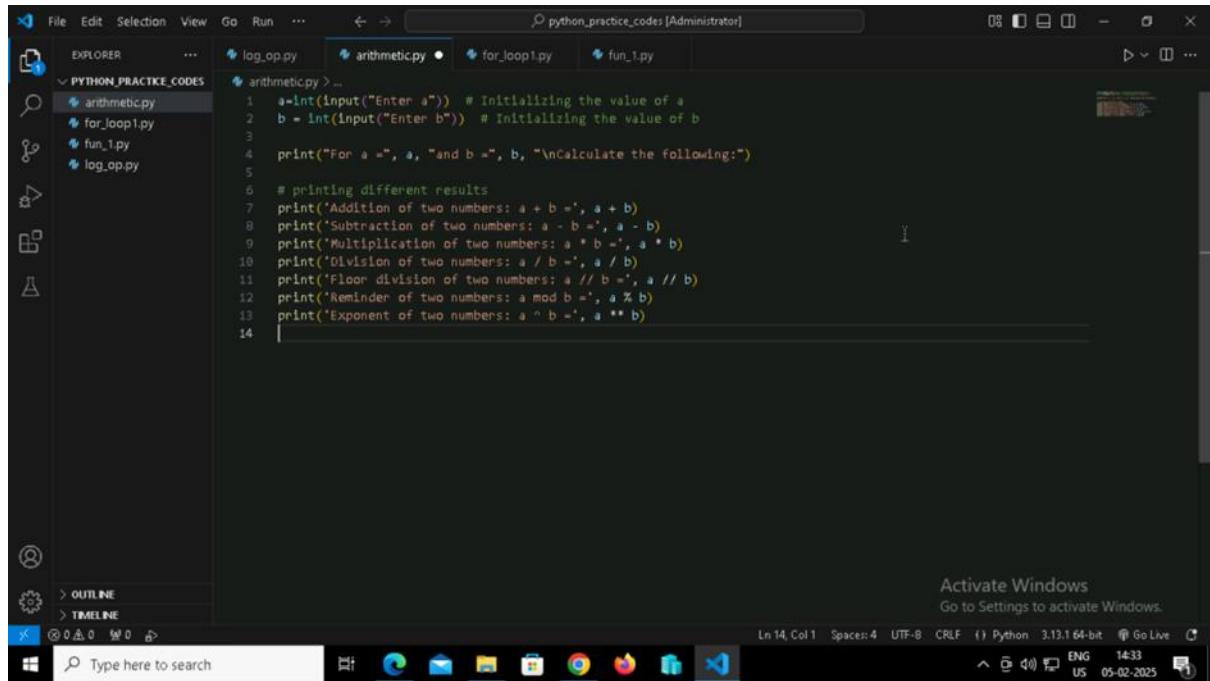


Operators in Python

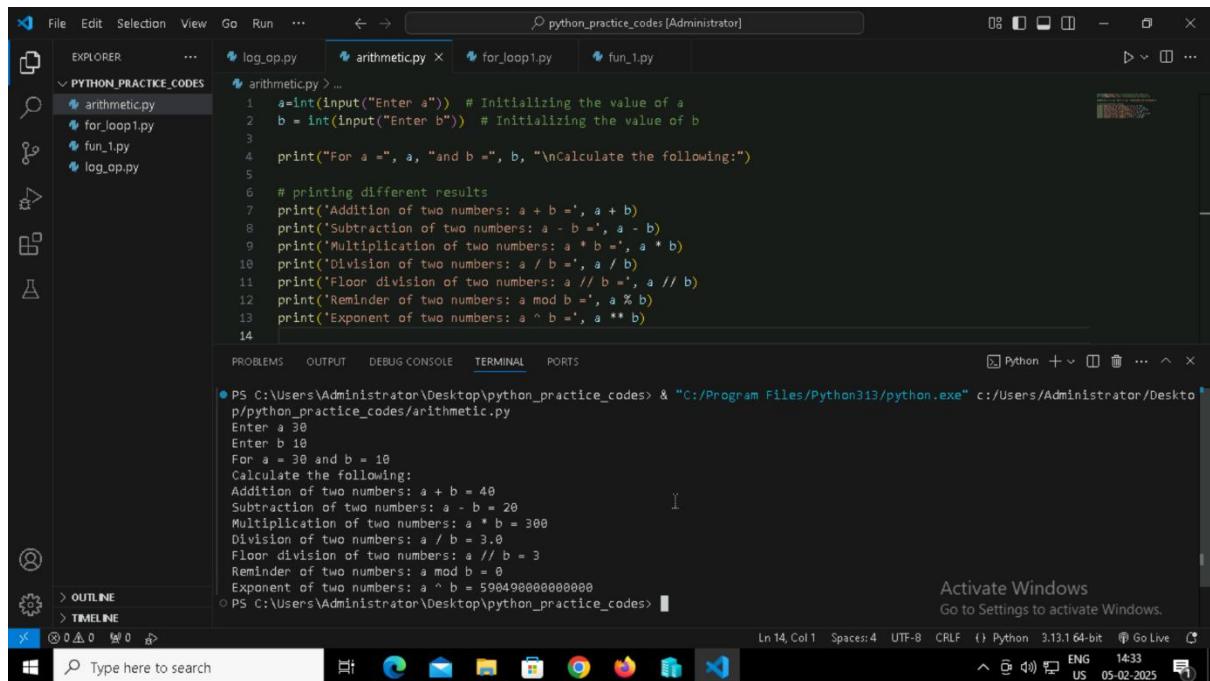
Arithmetic Operators

Code:



```
File Edit Selection View Go Run ... python_practice_codes [Administrator]
EXPLORER ...
PYTHON_PRACTICE_CODES
arithmetic.py
for_loop1.py
fun_1.py
log_op.py
arithmetic.py > ...
1 a=int(input("Enter a")) # Initializing the value of a
2 b = int(input("Enter b")) # Initializing the value of b
3
4 print("For a =", a, "and b =", b, "\nCalculate the following:")
5
6 # printing different results
7 print('Addition of two numbers: a + b =', a + b)
8 print('Subtraction of two numbers: a - b =', a - b)
9 print('Multiplication of two numbers: a * b =', a * b)
10 print('Division of two numbers: a / b =', a / b)
11 print('Floor division of two numbers: a // b =', a // b)
12 print('Reminder of two numbers: a mod b =', a % b)
13 print('Exponent of two numbers: a ^ b =', a ** b)
14 |
```

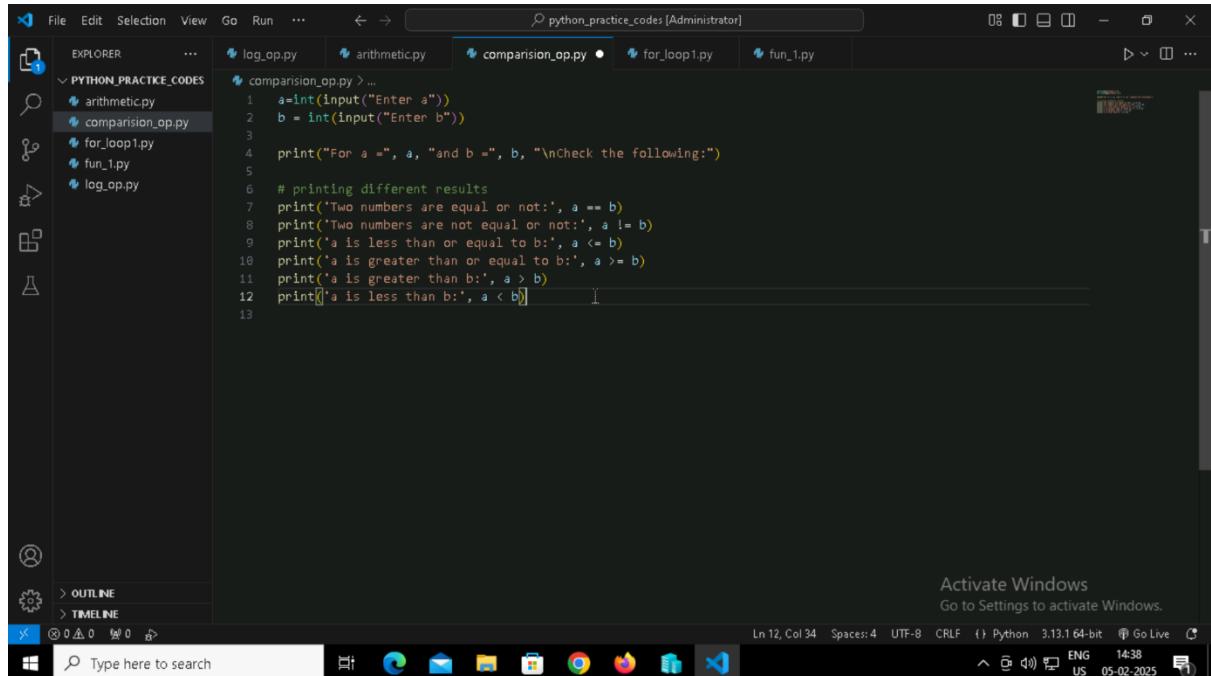
Output :



```
File Edit Selection View Go Run ... python_practice_codes [Administrator]
EXPLORER ...
PYTHON_PRACTICE_CODES
arithmetic.py
for_loop1.py
fun_1.py
log_op.py
arithmetic.py > ...
1 a=int(input("Enter a")) # Initializing the value of a
2 b = int(input("Enter b")) # Initializing the value of b
3
4 print("For a =", a, "and b =", b, "\nCalculate the following:")
5
6 # printing different results
7 print('Addition of two numbers: a + b =', a + b)
8 print('Subtraction of two numbers: a - b =', a - b)
9 print('Multiplication of two numbers: a * b =', a * b)
10 print('Division of two numbers: a / b =', a / b)
11 print('Floor division of two numbers: a // b =', a // b)
12 print('Reminder of two numbers: a mod b =', a % b)
13 print('Exponent of two numbers: a ^ b =', a ** b)
14 |
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/arithmetic.py
p/python_practice_codes/arithmetic.py
Enter a 30
Enter b 10
For a = 30 and b = 10
Calculate the following:
Addition of two numbers: a + b = 40
Subtraction of two numbers: a - b = 20
Multiplication of two numbers: a * b = 300
Division of two numbers: a / b = 3.0
Floor division of two numbers: a // b = 3
Reminder of two numbers: a mod b = 0
Exponent of two numbers: a ^ b = 5964900000000000
PS C:\Users\Administrator\Desktop\python_practice_codes>
```

Comparison Operators

Code:

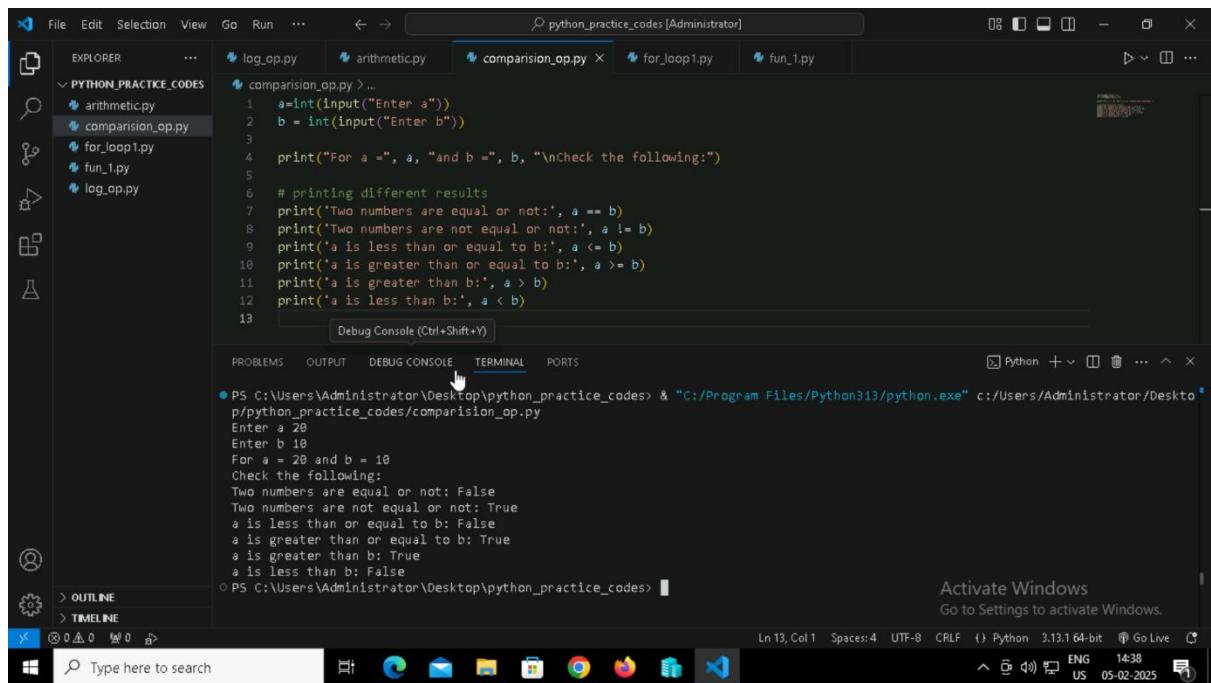


```
a=int(input("Enter a"))
b = int(input("Enter b"))

print("For a =", a, "and b =", b, "\nCheck the following:")

# printing different results
print('Two numbers are equal or not:', a == b)
print('Two numbers are not equal or not:', a != b)
print('a is less than or equal to b:', a <= b)
print('a is greater than or equal to b:', a >= b)
print('a is greater than b:', a > b)
print('a is less than b:', a < b)
```

