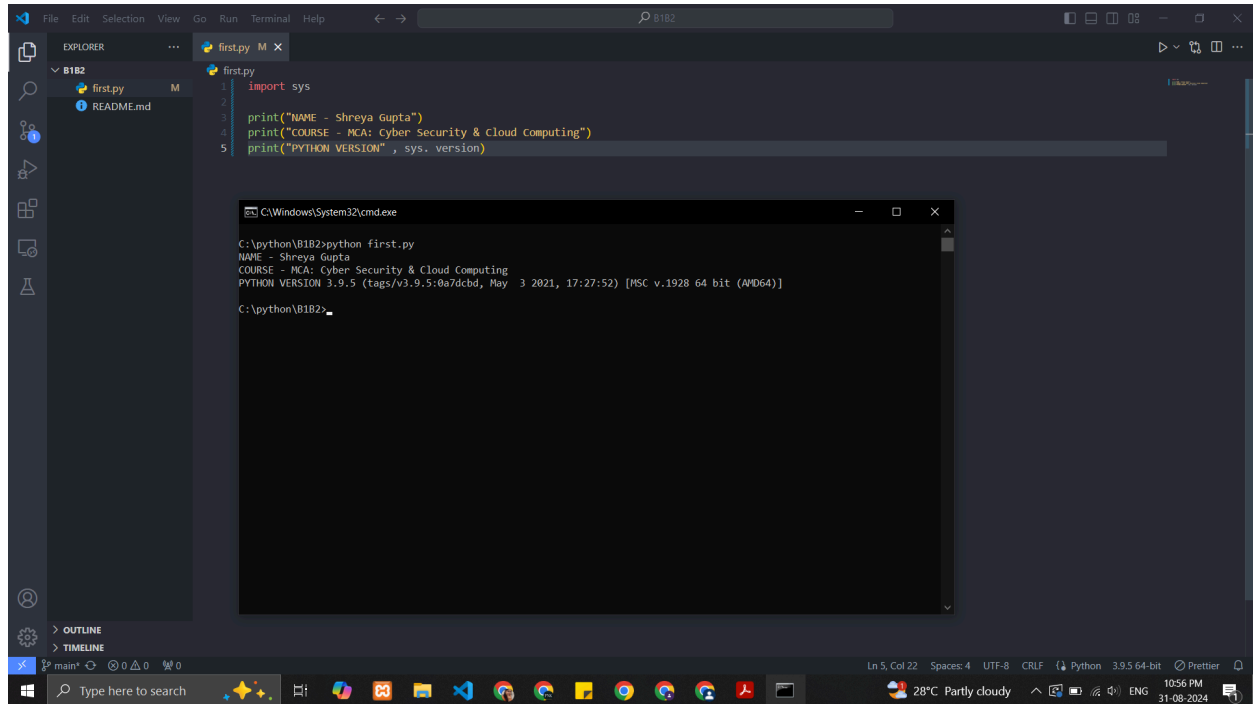


PYTHON ASSIGNMENT SCREENSHOT FILE

Name- Shreya Gupta

SAP ID- 590016431

1. Python Installation & Execution



The screenshot shows the Visual Studio Code interface with a file explorer on the left showing a project named 'B1B2' containing 'first.py' and 'README.md'. The main editor displays the code in 'first.py':

```
1 import sys
2
3 print("NAME - Shreya Gupta")
4 print("COURSE - MCA: Cyber Security & Cloud Computing")
5 print("PYTHON VERSION", sys.version)
```

