

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
<b>Program Name:</b> B. Tech		<b>Assignment Type:</b> Lab	
<b>Course Coordinator Name</b>		Dr. Rishabh Mittal	
<b>Course Code</b>	23CS002PC304	<b>Course Title</b>	AI Assisted Coding
<b>Year/Sem</b>	III/II	<b>Regulation</b>	R23
<b>Date and Day of Assignment</b>	Week3 – Wednesday	<b>batch</b>	47_B
<b>NAME</b>	V.SUSHMITHA	<b>HALLTICKET. NO</b>	2303A54055
<b>AssignmentNumber: 6.3</b>			

.		
	<h2>Lab 6: AI-Based Code Completion – Classes, Loops, and Conditionals</h2>	
1	<p><b>Task Description #1: Classes (Student Class)</b></p> <p><b>Scenario</b> You are developing a simple student information management module.</p> <p><b>Task:</b> Create a Student class with name, roll number, and branch, and display student details.</p> <p><b>Prompt Used</b> Generate a Python class named Student with attributes name, roll number, and branch. Add a method to display student details.</p> <p><b>Sample Input:</b> Name = Roll Number = Branch =</p> <p><b>Sample Output:</b> Name: Sushmitha Roll Number: 101 Branch: CSE</p> <p><b>Short Explanation:</b> This program uses a class to store student details the constructor initializes values and the method print them neatly</p>	Week3 - Wednesday

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows files in the "AI CODING" folder, including "task1.py", "task2.py", "task3.py", "task4.py", "task5.py", and "assignment3.1.py".
- Code Editor:** The active file is "task1.py", containing the following Python code:

```
#task1: Generate a Python class named Student with attributes name, roll number, and branch. Add a method to display student details.
class Student:
    def __init__(self, name, roll_no, branch):
        self.name = name
        self.roll_no = roll_no
        self.branch = branch
    def display_details(self):
        print("Name:", self.name)
        print("Roll Number:", self.roll_no)
        print("Branch:", self.branch)
# Object creation
s1 = Student("Sushmitha", 101, "CSE")
s1.display_details()
```
- Terminal:** Shows the output of running "task1.py" in Python 3.14.64. The output is:

```
PS C:\Users\bindu\Desktop\AI CODING & C:/Users/bindu/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/bindu/Desktop/AI CODING/task1.py"
Name: Sushmitha
Roll Number: 101
Branch: CSE
PS C:\Users\bindu\Desktop\AI CODING>
```
- Status Bar:** Displays "Line 16, Col 1" and "Python 3.14.2".

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

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

**Task:** Print the first 10 multiples of a given number using loops.

**Prompt Used:** Generate a Python function to print first 10 multiples of a number using for loop and while loop.

**Sample Input:**  
**5**

**Sample Output:**

```
5
10
15
20
25
30
35
40
45
50
```

**Short Explanation:** Both loops repeat the same task.  
The for loop is shorter, while the while loop gives more control over conditions.

The screenshot shows a VS Code interface with the following details:

