 = 21 7 x 4 = 28 7 x 5 = 35 7 x 6 = 42 7 x 7 = 49 7 x 8 = 56 7 x 9 = 63 7 x 10 = 70

Analysis of Looping Approaches:
- FOR loop: Best for known iterations, cleaner syntax, more Pythonic

68°F Sunny
Search
Ln 10, Col 5 Spaces: 4 UTF-8 CRLF Python 3.14.3 09:56
ENG IN 04-02-2026
```

Task Description #3: Conditional Statements (Age Classification)

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).

Prompt:conditional statements and age qualification using nested if-elif-else

explanation:The AI generated a function using `if-elif-else` statements to classify age into child, teenager, adult, and senior categories. The conditions were logically ordered and easy to understand. An alternative approach showed how the same classification could be done using simplified or dictionary-based logic.

```
C:\> Users>appnam>AppData>Local>Packages>5319275AWhatsAppDesktop_cvg1gvanyjgm>LocalStorage>sessions>ADE69961C7978A860A3B495981C384FAD2CC1095>transfers>2026-05>assignment 6.3.py>Student  
75 print("=>76)  
76  
77 # Task 3: Conditional Statements - Age Classification  
78  
79 # Approach 1: Using nested if-elif-else  
80 def classify_age_nested(age):  
81     """Classify age into groups using nested if-elif-else statements."""  
82     if age < 0:  
83         return "Invalid age"  
84     elif age < 13:  
85         return "Child"  
86     elif age < 18:  
87         return "Teenager"  
88     elif age < 60:  
89         return "Adult"  
90     else:  
91         return "Senior"  
92  
93 # Approach 2: Using dictionary-based logic  
94 def classify_age_dict(age):  
95     """Classify age into groups using dictionary and conditional checks."""  
96     age_groups = {  
97         0: "Child",  
98         13: "Teenager",  
99         60: "Adult",  
100        65: "Senior"}  
101  
102 age_classification_nested = classify_age_nested  
103 age_classification_dict = classify_age_dict  
104  
105 age_classification_nested(5)  
106 age_classification_nested(15)  
107 age_classification_nested(25)  
108 age_classification_nested(65)  
109  
110 age_classification_dict(5)  
111 age_classification_dict(15)  
112 age_classification_dict(25)  
113 age_classification_dict(65)  
114  
115 Ln 10, Col 5 Spaces: 4 UTF-8 CRLF Python 3.14.3 1001 ENG IN 04-02-2026
```

Task Description #4: For and While Loops (Sum of First n Numbers)

t

task4.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.

Prompt:give first sum of n natural numbers using for loop and while loop

explanation:AI was used to generate a function that calculates the sum of the first n natural numbers using a `for` loop. Alternative solutions using a `while` loop and a mathematical formula were also suggested. The formula-based approach was the most efficient, while loop-based solutions were easier for beginners.

```
File Edit Selection View Go Run Terminal Help ↻ 🔍 Search ⌂ Assignment_7.ipynb assignment 6.3.py X C: > Users > appam > AppData > Local > Packages > 5319275A.WhatsAppDesktop_cv1g1gvanyjgm > LocalState > sessions > ADE69961C7978A860A3B495981C384FAD2CC1095 > transfers > 2026-05 > assignment 6.3.py > Student 128 # Task 4: Sum of first n natural numbers 129 130 # Approach 1: Using for loop 131 def sum_to_n_for(n): 132     """Calculate sum of first n natural numbers using for loop.""" 133     total = 0 134     for i in range(1, n + 1): 135         total += i 136     return total 137 138 # Approach 2: Using while loop 139 def sum_to_n_while(n): 140     """Calculate sum of first n natural numbers using while loop.""" 141     total = 0 142     i = 1 143     while i <= n: 144         total += i 145         i += 1 146     return total 147 148 # Approach 3: Using mathematical formula 149 def sum_to_n_formula(n): PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL PORTS - Dictionary-based: More maintainable, easier to modify ranges, more scalable TASK 4: FOR AND WHILE LOOPS (SUM OF FIRST N NUMBERS) ----- Sum of First N Numbers: n = 5: FOR=15, WHILE=15, FORMULA=15 n = 10: FOR=55, WHILE=55, FORMULA=55 n = 20: FOR=210, WHILE=210, FORMULA=210 Comparison of Approaches: - FOR loop: Clean syntax, preferred for known iterations Ln 10, Col 5 Spaces: 4 UTF-8 CRLF Python 3.14.3 powershell Python Deb... 69°F Sunny Search ENG IN 10:06 04-02-2026
```

task5:Classes (Bank Account Class)

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

Prompt:create bank account class and demonstrate

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

- File Explorer:** Shows a file named "Assignment_7.ipynb" and a tab titled "assignment 6.3.py x".
- Code Editor:** Displays Python code for a "BankAccount" class. The code includes methods for initializing the account, making deposits, and performing withdrawals. It also maintains a transaction history.
- Terminal:** Shows the command line path: C:\> Users\appam\AppData\Local\Packages\5319275A.WhatsAppDesktop_cv1g1gvanyjgm\LocalState\sessions\ADE69961C7978A860A3B495981C384FAD2CC1095\transfers>2026-05>. The terminal also displays the output of the executed code.
- Output:** Shows the creation of a bank account for John Smith with an initial balance of \$1000, followed by a series of banking operations: depositing \$500.00, withdrawing \$200.00, depositing \$150.00, and withdrawing \$600.00, resulting in a current balance of \$850.00.
- Status Bar:** Provides information about the current file (Assignment_7.ipynb), the terminal (powershell), the Python interpreter (Python 3.14.3), and the system status (CPU 60%, 4 cores, 100% usage).

The screenshot shows a code editor interface with a dark theme. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a search bar. On the left, there are various icons for file operations like Open, Save, Find, and Copy/Paste. The main workspace displays a Python script named `assignment_6.3.py`. The code performs several banking operations on an account object, including deposits and withdrawals, and handles invalid operations. It also prints the final account details and a success message. Below the code, tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS are visible. A terminal window titled "powershell" shows the execution of the script. The bottom status bar indicates the current file is `assignment_6.3.py`, with 1010 lines of code, and provides information about the current session.

```
C:\> Users > appm > AppData > Local > Packages > 5319275A-WhatsAppDesktop_cvg1gvanyjgm > LocalState > sessions > ADE69961C7978A860A3B495981C384FAD2CC1095 > transfers > 2026-05 > assignment_6.3.py > Student

257 print("\nPerforming Banking Operations:")
258 print("-" * 50)
259
260 account.deposit(500)
261 account.withdraw(200)
262 account.deposit(150)
263 account.check_balance()
264 account.withdraw(600)
265 account.check_balance()
266
267 # Try invalid operations
268 print("\nAttempting Invalid operations:")
269 account.withdraw(10000) # Insufficient balance
270 account.deposit(-100) # Invalid amount
271
272 # Display final account details
273 account.display_account_details()
274
275 print("=-*70")
276 print("LAB 6 COMPLETED - ALL TASKS EXECUTED SUCCESSFULLY")
277 print("=-*70")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + × ... | x powershell
Account Holder: John Smith
Current Balance: $550.00
=====
Transaction History:
• Initial deposit: $1000.00
• Deposit: $500.00
• Withdrawal: $200.00
• Deposit: $150.00
• Withdrawal: $600.00
=====
```

