

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: B. Tech		Assignment Type: Lab	
Course Coordinator Name		Dr. Rishabh Mittal	
Instructor(s) Name		Mr. S Naresh Kumar Ms. B. Swathi Dr. Sasanko Shekhar Gantayat Mr. Md Sallauddin Dr. Mathivanan Mr. Y Srikanth Ms. N Shilpa Dr. Rishabh Mittal (Coordinator) Dr. R. Prashant Kumar Mr. Ankushavali MD Mr. B Viswanath Ms. Sujitha Reddy Ms. A. Anitha Ms. M.Madhuri Ms. Katherashala Swetha Ms. Velpula sumalatha Mr. Bingi Raju	
Course Code	23CS002PC304	Course Title	AI Assisted Coding
Year/Sem	III/II	Regulation	R23
Date and Day of Assignment	Week3 – Wednesday	Time(s)	23CSBTB01 To 23CSBTB52
Duration	2 Hours	Applicable to Batches	All batches
AssignmentNumber: 6.3(Present assignment number)/24(Total number of assignments)			
Q. No .	Question		Expected Time to complete

Lab 6: AI-Based Code Completion – Classes, Loops, and Conditionals

Lab Objectives

- To explore AI-powered auto-completion features for core Python constructs such as classes, loops, and conditional statements.
- To analyze how AI tools suggest logic for object-oriented programming and control structures.
- To evaluate the correctness, readability, and completeness of AI-generated Python code.

Lab Outcomes (LOs)

After completing this lab, students will be able to:

- Use AI tools to generate and complete Python class definitions and methods.
- Understand and assess AI-suggested loop constructs for iterative tasks.
- Generate and evaluate conditional statements using AI-driven prompts.
- Critically analyze AI-assisted code for correctness, clarity, and efficiency.

Task Description #1: Classes (Student Class)

Scenario

You are developing a simple student information management module.

Task

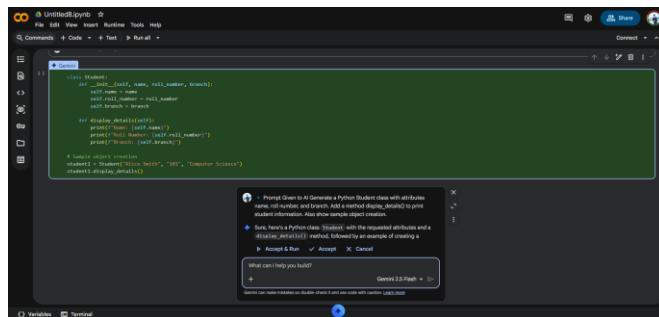
- Use an AI tool (GitHub Copilot / Cursor AI / Gemini) to complete a Student class.
- The class should include attributes such as name, roll number, and branch.
- Add a method display_details() to print student information.
- Execute the code and verify the output.
- Analyze the code generated by the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

Expected Output #1

- A Python class with a constructor (`__init__`) and a `display_details()` method.
- Sample object creation and output displayed on the console.
- Brief analysis of AI-generated code.

Prompt

Generate a Python Student class with attributes name, roll number, and branch. Add a method `display_details()` to print student information. Also show sample object creation.



```
class Student:
    def __init__(self, name, roll_number, branch):
        self.name = name
        self.roll_number = roll_number
        self.branch = branch

    def display_details(self):
        print("Name: ", self.name)
        print("Roll Number: ", self.roll_number)
        print("Branch: ", self.branch)

# sample object creation
student = Student("Alice Smith", "101", "Computer Science")
student.display_details()
```

Week3 -
Wednesday

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).

Expected Output #2

- Correct loop-based Python implementation.
- Output showing the first 10 multiples of a number.
- Comparison and analysis of different looping approaches.

Prompt

Write a function in Python to print first 10 multiples of a number using a for loop.

The screenshot shows a code editor window with the following Python code:

```
def print_multiples(number):
    print(f"Print first 10 multiples of {number}:")
    for i in range(1, 11): # (loop from 1 to 10 (inclusive)
        print(number * i)

# Simple function call
print_multiples(7)
```

The output window below the code shows the generated output:

```
First 10 multiples of 7:
7
14
21
28
35
42
49
56
63
70
```

A tooltip from the AI tool provides the task description:

Write a function in Python to print first 10 multiples of a number using a for loop.
+ Generates a Python function that takes a number as input and prints its first 10 multiples using a 'for' loop.
Working...

- Both implementations correctly print the first 10 multiples.
- **for** loop is simpler and more readable.
- **while** loop gives more control over the iteration variable.
- AI-generated logic is correct and efficient.

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).

Expected Output #3

- A Python function that classifies age into appropriate groups.
- Clear and correct conditional logic.
- Explanation of how the conditions work.