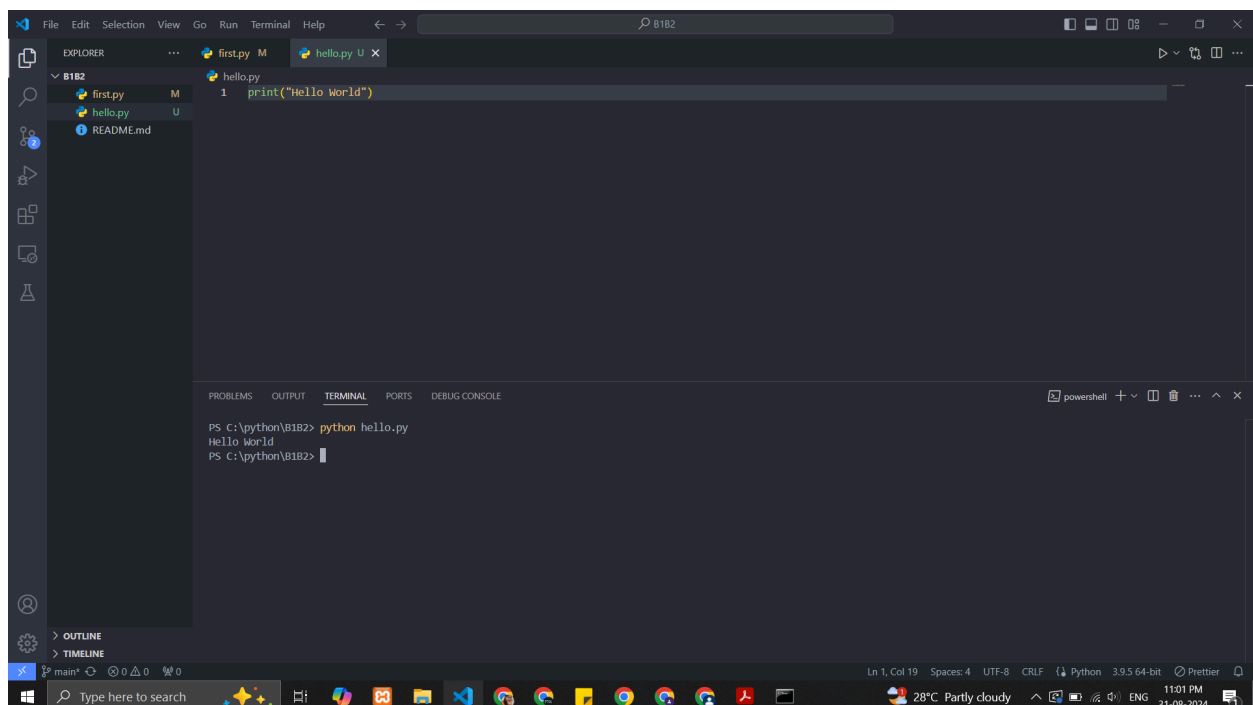
Below the code editor, a terminal window is open, showing the command prompt output of running 'python first.py' in the directory 'C:\python\B1B2':

```
C:\python\B1B2>python first.py
NAME - Shreya Gupta
COURSE - MCA: Cyber Security & Cloud Computing
PYTHON VERSION 3.9.5 (tags/v3.9.5:0a7dcdb, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)]

C:\python\B1B2>
```

The status bar at the bottom indicates the file is 'first.py' at line 5, column 22, using UTF-8 encoding, CRLF line endings, and the Python 3.9.5 64-bit interpreter.

2. VS Code Execution



The screenshot shows the Visual Studio Code interface with a file explorer on the left showing a project named 'B1B2' containing 'first.py', 'hello.py', and 'README.md'. The main editor displays the code in 'hello.py':

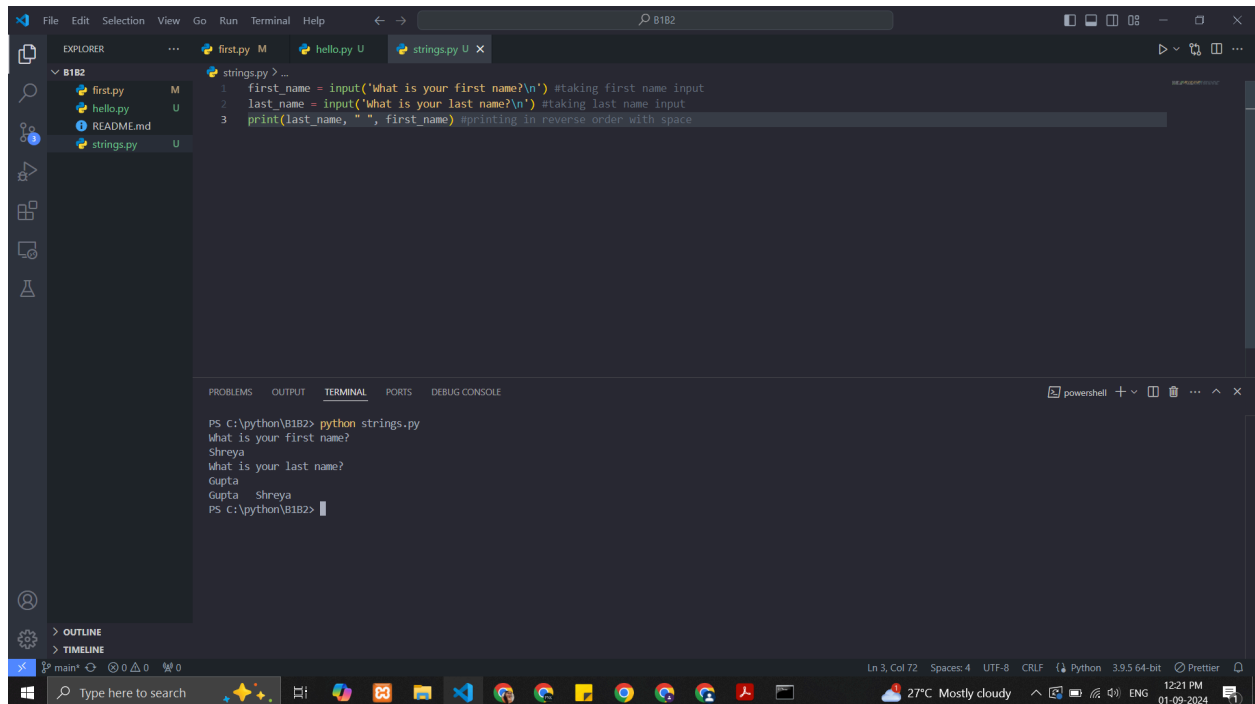
```
1 print("Hello World")
```

Below the code editor, a terminal window is open, showing the command prompt output of running 'python hello.py' in the directory 'C:\python\B1B2':

```
PS C:\python\B1B2> python hello.py
Hello World
PS C:\python\B1B2>
```

The status bar at the bottom indicates the file is 'hello.py' at line 1, column 19, using UTF-8 encoding, CRLF line endings, and the Python 3.9.5 64-bit interpreter.