Output:



```
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/comparision_op.py
Enter a 20
Enter b 10
For a = 20 and b = 10
Check the following:
Two numbers are equal or not: False
Two numbers are not equal or not: True
a is less than or equal to b: False
a is greater than or equal to b: True
a is greater than b: True
a is less than b: False
```

Assignment Operators

Code:

The screenshot shows the Visual Studio Code interface with the title bar "python_practice_codes [Administrator]". The Explorer sidebar on the left lists files in the "PYTHON_PRACTICE_CODES" folder: arithmetic.py, assign_op.py, comparison_op.py, for_loop1.py, fun_1.py, and log_op.py. The "assign_op.py" file is selected and open in the main editor area. The code prints various assignment operations (+=, -=, *=, /=, %=, **=, //=) with their results. The status bar at the bottom shows "Ln 2, Col 25" and "Python 3.13.1 64-bit".

```
a=int(input(" enter a"))
b=int(input(" enter b"))
print(' a += b:', a + b)
print(' a -= b:', a - b)
print(' a *= b:', a * b)
print(' a /= b:', a / b)
print(' a %= b:', a % b)
print(' a **= b:', a ** b)
print(' a //= b:', a // b)
```

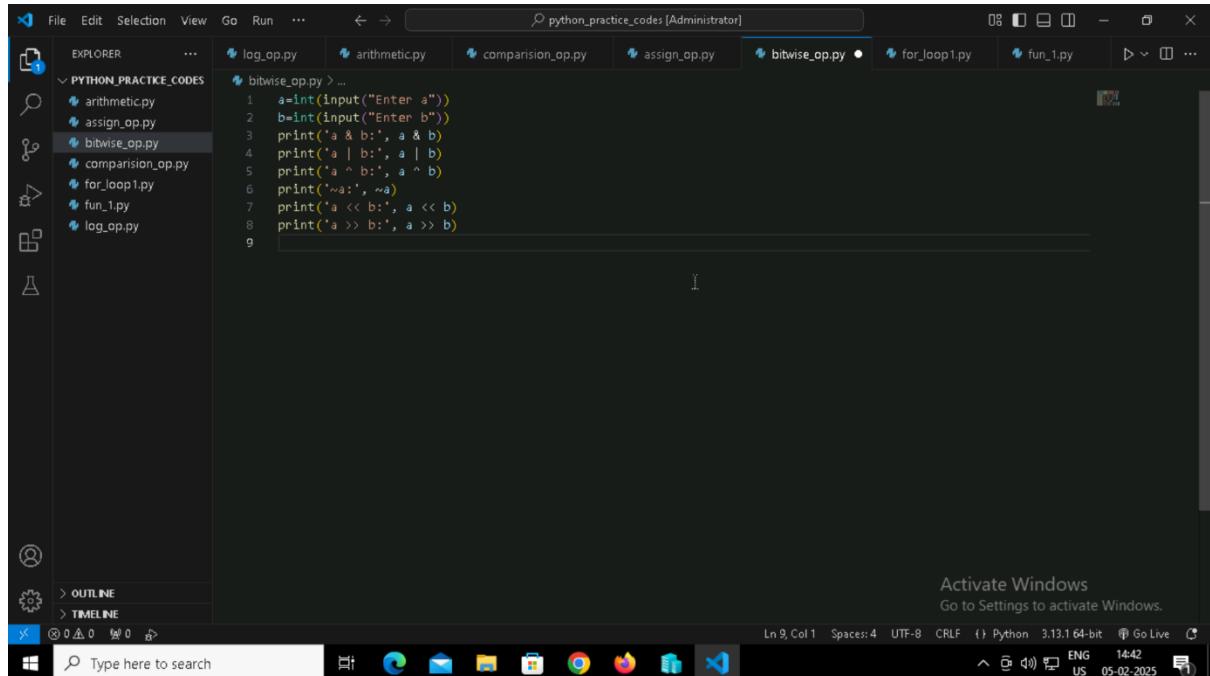
Output:

The screenshot shows the Visual Studio Code interface with the title bar "python_practice_codes [Administrator]". The Explorer sidebar on the left lists files in the "PYTHON_PRACTICE_CODES" folder: arithmetic.py, assign_op.py, comparison_op.py, for_loop1.py, fun_1.py, and log_op.py. The "assign_op.py" file is selected and open in the main editor area. Below the editor, the terminal window displays the execution of the script and its output. The status bar at the bottom shows "Ln 8, Col 26" and "Python 3.13.1 64-bit".

```
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/assign_op.py
enter a 12
enter b 14
a += b: 26
a -= b: -2
a *= b: 168
a /= b: 0.8571428571428571
a %= b: 12
a **= b: 1283918464548864
a //= b: 0
```

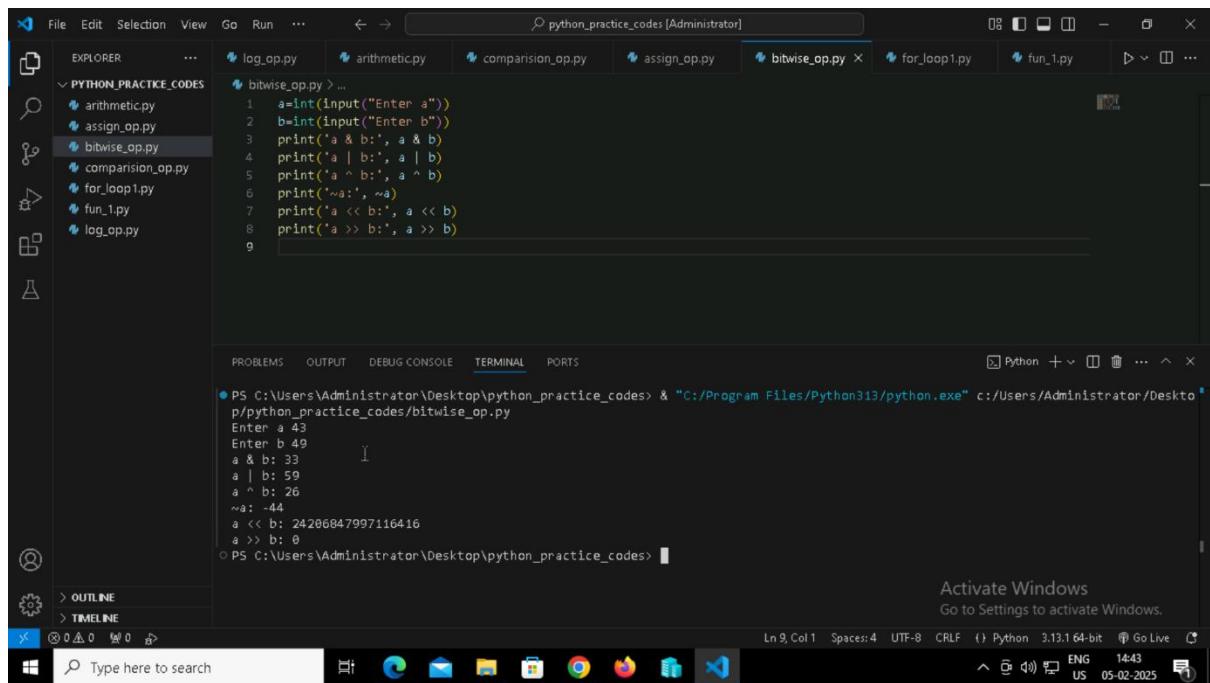
Bitwise Operators

Code:



```
a=int(input("Enter a"))
b=int(input("Enter b"))
print('a & b:', a & b)
print('a | b:', a | b)
print('a ^ b:', a ^ b)
print('~a:', ~a)
print('a << b:', a << b)
print('a >> b:', a >> b)
```

Output:



```
a=int(input("Enter a"))
b=int(input("Enter b"))
print('a & b:', a & b)
print('a | b:', a | b)
print('a ^ b:', a ^ b)
print('~a:', ~a)
print('a << b:', a << b)
print('a >> b:', a >> b)
```

PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/bitwise_op.py
Enter a 43
Enter b 49
a & b: 33
a | b: 59
a ^ b: 26
~a: -44
a << b: 24206847997116416
a >> b: 0

Logical Operators

Code:

The screenshot shows the Visual Studio Code interface with the title bar "python_practice_codes [Administrator]". The Explorer sidebar on the left lists files in the "PYTHON_PRACTICE_CODES" folder: arithmetic.py, assign_op.py, bitwise_op.py, comparision_op.py, for_loop1.py, fun_1.py, log_op.py, and logical_op.py. The logical_op.py file is open in the main editor area. The code prints conditions for a variable 'a' (7) and their results:

```
a = int(input("Enter a "))
print("For a = 7, checking whether the following conditions are True or False:")
print("a > 5 and a < 7 =>", a > 5 and a < 7)
print("a > 5 or a < 7 =>", a > 5 or a < 7)
print("not (a > 5 and a < 7)" =>, not(a > 5 and a < 7))
```

The status bar at the bottom shows "Ln 8, Col 1 Spaces:4 UTF-8 CRLF Python 3.13.1 64-bit ENG 14:44 US 05-02-2025".

Output:

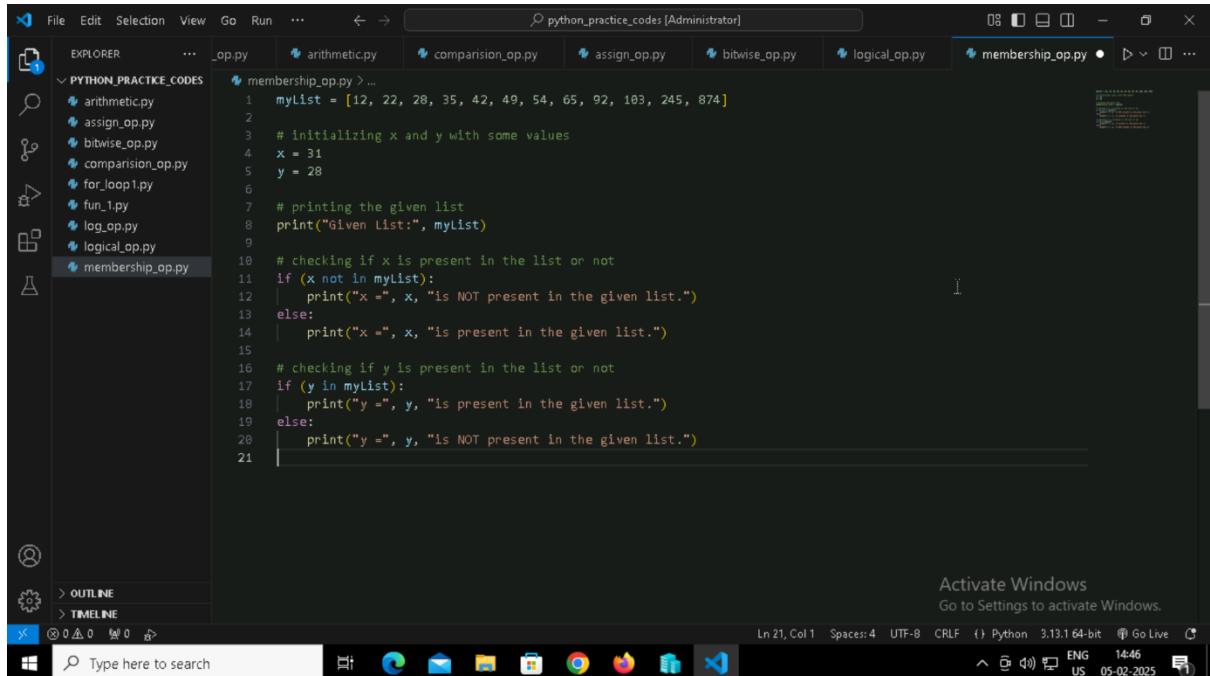
The screenshot shows the Visual Studio Code interface with the title bar "python_practice_codes [Administrator]". The Explorer sidebar on the left lists files in the "PYTHON_PRACTICE_CODES" folder: arithmetic.py, assign_op.py, bitwise_op.py, comparision_op.py, for_loop1.py, fun_1.py, log_op.py, and logical_op.py. The logical_op.py file is open in the main editor area. The terminal tab at the bottom shows the command line output of running the script:

```
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/logical_op.py
Enter a 14
For a = 7, checking whether the following conditions are True or False:
"a > 5 and a < 7" => False
"a > 5 or a < 7" => True
"not (a > 5 and a < 7)" => True
```

The status bar at the bottom shows "Ln 8, Col 1 Spaces:4 UTF-8 CRLF Python 3.13.1 64-bit ENG 14:44 US 05-02-2025".