- The nested if-elif-else structure is clear and easy to understand.
- The alternative version uses a list of conditions, making it flexible and cleaner.
- AI-generated conditions are logically correct.

- The nested if-elif-else structure is clear

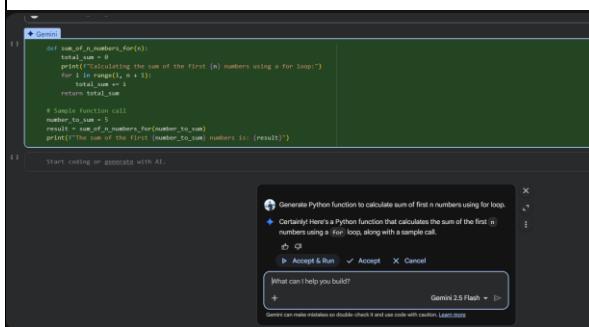
Task Description #4: For and While Loops (Sum of First n Numbers)

Scenario

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

Prompt

Write the same program using a while loop.



```

1 | 1 def sum_of_numbers_for(n):
2 | 2     total_sum = 0
3 | 3     print("Calculating the sum of the first (n) numbers using a for loop:")
4 | 4     for i in range(1, n + 1):
5 | 5         total_sum += i
6 | 6     return total_sum
7 |
8 | 8 # Sample function call
9 | 9 number_to_sum = 10
10|10 result = sum_of_numbers_for(number_to_sum)
11|11 print("The sum of the first (number_to_sum) numbers is: (result)")

```

The screenshot shows the Gemini AI interface. At the top, there's a code editor window with the above Python code. Below it is a modal dialog box with the title "Generate Python function to calculate sum of first n numbers using for loop." It contains the generated code and a message: "Certainly! Here's a Python function that calculates the sum of the first (n) numbers using a 'for' loop, along with a sample call." There are three buttons at the bottom of the dialog: "Accept & Run" (highlighted), "Accept", and "Cancel". At the very bottom of the interface, there's a footer note: "Gemini can make mistakes so double check it and use code with caution. Learn more".

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.

Expected Output #4

- Python function to compute the sum of first n numbers.
- Correct output for sample inputs.
- Explanation and comparison of different approaches.

Prompt

```

def sum_of_n_numbers_for(n):
    total_sum = 0
    print("Calculating the sum of the first (n) numbers using a for loop:")
    for i in range(1, n + 1):
        total_sum += i
    return total_sum

# Single function call
number_to_sum = 5
result = sum_of_n_numbers_for(function_to_sum)
print(f"the sum of the first {number_to_sum} numbers is: {result}")


```

Start coding or generate with AI...

Generate Python function to calculate sum of first n numbers using for loop.

Certainly! Here's a Python function that calculates the sum of the first (n) numbers using a for-loop, along with a sample call.

Accept & Run Accept Cancel

What can I help you build?

Gemini 2.5 Flash

Task Description #5: 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.

Expected Output #5

- Complete Python Bank Account class.
- Demonstration of deposit and withdrawal operations with updated balance.
- Well-commented code with a clear explanation.

Note: Report should be submitted as a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots.

Prompt

write a class for bank application with deposit and withdraw methods

```

class BankAccount:
    def __init__(self, account_number, account_holder_name, initial_balance=0):
        self.account_number = account_number
        self.account_holder_name = account_holder_name
        self.initial_balance = initial_balance
        print(f"Account {self.account_number} created for {self.account_holder_name} with initial balance: ${self.balance}")

    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            print(f"Deposited ${amount}. New balance: ${self.balance}")
        else:
            print("Deposit amount must be positive!")

    def withdraw(self, amount):
        if amount > 0:
            if amount < self.balance:
                self.balance -= amount
                print(f"Withdrew ${amount}. New balance: ${self.balance}")
            else:
                print("Insufficient funds. Current balance: ${self.balance}")
        else:
            print("Withdrawal amount must be positive!")

    def get_balance(self):
        return self.balance

    def display_account_info(self):
        print(f"Account Number: {self.account_number}")
        print(f"Account Holder Name: {self.account_holder_name}")
        print(f"Current Balance: ${self.balance}")


```

Write a class for bank application with deposit and withdraw methods.

Certainly! Here's a Python class `BankAccount` with `deposit()` and `withdraw()` methods, along with a sample of how to create an account and perform some transactions.

Accept & Run Accept Cancel

What can I help you build?

Gemini 2.5 Flash

- | | | |
|--|---|--|
| | <ul style="list-style-type: none">• AI tools generated correct and readable Python code for classes, loops, and conditionals.• The logic produced was mostly optimal and easy to understand.• Alternative approaches (for vs while, if-else vs simplified logic, formula vs loops) helped in understanding efficiency.• AI assistance speeds up development but human verification is required to ensure correctness and edge-case handling. | |
|--|---|--|