PYTHON PROGRAMMING LANGUAGE

ASSIGNMENT

OPERATORS

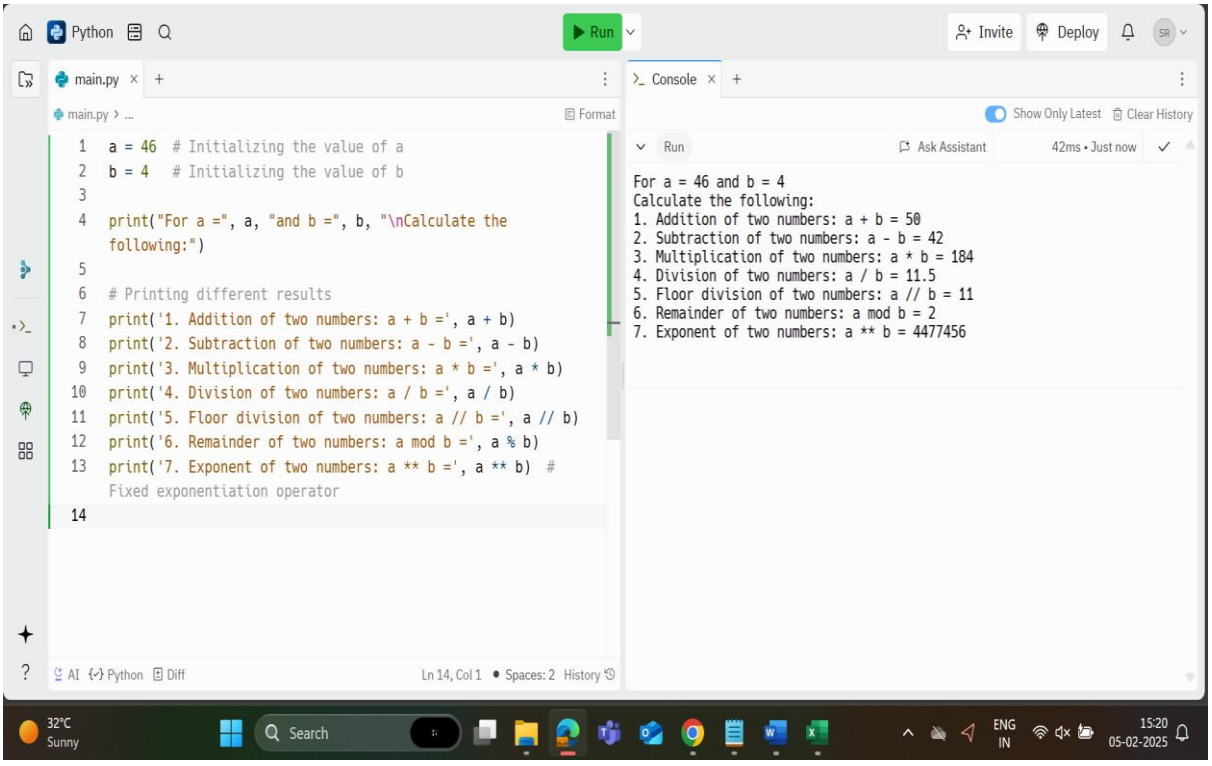
Operators are the symbols used to perform a specific operation on different values and variables. These values and variables are considered as the Operands, on which the operator is applied. Operators serve as the foundation upon which logic is constructed in a program in a particular programming language. In every programming language, some operators perform several tasks.

Arithmetic Operators

- Python Arithmetic Operators are used on two operands to perform basic mathematical operators like addition, subtraction, multiplication, and division.
- There are different types of arithmetic operators available in Python including the '+' operator for addition, '-' operator for subtraction, '*' for multiplication, '/' for division,
- '%' for modulus, '**' for exponent and '//' for floor division.

Example 1: Using all Arithmetic Operators:

Output:

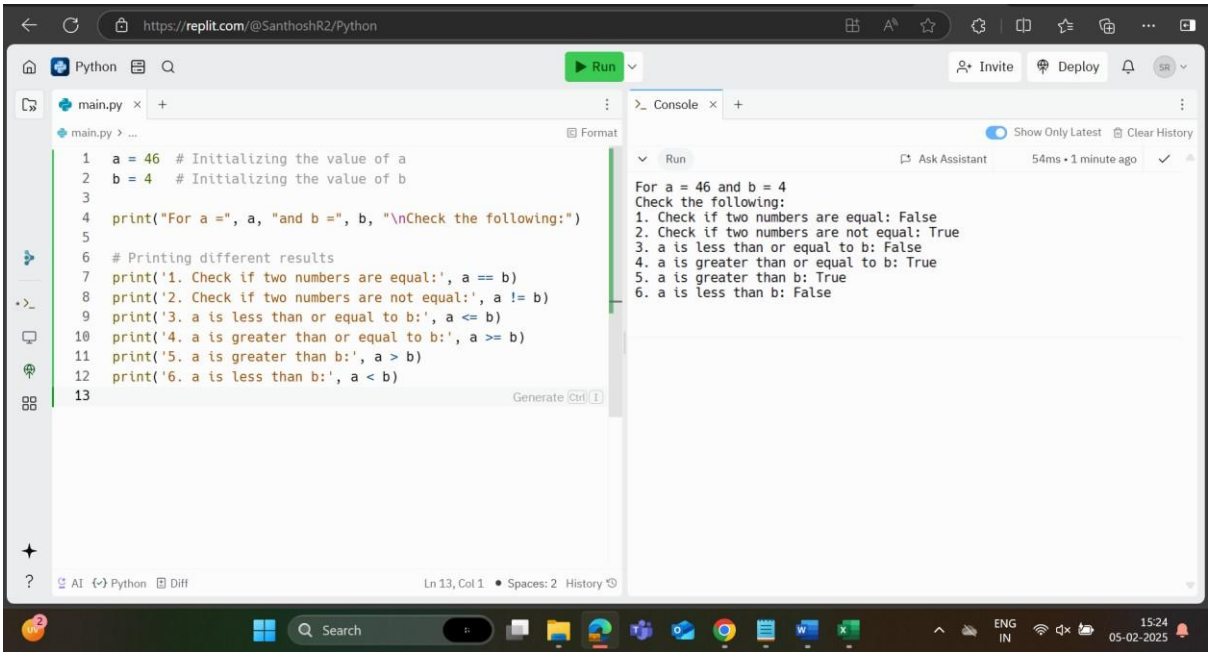


Comparison Operators

Python Comparison operators are mainly used for the purpose of comparing two values or variables (operands) and return a Boolean value as either True or False accordingly. There are various types of comparison operators available in Python including the '==', '!=', '<=', '>=', '<', and '>'.

Example 2: Using all Comparison Operators:

Output:

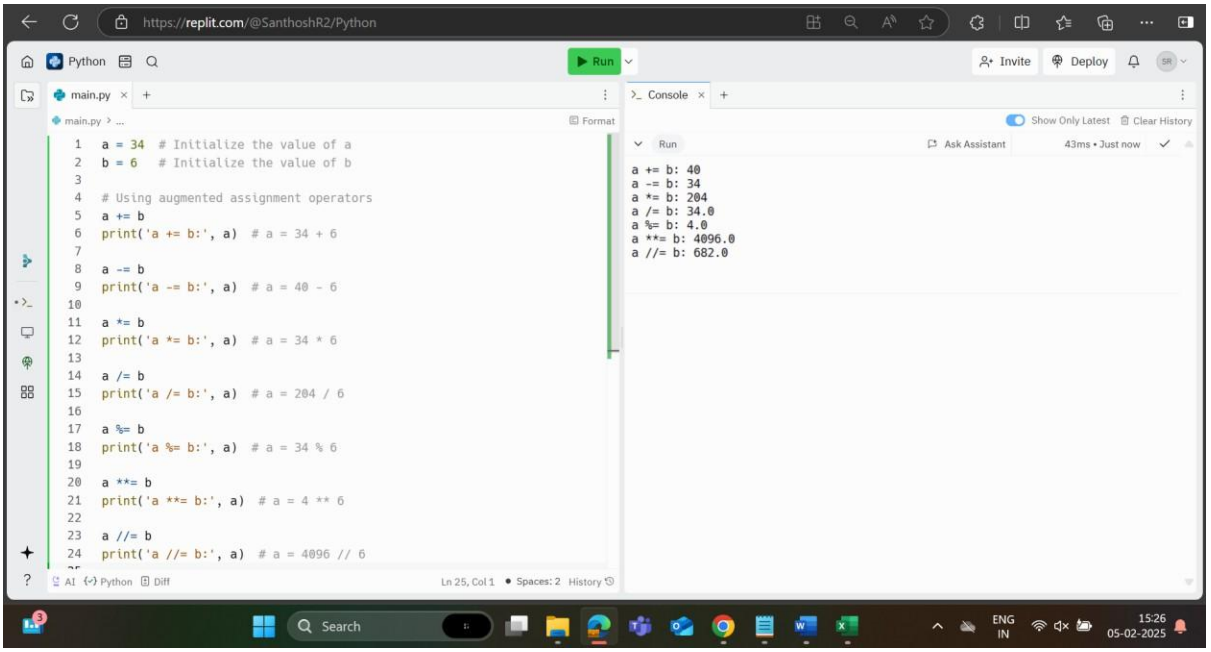


Assignment Operators

Using the assignment operators, the right expression's value is assigned to the left operand. Python offers different assignment operators to assign values to the variable. These assignment operators include '=', '+=', '-=', '*=', '/=', '%=', '//=', '**=', '&=', '|=', '^=', '>>=', and '<<='.

Example 3: Using all Assignment Operators:

Output:

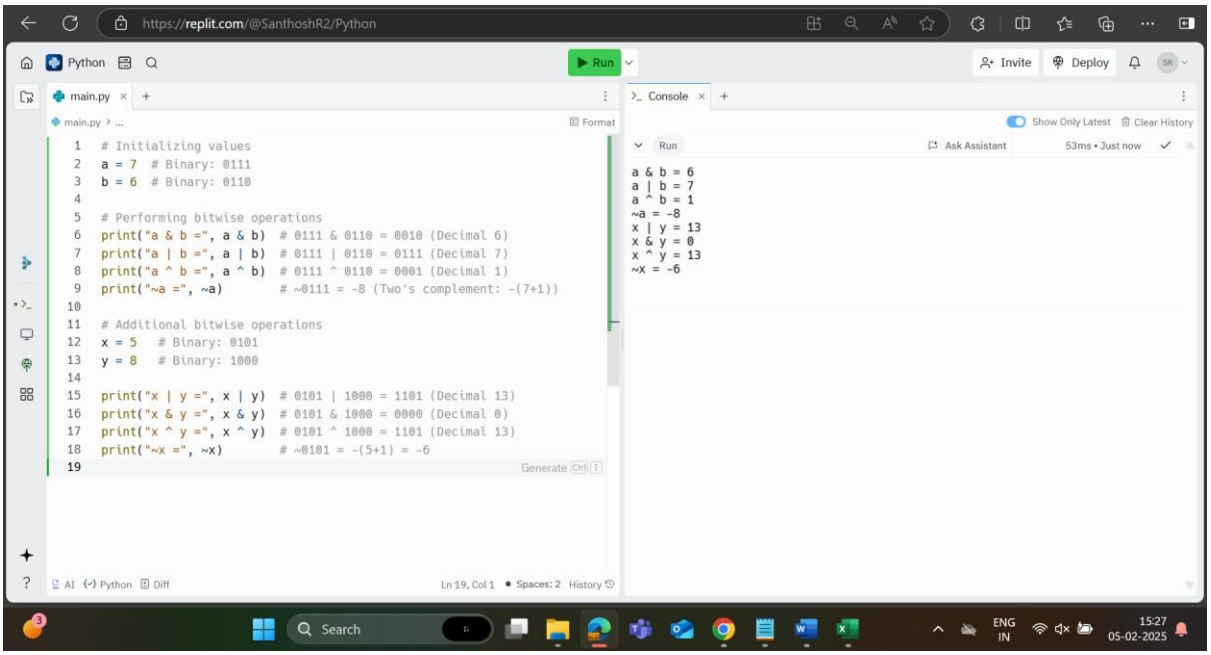


Bitwise Operators

The two operands' values are processed bit by bit by the bitwise operators. There are various Bitwise operators used in Python, such as bitwise OR (|), bitwise AND (&), bitwise XOR (^), negation (~), Left shift (<>). Consider the case below.

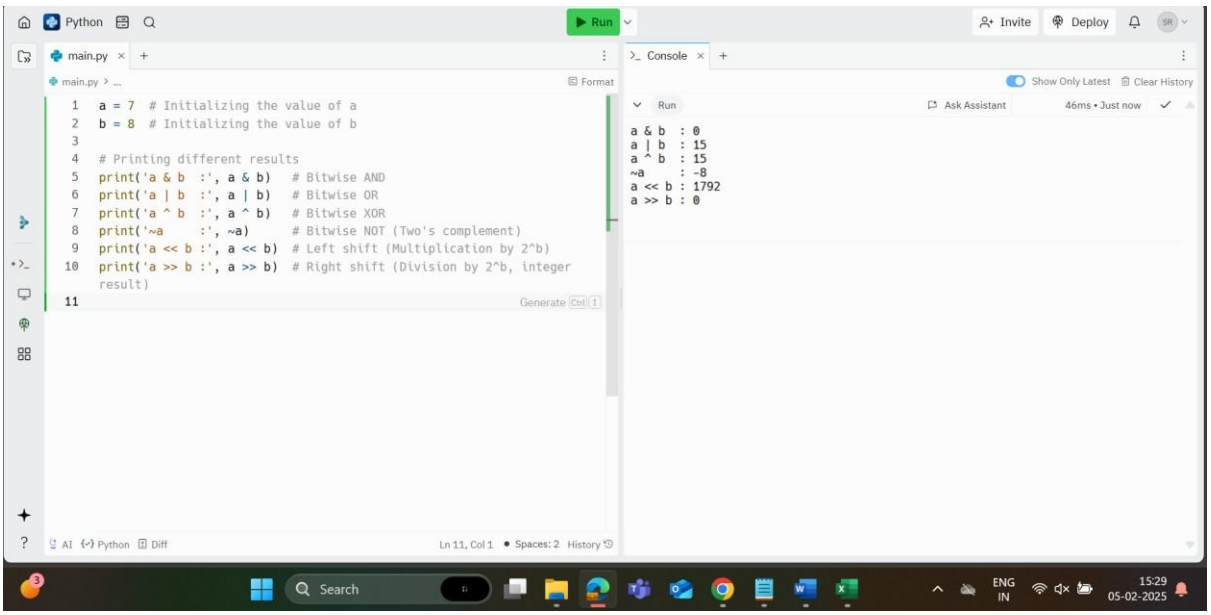
Example 4: Using all Bitwise Operators:

Output:



Example 5: Using all Bitwise Operators:

Output:

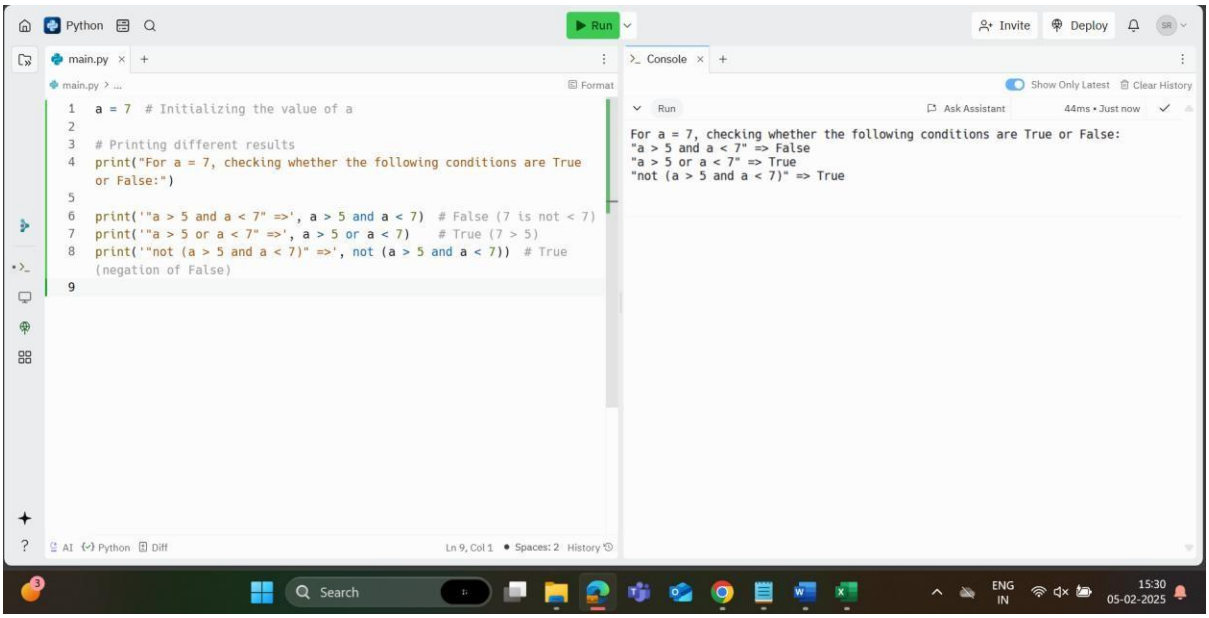


Logical Operators

The assessment of expressions to make decisions typically uses logical operators. Python offers different types of logical operators such as and, or, and not. In the case of the logical AND, if the first one is 0, it does not depend upon the second one. In the case of the logical OR, if the first one is 1, it does not depend on the second one.

Example 6: Using all Logical Operators:

Output:

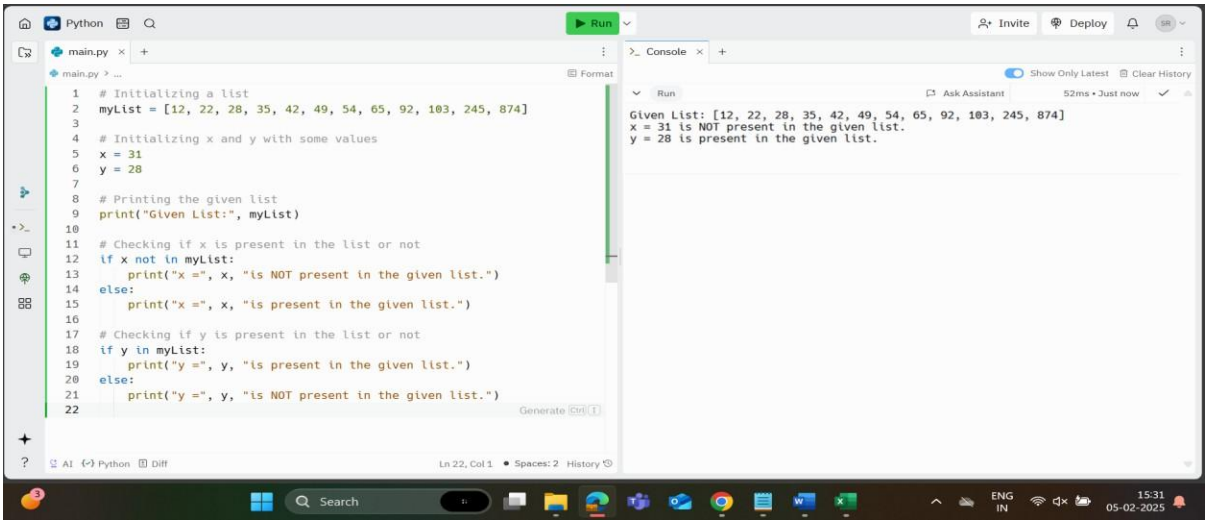


Membership Operators

We can verify the membership of a value inside a Python data structure using the Python membership operators. The result is said to be true if the value or variable is in the sequence (list, tuple, or dictionary); otherwise, it returns false.