Membership Operators

Code:



The screenshot shows the Microsoft Visual Studio Code interface. The left sidebar displays a tree view of files under 'PYTHON_PRACTICE_CODES' with 'membership_op.py' selected. The main code editor window contains the following Python script:

```
myList = [12, 22, 28, 35, 42, 49, 54, 65, 82, 103, 245, 874]
# initializing x and y with some values
x = 31
y = 28

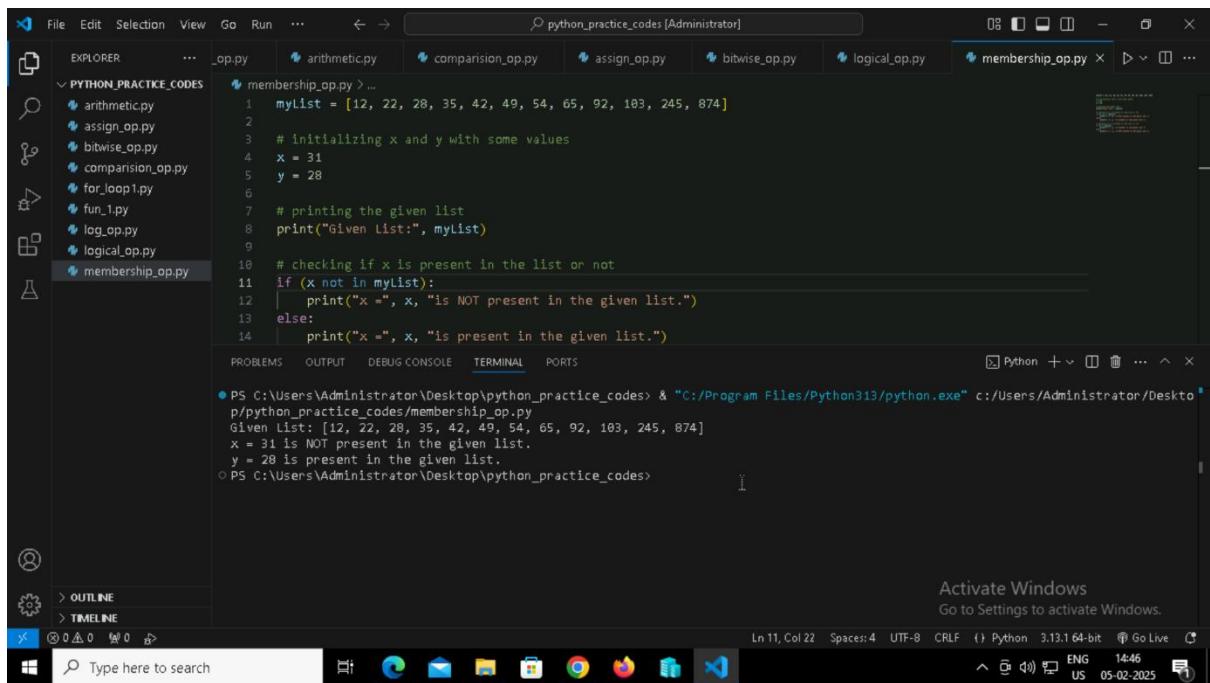
# printing the given list
print("Given List:", myList)

# checking if x is present in the list or not
if (x not in myList):
    print("x =", x, "is NOT present in the given list.")
else:
    print("x =", x, "is present in the given list.")

# checking if y is present in the list or not
if (y in myList):
    print("y =", y, "is present in the given list.")
else:
    print("y =", y, "is NOT present in the given list.")
```

The status bar at the bottom indicates 'Ln 21, Col 1' and 'Python 3.13.1 64-bit'. A message 'Activate Windows' with a link to settings is also visible.

Output:



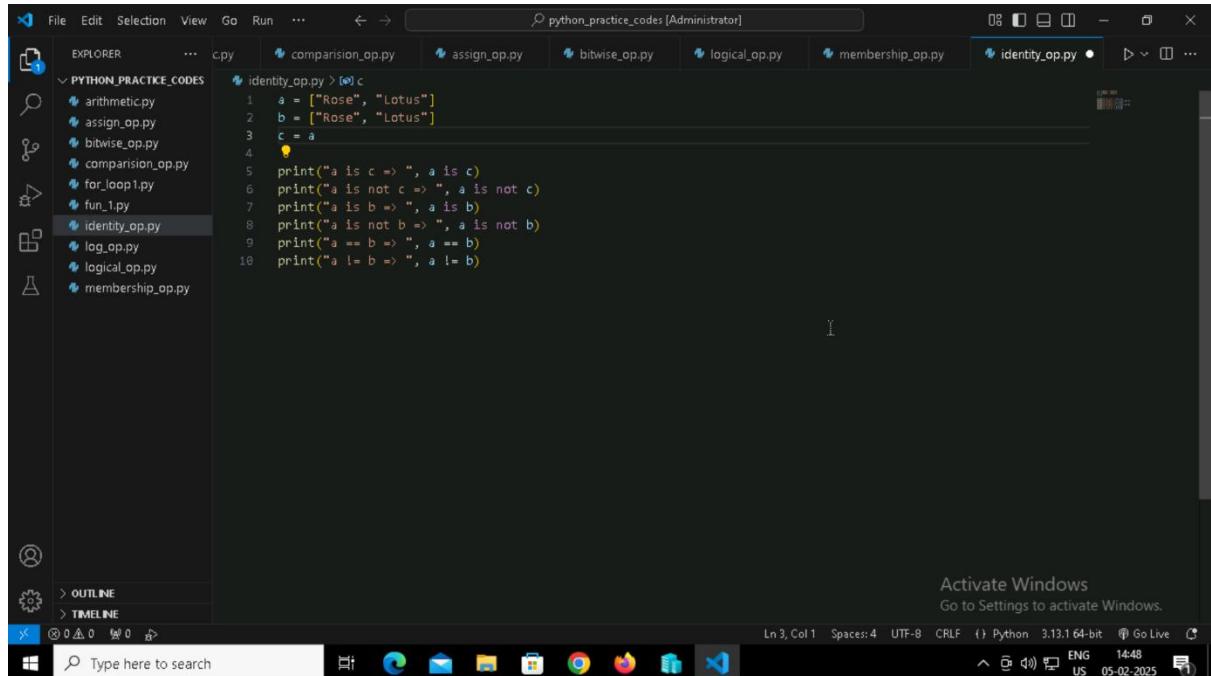
The screenshot shows the Microsoft Visual Studio Code interface with the 'TERMINAL' tab selected. The terminal window displays the command-line output of running the 'membership_op.py' script:

```
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/membership_op.py
Given List: [12, 22, 28, 35, 42, 49, 54, 65, 82, 103, 245, 874]
x = 31 is NOT present in the given list.
y = 28 is present in the given list.
```

The status bar at the bottom indicates 'Ln 11, Col 22' and 'Python 3.13.1 64-bit'. A message 'Activate Windows' with a link to settings is also visible.

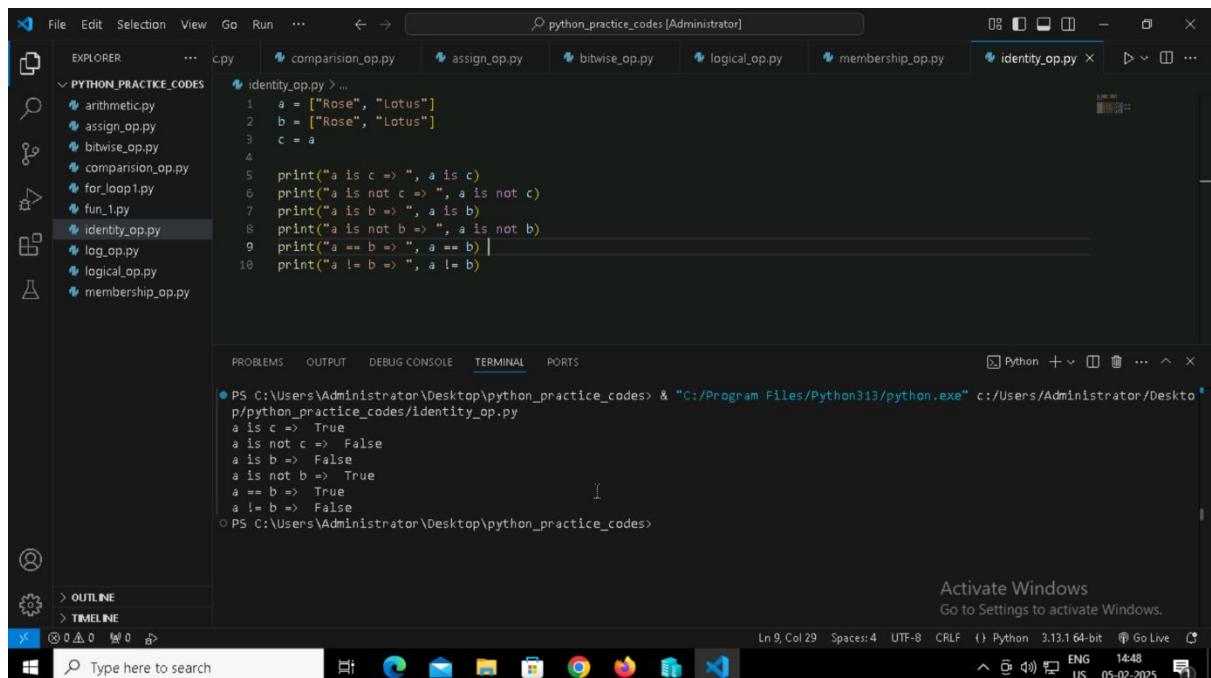
Identity Operators

Code:



```
a = ["Rose", "Lotus"]
b = ["Rose", "Lotus"]
c = a
print("a is c => ", a is c)
print("a is not c => ", a is not c)
print("a is b => ", a is b)
print("a is not b => ", a is not b)
print("a == b => ", a == b)
print("a != b => ", a != b)
```

Output:



```
a = ["Rose", "Lotus"]
b = ["Rose", "Lotus"]
c = a
print("a is c => ", a is c)
print("a is not c => ", a is not c)
print("a is b => ", a is b)
print("a is not b => ", a is not b)
print("a == b => ", a == b)
print("a != b => ", a != b)

PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/identity_op.py
a is c => True
a is not c => False
a is b => False
a is not b => True
a == b => True
a != b => False
```

Read a csv file :

Code :

The screenshot shows the Visual Studio Code interface. The left sidebar displays a folder named 'PYTHON_PRACTICE_CODES' containing several Python files: arithmetic.py, assign_op.py, bitwise_op.py, comparison_op.py, for_loop1.py, fun_1.py, identity_op.py, log_op.py, logical_op.py, membership_op.py, and readcsv.py. The 'readcsv.py' file is currently open in the main editor area. The code reads a CSV file named 'sample.csv' located at 'C:\Users\Administrator\Desktop\demo'. It uses the 'csv' module to read the file and prints the column names and each row's data. The status bar at the bottom indicates the file is saved in Python 3.13.1 64-bit.

```
# Importing the csv module
import csv

# Open the CSV file by passing the file path
with open('C:/Users/Administrator/Desktop/demo/sample.csv') as csv_file:
    csv_read = csv.reader(csv_file, delimiter=',') # Delimiter is comma

    count_line = 0

    # Iterate over each row in the file
    for row in csv_read:
        if count_line == 0:
            print(f'Column names are {" ".join(row)}')
        else:
            print(f'\t{row[0]} roll number is: {row[1]} and department is: {row[2]}')
        count_line += 1

print(f'Processed {count_line} lines.') # This line will print the number of lines in the file
```

Output:

The screenshot shows the Visual Studio Code interface with the terminal tab active. The terminal window displays the execution of the 'readcsv.py' script. It shows the column names 'Name', 'Age', and 'City', followed by five rows of data: John (roll number 28, department New York), Alice (roll number 24, department Los Angeles), Bob (roll number 35, department Chicago), Eve (roll number 40, department Boston), and Charlie (roll number 22, department San Francisco). The script concludes with a message indicating it processed 6 lines. The status bar at the bottom indicates the file is saved in Python 3.13.1 64-bit.

```
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/readcsv.py
Column names are Name, Age, City
    John roll number is: 28 and department is: New York.
    Alice roll number is: 24 and department is: Los Angeles.
    Bob roll number is: 35 and department is: Chicago.
    Eve roll number is: 40 and department is: Boston.
    Charlie roll number is: 22 and department is: San Francisco.
Processed 6 lines.
```

reverse the given string using for loop:

code:

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder named "PYTHON_PRACTICE_CODES" containing files: arithmetic.py, assign_op.py, bitwise_op.py, comparision_op.py, for_loop1.py, fun_1.py, identity_op.py, log_op.py, logical_op.py, membership_op.py, readcsv.py, and rstring1.py.
- Code Editor:** The active file is "rstring1.py". The code defines a function `reverse_string` that takes a string `str` and reverses it character by character using a for loop. It then prints the original and reversed strings.
- Status Bar:** Shows the current file is "rstring1.py", the line and column are "Ln 10, Col 1", and the editor is set to Python 3.13.1 64-bit.
- Taskbar:** Shows various pinned application icons.

Output:

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder named "PYTHON_PRACTICE_CODES" containing files: arithmetic.py, assign_op.py, bitwise_op.py, comparision_op.py, for_loop1.py, fun_1.py, identity_op.py, log_op.py, logical_op.py, membership_op.py, readcsv.py, and rstring1.py.
- Terminal:** The terminal window shows the command `python rstring1.py` being run, followed by the output: "The original string is: JavaTpoint" and "The reverse string is: dnoJavTaJ".
- Status Bar:** Shows the current file is "rstring1.py", the line and column are "Ln 10, Col 1", and the editor is set to Python 3.13.1 64-bit.
- Taskbar:** Shows various pinned application icons.

reverse a string using a while loop:

code:

The screenshot shows the Visual Studio Code interface. The left sidebar displays a file tree under 'EXPLORER' with several Python files listed. The main editor tab is titled 'rstring2.py' and contains the following code:

```
rsting2.py > ...
1 str = "JavaPoint"
2 print("The original string is : ", str)
3 reverse_String = ""
4 count = len(str)
5
6 while count > 0:
7     reverse_String += str[count - 1]
8     count = count - 1
9
10 print("The reversed string using a while loop is : ", reverse_String)
11
```

The status bar at the bottom indicates the code is in Python 3.13.1 64-bit mode, with the date and time as 05-02-2025.