3. String Operations



The screenshot shows a Visual Studio Code editor with a file explorer on the left containing 'first.py', 'hello.py', 'README.md', and 'strings.py'. The 'strings.py' file is open in the editor, containing the following Python code:

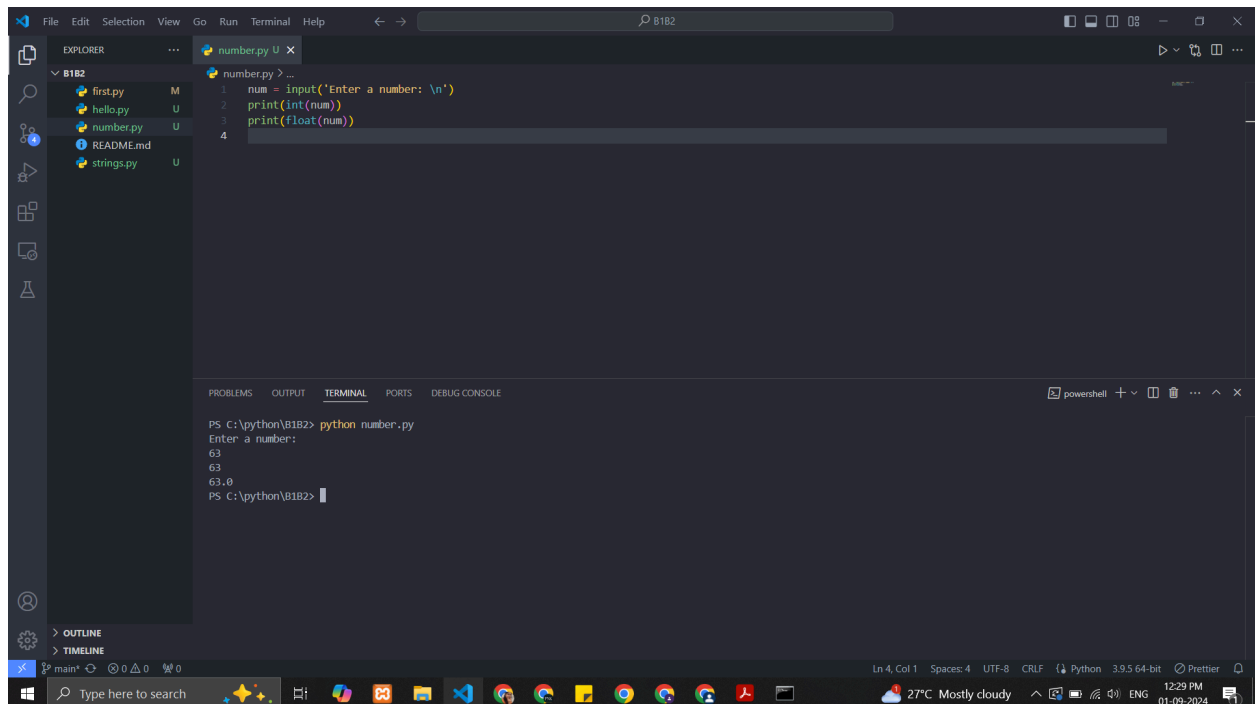
```
1 first_name = input('What is your first name?\n') #taking first name input
2 last_name = input('What is your last name?\n') #taking last name input
3 print(last_name, " ", first_name) #printing in reverse order with space
```

The terminal at the bottom shows the execution of the script:

```
PS C:\python\B1B2> python strings.py
What is your first name?
Shreya
What is your last name?
Gupta
Gupta Shreya
PS C:\python\B1B2>
```

The status bar at the bottom indicates the file is at line 3, column 72, with 4 spaces, UTF-8 encoding, CRLF line endings, and is using Python 3.9.5 64-bit with Prettier formatting.