- File Explorer:** Shows files under "AI CODING" folder:
  - assignment 4.1.py
  - assignment 6.3
  - task1.py
  - task2.py (highlighted)
  - task3.py
  - task4.py
  - task5.py
  - task6.py
  - task7.py
  - task8.py
  - task9.py
  - task10.py
  - task11.py
  - task12.py
  - task13.py
  - task14.py
  - task15.py
  - task16.py
  - task17.py
  - task18.py
  - task19.py
  - task20.py
  - task21.py
  - task22.py
  - task23.py
  - task24.py
  - task25.py
  - task26.py
  - task27.py
  - task28.py
  - task29.py
  - task30.py
  - task31.py
  - task32.py
  - task33.py
  - task34.py
  - task35.py
  - task36.py
  - task37.py
  - task38.py
  - task39.py
  - task40.py
  - task41.py
  - task42.py
  - task43.py
  - task44.py
  - task45.py
  - task46.py
  - task47.py
  - task48.py
  - task49.py
  - task50.py
  - task51.py
  - task52.py
  - task53.py
  - task54.py
  - task55.py
  - task56.py
  - task57.py
  - task58.py
  - task59.py
  - task60.py
  - task61.py
  - task62.py
  - task63.py
  - task64.py
  - task65.py
  - task66.py
  - task67.py
  - task68.py
  - task69.py
  - task70.py
  - task71.py
  - task72.py
  - task73.py
  - task74.py
  - task75.py
  - task76.py
  - task77.py
  - task78.py
  - task79.py
  - task80.py
  - task81.py
  - task82.py
  - task83.py
  - task84.py
  - task85.py
  - task86.py
  - task87.py
  - task88.py
  - task89.py
  - task90.py
  - task91.py
  - task92.py
  - task93.py
  - task94.py
  - task95.py
  - task96.py
  - task97.py
  - task98.py
  - task99.py
  - task100.py
  - task101.py
  - task102.py
  - task103.py
  - task104.py
  - task105.py
  - task106.py
  - task107.py
  - task108.py
  - task109.py
  - task110.py
  - task111.py
  - task112.py
  - task113.py
  - task114.py
  - task115.py
  - task116.py
  - task117.py
  - task118.py
  - task119.py
  - task120.py
  - task121.py
  - task122.py
  - task123.py
  - task124.py
  - task125.py
  - task126.py
  - task127.py
  - task128.py
  - task129.py
  - task130.py
  - task131.py
  - task132.py
  - task133.py
  - task134.py
  - task135.py
  - task136.py
  - task137.py
  - task138.py
  - task139.py
  - task140.py
  - task141.py
  - task142.py
  - task143.py
  - task144.py
  - task145.py
  - task146.py
  - task147.py
  - task148.py
  - task149.py
  - task150.py
  - task151.py
  - task152.py
  - task153.py
  - task154.py
  - task155.py
  - task156.py
  - task157.py
  - task158.py
  - task159.py
  - task160.py
  - task161.py
  - task162.py
  - task163.py
  - task164.py
  - task165.py
  - task166.py
  - task167.py
  - task168.py
  - task169.py
  - task170.py
  - task171.py
  - task172.py
  - task173.py
  - task174.py
  - task175.py
  - task176.py
  - task177.py
  - task178.py
  - task179.py
  - task180.py
  - task181.py
  - task182.py
  - task183.py
  - task184.py
  - task185.py
  - task186.py
  - task187.py
  - task188.py
  - task189.py
  - task190.py
  - task191.py
  - task192.py
  - task193.py
  - task194.py
  - task195.py
  - task196.py
  - task197.py
  - task198.py
  - task199.py
  - task200.py
  - task201.py
  - task202.py
  - task203.py
  - task204.py
  - task205.py
  - task206.py
  - task207.py
  - task208.py
  - task209.py
  - task210.py
  - task211.py
  - task212.py
  - task213.py
  - task214.py
  - task215.py
  - task216.py
  - task217.py
  - task218.py
  - task219.py
  - task220.py
  - task221.py
  - task222.py
  - task223.py
  - task224.py
  - task225.py
  - task226.py
  - task227.py
  - task228.py
  - task229.py
  - task230.py
  - task231.py
  - task232.py
  - task233.py
  - task234.py
  - task235.py
  - task236.py
  - task237.py
  - task238.py
  - task239.py
  - task240.py
  - task241.py
  - task242.py
  - task243.py
  - task244.py
  - task245.py
  - task246.py
  - task247.py
  - task248.py
  - task249.py
  - task250.py