Example 7: Using all Membership Operators:

Output:

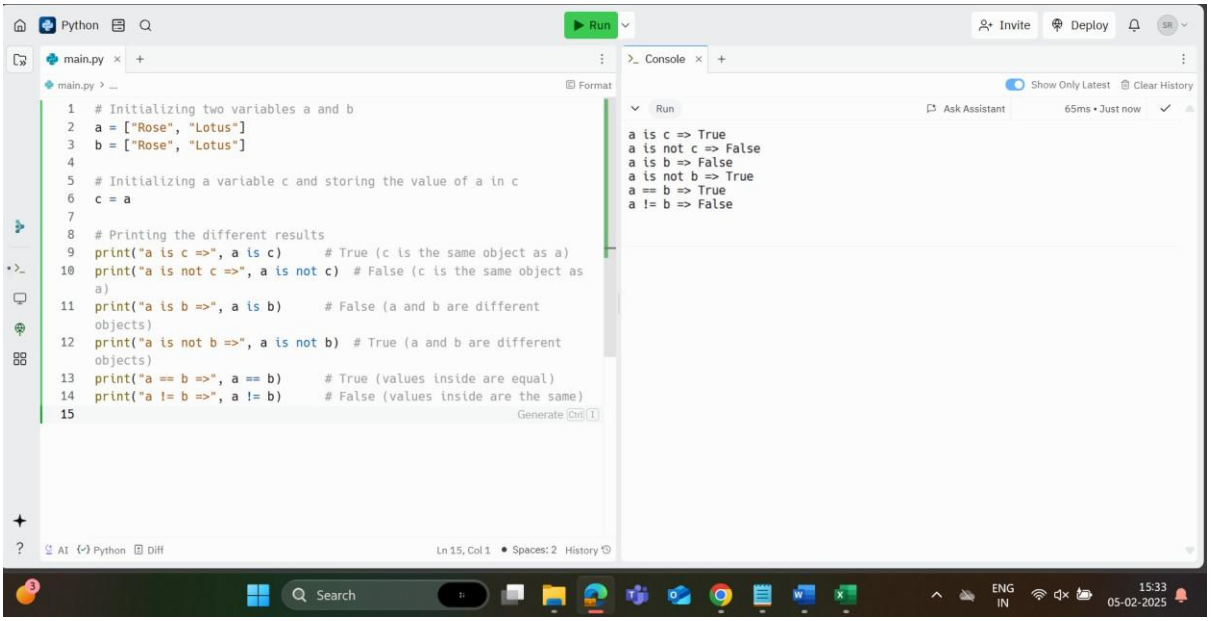


Identity Operators

Python offers two identity operators as is and is not, that are used to check if two values are located on the same part of the memory. Two variables that are equal do not imply that they are identical.

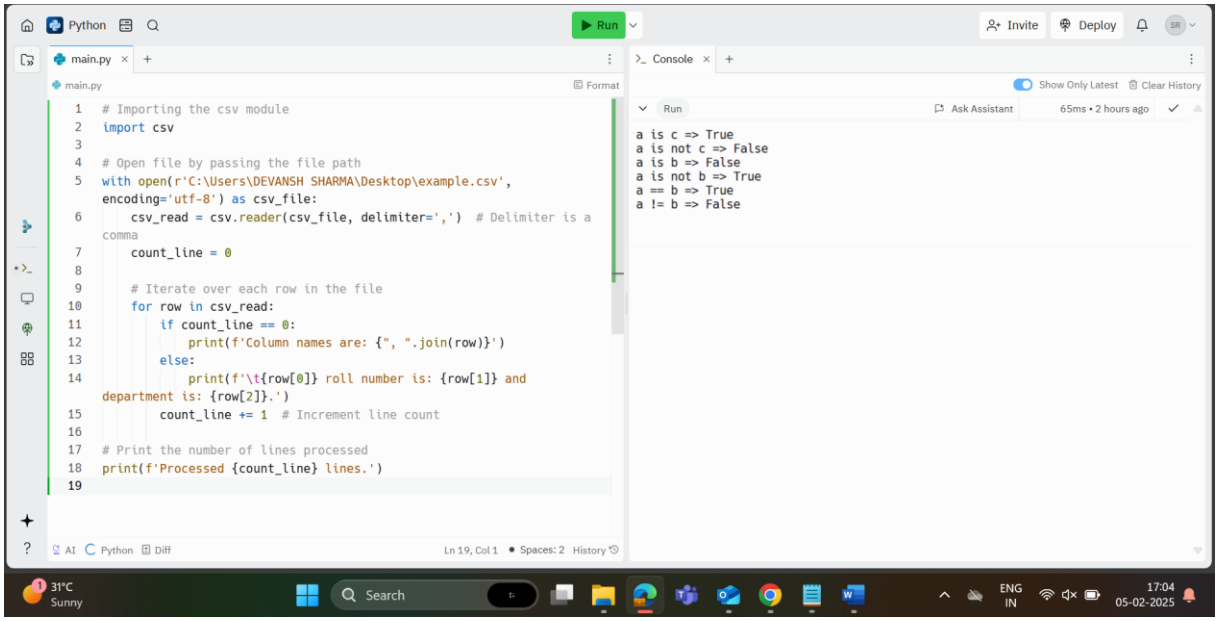
Example 8: Using all Identity Operators:

Output:



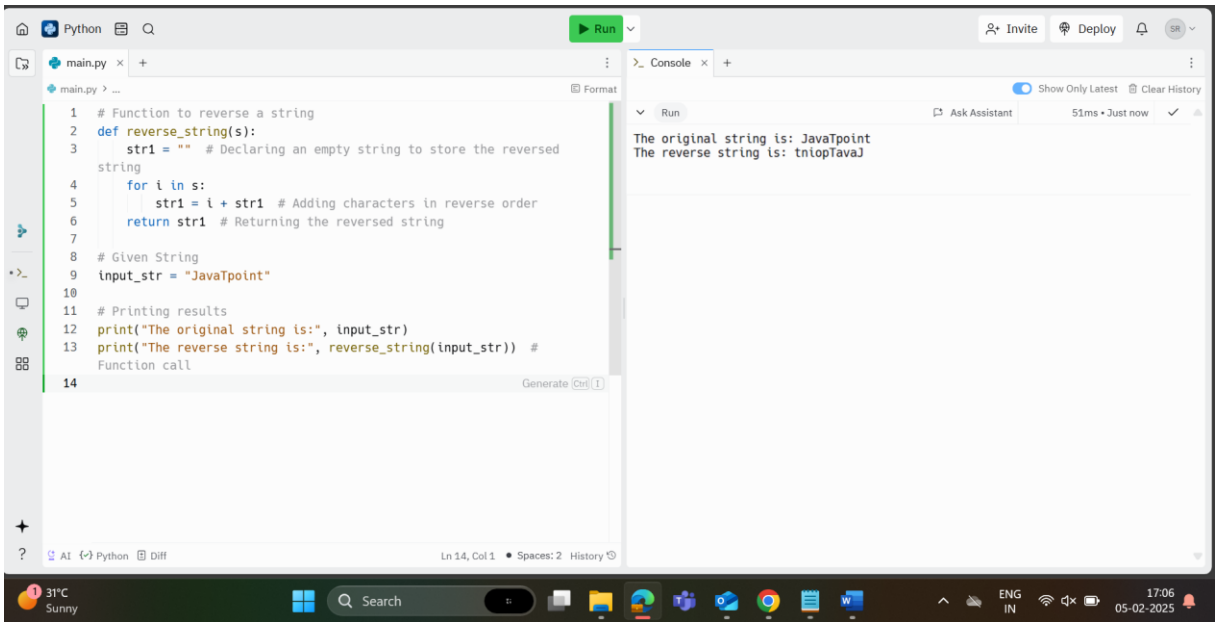
How to read CSV file in Python

The CSV file stands for a comma-separated values file. It is a type of plain text file where the information is organized in the tabular form. It can contain only the actual text data. The textual data don't need to be separated by the commas (,). There are also many separator characters such as tab (\t), colon(:), and semi-colon(;), which can be used as a separator. Let's understand the following example.



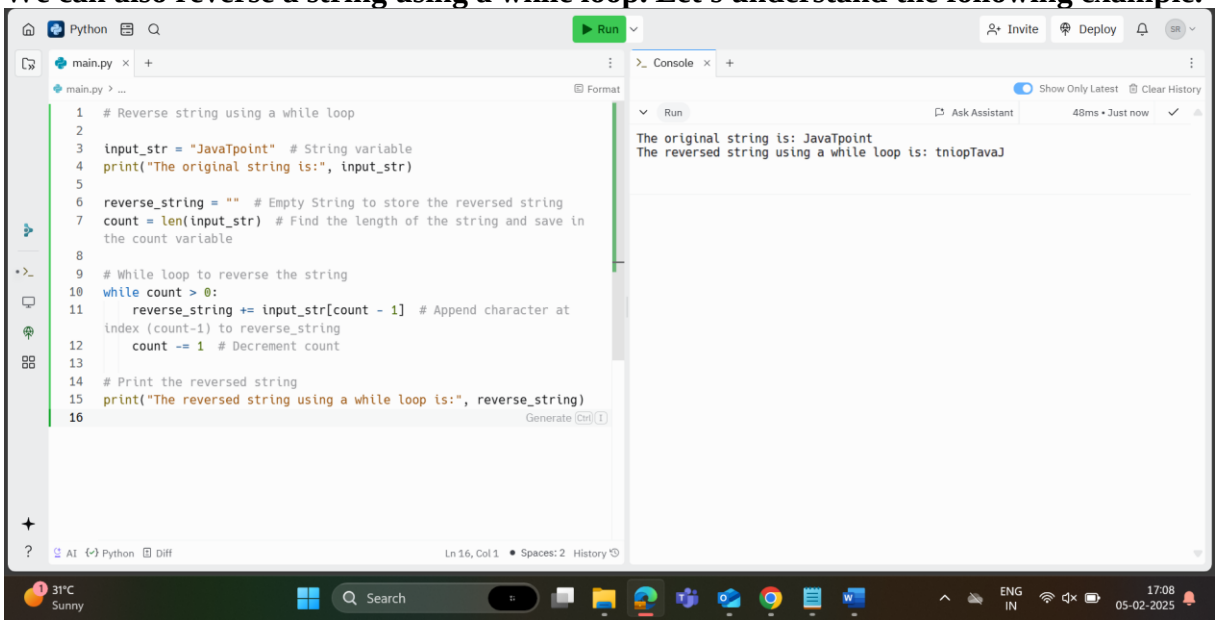
How to reverse a string in Python

The collection of Unicode characters is Python String. Python has various capabilities for string control, yet Python string library doesn't uphold the in-constructed "switch()" capability. However, there are numerous methods for reversing the string. The following reverse Python String method is being defined.



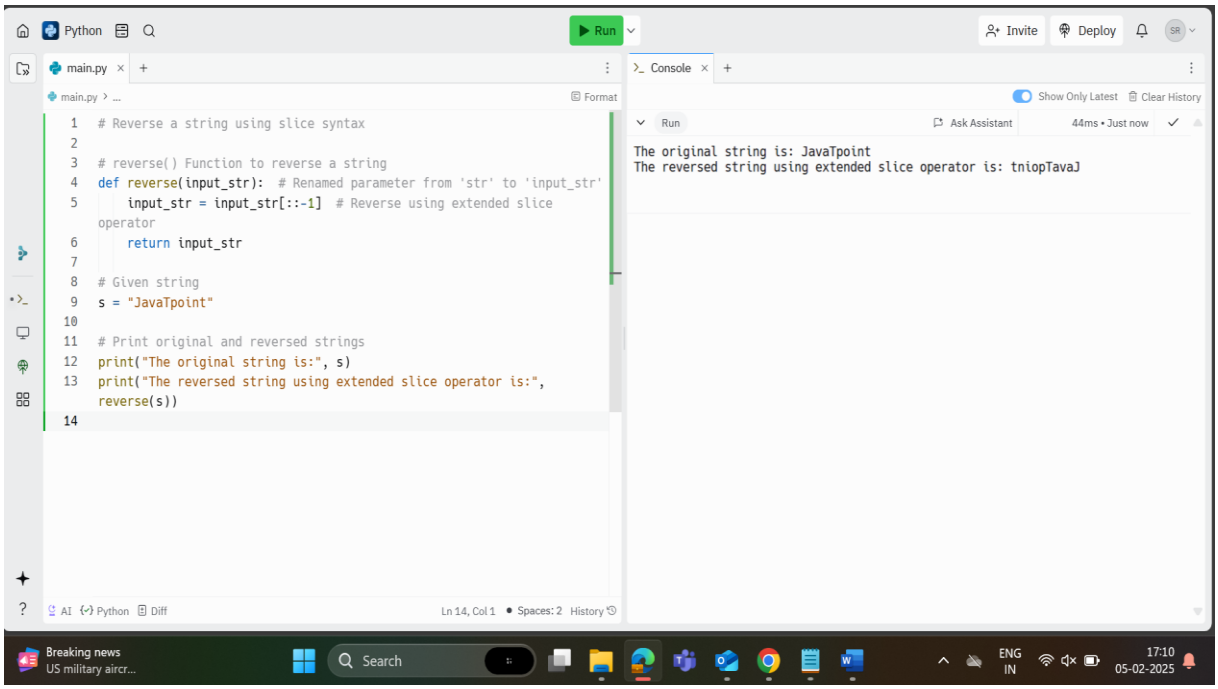
Using while loop

We can also reverse a string using a while loop. Let's understand the following example.



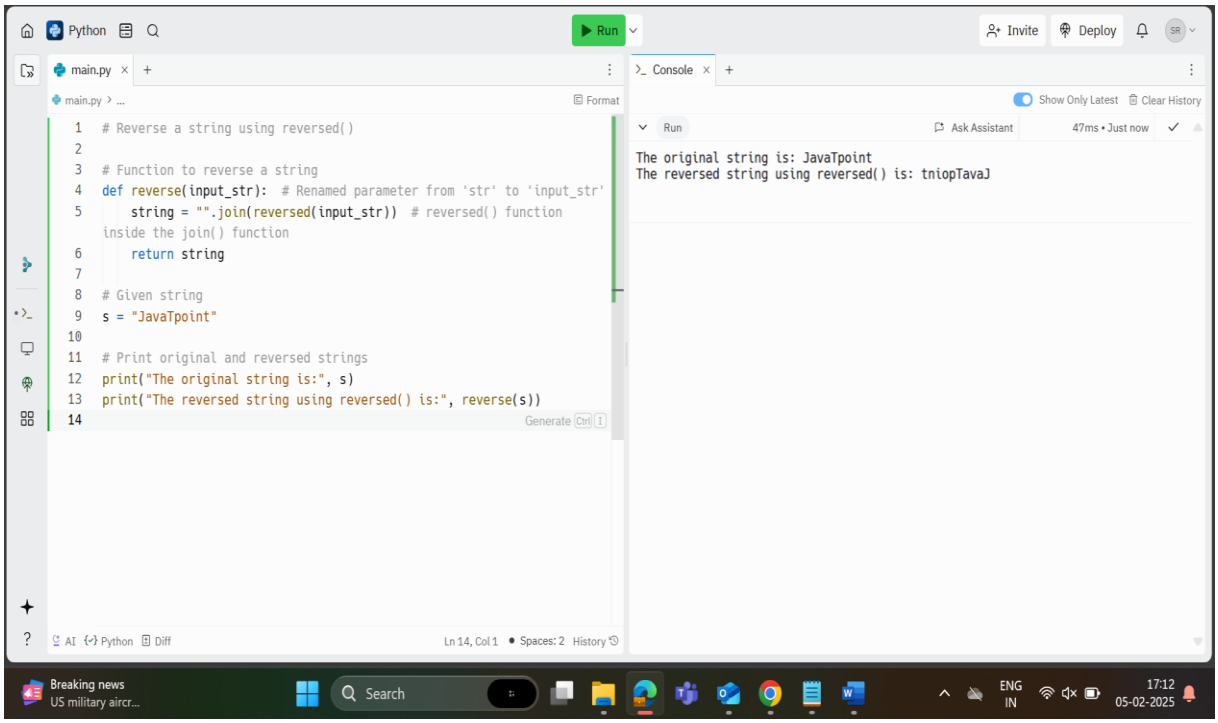
Using the slice ([]) operator

We can also reverse the given string using the extended slice operator. Let's understand the following example.



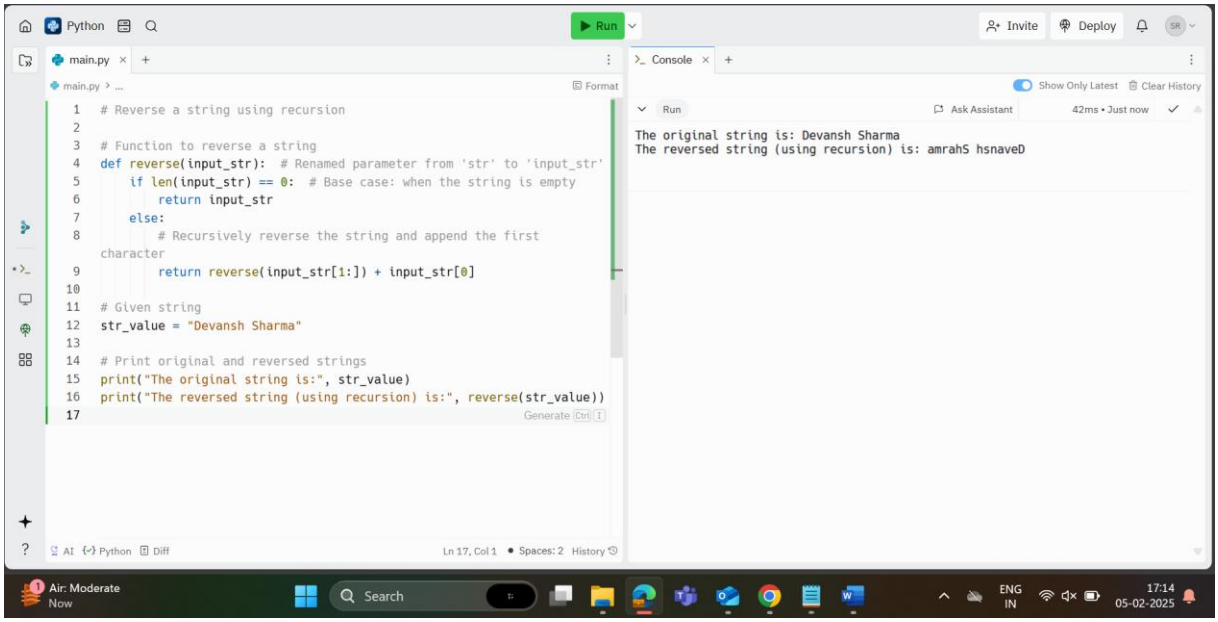
Using reverse function with join

Python provides the reversed() function to reverse the string. Let's understand the following example.