4. Numeric Data Types and Conversion Functions



The screenshot shows a Visual Studio Code editor with a file explorer on the left containing 'first.py', 'hello.py', 'number.py', 'README.md', and 'strings.py'. The 'number.py' file is open in the editor, containing the following Python code:

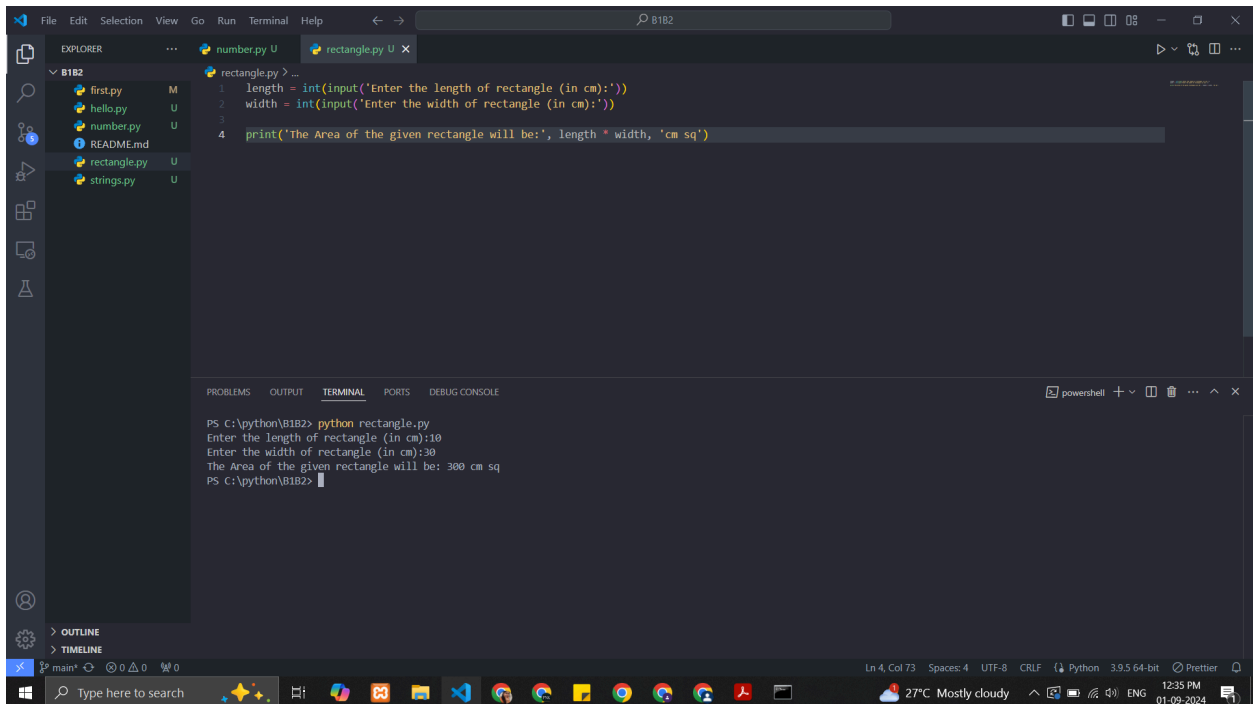
```
1 num = input('Enter a number: \n')
2 print(int(num))
3 print(float(num))
4
```

The terminal at the bottom shows the execution of the script:

```
PS C:\python\B1B2> python number.py
Enter a number:
63
63
63.0
PS C:\python\B1B2>
```

The status bar at the bottom indicates the file is at line 4, column 1, with 4 spaces, UTF-8 encoding, CRLF line endings, and is using Python 3.9.5 64-bit with Prettier formatting.