  - task251.py
  - task252.py
  - task253.py
  - task254.py
  - task255.py
  - task256.py
  - task257.py
  - task258.py
  - task259.py
  - task260.py
  - task261.py
  - task262.py
  - task263.py
  - task264.py
  - task265.py
  - task266.py
  - task267.py
  - task268.py
  - task269.py
  - task270.py
  - task271.py
  - task272.py
  - task273.py
  - task274.py
  - task275.py
  - task276.py
  - task277.py
  - task278.py
  - task279.py
  - task280.py
  - task281.py
  - task282.py
  - task283.py
  - task284.py
  - task285.py
  - task286.py
  - task287.py
  - task288.py
  - task289.py
  - task290.py
  - task291.py
  - task292.py
  - task293.py
  - task294.py
  - task295.py
  - task296.py
  - task297.py
  - task298.py
  - task299.py
  - task300.py
  - task301.py
  - task302.py
  - task303.py
  - task304.py
  - task305.py
  - task306.py
  - task307.py
  - task308.py
  - task309.py
  - task310.py
  - task311.py
  - task312.py
  - task313.py
  - task314.py
  - task315.py
  - task316.py
  - task317.py
  - task318.py
  - task319.py
  - task320.py
  - task321.py
  - task322.py
  - task323.py
  - task324.py
  - task325.py
  - task326.py
  - task327.py
  - task328.py
  - task329.py
  - task330.py
  - task331.py
  - task332.py
  - task333.py
  - task334.py
  - task335.py
  - task336.py
  - task337.py
  - task338.py
  - task339.py
  - task340.py
  - task341.py
  - task342.py
  - task343.py
  - task344.py
  - task345.py
  - task346.py
  - task347.py
  - task348.py
  - task349.py
  - task350.py
  - task351.py
  - task352.py
  - task353.py
  - task354.py
  - task355.py
  - task356.py
  - task357.py
  - task358.py
  - task359.py
  - task360.py
  - task361.py
  - task362.py
  - task363.py
  - task364.py
  - task365.py
  - task366.py
  - task367.py
  - task368.py
  - task369.py
  - task370.py
  - task371.py
  - task372.py
  - task373.py
  - task374.py
  - task375.py
  - task376.py
  - task377.py
  - task378.py
  - task379.py
  - task380.py
  - task381.py
  - task382.py
  - task383.py
  - task384.py
  - task385.py
  - task386.py
  - task387.py
  - task388.py
  - task389.py
  - task390.py
  - task391.py
  - task392.py
  - task393.py
  - task394.py
  - task395.py
  - task396.py
  - task397.py
  - task398.py
  - task399.py
  - task400.py
  - task401.py
  - task402.py
  - task403.py
  - task404.py
  - task405.py
  - task406.py
  - task407.py
  - task408.py
  - task409.py
  - task410.py
  - task411.py
  - task412.py
  - task413.py
  - task414.py
  - task415.py
  - task416.py
  - task417.py
  - task418.py
  - task419.py
  - task420.py
  - task421.py
  - task422.py
  - task423.py
  - task424.py
  - task425.py
  - task426.py
  - task427.py
  - task428.py
  - task429.py
  - task430.py
  - task431.py
  - task432.py
  - task433.py
  - task434.py
  - task435.py
  - task436.py
  - task437.py
  - task438.py
  - task439.py
  - task440.py
  - task441.py
  - task442.py
  - task443.py
  - task444.py
  - task445.py
  - task446.py
  - task447.py
  - task448.py
  - task449.py
  - task450.py
  - task451.py
  - task452.py
  - task453.py
  - task454.py
  - task455.py
  - task456.py
  - task457.py
  - task458.py
  - task459.py
  - task460.py
  - task461.py
  - task462.py
  - task463.py
  - task464.py
  - task465.py
  - task466.py
  - task467.py
  - task468.py
  - task469.py
  - task470.py
  - task471.py
  - task472.py
  - task473.py
  - task474.py
  - task475.py
  - task476.py
  - task477.py
  - task478.py
  - task479.py
  - task480.py
  - task481.py
  - task482.py
  - task483.py
  - task484.py
  - task485.py
  - task486.py
  - task487.py
  - task488.py
  - task489.py
  - task490.py
  - task491.py
  - task492.py
  - task493.py
  - task494.py
  - task495.py
  - task496.py
  - task497.py
  - task498.py
  - task499.py
  - task500.py