Output:

The screenshot shows the Visual Studio Code interface with the terminal tab active. The terminal window displays the output of the 'rstring2.py' script:

```
rsting2.py > ...
1 str = "JavaPoint"
2 print("The original string is : ", str)
3 reverse_String = ""
4 count = len(str)
5
6 while count > 0:
7     reverse_String += str[count - 1]
8     count = count - 1
9
10 print("The reversed string using a while loop is : ", reverse_String)
11

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/rstring2.py
The original string is : JavaPoint
The reversed string using a while loop is : tniopTavaJ
PS C:\Users\Administrator\Desktop\python_practice_codes>
```

The status bar at the bottom indicates the code is in Python 3.13.1 64-bit mode, with the date and time as 05-02-2025.

reverse the given string using the extended slice operator:

code:

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder named "PYTHON_PRACTICE_CODES" containing several Python files: arithmetic.py, assign_op.py, bitwise_op.py, comparison_op.py, for_loop1.py, fun_1.py, identity_op.py, log_op.py, logical_op.py, membership_op.py, readcsv.py, rstring1.py, rstring2.py, and rstring3.py.
- Code Editor:** The file "rstring3.py" is open, displaying the following Python code:

```
def reverse(str):
    str = str[::-1]
    return str

s = "JavaPoint"
print("The original string is : ", s)
print("The reversed string using extended slice operator is : ", reverse(s))
```
- Status Bar:** Shows "Ln 8, Col 1" and "Python 3.13.1 64-bit".
- Taskbar:** Shows icons for various applications including File Explorer, Task Manager, and a search bar.

Output:

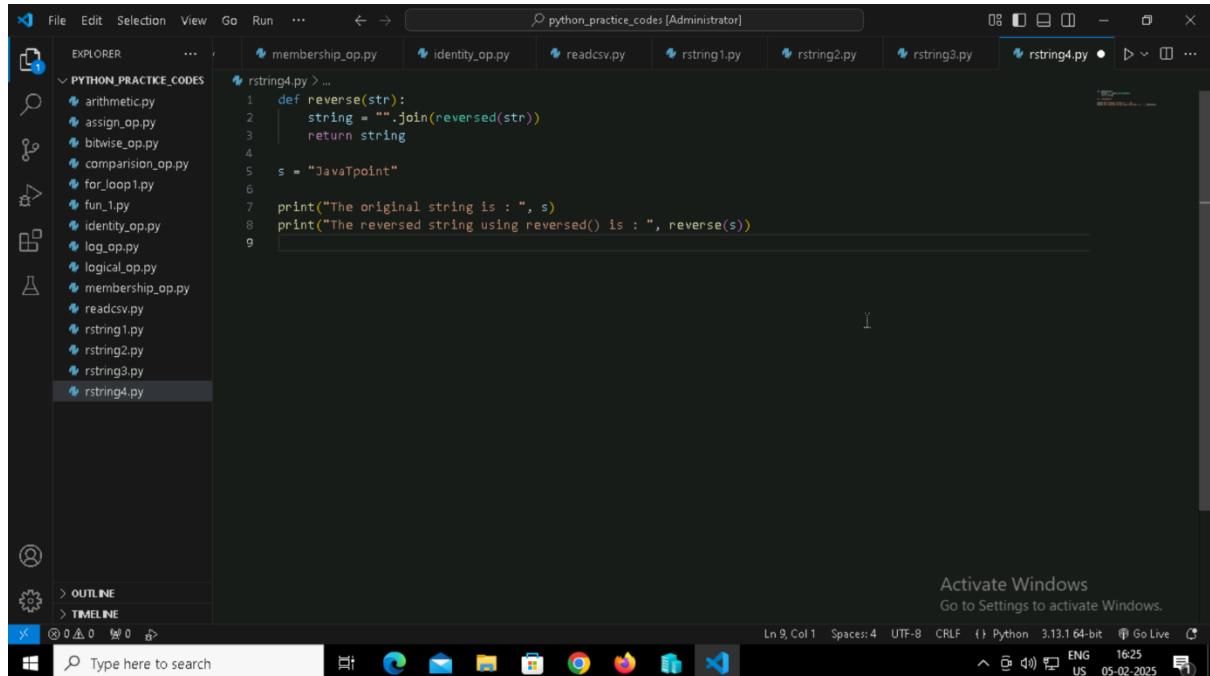
The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder named "PYTHON_PRACTICE_CODES" containing several Python files: arithmetic.py, assign_op.py, bitwise_op.py, comparison_op.py, for_loop1.py, fun_1.py, identity_op.py, log_op.py, logical_op.py, membership_op.py, readcsv.py, rstring1.py, rstring2.py, and rstring3.py.
- Terminal:** The terminal window shows the output of running the "rstring3.py" script:

```
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/rstring3.py
The original string is : JavaPoint
The reversed string using extended slice operator is : tniopTavaJ
PS C:\Users\Administrator\Desktop\python_practice_codes>
```
- Status Bar:** Shows "Ln 8, Col 1" and "Python 3.13.1 64-bit".
- Taskbar:** Shows icons for various applications including File Explorer, Task Manager, and a search bar.

Using reverse function with join :

Code:

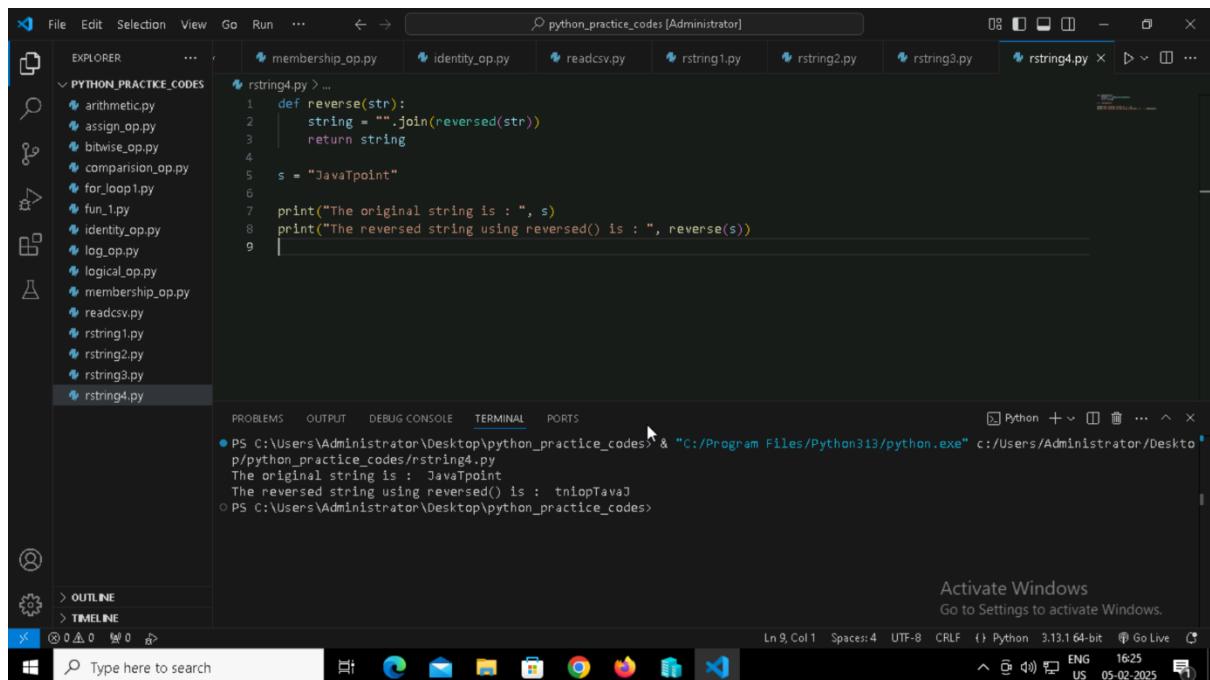


```
File Edit Selection View Go Run ... ← → python_practice_codes [Administrator]
EXPLORER PYTHON_PRACTICE_CODES
rstring4.py > ...
1 def reverse(str):
2     string = "".join(reversed(str))
3     return string
4
5 s = "JavaPoint"
6
7 print("The original string is : ", s)
8 print("The reversed string using reversed() is : ", reverse(s))
9
```

Activate Windows
Go to Settings to activate Windows.

Ln 9, Col 1 Spaces:4 UTF-8 CRLF Python 3.13.1 64-bit 16:25 ENG US 05-02-2025

Output:



```
File Edit Selection View Go Run ... ← → python_practice_codes [Administrator]
EXPLORER PYTHON_PRACTICE_CODES
rstring4.py > ...
1 def reverse(str):
2     string = "".join(reversed(str))
3     return string
4
5 s = "JavaPoint"
6
7 print("The original string is : ", s)
8 print("The reversed string using reversed() is : ", reverse(s))
9
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/rstring4.py
The original string is : JavaPoint
The reversed string using reversed() is : tniopTavaJ
PS C:\Users\Administrator\Desktop\python_practice_codes>
```

Activate Windows
Go to Settings to activate Windows.

Ln 9, Col 1 Spaces:4 UTF-8 CRLF Python 3.13.1 64-bit 16:25 ENG US 05-02-2025

Using recursion():

Code:

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder named "PYTHON_PRACTICE_CODES" containing various Python files like arithmetic.py, assign_op.py, etc., and "rstring5.py".
- Code Editor:** Displays the content of "rstring5.py". The code defines a recursive function "reverse" that takes a string "str" and returns its reverse. It handles the base case where the length of the string is 0, returning the string as is. For other cases, it returns the reverse of the substring from index 1 followed by the character at index 0.
- Terminal:** Shows the command "PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/rstring5.py". The output shows the original string "Devansh Sharma" and the reversed string "amrahs hsnaveD".
- Status Bar:** Shows the file path "python_practice_codes [Administrator]", line 10, column 1, spaces: 4, encoding: UTF-8, Python 3.13.1 64-bit, and the date/time "05-02-2025".

Output:

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder named "PYTHON_PRACTICE_CODES" containing various Python files like arithmetic.py, assign_op.py, etc., and "rstring5.py".
- Code Editor:** Displays the content of "rstring5.py". The code defines a recursive function "reverse" that takes a string "str" and returns its reverse. It handles the base case where the length of the string is 0, returning the string as is. For other cases, it returns the reverse of the substring from index 1 followed by the character at index 0.
- Terminal:** Shows the command "PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/rstring5.py". The output shows the original string "Devansh Sharma" and the reversed string "amrahs hsnaveD".
- Status Bar:** Shows the file path "python_practice_codes [Administrator]", line 10, column 1, spaces: 4, encoding: UTF-8, Python 3.13.1 64-bit, and the date/time "05-02-2025".

If condition:

Simple Python program to understand the if statement

Code:

The screenshot shows the Visual Studio Code interface. The left sidebar displays a tree view of files under 'PYTHON_PRACTICE_CODES'. The current file, 'if1.py', is selected and shown in the main editor area. The code contains three lines of Python: 'n=int(input("Enter a number"))', 'if n%2==0:', and 'print("the number is even")'. The status bar at the bottom right indicates the file is saved and shows the date and time as 06-02-2025.

```
n=int(input("Enter a number"))
if n%2==0:
    print("the number is even")
```

Output:

The screenshot shows the Visual Studio Code interface with the 'OUTPUT' tab selected in the bottom navigation bar. The terminal window displays the command 'python if1.py' being run and its output. The user enters '8' when prompted, and the program prints 'the number is even'. The status bar at the bottom right shows the file is saved and the date and time as 06-02-2025.

```
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/if1.py
Enter a number 8
the number is even
```

Program to print the largest of the three numbers.

Code:

The screenshot shows a code editor window titled "python_practice_codes [Administrator]". The file "if2.py" is open, containing the following Python code:

```
1 # Simple Python Program to print the largest of the three numbers.
2 a=int(input("Enter a"))
3 b=int(input("Enter b"))
4 c=int(input("Enter c"))
5 if (a>b and a>c):
6     print("a is the largest number")
7 if (b>a and b>c):
8     print("b is the largest number")
9 if (c>a and c>b):
10    print("c is the largest number")
```

The code uses three nested if statements to compare three integers (a, b, and c) and print the largest one. The code editor interface includes a sidebar with files like "arithmetic.py", "assign_op.py", etc., and a bottom status bar showing "Ln 10, Col 36" and "Python 3.13.1 64-bit".

Output:

The screenshot shows the same code editor window after running the "if2.py" script. The terminal tab at the bottom displays the following output:

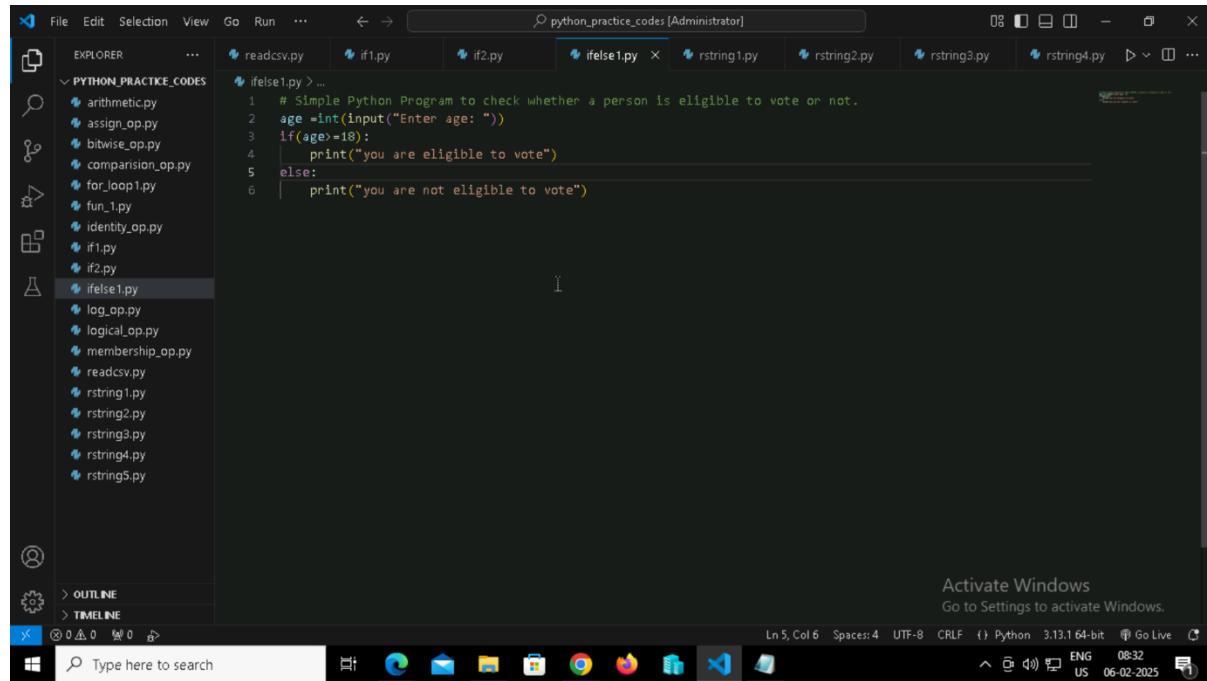
```
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/if2.py
Enter a 10
Enter b 12
Enter c 14
c is the largest number
```

The terminal also shows the command to run the script and the user inputs for a, b, and c. The status bar at the bottom indicates "08:29" and "06-02-2025".