5. Basic input/output

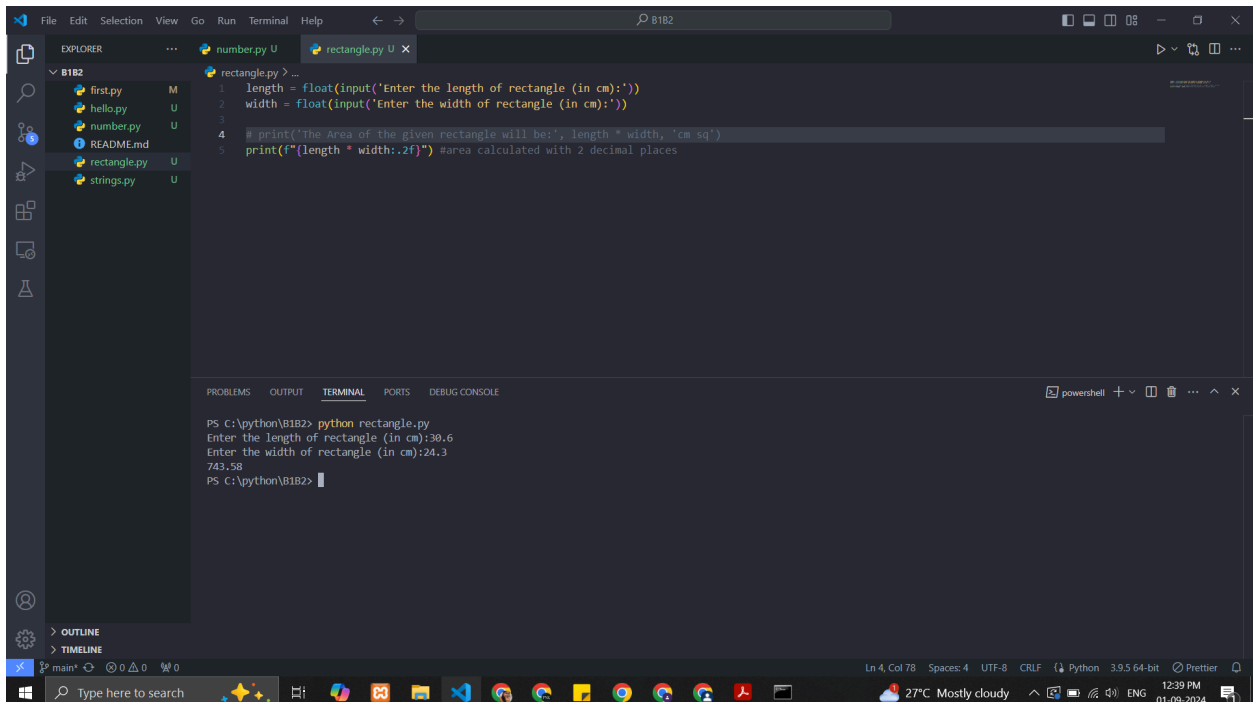


The screenshot shows the Visual Studio Code editor with a file explorer on the left containing files like first.py, hello.py, number.py, README.md, rectangle.py, and strings.py. The main editor window displays the code for rectangle.py, which uses the int() function for input and print() for output. The terminal at the bottom shows the execution of the script, where the user enters 10 for length and 30 for width, resulting in an output of 'The Area of the given rectangle will be: 300 cm sq'.

```
rectangle.py > --
1 length = int(input('Enter the length of rectangle (in cm):'))
2 width = int(input('Enter the width of rectangle (in cm):'))
3
4 print('The Area of the given rectangle will be:', length * width, 'cm sq')
```

```
PS C:\python\B1B2> python rectangle.py
Enter the length of rectangle (in cm):10
Enter the width of rectangle (in cm):30
The Area of the given rectangle will be: 300 cm sq
PS C:\python\B1B2>
```