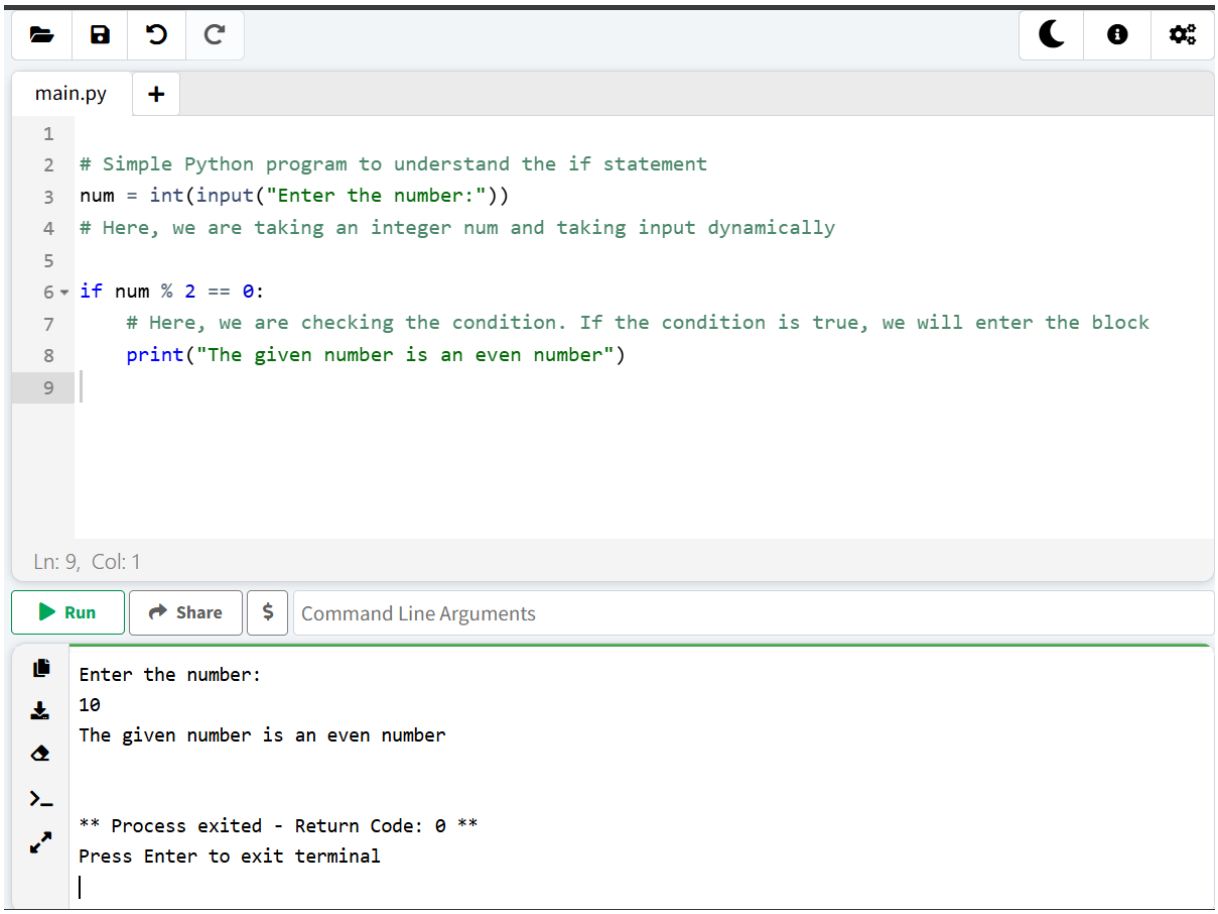
Using recursion()

The recursion can also be used to turn the string around. Recursion is a cycle where capability calls itself. Look at the following example.

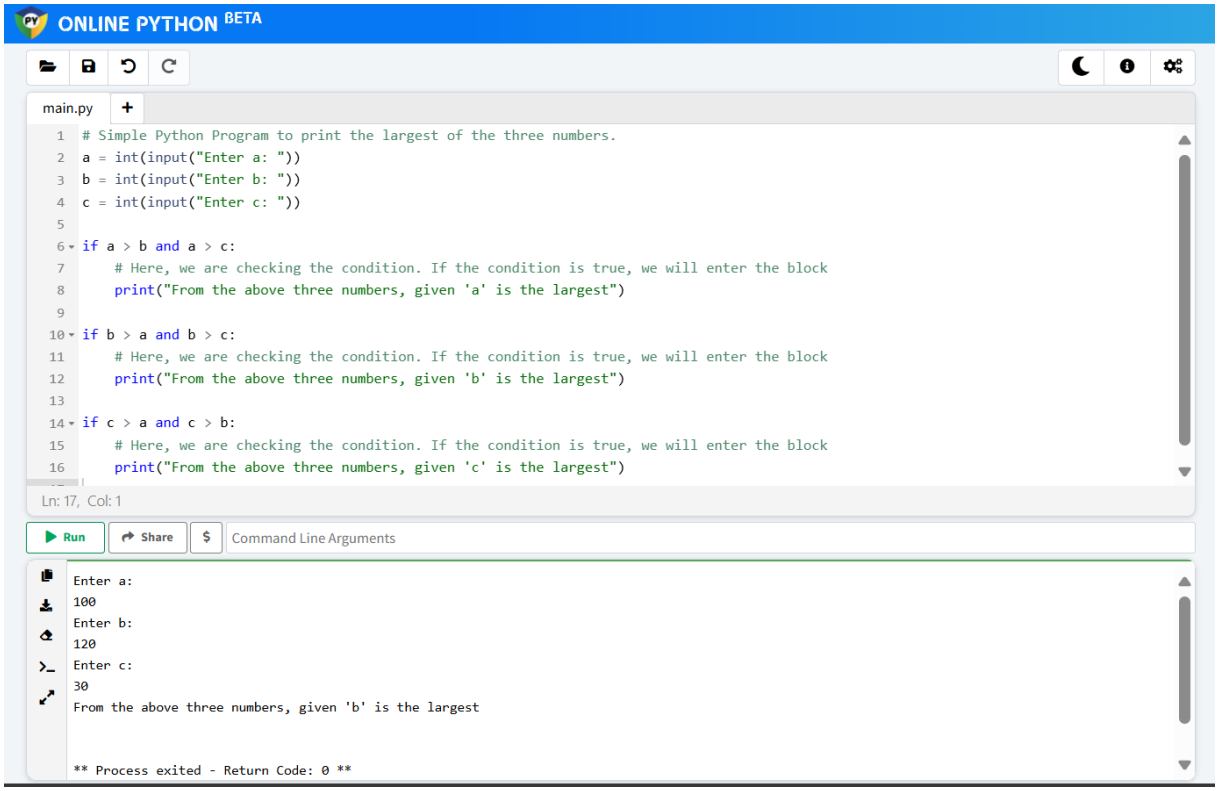


Python If-else statements

Decision making is the most important aspect of almost all the programming languages. As the name implies, decision making allows us to run a particular block of code for a particular decision. Here, the decisions are made on the validity of the conditions. Condition checking is the backbone of decision making.



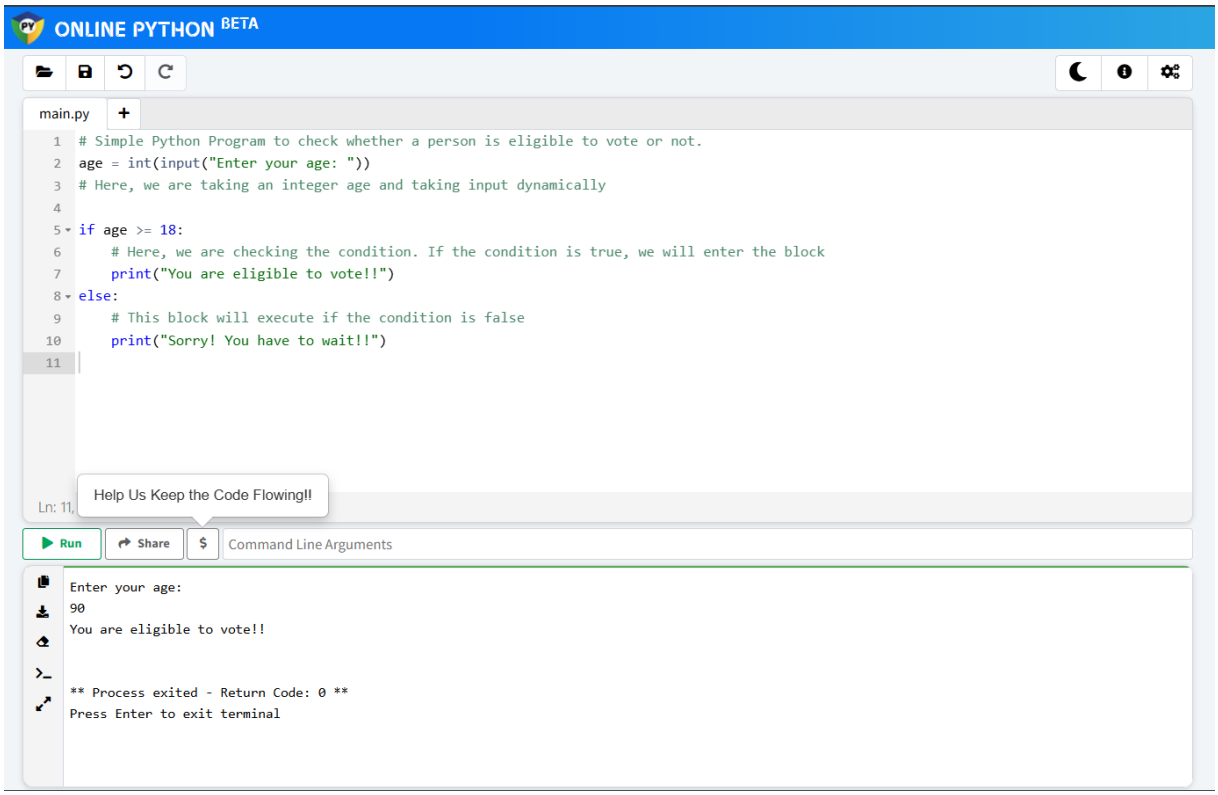
Program to print the largest of the three numbers.



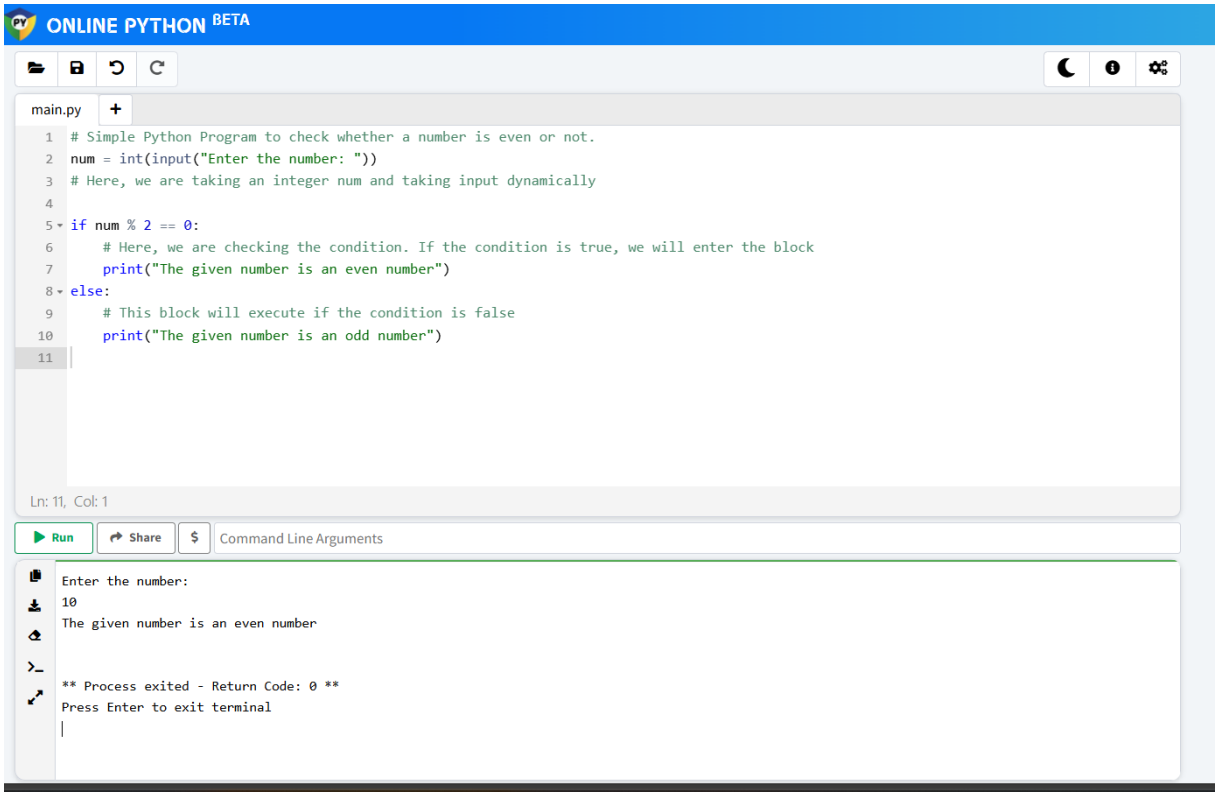
The if-else statement

The if-else statement provides an else block combined with the if statement which is executed in the false case of the condition.

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

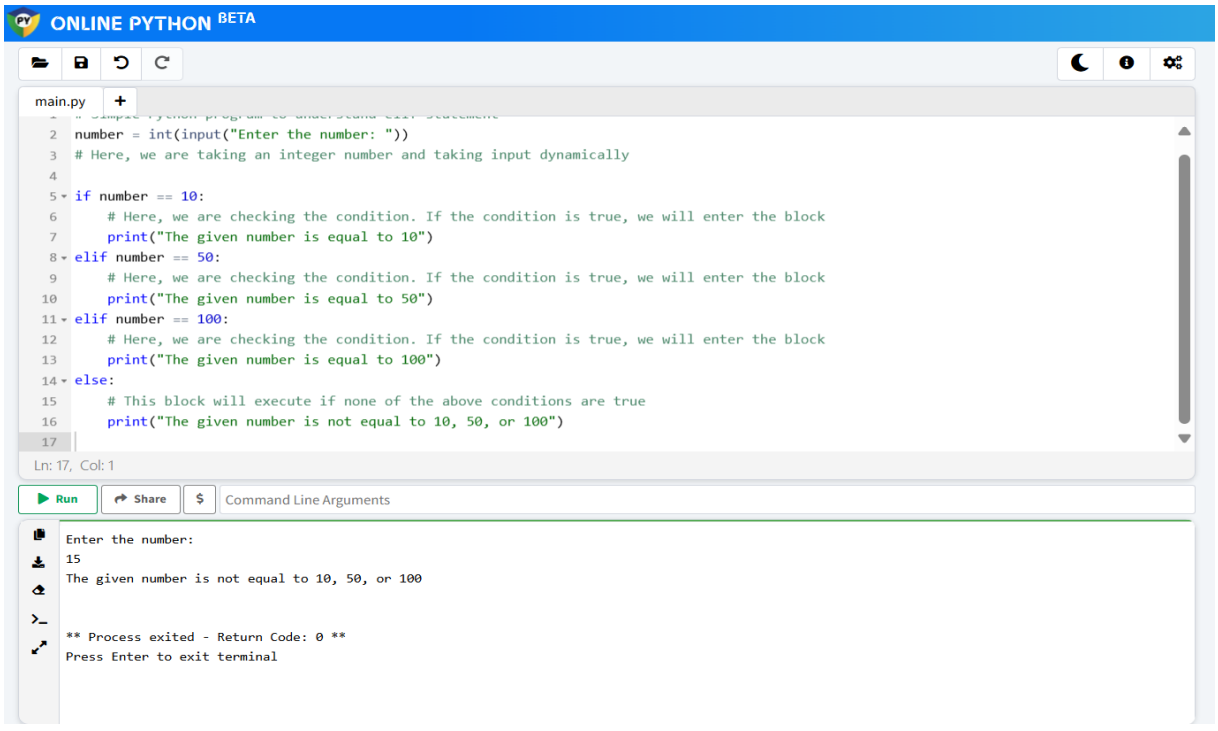


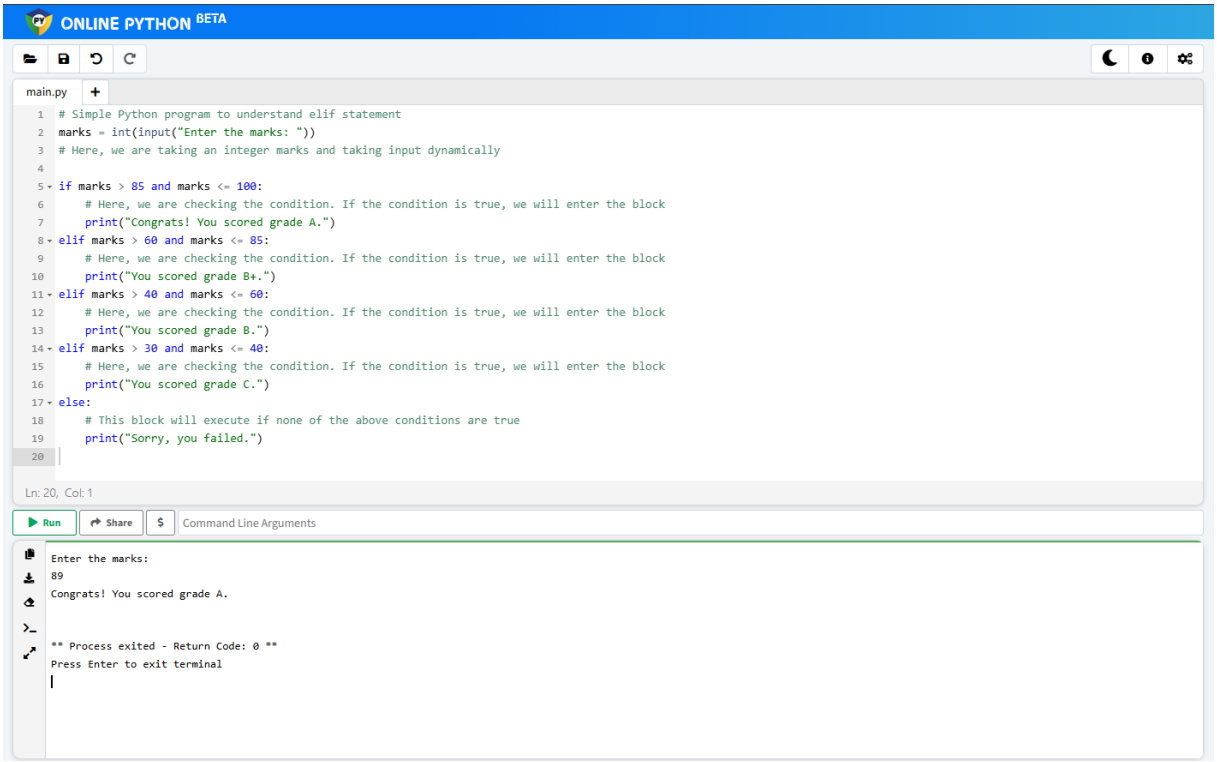
Program to check whether a number is even or not.