If Else:

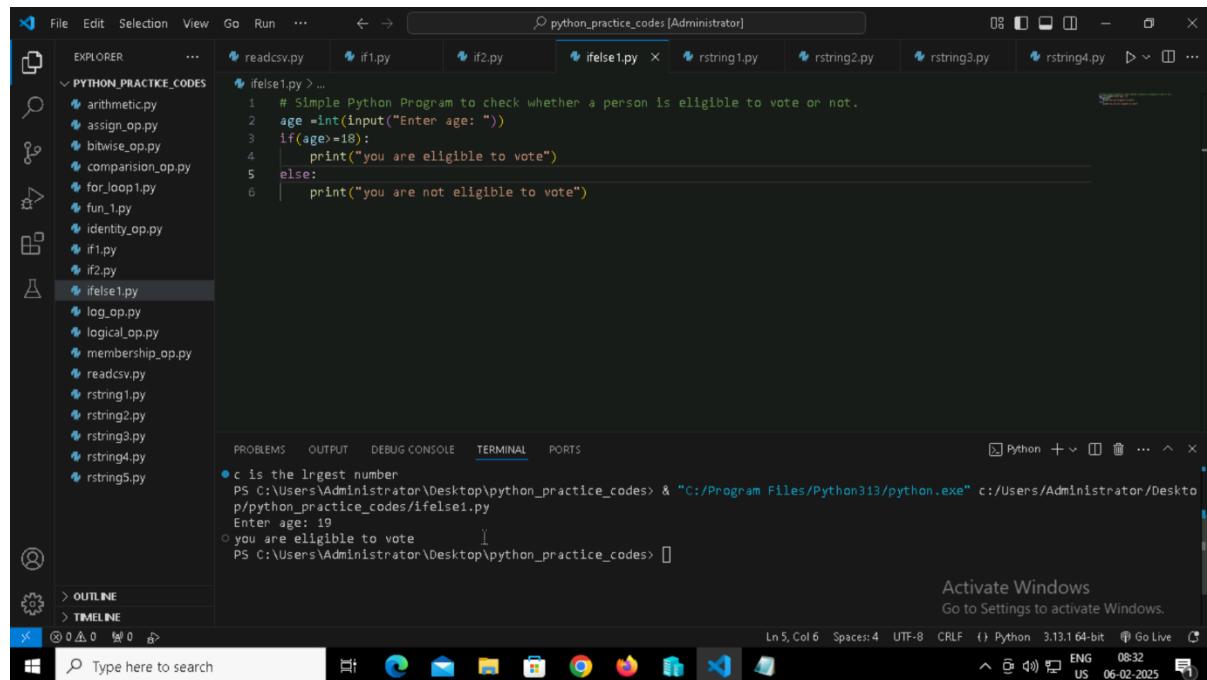
Program to check whether a person is eligible to vote or not.

Code:



```
# Simple Python Program to check whether a person is eligible to vote or not.
age = int(input("Enter age: "))
if(age>=18):
    print("you are eligible to vote")
else:
    print("you are not eligible to vote")
```

Output:



```
c is the largest number
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/ifelse1.py
Enter age: 19
you are eligible to vote
PS C:\Users\Administrator\Desktop\python_practice_codes>
```

Program to check whether a number is even or not.

Code:

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder named "PYTHON_PRACTICE_CODES" containing various Python files like arithmetic.py, assign_op.py, etc., and two specific files highlighted: "ifelse1.py" and "ifelse2.py".
- Code Editor:** Displays the content of "ifelse2.py":

```
1 # Simple Python Program to check whether a number is even or not.
2 num=int(input("Enter a number"))
3 if(num%2==0):
4     print("Number is even")
5 else:
6     print("Number is not even")
```
- Status Bar:** Shows "Ln 6, Col 32" and "Python 3.13.1 64-bit".
- Taskbar:** Shows the Windows taskbar with various pinned icons.

Output:

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the same "PYTHON_PRACTICE_CODES" folder as the previous screenshot.
- Code Editor:** Displays the same "ifelse2.py" code as before.
- Terminal:** Shows the command-line output:

```
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/ifelse2.py
p/python_practice_codes/ifelse2.py
p/python_practice_codes/ifelse2.py
> Enter a number 12
Number is even
PS C:\Users\Administrator\Desktop\python_practice_codes>
```
- Status Bar:** Shows "Ln 6, Col 32" and "Python 3.13.1 64-bit".
- Taskbar:** Shows the Windows taskbar with various pinned icons.

El if:

Simple Python program to understand elif statement

Code:

The screenshot shows the Visual Studio Code interface with the title bar "python_practice_codes [Administrator]". The Explorer sidebar on the left lists various Python files in the "PYTHON_PRACTICE_CODES" folder. The "ELIF1.PY" file is selected and shown in the main editor area. The code contains an if-elif-else conditional statement. The status bar at the bottom right shows "Ln 11, Col 1" and "Python 3.13.1 64-bit".

```
number = int(input("Enter the number?"))
if number == 10:
    print("The given number is equal to 10")
elif number == 50:
    print("The given number is equal to 50")
elif number == 100:
    print("The given number is equal to 100")
else:
    print("The given number is not equal to 10, 50, or 100")
```

Output:

The screenshot shows the Visual Studio Code interface with the title bar "python_practice_codes [Administrator]". The Explorer sidebar on the left lists various Python files in the "PYTHON_PRACTICE_CODES" folder. The "ELIF1.PY" file is selected and shown in the main editor area. Below the editor is the Terminal tab, which displays the command-line session. The user runs the script and enters a value, receiving the expected output. The status bar at the bottom right shows "Ln 11, Col 1" and "Python 3.13.1 64-bit".

```
PS C:\Users\Administrator\Desktop\python_practice_codes> ^C
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/elif1.py
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/elif1.py
Enter the number? 20
The given number is not equal to 10, 50, or 100
PS C:\Users\Administrator\Desktop\python_practice_codes>
```

Program to calculate the grade

Code:

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

- File Menu:** File, Edit, Selection, View, Go, Run, ...
- Title Bar:** python_practice_codes [Administrator]
- Left Sidebar (Explorer):** Shows a tree view of files in the "PYTHON_PRACTICE_CODES" folder. The "elif2.py" file is currently selected.
- Code Editor:** Displays the following Python code for calculating grades based on marks:

```
1  marks = int(input("Enter the marks? "))
2
3  if marks > 85 and marks <= 100:
4      print("Congrats! You scored grade A ...")
5  elif marks > 60 and marks <= 85:
6      print("You scored grade B+ ...")
7  elif marks > 40 and marks <= 60:
8      print("You scored grade B ...")
9  elif marks > 30 and marks <= 40:
10     print("You scored grade C ...")
11 else:
12     print("Sorry, you have failed.")
```

- Bottom Status Bar:** Shows Ln 12, Col 37, Spaces:4, UTF-8, CRLF, Python, 3.13.1 64-bit, 08:38, ENG US 06-02-2025.
- Bottom Taskbar:** Shows icons for File Explorer, Command Line, Mail, Task List, and VS Code.
- Bottom Search Bar:** Type here to search.

Output:

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The top bar includes File, Edit, Selection, View, Go, Run, and a search bar for "python_practice_codes [Administrator]". The left sidebar has sections for EXPLORER, PYTHON_PRACTICE_CODES (containing files like arithmetic.py, assign_op.py, etc.), OUTLINE, and TIMELINE. The main editor area displays a Python script named elif2.py:

```
marks = int(input("Enter the marks? "))
if marks > 85 and marks <= 100:
    print("Congrats! You scored grade A ...")
elif marks > 60 and marks <= 85:
    print("You scored grade B+ ...")
elif marks > 40 and marks <= 60:
    print("You scored grade B ...")
elif marks > 30 and marks <= 40:
    print("You scored grade C ...")
else:
    print("Sorry, you have failed.")
```

The bottom navigation bar includes PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is active), and PORTS. The terminal shows the following session:

```
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/elif2.py
Enter the marks? 76
You scored grade B+ ...
PS C:\Users\Administrator\Desktop\python_practice_codes>
```

A watermark for "Activate Windows" is visible in the bottom right corner.

For Loop:

Python program to show how the for loop works

Code:

A screenshot of the Visual Studio Code (VS Code) interface. The title bar says "python_practice_codes [Administrator]". The Explorer sidebar shows a folder named "PYTHON_PRACTICE_CODES" containing many Python files like arithmetic.py, assign_op.py, etc. The main editor tab is "forloop1.py". The code in the editor is:

```
1 numbers = [4, 2, 6, 7, 3, 5, 8, 10, 6, 1, 9, 2]
2
3 square = 0
4 squares = []
5
6 for value in numbers:
7     square = value ** 2
8     squares.append(square)
9
10 print("The list of squares is", squares)
```

The status bar at the bottom shows "Ln 10, Col 41" and "Python 3.13.1 64-bit".

Output:

A screenshot of the Visual Studio Code (VS Code) interface, similar to the previous one but with a terminal tab open. The title bar says "python_practice_codes [Administrator]". The Explorer sidebar shows the same folder structure. The main editor tab is "forloop1.py". The terminal tab shows the command line output:

```
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/forloop1.py
The list of squares is [16, 4, 36, 49, 9, 25, 64, 100, 36, 1, 81, 4]
```

The status bar at the bottom shows "Ln 11, Col 1" and "Python 3.13.1 64-bit".

Python program to show how if-else statements work inside for:

Code:

The screenshot shows the Visual Studio Code interface. The code editor displays the following Python script:

```
forloop2.py > ...
1 string = "Python Loop"
2
3 for s in string:
4     if s == "o":
5         print("If block")
6     else:
7         print(s)
```

The file is saved as `forloop2.py`. The status bar at the bottom indicates the file is 3.13.1 64-bit, Python 3.13.1 64-bit, and the date is 06-02-2025.

Output:

The screenshot shows the Visual Studio Code interface with the terminal tab selected. The terminal window displays the output of running the `forloop2.py` script in a PowerShell window:

```
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/forloop2.py
P
y
t
h
i
f
b
l
o
c
k
n
I
f
b
l
o
c
k
I
f
b
l
o
c
k
P
```

The status bar at the bottom indicates the file is 3.13.1 64-bit, Python 3.13.1 64-bit, and the date is 06-02-2025.

Python program to show how to use else statement with for loop

Code:

The screenshot shows the Visual Studio Code interface with the title bar "python_practice_codes [Administrator]". The Explorer sidebar on the left lists files in the "PYTHON_PRACTICE_CODES" folder, including "forloop3.py". The main editor tab contains the following Python code:

```
forloop3.py > ...
1 # Creating a sequence
2 tuple_ = (3, 4, 6, 8, 9, 2, 3, 8, 9, 7)
3
4 # Initiating the loop
5 for value in tuple_:
6     if value % 2 != 0:
7         print(value)
8 # The else block should be outside the loop
9 else:
10    print("These are the odd numbers present in the tuple")
```

The status bar at the bottom indicates "Ln 10, Col 60" and "Python 3.13.1 64-bit".

Output:

The screenshot shows the Visual Studio Code interface with the title bar "python_practice_codes [Administrator]". The Explorer sidebar on the left lists files in the "PYTHON_PRACTICE_CODES" folder, including "forloop3.py". The terminal tab at the bottom shows the command line output:

```
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/forloop3.py
3
9
3
9
7
These are the odd numbers present in the tuple
```

The status bar at the bottom indicates "Ln 11, Col 1" and "Python 3.13.1 64-bit".

Python program to iterate over a sequence with the help of indexing

Code:

A screenshot of the Visual Studio Code (VS Code) interface. The title bar says "python_practice_codes [Administrator]". The Explorer sidebar shows a folder named "PYTHON_PRACTICE_CODES" containing numerous Python files like arithmetic.py, assign_op.py, etc. The main editor tab is "forloop4.py" which contains the following code:

```
tuple_ = ("Python", "Loops", "Sequence", "Condition", "Range")
for iterator in range(len(tuple_)):
    print(tuple_[iterator].upper())
```

The status bar at the bottom shows "Ln 5, Col 36" and "Python 3.13.1 64-bit". A watermark "Activate Windows" with "Go to Settings to activate Windows." is visible.