6. Using format method

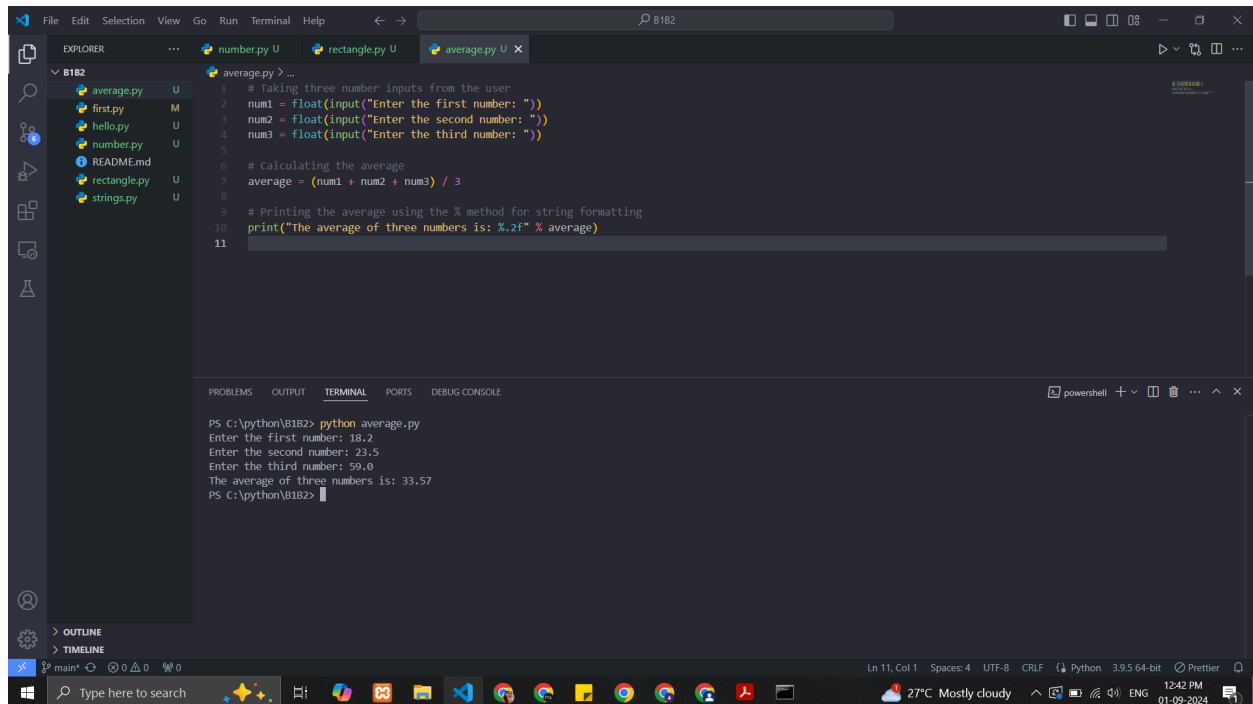


The screenshot shows the Visual Studio Code editor with the same file explorer as the previous image. The main editor window displays the code for rectangle.py, which uses the float() function for input and the format method for output. The terminal at the bottom shows the execution of the script, where the user enters 30.6 for length and 24.3 for width, resulting in an output of '743.58'.

```
rectangle.py > --
1 length = float(input('Enter the length of rectangle (in cm):'))
2 width = float(input('Enter the width of rectangle (in cm):'))
3
4 # print('The Area of the given rectangle will be:', length * width, 'cm sq')
5 print(f'{length * width:.2f}') #area calculated with 2 decimal places
```

```
PS C:\python\B1B2> python rectangle.py
Enter the length of rectangle (in cm):30.6
Enter the width of rectangle (in cm):24.3
743.58
PS C:\python\B1B2>
```

7. The % method & print method



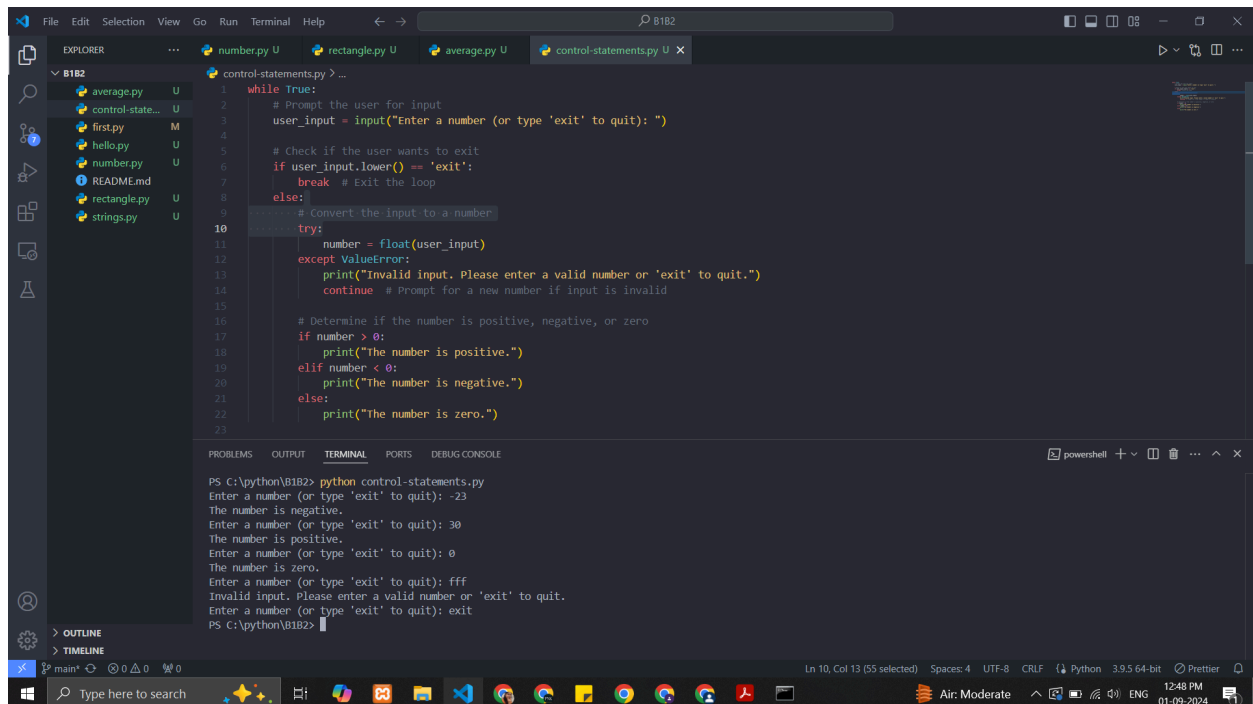
The screenshot shows a Visual Studio Code editor with a file explorer on the left containing files like `average.py`, `first.py`, `hello.py`, `number.py`, `rectangle.py`, and `strings.py`. The main editor displays the content of `average.py`:

```
1 # Taking three number inputs from the user
2 num1 = float(input("Enter the first number: "))
3 num2 = float(input("Enter the second number: "))
4 num3 = float(input("Enter the third number: "))
5
6 # Calculating the average
7 average = (num1 + num2 + num3) / 3
8
9 # Printing the average using the % method for string formatting
10 print("The average of three numbers is: %.2f" % average)
11
```

Below the editor, the terminal window shows the execution of the script:

```
PS C:\python\B1B2> python average.py
Enter the first number: 18.2
Enter the second number: 23.5
Enter the third number: 59.0
The average of three numbers is: 33.57
PS C:\python\B1B2>
```