The elif statement

The elif statement enables us to check multiple conditions and execute the specific block of statements depending upon the true condition among them. We can have any number of elif statements in our program depending upon our need. However, using elif is optional.



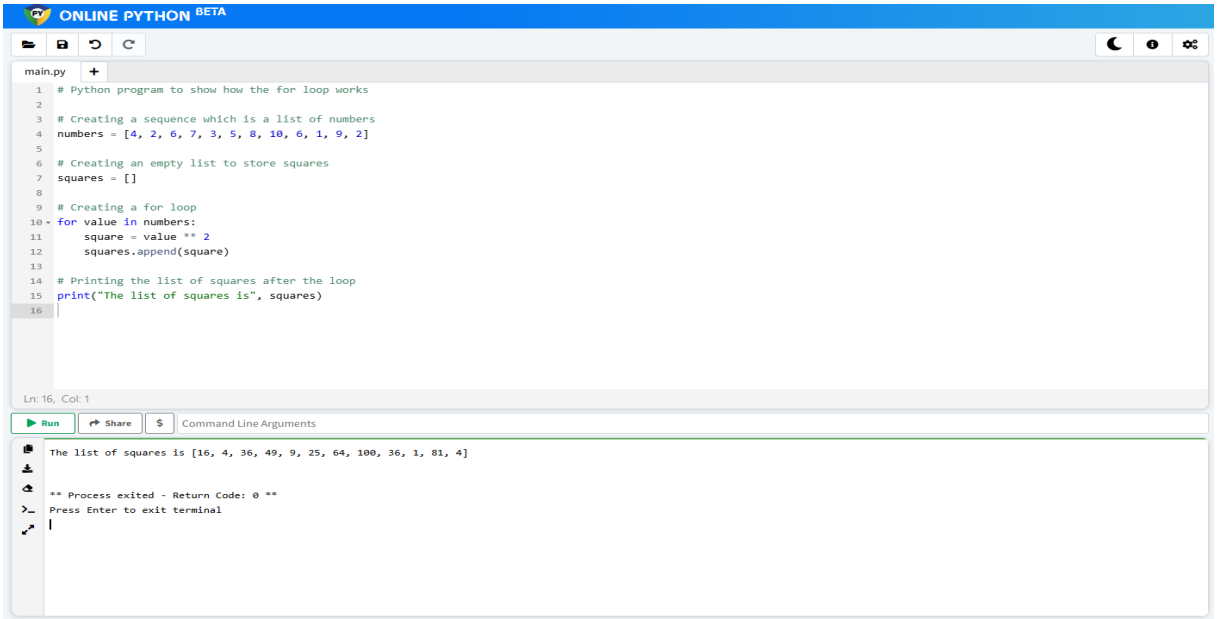


Python Loops

The following loops are available in Python to fulfil the looping needs. Python offers 3 choices for running the loops. The basic functionality of all the techniques is the same, although the syntax and the amount of time required for checking the condition differ.

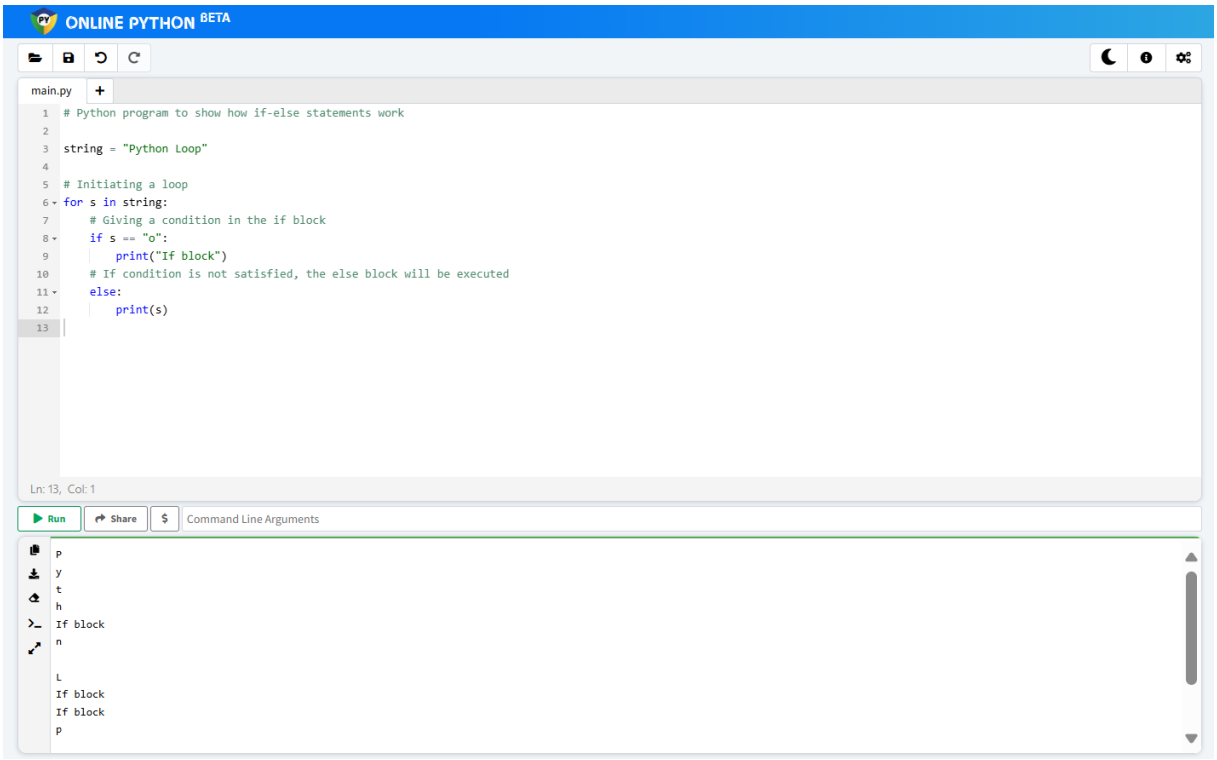
For Loop

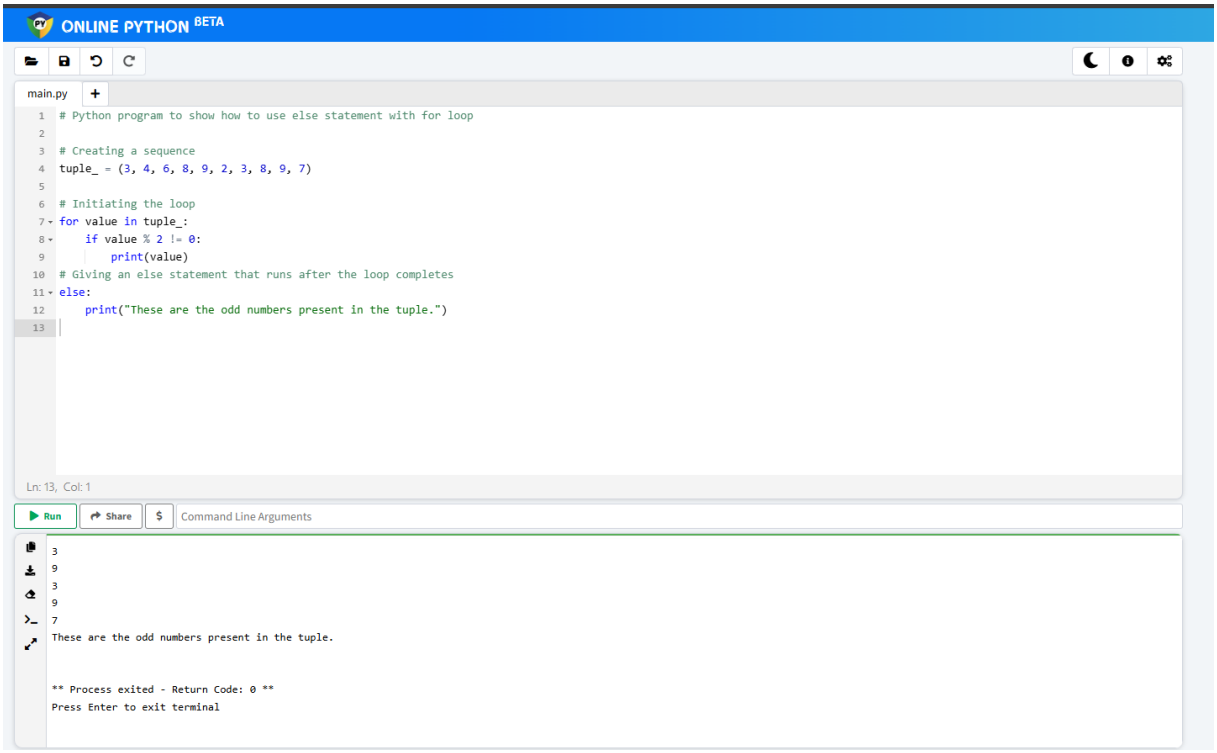
Python's for loop is designed to repeatedly execute a code block while iterating through a list, tuple, dictionary, or other iterable objects of Python. The process of traversing a sequence is known as iteration.



Using else Statement with for Loop

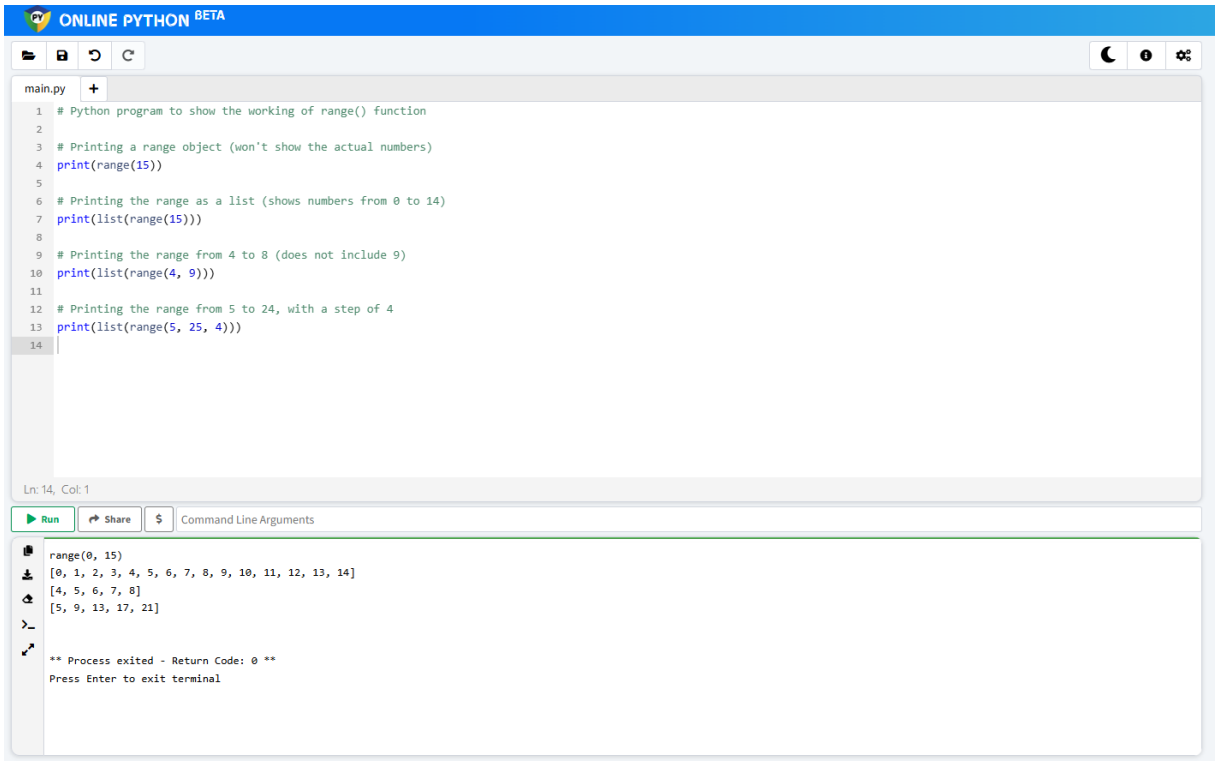
As already said, a for loop executes the code block until the sequence element is reached. The statement is written right after the for loop is executed after the execution of the for loop is complete.

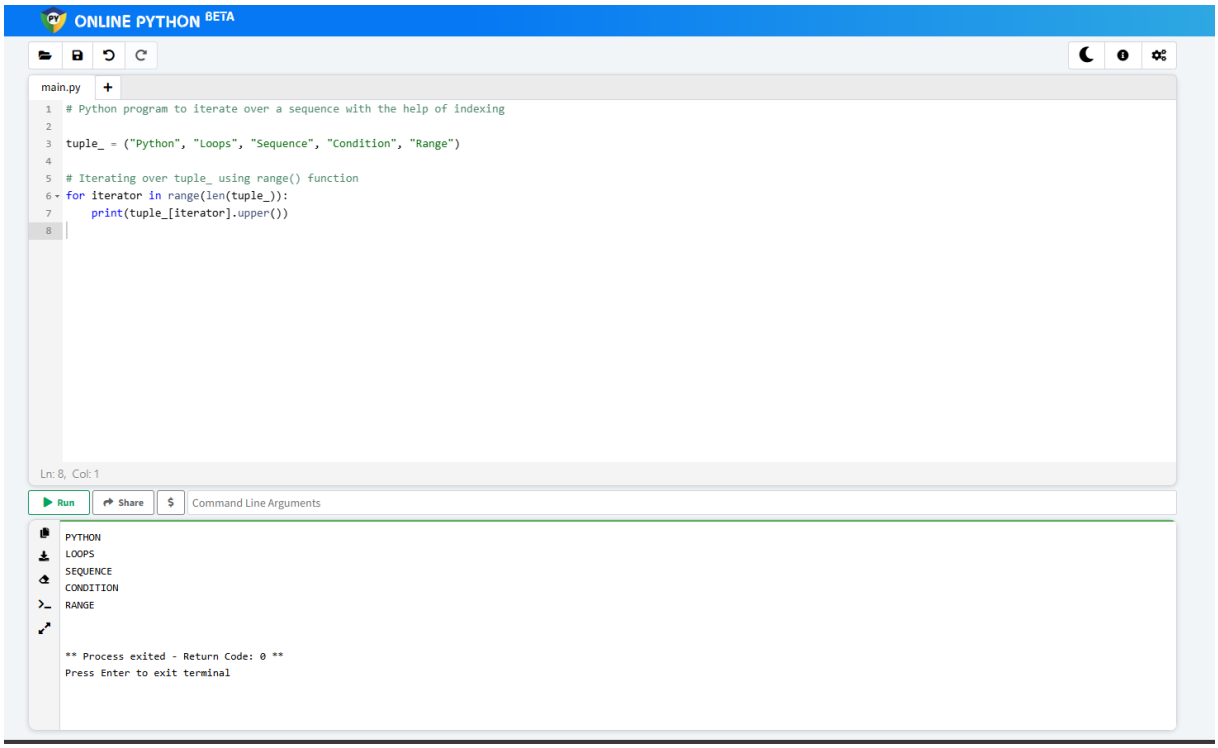




The range() Function

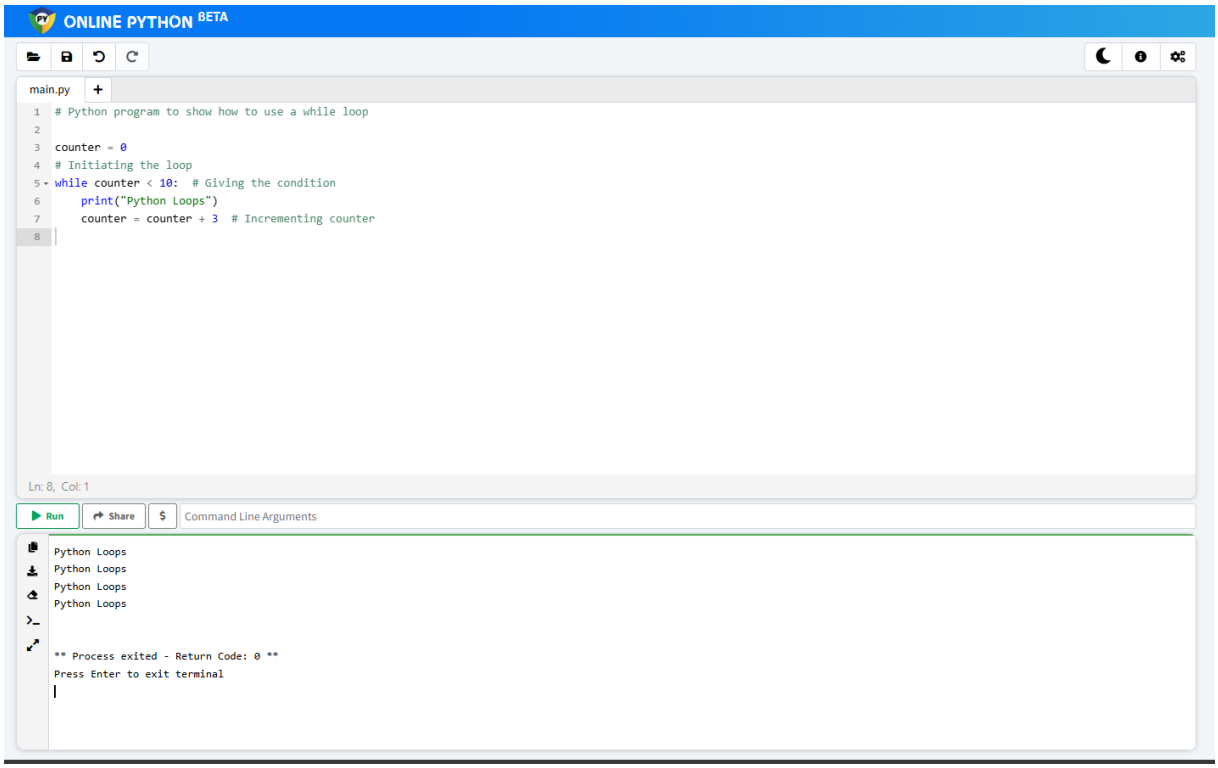
With the help of the range() function, we may produce a series of numbers. range(10) will produce values between 0 and 9. (10 numbers).