## Task Description #3: Conditional Statements (Age Classification)

## Scenario

You are building a basic classification system based on age.

## Task

Classify age into child, teenager, adult, or senior.

**Prompt Used:** Write Python code using if-elif-else to classify age groups.

### Sample Input: 20

## Sample Output: Adult

**Short Explanation:** The program checks age step by step using conditions and prints the correct group.

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows a tree view of files under "All CODING".
  - assignment 6.3
  - task1.py
  - task2.py
  - task3.py
  - task4.py
  - task5.py
  - assignment5.1.py
  - task1.py
  - task2.py
  - task3.py
  - task4.py
  - task5.py
  - assignment1
  - assignment8.1.py
  - task 2.py
  - task1.py
  - task3.py
  - task5.py
  - user\_data.txt
- PROBLEMS, OUTPUT, TERMINAL, PORTS:** Standard VS Code tabs.
- DEBUG CONSOLE:** Shows the command PS C:\Users\bhindu\OneDrive\Desktop\All CODING & C:/Users/bhindu/AppData/Local/Python/pythoncore-3.14-64/python.exe "C:/Users/bhindu/OneDrive/Desktop/All CODING\Assignment 6.3\task3.py". The output shows age input and classification results (Child, Teenager, Adult, Senior).
- OUTLINE, TIMELINE:** Standard VS Code navigation tabs.
- Bottom Status Bar:** Shows file paths (master: 01.11), status icons (0 0 0 0), and system information (In T, Col 8 (0 selected) Spaces 4 JTF-8 CRLF ( ) Python 314.2).

## 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.

### Task

Calculate the sum of first n natural numbers.

**Prompt Used:** Generate a Python function to calculate sum of first n numbers using for loop and while loop.

**Sample Input:** Sum of first 10 numbers using for loop: 55

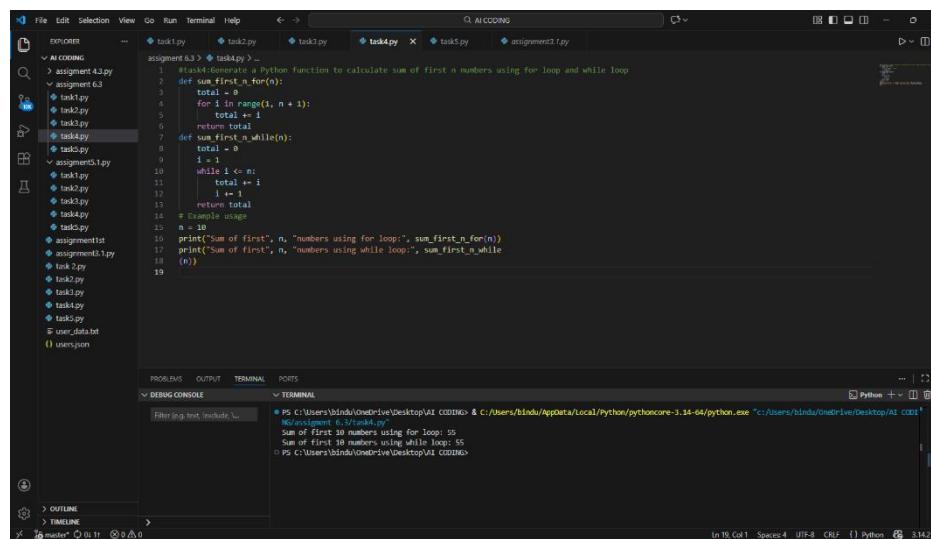
### Sample Output

Sum of first 10 numbers using while loop: 55

### Short Explanation

Both methods add numbers from 1 to n.

The logic is simple accumulation using loops.



A screenshot of a code editor (VS Code) showing two Python files: `task4.py` and `task4.py`. Both files contain the same code, which is a function to calculate the sum of the first  $n$  natural numbers using either a for loop or a while loop. The code is as follows:

```
#task4: Generate a Python function to calculate sum of first n numbers using for loop and while loop
assignment 6.3 > task4.py ...
1 #task4: Generate a Python function to calculate sum of first n numbers using for loop and while loop
2 def sum_first_n_for(n):
3     total = 0
4     for i in range(1, n + 1):
5         total += i
6     return total
7
8 def sum_first_n_while(n):
9     total = 0
10    i = 1
11    while i <= n:
12        total += i
13        i += 1
14    return total
15
16 # Example usage
17 n = 10
18 print("Sum of first", n, "numbers using for loop:", sum_first_n_for(n))
19 print("Sum of first", n, "numbers using while loop:", sum_first_n_while(n))
```

The terminal window shows the output of running the code in both ways, resulting in the sum of the first 10 natural numbers being 55.