8. Control Flow (if statements & loops)



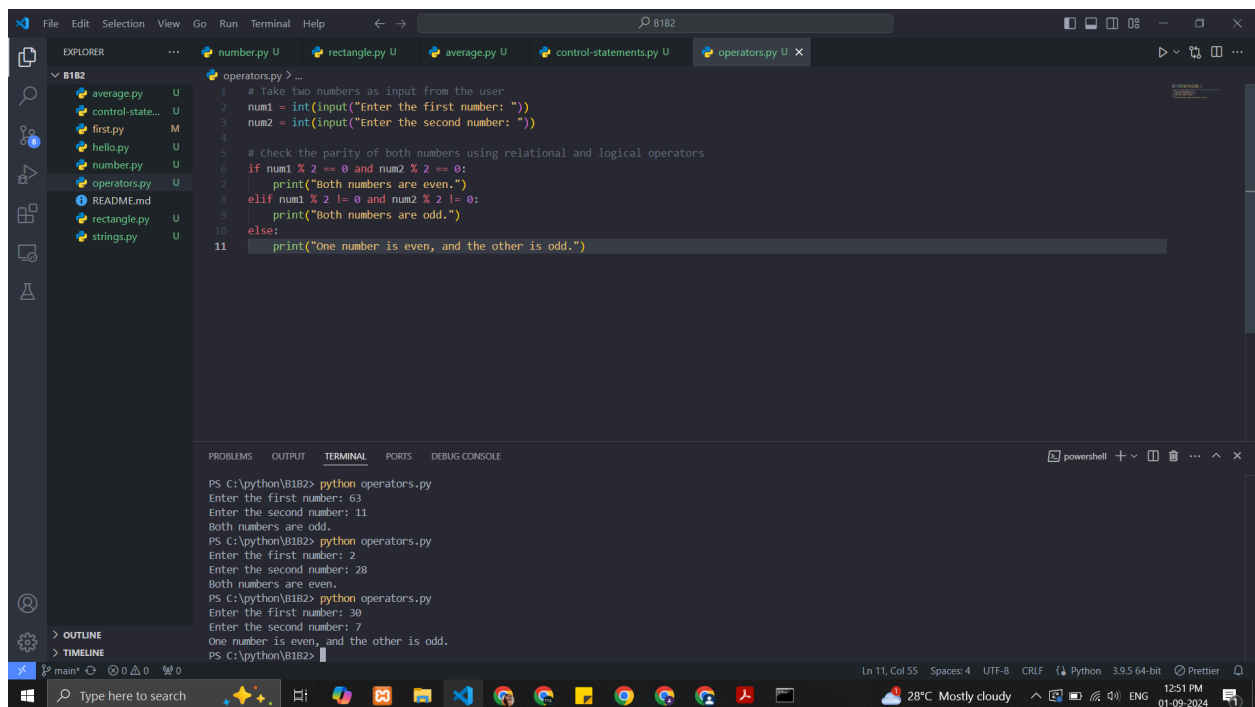
The screenshot shows a Visual Studio Code editor with a file explorer on the left containing files like `average.py`, `control-state...`, `first.py`, `hello.py`, `number.py`, `rectangle.py`, and `strings.py`. The main editor displays the content of `control-statements.py`:

```
1 while True:
2     # Prompt the user for input
3     user_input = input("Enter a number (or type 'exit' to quit): ")
4
5     # Check if the user wants to exit
6     if user_input.lower() == 'exit':
7         break # Exit the loop
8     else:
9         # Convert the input to a number
10        try:
11            number = float(user_input)
12        except ValueError:
13            print("Invalid input. Please enter a valid number or 'exit' to quit.")
14            continue # Prompt for a new number if input is invalid
15
16        # Determine if the number is positive, negative, or zero
17        if number > 0:
18            print("The number is positive.")
19        elif number < 0:
20            print("The number is negative.")
21        else:
22            print("The number is zero.")
23
```

Below the editor, the terminal window shows the execution of the script:

```
PS C:\python\B1B2> python control-statements.py
Enter a number (or type 'exit' to quit): -23
The number is negative.
Enter a number (or type 'exit' to quit): 30
The number is positive.
Enter a number (or type 'exit' to quit): 0
The number is zero.
Enter a number (or type 'exit' to quit): fff
Invalid input. Please enter a valid number or 'exit' to quit.
Enter a number (or type 'exit' to quit): exit
PS C:\python\B1B2>
```