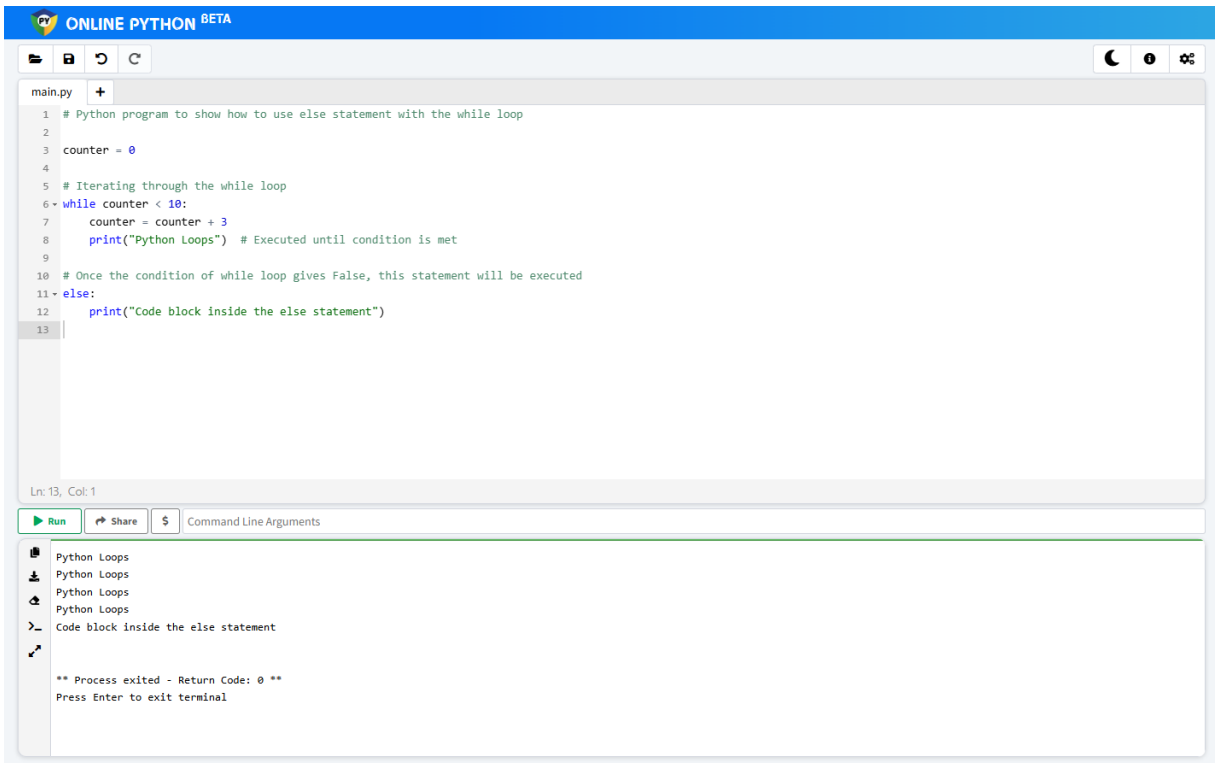
While Loop

While loops are used in Python to iterate until a specified condition is met. However, the statement in the program that follows the while loop is executed once the condition changes to false.



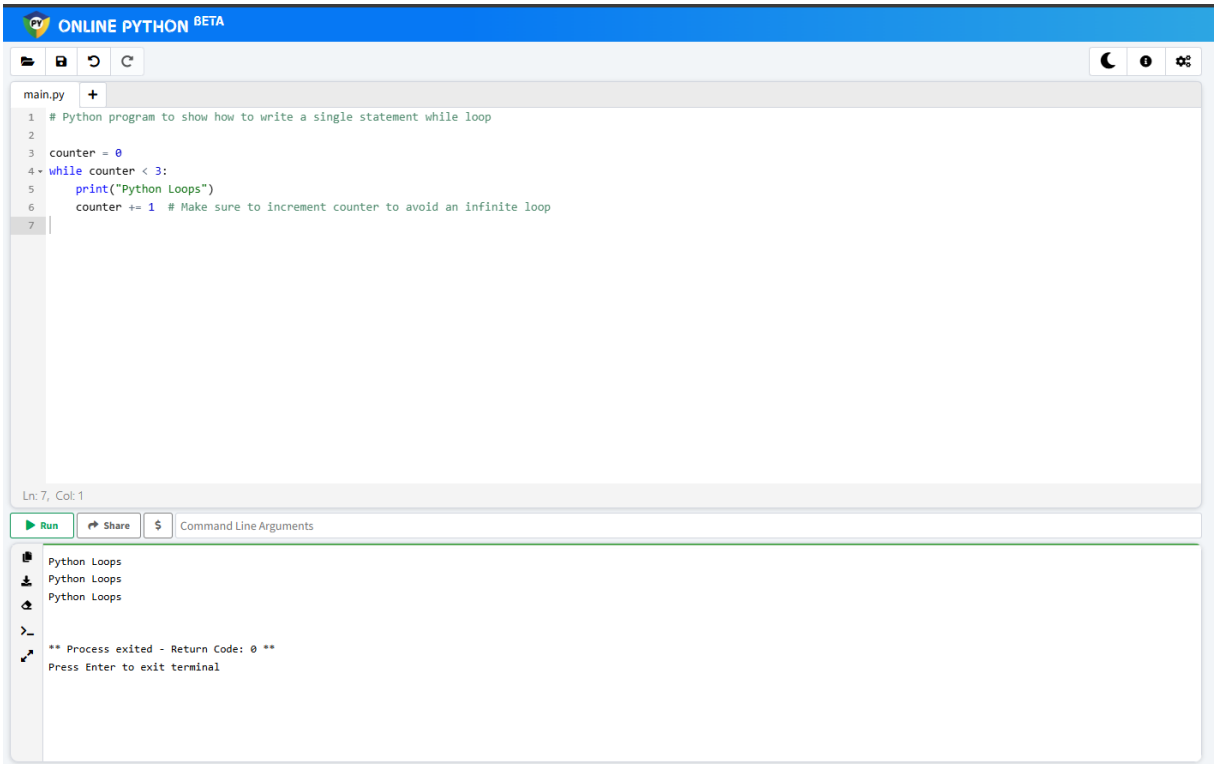
Using else Statement with while Loops

As discussed earlier in the for loop section, we can use the else statement with the while loop also. It has the same syntax.



Single statement while Block

The loop can be declared in a single statement, as seen below. This is similar to the if-else block, where we can write the code block in a single line.

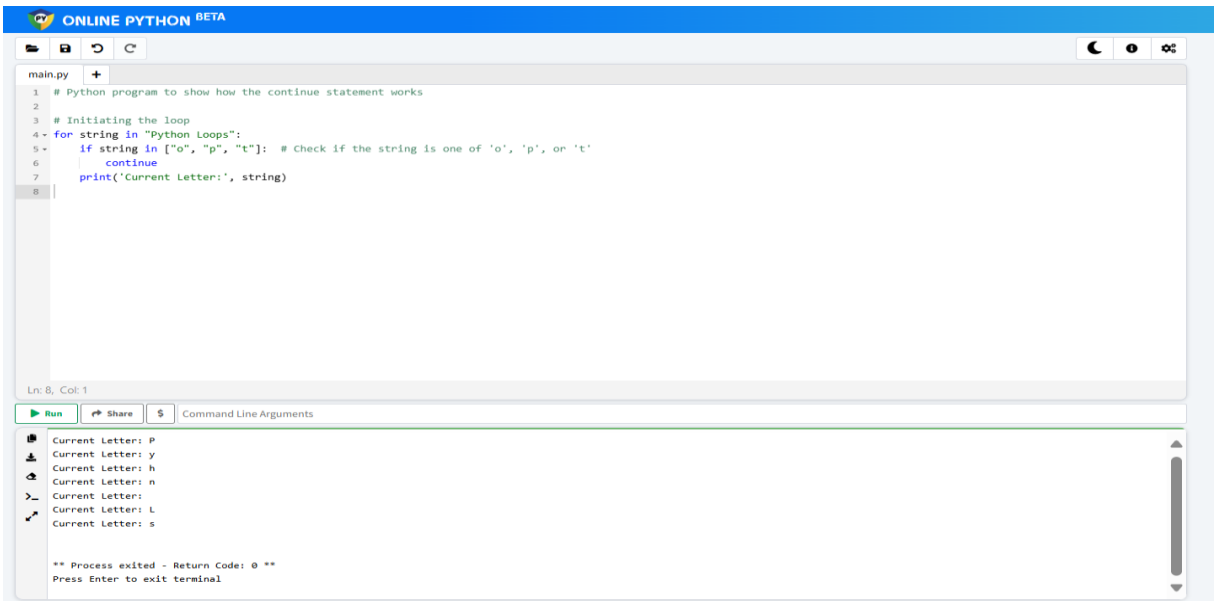


Loop Control Statements

Now we will discuss the loop control statements in detail. We will see an example of each control statement.

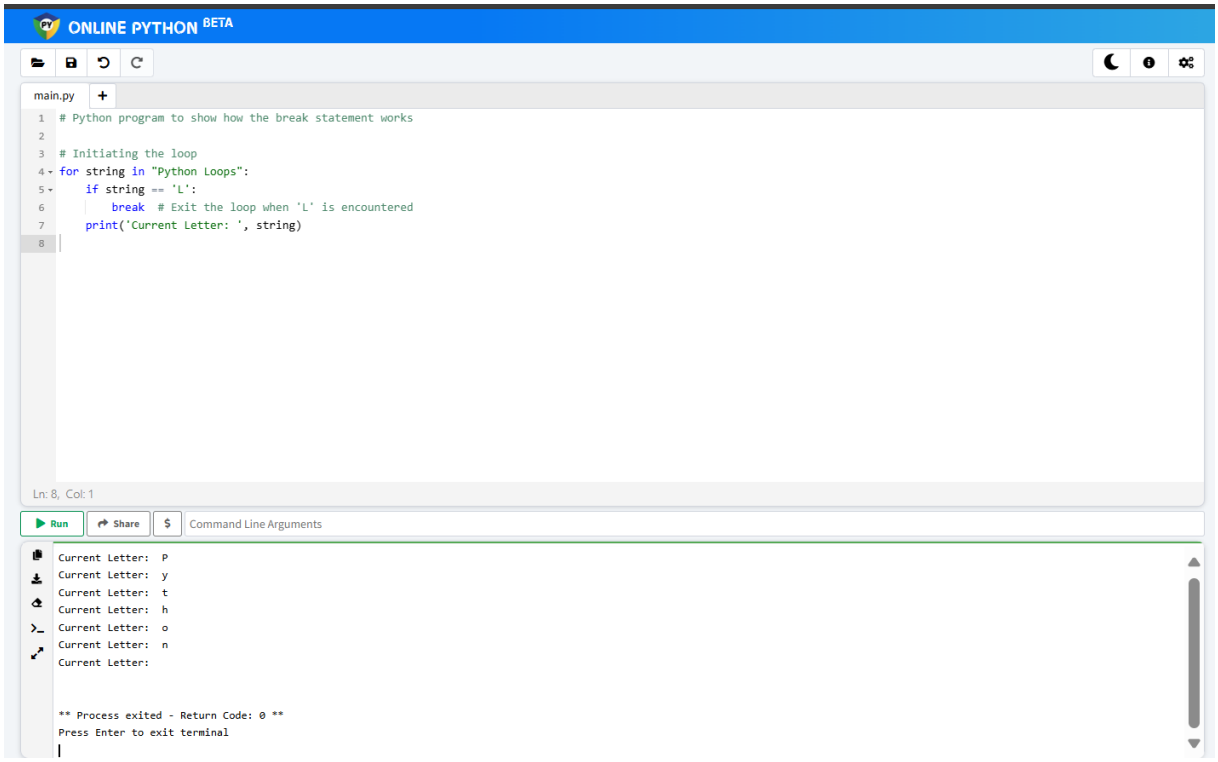
Continue Statement

It returns the control to the beginning of the loop



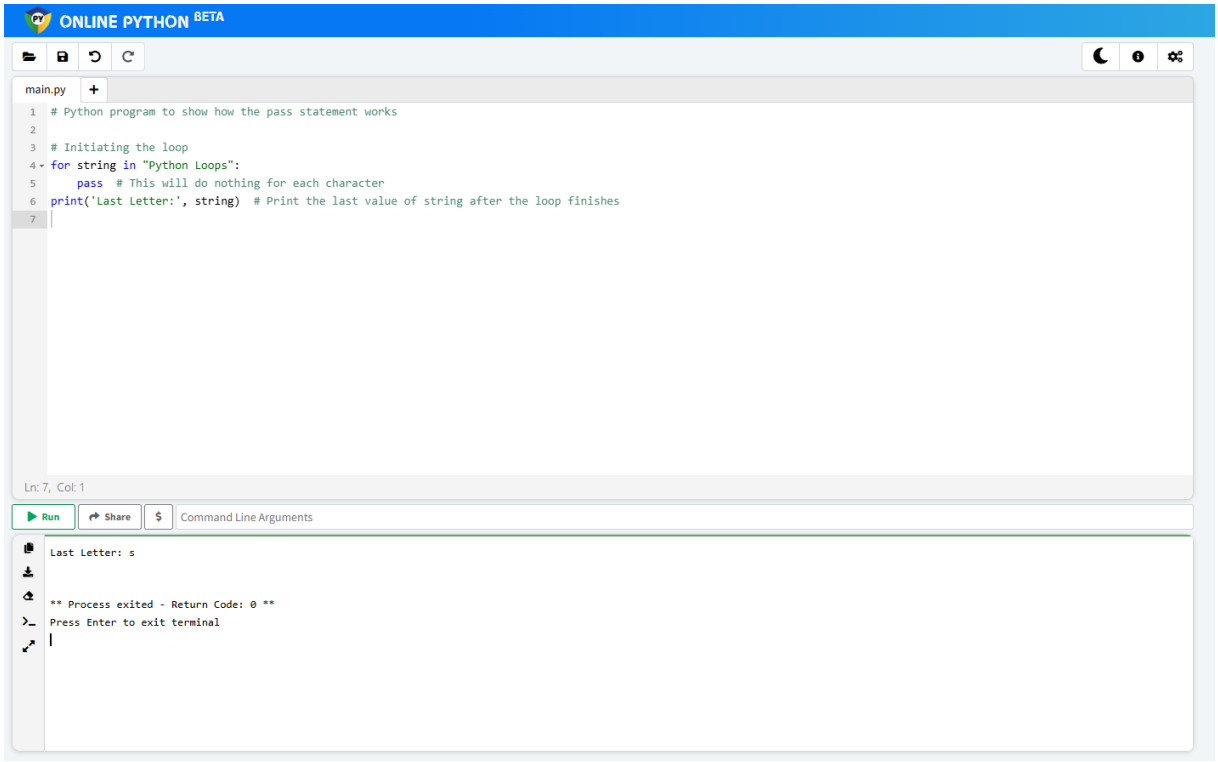
Break Statement

It stops the execution of the loop when the break statement is reached.

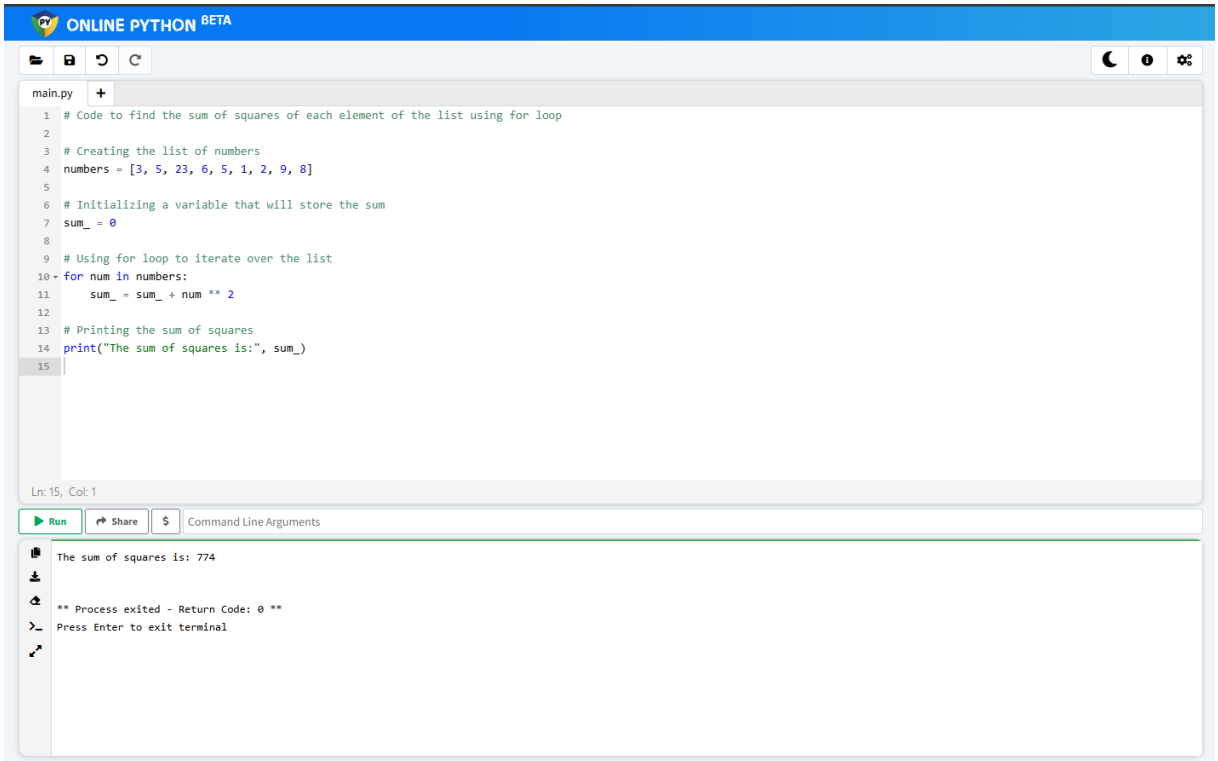


Pass Statement

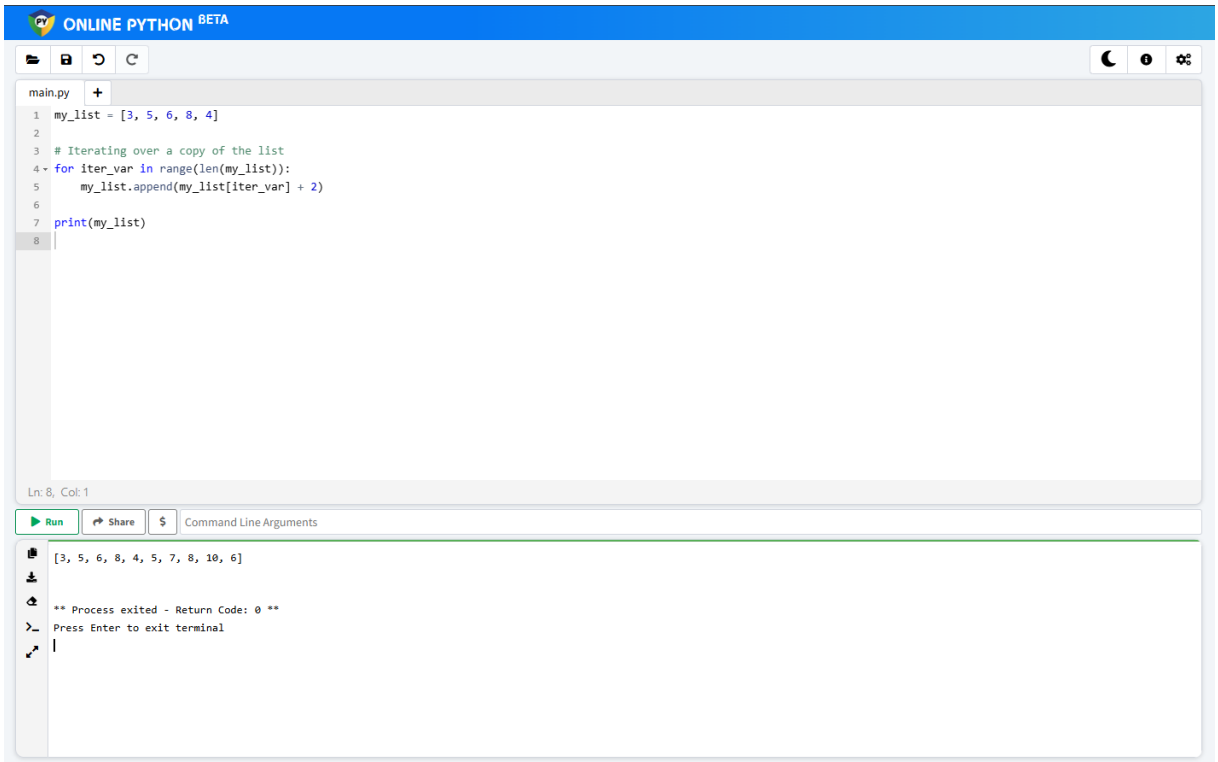
Pass statements are used to create empty loops. Pass statement is also employed for classes, functions, and empty control statements.