Output:

A screenshot of the Visual Studio Code (VS Code) interface, similar to the previous one but with a different terminal output. The title bar says "python_practice_codes [Administrator]". The Explorer sidebar shows the same "PYTHON_PRACTICE_CODES" folder. The main editor tab is "forloop4.py" with the same code as before. The terminal tab is active and shows the following output:

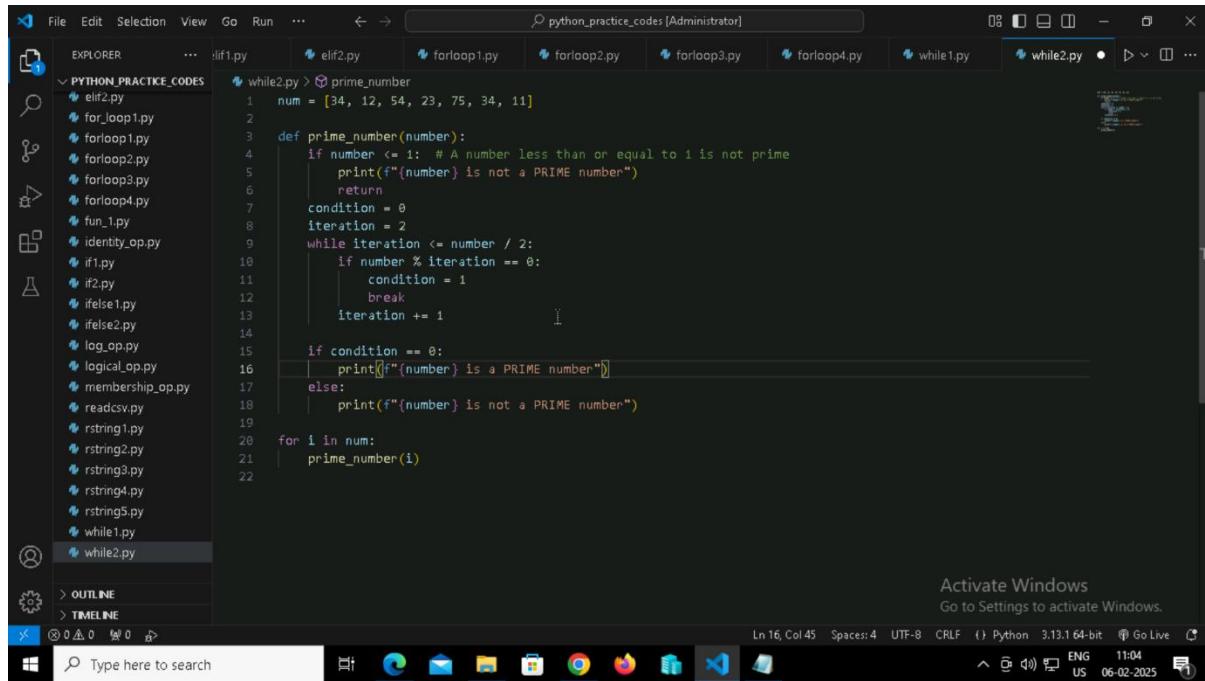
```
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/forloop4.py
PYTHON
LOOPS
SEQUENCE
CONDITION
RANGE
```

The status bar at the bottom shows "Ln 6, Col 1" and "Python 3.13.1 64-bit". A watermark "Activate Windows" with "Go to Settings to activate Windows." is visible.

While loop:

Prime number with while loop

Code:



```
File Edit Selection View Go Run ... 🔍 python_practice_codes [Administrator]
EXPLORER ... elif1.py elif2.py forloop1.py forloop2.py forloop3.py forloop4.py while1.py while2.py
PYTHON_PRACTICE_CODES
  prime_number.py
    num = [34, 12, 54, 23, 75, 34, 11]
    def prime_number(number):
        if number <= 1: # A number less than or equal to 1 is not prime
            print(f"{number} is not a PRIME number")
            return
        condition = 0
        iteration = 2
        while iteration <= number / 2:
            if number % iteration == 0:
                condition = 1
                break
            iteration += 1
        if condition == 0:
            print(f"{number} is a PRIME number")
        else:
            print(f"{number} is not a PRIME number")
    for i in num:
        prime_number(i)

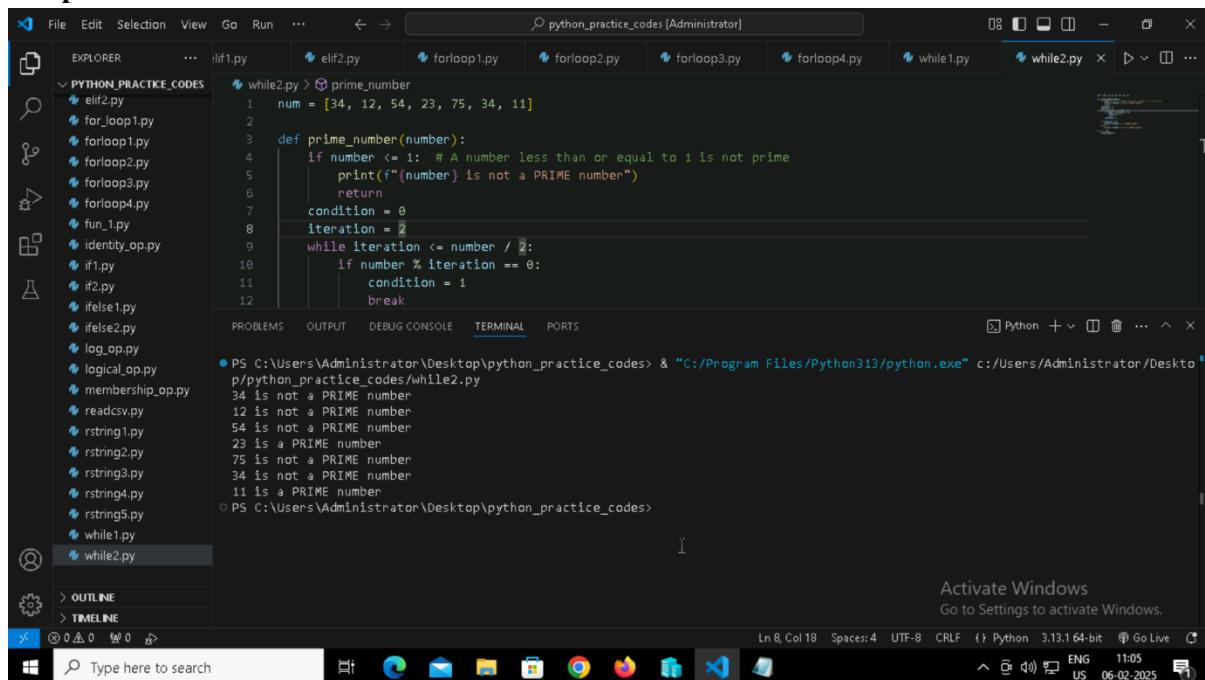
```

Activate Windows
Go to Settings to activate Windows.

Ln 16, Col 45 Spaces:4 UTF-8 CRLF Python 3.13.1 64-bit Go Live ⚡

Windows Type here to search ⌂ ENG 11:04 US 06-02-2025

Output:



```
File Edit Selection View Go Run ... 🔍 python_practice_codes [Administrator]
EXPLORER ... elif1.py elif2.py forloop1.py forloop2.py forloop3.py forloop4.py while1.py while2.py
PYTHON_PRACTICE_CODES
  prime_number.py
    num = [34, 12, 54, 23, 75, 34, 11]
    def prime_number(number):
        if number <= 1: # A number less than or equal to 1 is not prime
            print(f"{number} is not a PRIME number")
            return
        condition = 0
        iteration = 2
        while iteration <= number / 2:
            if number % iteration == 0:
                condition = 1
                break
            iteration += 1
        if condition == 0:
            print(f"{number} is a PRIME number")
        else:
            print(f"{number} is not a PRIME number")
    for i in num:
        prime_number(i)

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

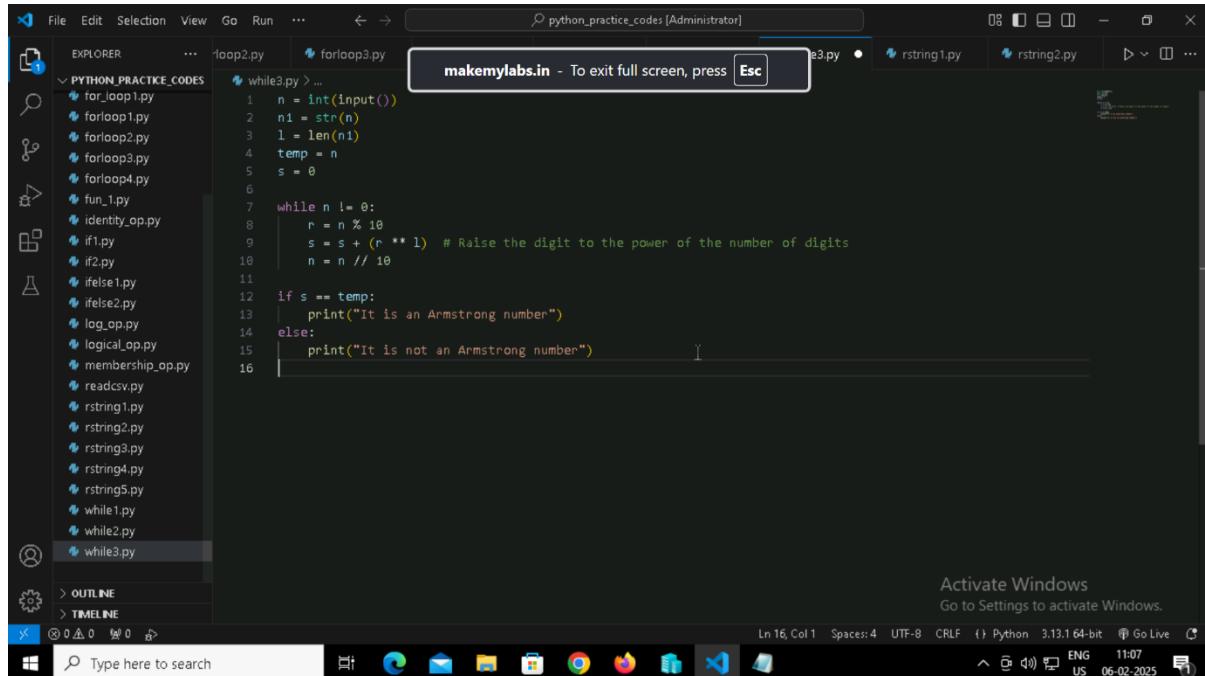
Activate Windows
Go to Settings to activate Windows.

Ln 8, Col 18 Spaces:4 UTF-8 CRLF Python 3.13.1 64-bit Go Live ⚡

Windows Type here to search ⌂ ENG 11:05 US 06-02-2025

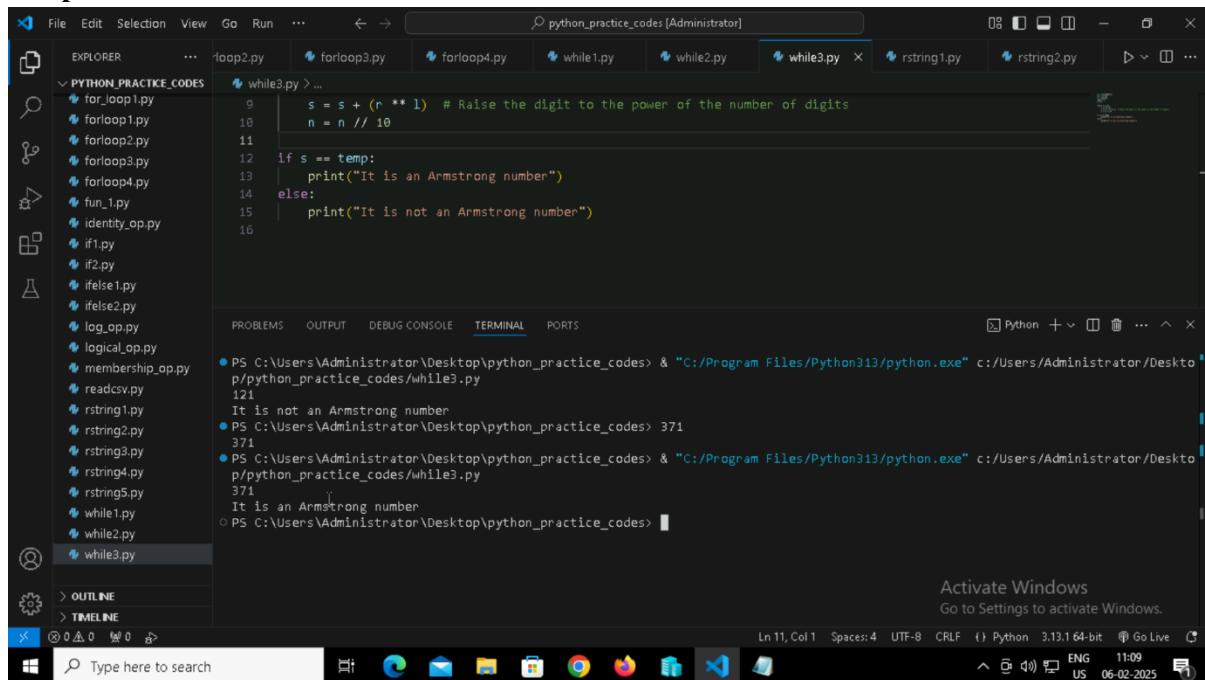
Armstrong number:

Code:



```
while3.py
while3.py > ...
1 n = int(input())
2 n1 = str(n)
3 l = len(n1)
4 temp = n
5 s = 0
6
7 while n != 0:
8     r = n % 10
9     s = s + (r ** l) # Raise the digit to the power of the number of digits
10    n = n // 10
11
12 if s == temp:
13     print("It is an Armstrong number")
14 else:
15     print("It is not an Armstrong number")
16
```

Output:



```
while3.py
while3.py > ...
9 s = s + (r ** l) # Raise the digit to the power of the number of digits
10 n = n // 10
11
12 if s == temp:
13     print("It is an Armstrong number")
14 else:
15     print("It is not an Armstrong number")
16
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/while3.py
121
It is not an Armstrong number
PS C:\Users\Administrator\Desktop\python_practice_codes> 371
371
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/while3.py
371
It is an Armstrong number
PS C:\Users\Administrator\Desktop\python_practice_codes>
```

Python for determine odd and even number from every number of a list

Code:

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder named "PYTHON_PRACTICE_CODES" containing various Python files like forloop1.py, forloop2.py, etc., and some other files like if1.py, if2.py, etc.
- Code Editor:** The active file is "while4.py". The code initializes a list with values [3, 4, 8, 10, 34, 45, 67, 80] and iterates through it using a while loop. It prints each element, checking if it's even or odd using the modulo operator (%). Even numbers are printed as "{element} is an even number", and odd numbers are printed as "{element} is an odd number".