9. Relational & Logical Operators



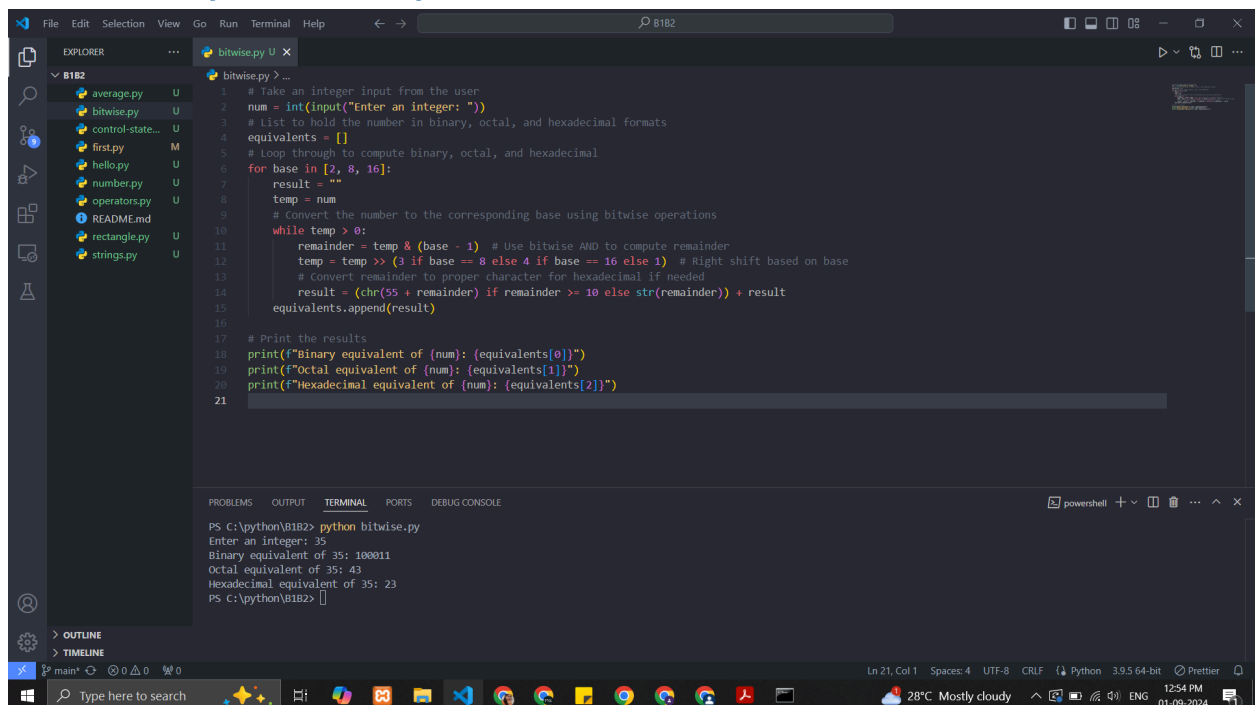
The screenshot shows a Visual Studio Code editor window with a Python file named `operators.py`. The script takes two numbers as input and checks their parity using relational and logical operators. The terminal shows the execution of the script with three test cases.

```
1 # Take two numbers as input from the user
2 num1 = int(input("Enter the first number: "))
3 num2 = int(input("Enter the second number: "))
4
5 # Check the parity of both numbers using relational and logical operators
6 if num1 % 2 == 0 and num2 % 2 == 0:
7     print("Both numbers are even.")
8 elif num1 % 2 != 0 and num2 % 2 != 0:
9     print("Both numbers are odd.")
10 else:
11     print("One number is even, and the other is odd.")
```

Terminal Output:

```
PS C:\python\B1B2> python operators.py
Enter the first number: 63
Enter the second number: 11
Both numbers are odd.
PS C:\python\B1B2> python operators.py
Enter the first number: 2
Enter the second number: 28
Both numbers are even.
PS C:\python\B1B2> python operators.py
Enter the first number: 30
Enter the second number: 7
One number is even, and the other is odd.
PS C:\python\B1B2>
```

10. For loop & Bitwise Operator



The screenshot shows a Visual Studio Code editor window with a Python file named `bitwise.py`. The script takes an integer input and computes its binary, octal, and hexadecimal equivalents using a for loop and bitwise operations. The terminal shows the execution of the script with a test case of 35.

```
1 # Take an integer input from the user
2 num = int(input("Enter an integer: "))
3 # List to hold the number in binary, octal, and hexadecimal formats
4 equivalents = []
5 # Loop through to compute binary, octal, and hexadecimal
6 for base in [2, 8, 16]:
7     result = ""
8     temp = num
9     # Convert the number to the corresponding base using bitwise operations
10    while temp > 0:
11        remainder = temp & (base - 1) # Use bitwise AND to compute remainder
12        temp = temp >> 1 # Right shift based on base
13        # Convert remainder to proper character for hexadecimal if needed
14        result = (chr(55 + remainder) if remainder >= 10 else str(remainder)) + result
15    equivalents.append(result)
16
17 # Print the results
18 print(f"Binary equivalent of {num}: {equivalents[0]}")
19 print(f"Octal equivalent of {num}: {equivalents[1]}")
20 print(f"Hexadecimal equivalent of {num}: {equivalents[2]}")
21
```

Terminal Output:

```
PS C:\python\B1B2> python bitwise.py
Enter an integer: 35
Binary equivalent of 35: 100011
Octal equivalent of 35: 43
Hexadecimal equivalent of 35: 23
PS C:\python\B1B2>
```