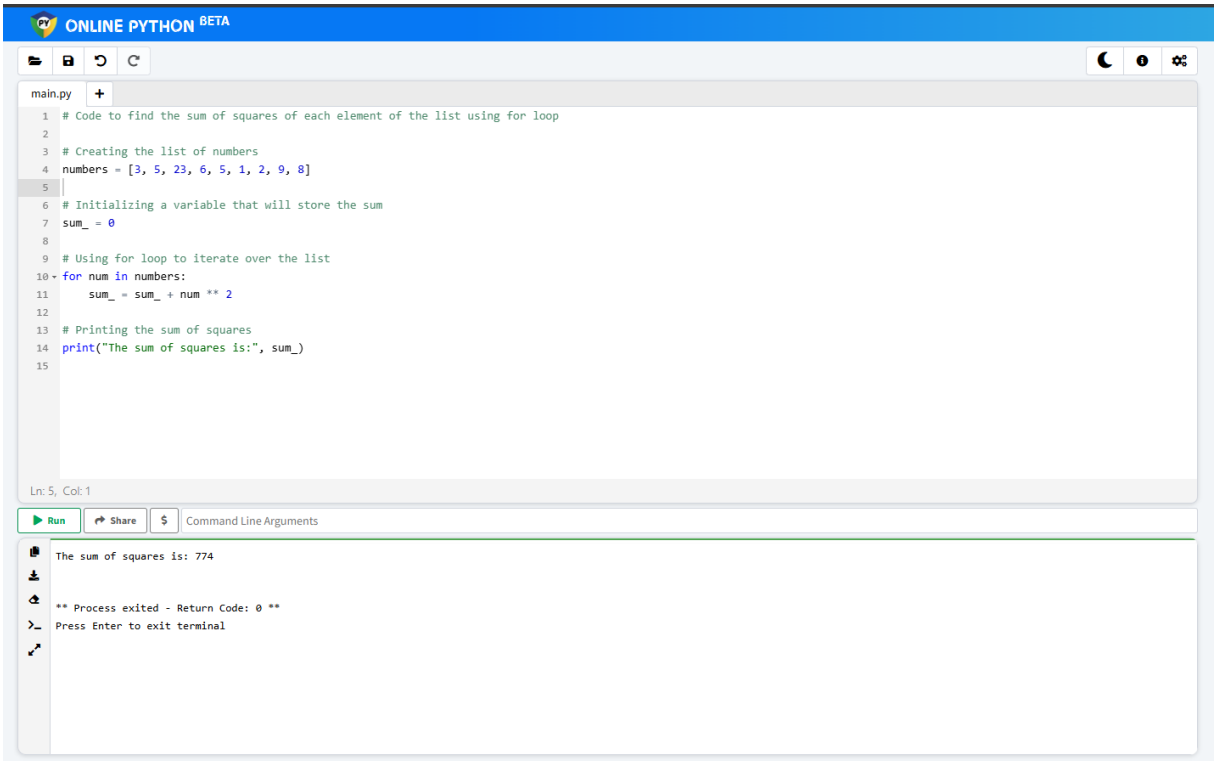
Example of Python for Loop



The range() Function

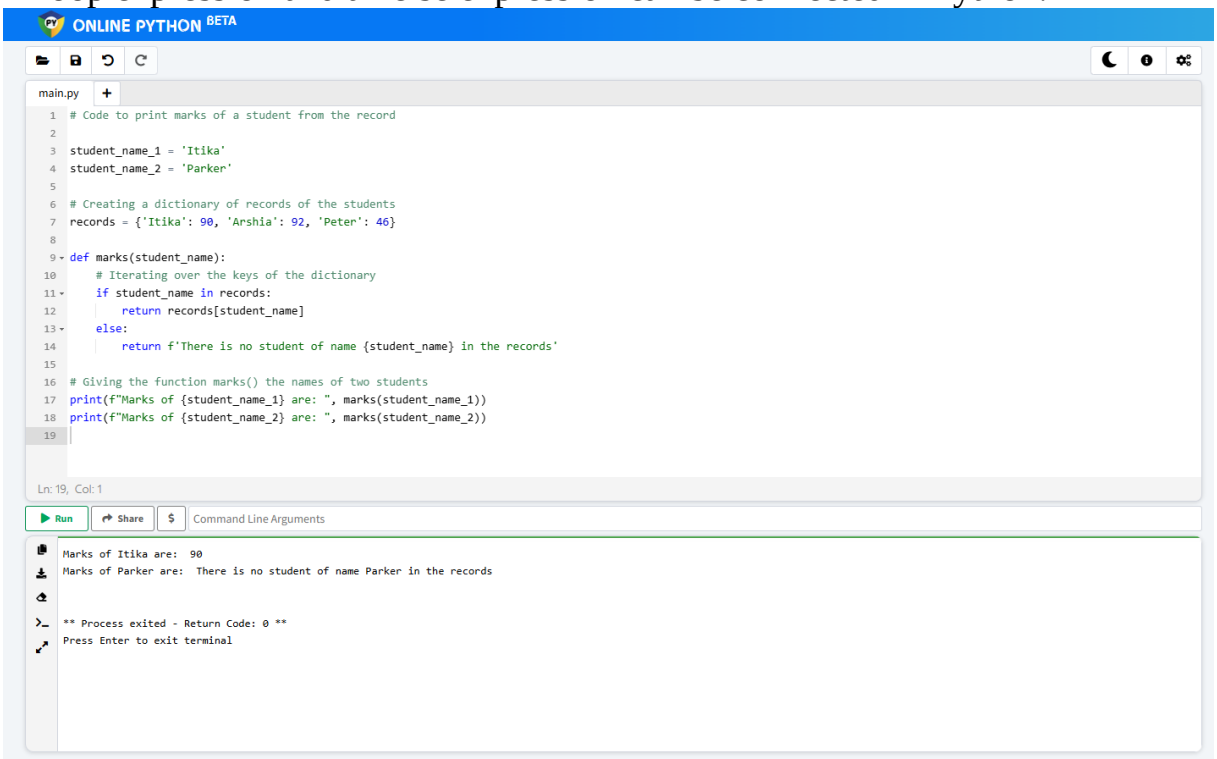


Iterating by Using Index of Sequence



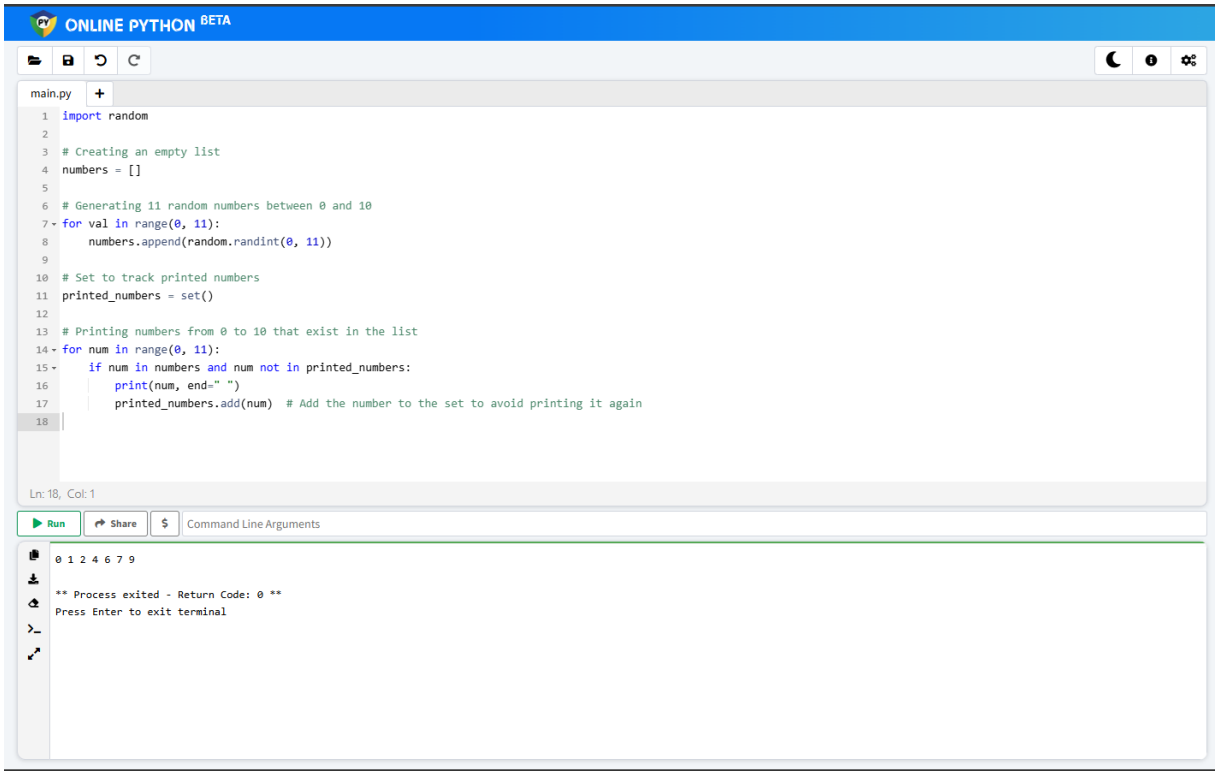
Using else Statement with for Loop

A loop expression and an else expression can be connected in Python.

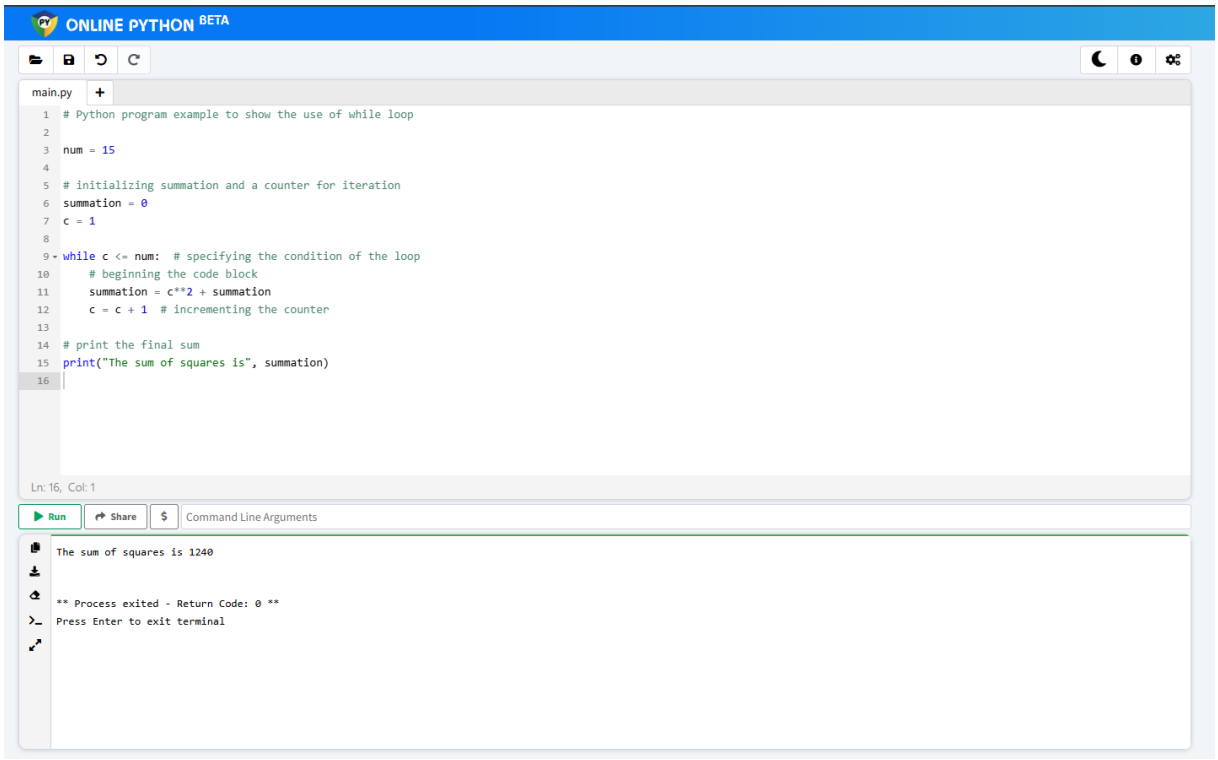


Nested Loops

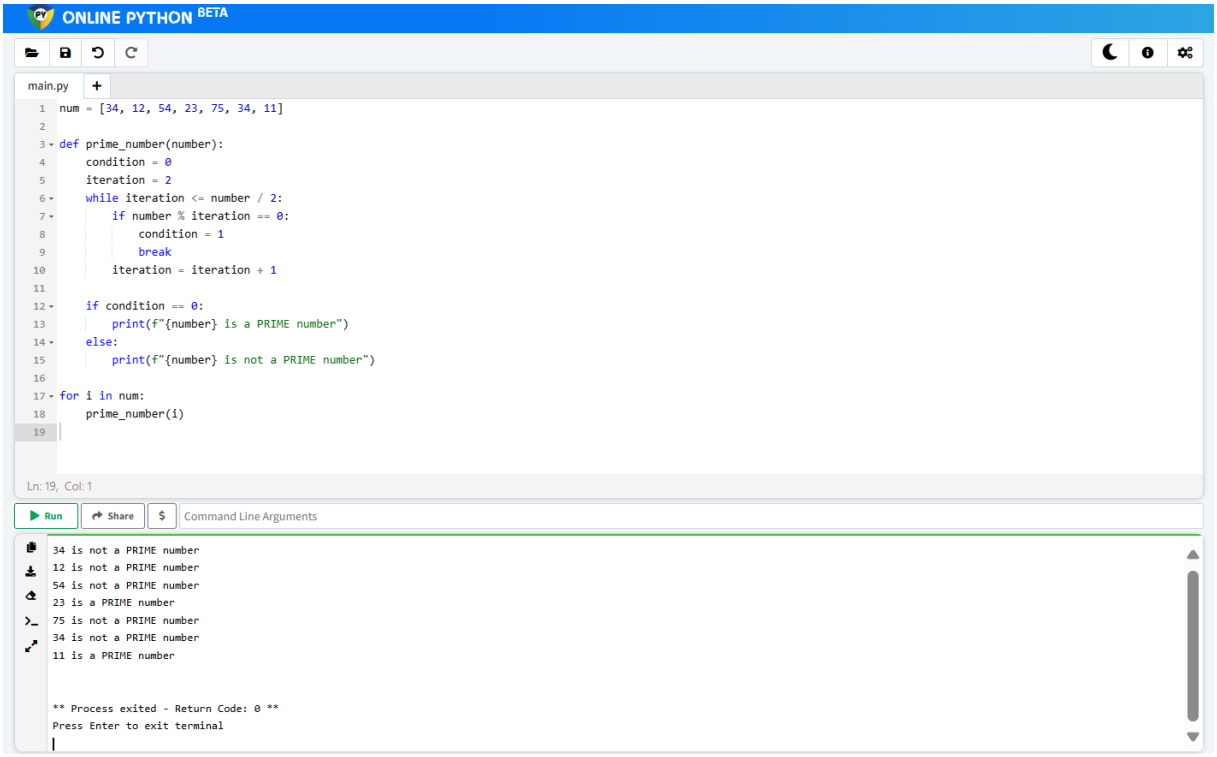
If we have a piece of content that we need to run various times and, afterward, one more piece of content inside that script that we need to run B several times, we utilize a "settled circle." While working with an iterable in the rundowns, Python broadly uses these.