```
while4.py > ...
1 list_ = [3, 4, 8, 10, 34, 45, 67, 80] # Initialize the list
2 index = 0
3
4 while index < len(list_):
5     element = list_[index]
6
7     if element % 2 == 0:
8         print(f"{element} is an even number") # Print the actual number if it's even
9     else:
10        print(f"{element} is an odd number") # Print the actual number if it's odd
11
12     index += 1
```
- Status Bar:** Shows "Activate Windows Go to Settings to activate Windows.", "Ln 13, Col 1 Spaces:4 UTF-8 CRLF Python 3.13.1 64-bit Go Live", and the date/time "06-02-2025 11:10".

Output:

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Same as the previous screenshot, showing "PYTHON_PRACTICE_CODES" folder.
- Terminal:** The terminal tab is active, showing the command PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/while4.py. The output of the script is displayed:

```
3 is an odd number
4 is an even number
8 is an even number
10 is an even number
34 is an even number
45 is an odd number
67 is an odd number
80 is an even number
```
- Status Bar:** Shows "Activate Windows Go to Settings to activate Windows.", "Ln 13, Col 1 Spaces:4 UTF-8 CRLF Python 3.13.1 64-bit Go Live", and the date/time "06-02-2025 11:10".

Strings:

Using operators and slicing

Code:

The screenshot shows the Visual Studio Code interface with the title bar "python_practice_codes [Administrator]". The Explorer sidebar on the left lists files in the "PYTHON_PRACTICE_CODES" folder, including "string1.py". The main editor pane displays the following Python code:

```
# Defining strings
str = "Hello"
stri = " world"

# String repetition
print(str * 3) # Output: HelloHelloHello

# String concatenation
print(str + stri) # Output: Hello world

# Accessing a character by index
print(stri[4]) # Output: o

# String slicing
print(stri[2:4]) # Output: ll

# Membership operator: checking if 'w' is in the string
print('w' in str) # Output: False

# Checking if 'wo' is NOT in str
print('wo' not in str) # Output: True

# Raw string (ignores escape sequences)
print(r'C:/python37') # Output: C:/python37

# String formatting using %
print("The string str : %s" % (str)) # Output: The string str : Hello
```

The status bar at the bottom shows "Ln 28, Col 1" and "Python 3.13.1 64-bit".

Output:

The screenshot shows the Visual Studio Code interface with the title bar "python_practice_codes [Administrator]". The Explorer sidebar on the left lists files in the "PYTHON_PRACTICE_CODES" folder, including "string1.py". The main editor pane displays the same Python code as before. Below the editor is the Terminal tab, which shows the command line output:

```
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/string1.py
HelloHelloHello
Hello world
o
ll
False
False
C://python37
The string str : Hello
○ PS C:\Users\Administrator\Desktop\python_practice_codes>
```

The status bar at the bottom shows "Ln 6, Col 42" and "Python 3.13.1 64-bit".

Using slicing:

Code:

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder named "PYTHON_PRACTICE_CODES" containing various Python files: forloop3.py, forloop4.py, fun_1.py, identity_op.py, if1.py, if2.py, ifelse1.py, ifelse2.py, log_op.py, logical_op.py, membership_op.py, readcsv.py, rstring1.py, rstring2.py, rstring3.py, rstring4.py, rstring5.py, string1.py, string2.py, while1.py, while2.py, while3.py, while4.py.
- Code Editor:** Displays the content of `string2.py`. The code demonstrates various string slicing operations on the string "JAVAATPOINT".
- Status Bar:** Shows "Ln 20, Col 1" and "Python 3.13.1 64-bit".
- Taskbar:** Shows icons for File Explorer, Task Manager, Mail, Photos, OneDrive, Edge, Google Chrome, Firefox, and VS Code.

Output:

The screenshot shows the Visual Studio Code interface with the following details:

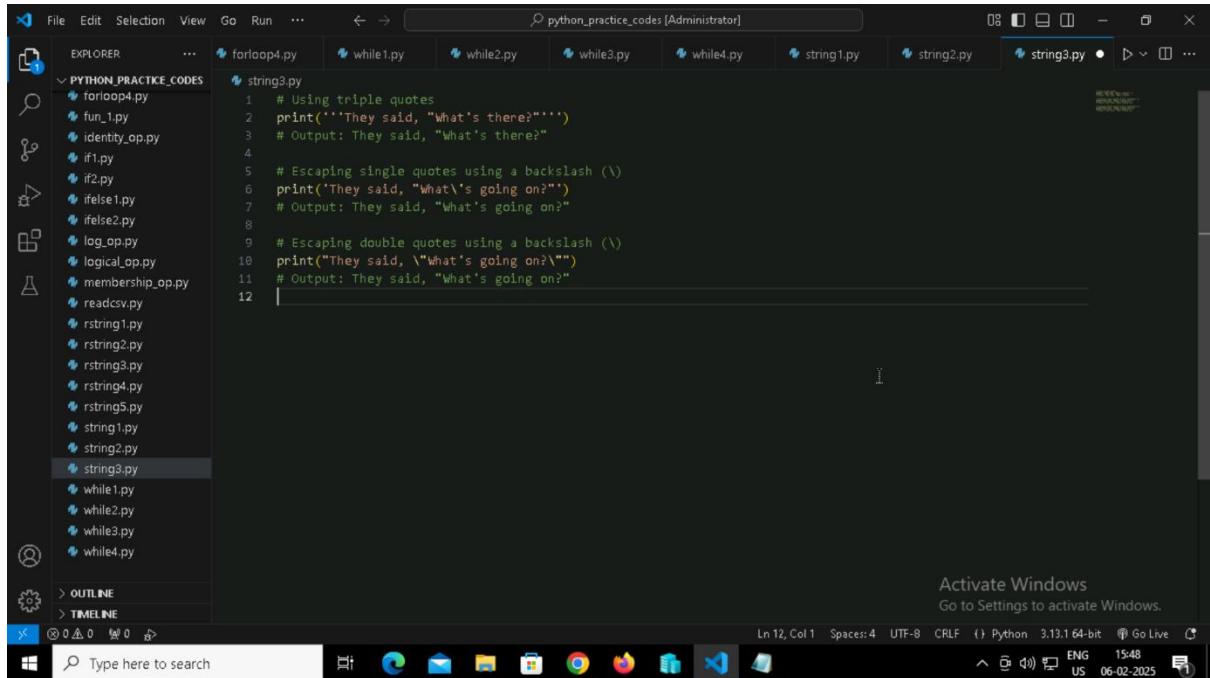
- File Explorer:** Same as the previous screenshot, showing the "PYTHON_PRACTICE_CODES" folder.
- Terminal:** Shows the command "python python_practice_codes/string2.py" being run, and the output of the code execution.
- Status Bar:** Shows "Ln 10, Col 30" and "Python 3.13.1 64-bit".
- Taskbar:** Same as the previous screenshot.

The terminal output is as follows:

```
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/string2.py
T
I
NT
OIN
ATPOI
TNIOPTAVAJ
```

String Formatting:

Code:

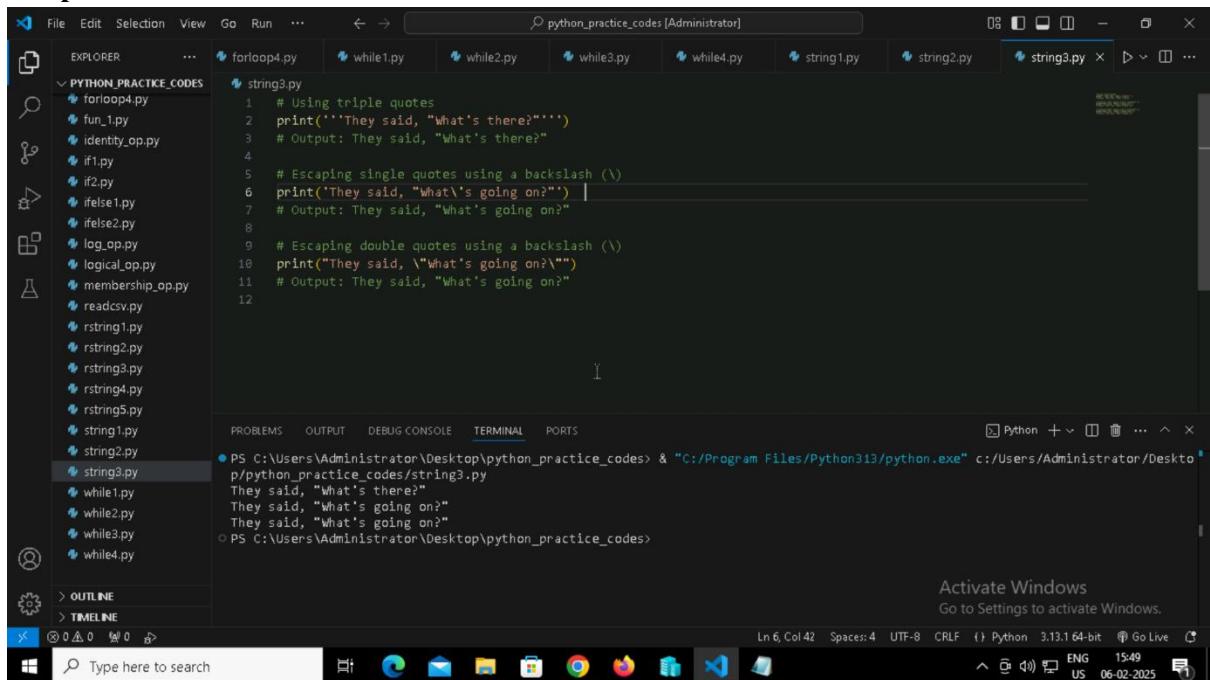


A screenshot of the Visual Studio Code (VS Code) interface. The title bar says "python_practice_codes [Administrator]". The Explorer sidebar shows a folder named "PYTHON_PRACTICE_CODES" containing several Python files. The "string3.py" file is open in the editor. The code demonstrates various string escaping techniques:

```
string3.py
1 # Using triple quotes
2 print('''They said, "What's there?'''')
3 # Output: They said, "What's there?"
4
5 # Escaping single quotes using a backslash (\)
6 print('They said, "What\'s going on?"')
7 # Output: They said, "What's going on?"
8
9 # Escaping double quotes using a backslash (\)
10 print("They said, \"What's going on?\"")
11 # Output: They said, "What's going on?"
```

The status bar at the bottom shows "Ln 12, Col 1" and "Python 3.13.1 64-bit".

Output:



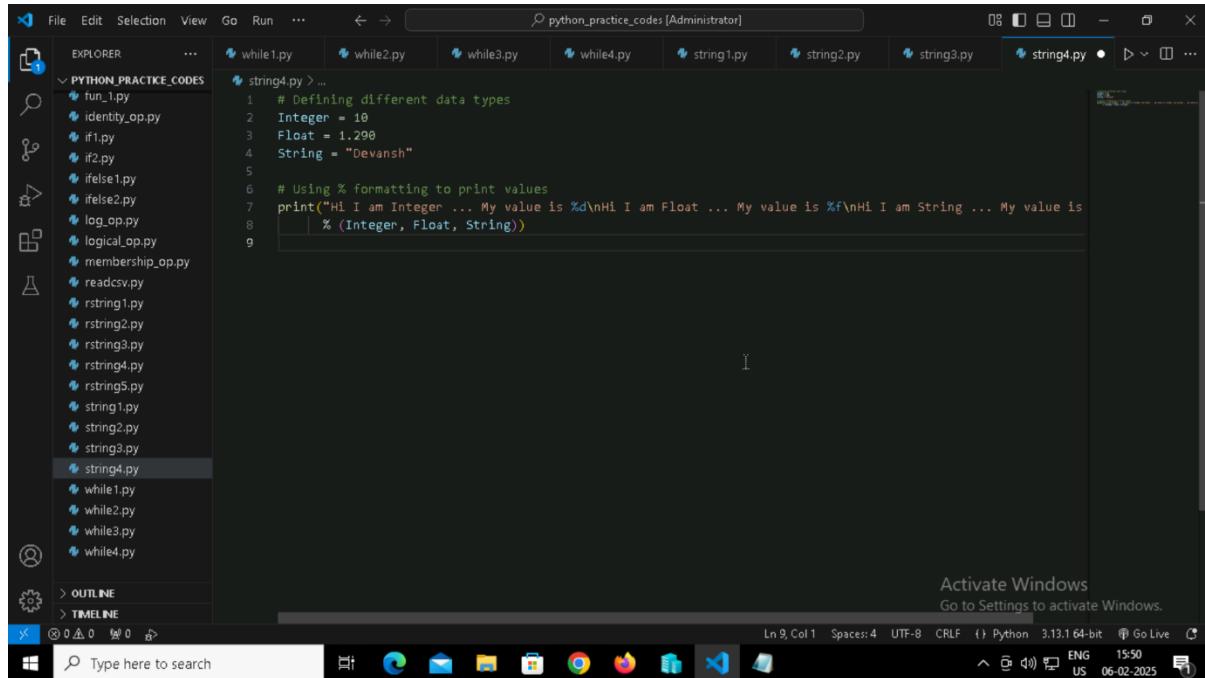
A screenshot of the Visual Studio Code interface, similar to the previous one but with a terminal tab open. The title bar says "python_practice_codes [Administrator]". The Explorer sidebar shows the same "PYTHON_PRACTICE_CODES" folder. The "string3.py" file is selected in the editor. The terminal tab shows the command "python3 string3.py" being run and its output:

```
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/string3.py
They said, "What's there?"
They said, "What's going on?"
```

The status bar at the bottom shows "Ln 6, Col 42" and "Python 3.13.1 64-bit".

Formatting using %

Code:



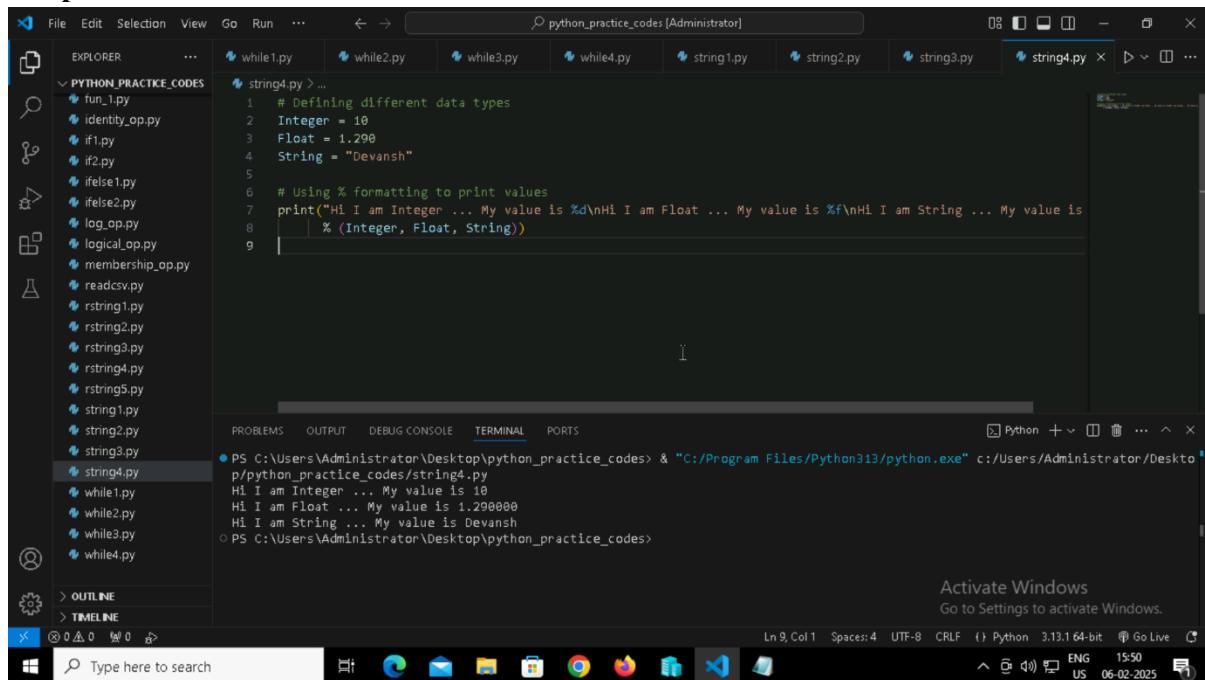
A screenshot of the Visual Studio Code (VS Code) interface. The title bar says "python_practice_codes [Administrator]". The Explorer sidebar shows a folder named "PYTHON_PRACTICE_CODES" containing numerous Python files. The main editor tab is "string4.py", which contains the following code:

```
# Defining different data types
Integer = 10
Float = 1.290
String = "Devanash"

# Using % formatting to print values
print("Hi I am Integer ... My value is %d\nHi I am Float ... My value is %f\nHi I am String ... My value is %s" % (Integer, Float, String))
```

The status bar at the bottom shows "Ln 9, Col 1 Spaces:4 UTF-8 CRLF Python 3.13.1 64-bit Go Live".

Output:



A screenshot of the Visual Studio Code interface, similar to the previous one but with the terminal tab active. The title bar says "python_practice_codes [Administrator]". The Explorer sidebar is the same. The terminal tab shows the command "PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/string4.py" followed by the output:

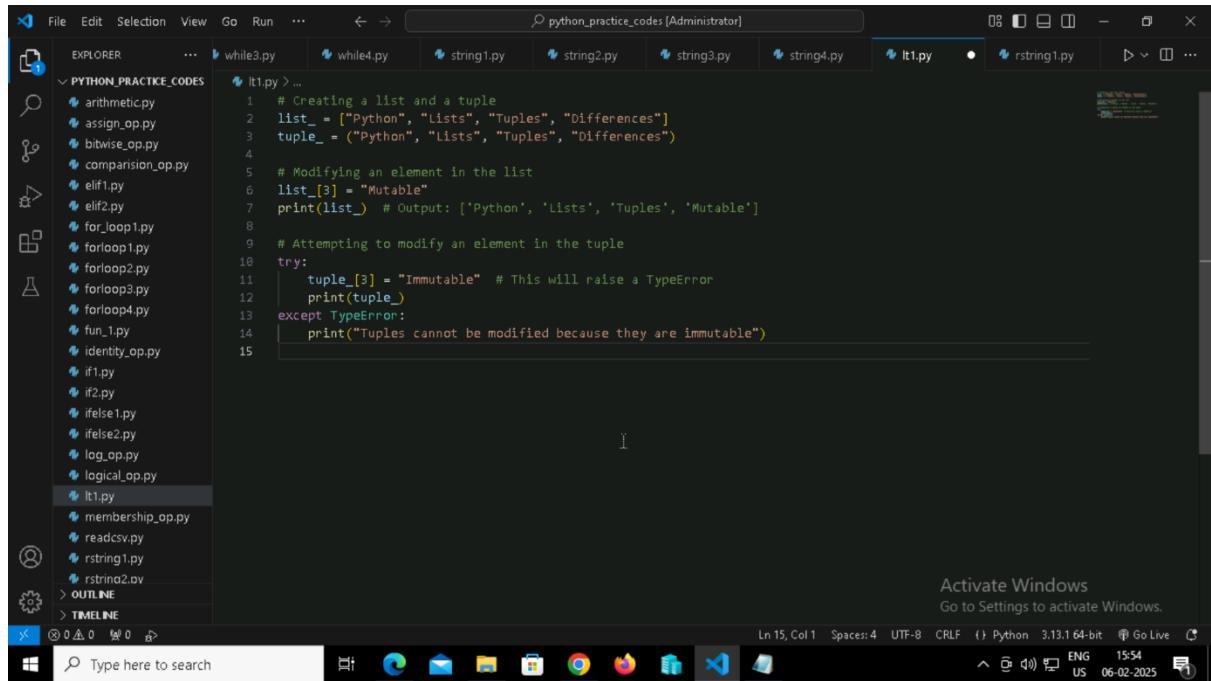
```
Hi I am Integer ... My value is 10
Hi I am Float ... My value is 1.290000
Hi I am String ... My value is Devanash
```

The status bar at the bottom shows "PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + ENG 15:50 US 06-02-2025".

List and Tuple :

Modifying element in list and tuple:

Code:

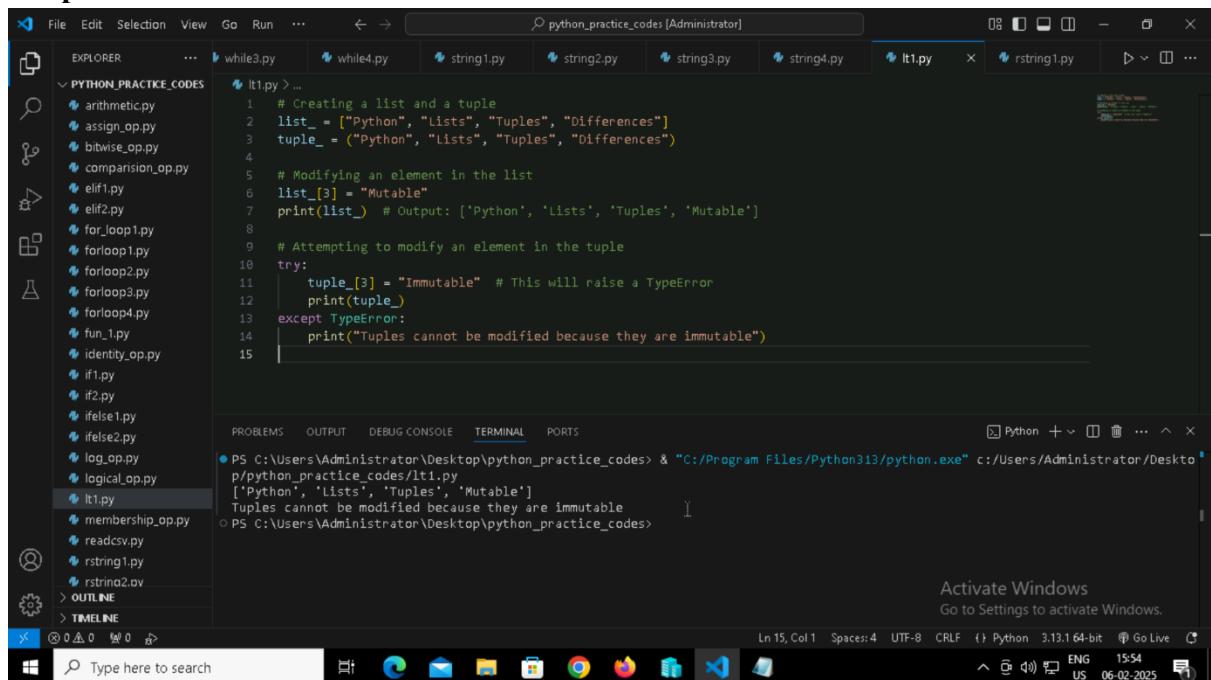


```
# Creating a list and a tuple
list_ = ["Python", "Lists", "Tuples", "Differences"]
tuple_ = ("Python", "Lists", "Tuples", "Differences")

# Modifying an element in the list
list_[3] = "Mutable"
print(list_) # Output: ['Python', 'Lists', 'Tuples', 'Mutable']

# Attempting to modify an element in the tuple
try:
    tuple_[3] = "Immutable" # This will raise a TypeError
    print(tuple_)
except TypeError:
    print("Tuples cannot be modified because they are immutable")
```

Output:



```
# Creating a list and a tuple
list_ = ["Python", "Lists", "Tuples", "Differences"]
tuple_ = ("Python", "Lists", "Tuples", "Differences")

# Modifying an element in the list
list_[3] = "Mutable"
print(list_) # Output: ['Python', 'Lists', 'Tuples', 'Mutable']

# Attempting to modify an element in the tuple
try:
    tuple_[3] = "Immutable" # This will raise a TypeError
    print(tuple_)
except TypeError:
    print("Tuples cannot be modified because they are immutable")
```

PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/lt1.py
['Python', 'Lists', 'Tuples', 'Mutable']
Tuples cannot be modified because they are immutable

Printing sizes:

Code:

The screenshot shows a Windows desktop environment. In the center is a Microsoft Visual Studio Code window titled "python_practice_codes [Administrator]". The Explorer sidebar on the left lists several Python files under the folder "PYTHON_PRACTICE_CODES", including arithmetic.py, assign_op.py, bitwise_op.py, comparison_op.py, elif1.py, elif2.py, for_loop1.py, forloop1.py, forloop2.py, forloop3.py, forloop4.py, fun_1.py, identity_op.py, if1.py, if2.py, ifelse1.py, ifelse2.py, log_op.py, logical_op.py, lt1.py, lt2.py, membership_op.py, readcsv.py, rstring1.py, and rstring1.ov. The terminal tab at the bottom of the VS Code window shows the command "PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/lt1.py" followed by the output "[‘Python’, ‘Lists’, ‘Tuples’, ‘Mutable’] Tuples cannot be modified because they are immutable". Below the terminal is a standard Windows taskbar with icons for File Explorer, Task View, Start, Taskbar settings, and the date/time (06-02-2025).

```
lt2.py > ...
1 # Creating a list and a tuple
2 list_ = ["Python", "Lists", "Tuples", "Differences"]
3 tuple_ = ("Python", "Lists", "Tuples", "Differences")
4
5 # Printing sizes
6 print("Size of tuple: ", tuple_.__sizeof__())
7 print("Size of list: ", list_.__sizeof__())
8
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/lt1.py
[‘Python’, ‘Lists’, ‘Tuples’, ‘Mutable’]
Tuples cannot be modified because they are immutable

PS C:\Users\Administrator\Desktop\python_practice_codes>

Type here to search

Activate Windows
Go to Settings to activate Windows.

Ln 8, Col 1 Spaces:4 UTF-8 CRLF Python 3.13.1 64-bit Go Live

15:55 ENG US 06-02-2025

Output:

This screenshot is identical to the one above, showing the same VS Code window with the "lt2.py" file open and its contents. The terminal output shows the sizes of the tuple and list objects. The Windows taskbar at the bottom is also identical.

```
lt2.py > ...
1 # Creating a list and a tuple
2 list_ = ["Python", "Lists", "Tuples", "Differences"]
3 tuple_ = ("Python", "Lists", "Tuples", "Differences")
4
5 # Printing sizes
6 print("Size of tuple: ", tuple_.__sizeof__())
7 print("Size of list: ", list_.__sizeof__())
8
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/lt2.py
Size of tuple: 56
Size of list: 72

PS C:\Users\Administrator\Desktop\python_practice_codes>

Type here to search

Activate Windows
Go to Settings to activate Windows.

Ln 8, Col 1 Spaces:4 UTF-8 CRLF Python 3.13.1 64-bit Go Live

15:56 ENG US 06-02-2025