## Task Description #5: Classes (Bank Account Class)

### Scenario

You are designing a basic banking application.

### Task

Create a Bank Account class with deposit, withdraw, and check balance.

### Prompt Used:

Generate a Python BankAccount class with deposit, withdraw, and balance checking methods.

### Sample Input

Initial Balance = 1000

Deposit = 500

Withdraw = 300

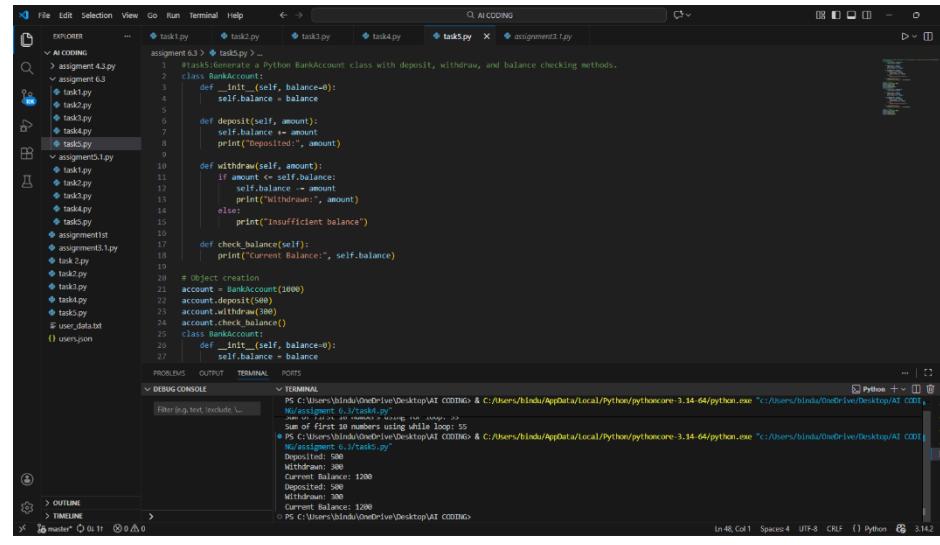
### Sample Output:

Deposited: 500  
Withdrawn: 300  
Current Balance: 1200

### Short Explanation

This class manages bank operations.

It updates balance safely and checks conditions before withdrawing money.



The screenshot shows the PyCharm IDE interface. The code editor displays a Python file named `task5.py` which contains a `BankAccount` class with methods for depositing and withdrawing funds while checking for insufficient balance. The terminal window shows the execution of the code and its output: deposited 500, withdrawn 300, and current balance 1200. The debug console shows the command to run the file and the resulting output.

```
#!/usr/bin/python
#task5: Generate a python BankAccount class with deposit, withdraw, and balance checking methods.
class BankAccount:
    def __init__(self, balance=0):
        self.balance = balance
    def deposit(self, amount):
        self.balance += amount
        print("Deposited:", amount)
    def withdraw(self, amount):
        if amount < self.balance:
            self.balance -= amount
            print("Withdrawn:", amount)
        else:
            print("Insufficient balance")
    def check_balance(self):
        print("Current Balance:", self.balance)
# Object creation
account = BankAccount(1000)
account.deposit(500)
account.withdraw(300)
account.check_balance()
class BankAccount:
    def __init__(self, balance=0):
        self.balance = balance
```

TERMINAL

```
PS C:\Users\Bindu\OneDrive\Desktop\AT\CODE & C:/Users/Bindu/AppData/Local/Python/PythonCore-3.14-64/python.exe "c:/Users/Bindu/OneDrive/Desktop/AT\CODE\Assignment_6\task5.py"
Sum of first 10 numbers using while loop: 55
PS C:\Users\Bindu\OneDrive\Desktop\AT\CODE & C:/Users/Bindu/AppData/Local/Python/PythonCore-3.14-64/python.exe "c:/Users/Bindu/OneDrive/Desktop/AT\CODE\Assignment_6\task5.py"
Deposited: 500
Withdrawn: 300
Current Balance: 1200
Deposited: 500
Withdrawn: 300
Current Balance: 1200
PS C:\Users\Bindu\OneDrive\Desktop\AT\CODE
```