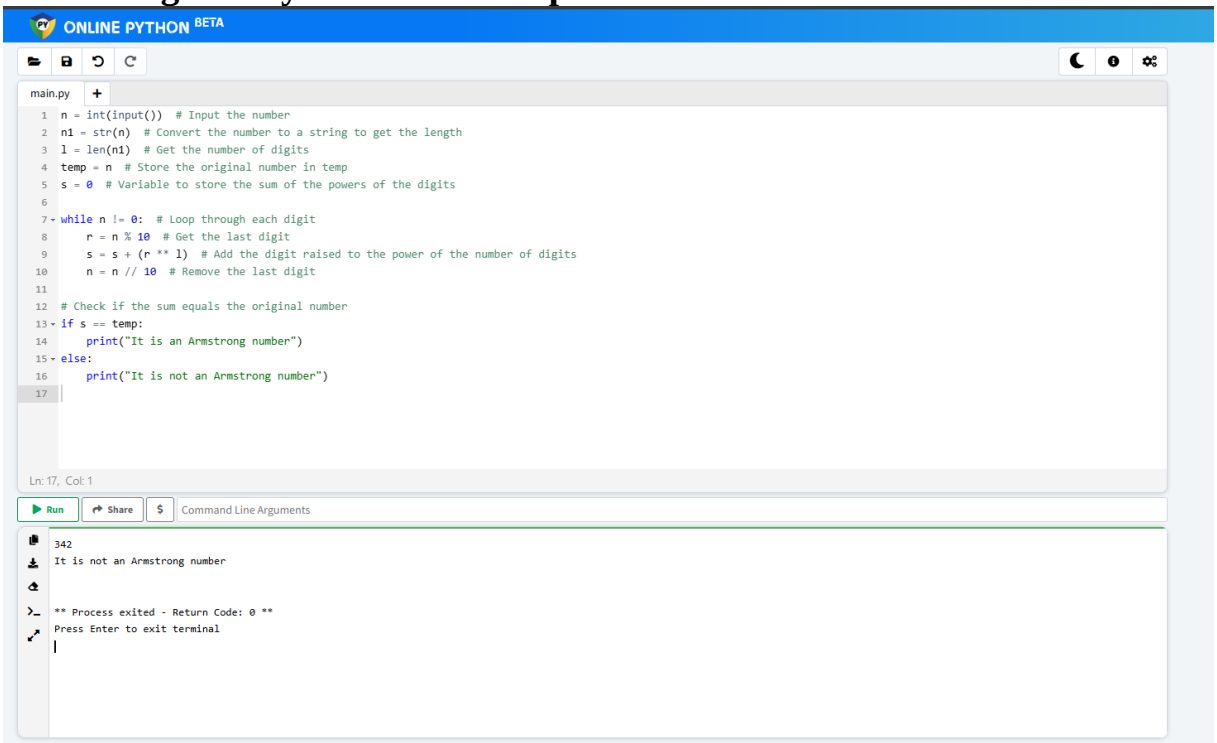
Python While Loops



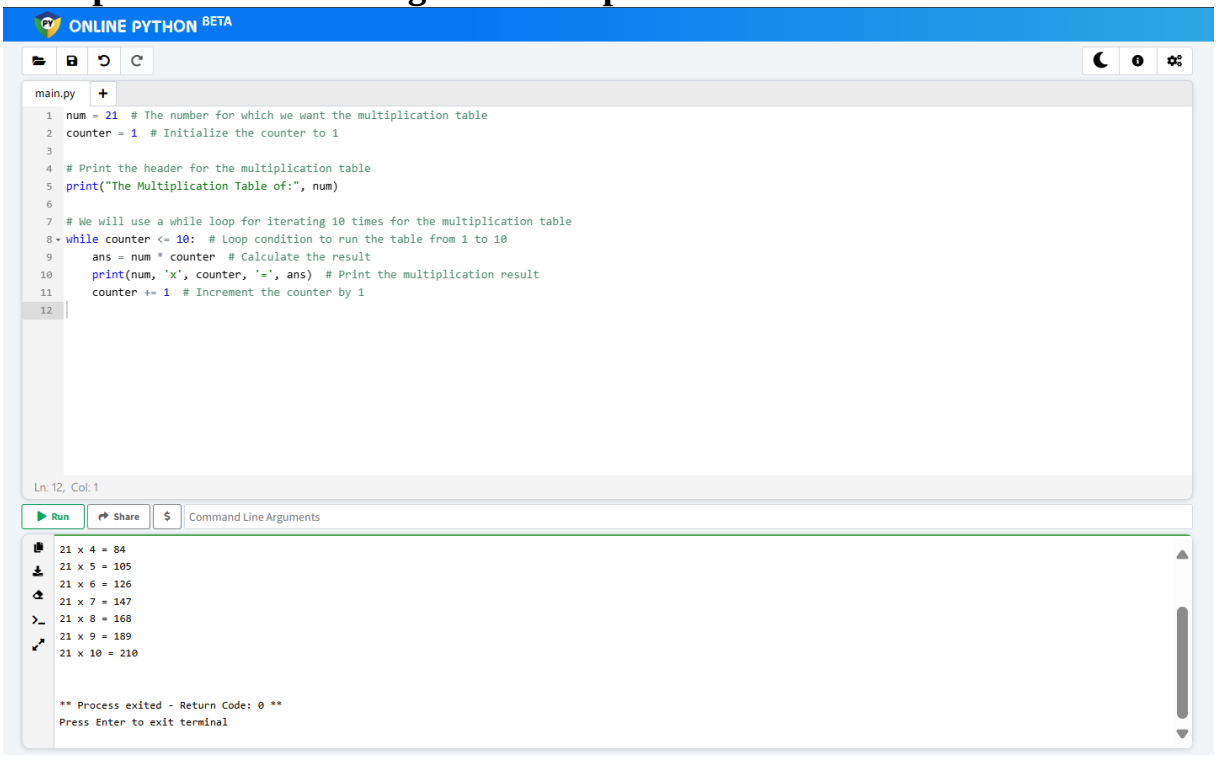
Prime Numbers and Python While Loop



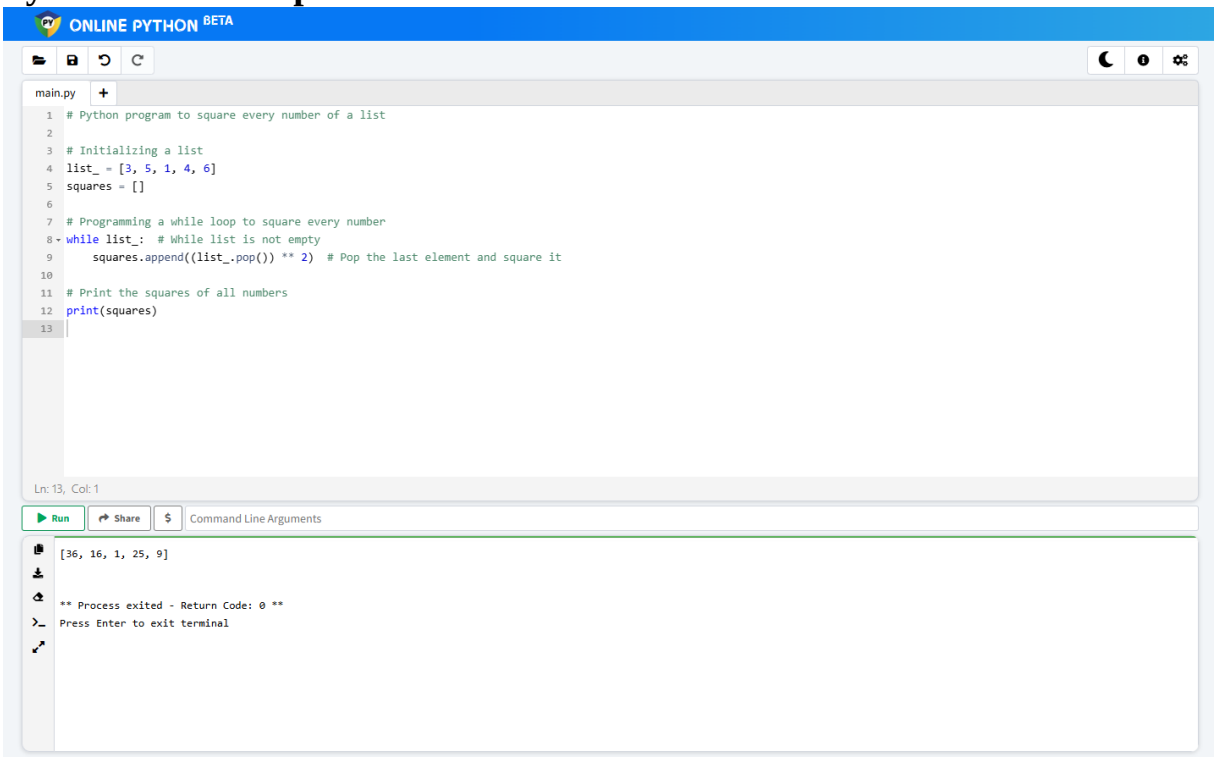
Armstrong and Python While Loop



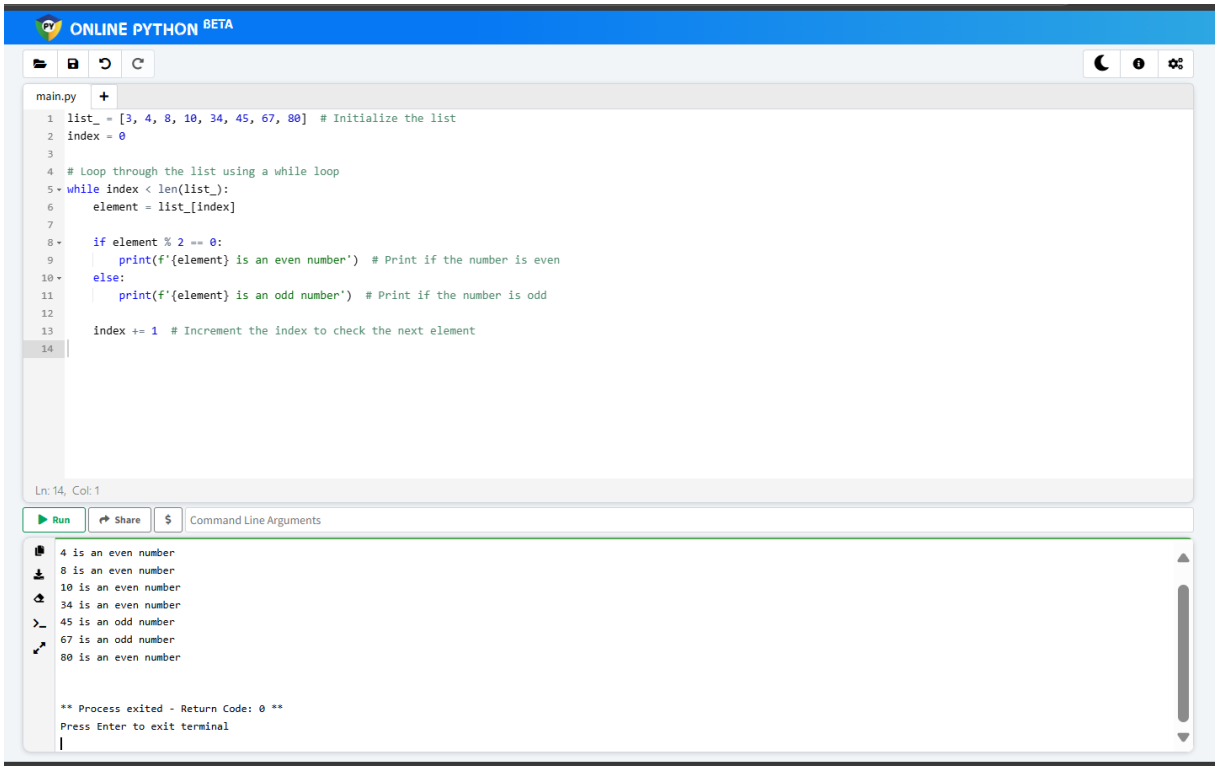
Multiplication Table using While Loop



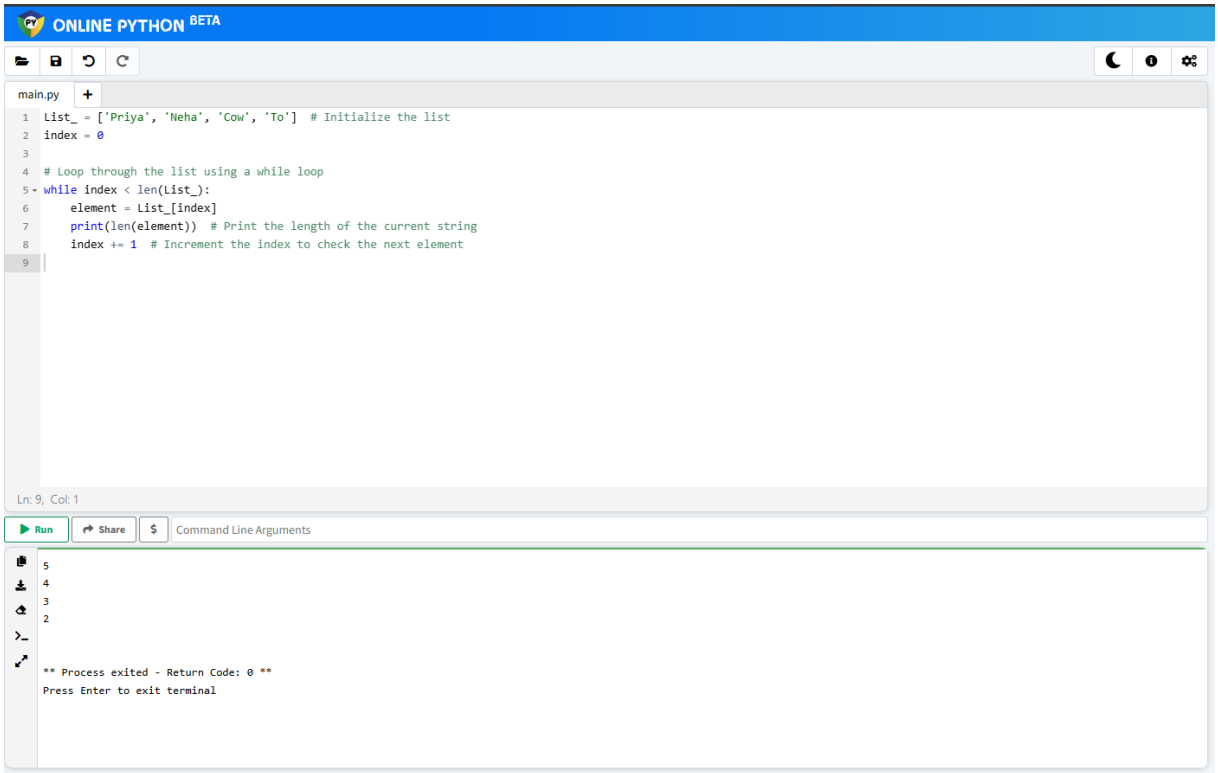
Python While Loop with List



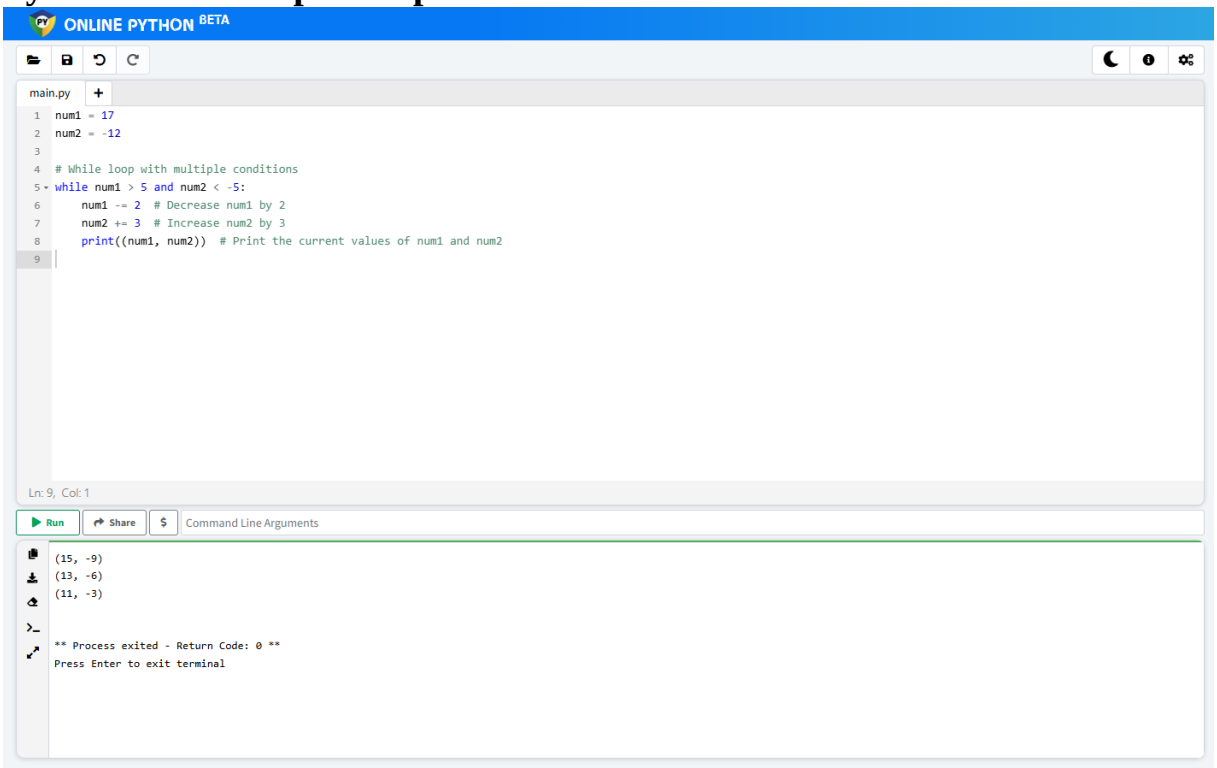
Now we give code examples of while loops in Python for determine odd and even number from every number of a list. The code is given below -

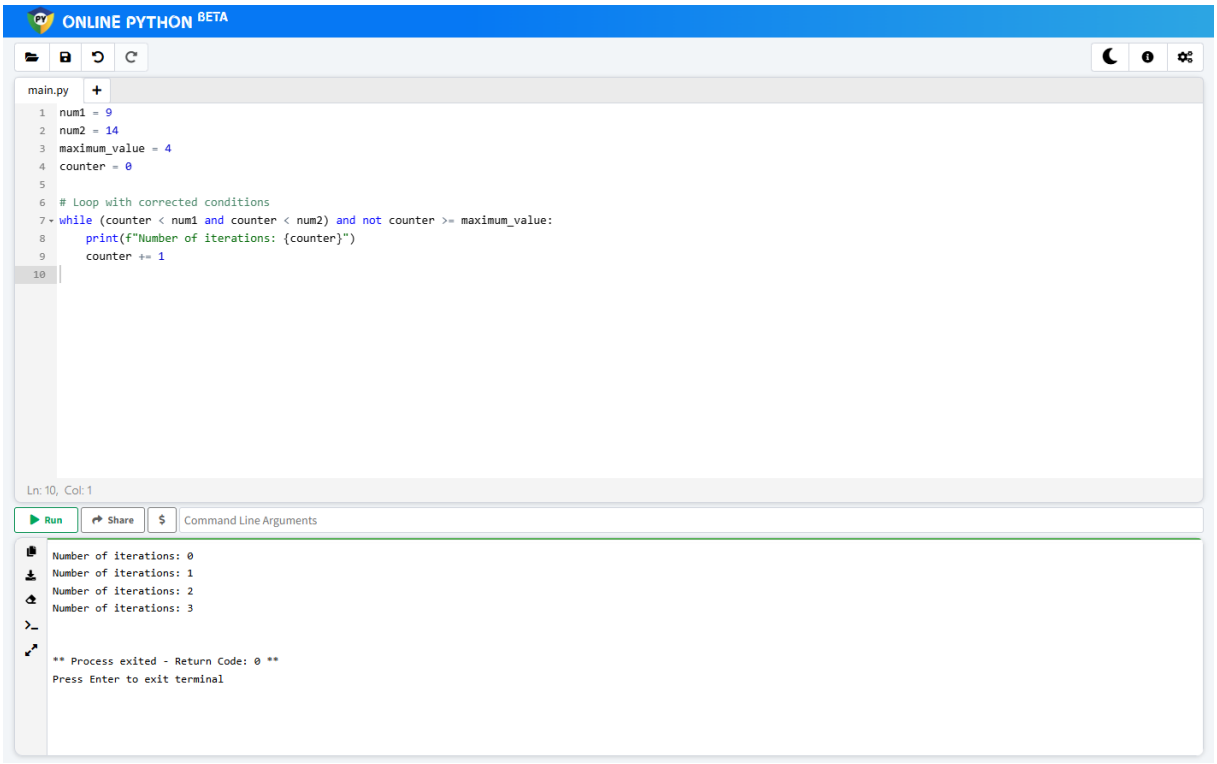


Now we give code examples of while loops in Python for determine the number letters of every word from the given list. The code is given below -

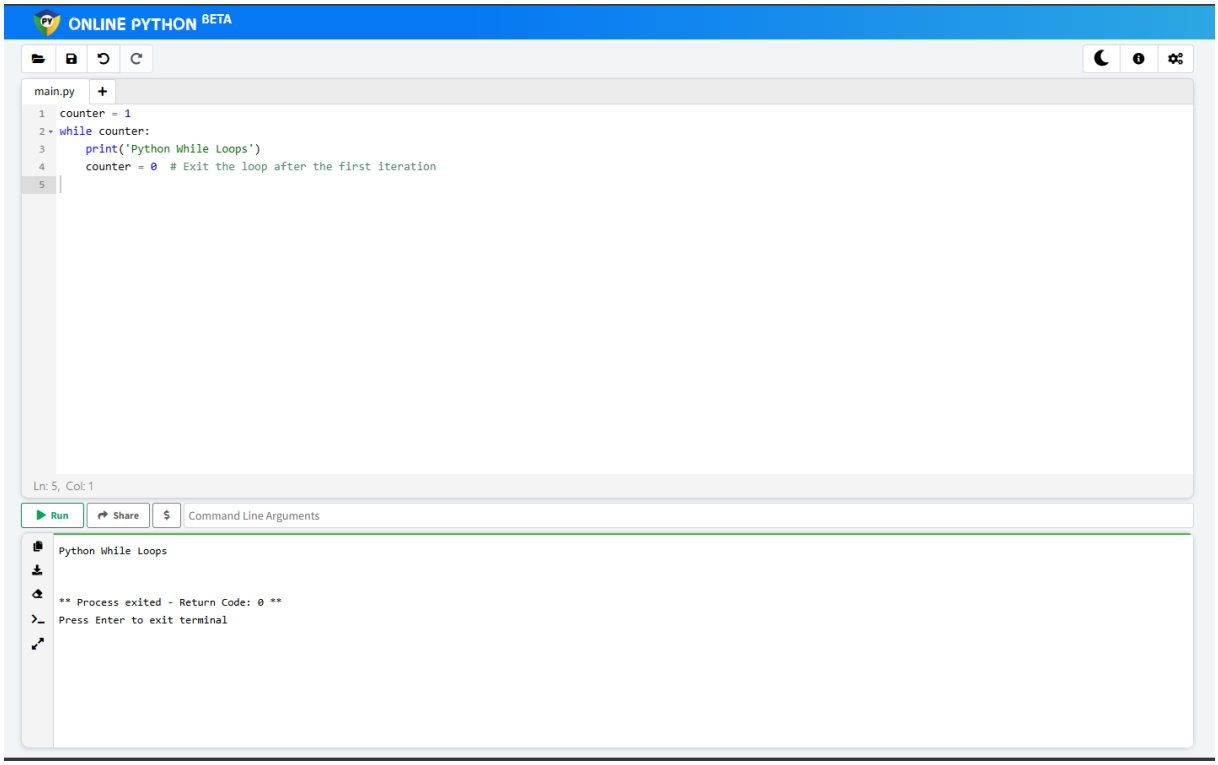


Python While Loop Multiple Conditions

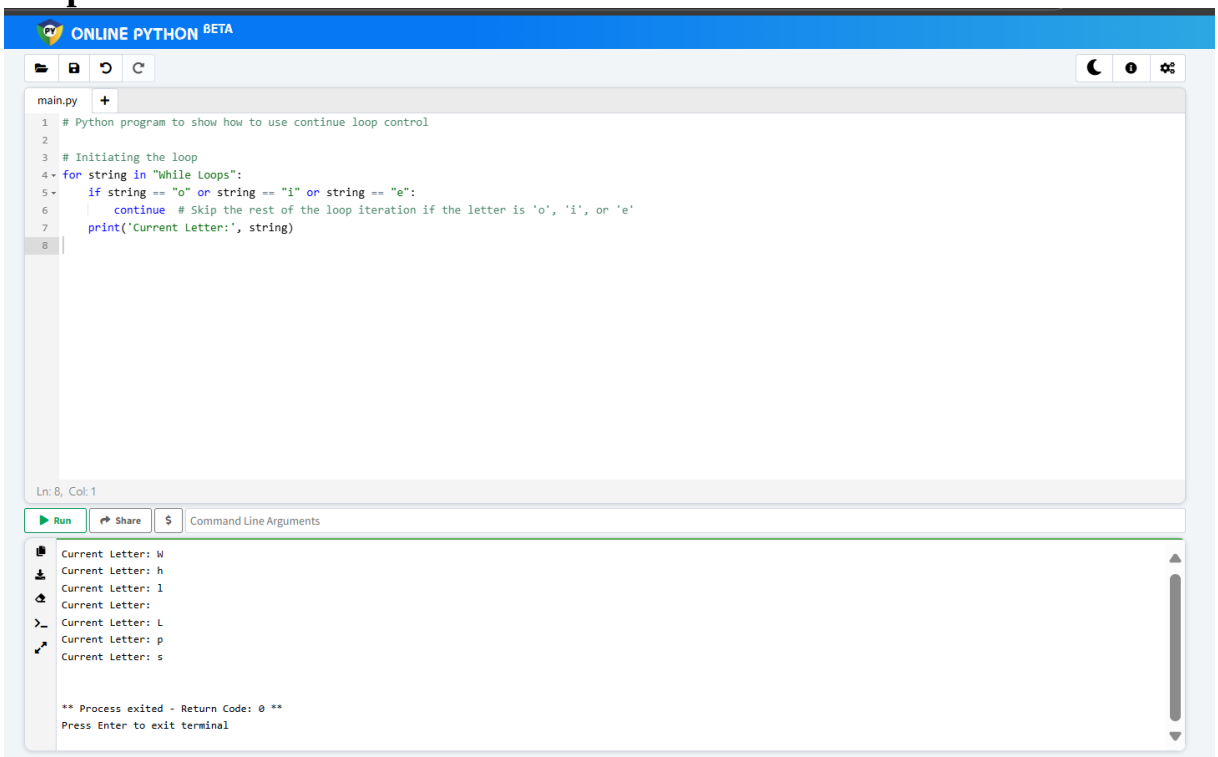




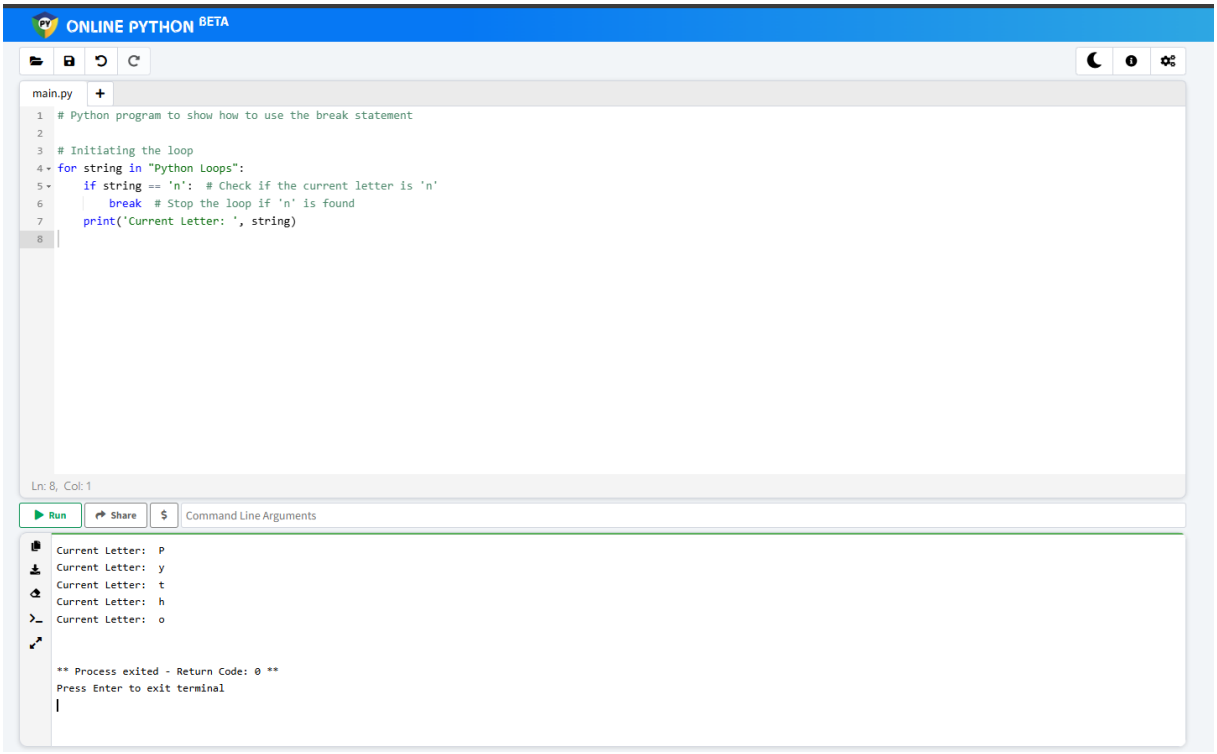
Single Statement While Loop



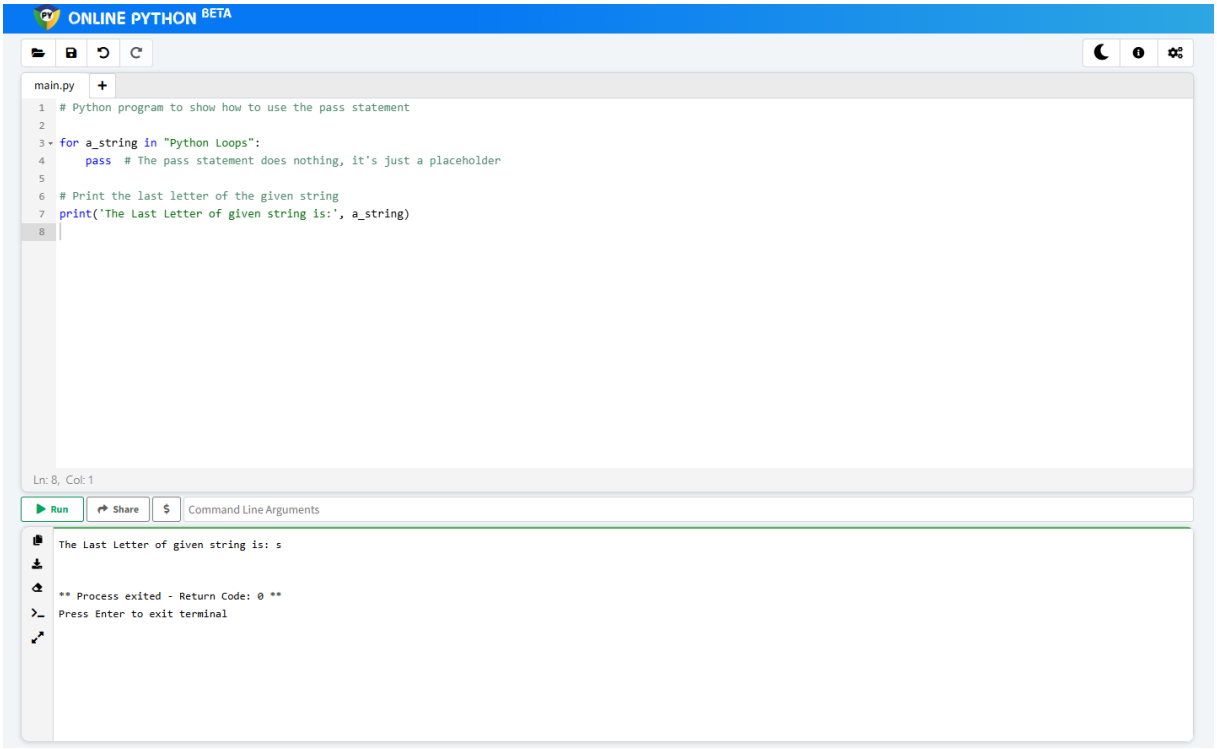
Loop Control Statements



Break Statement

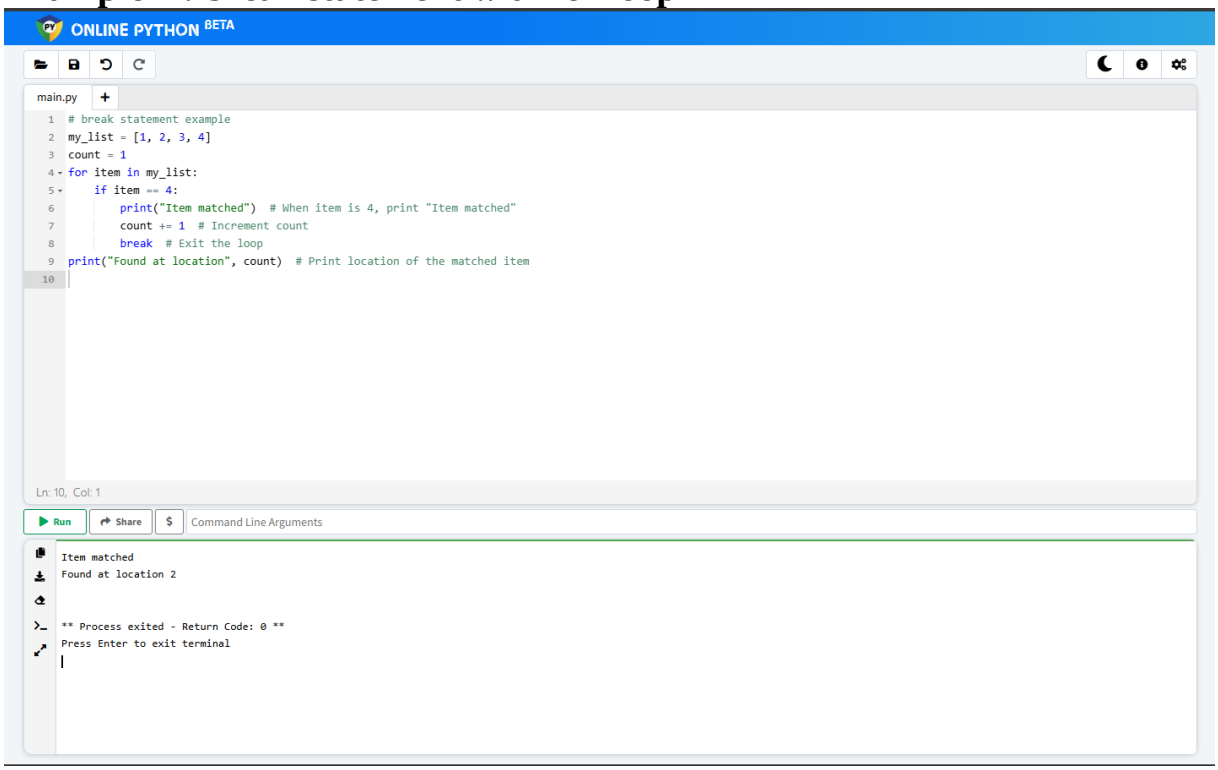


Pass Statement

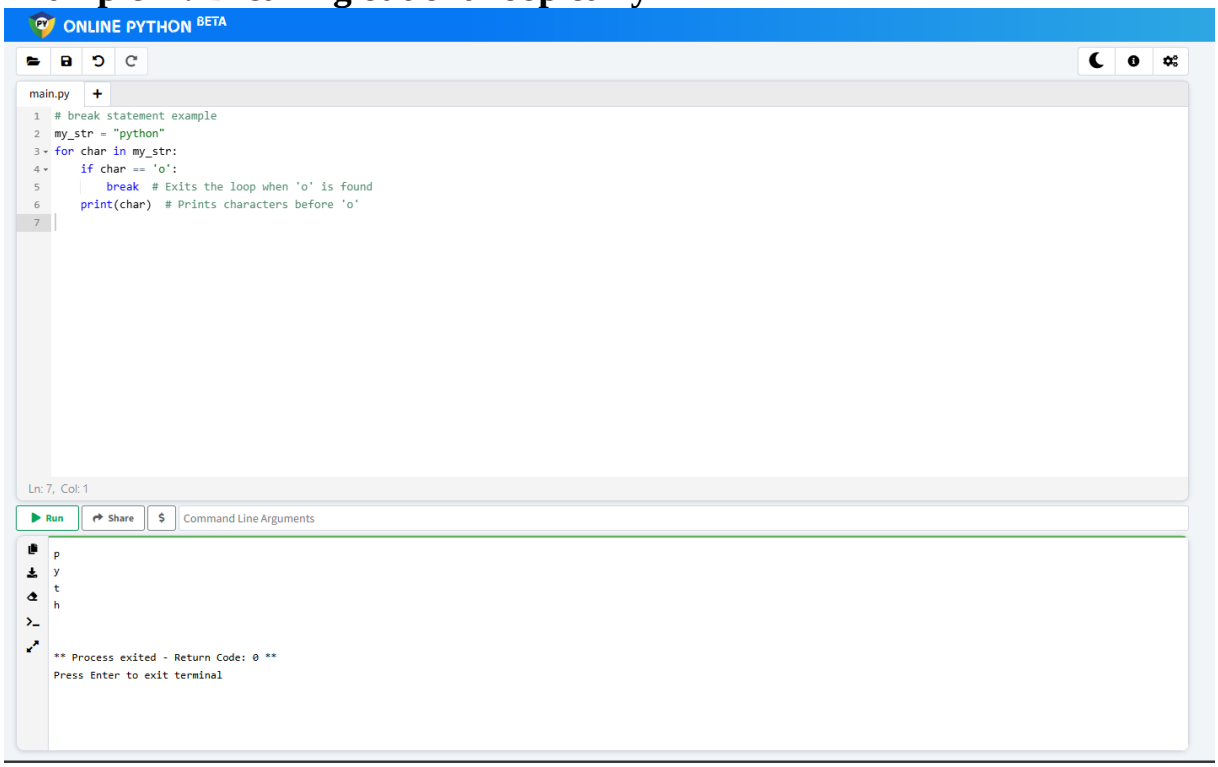


Python break statement

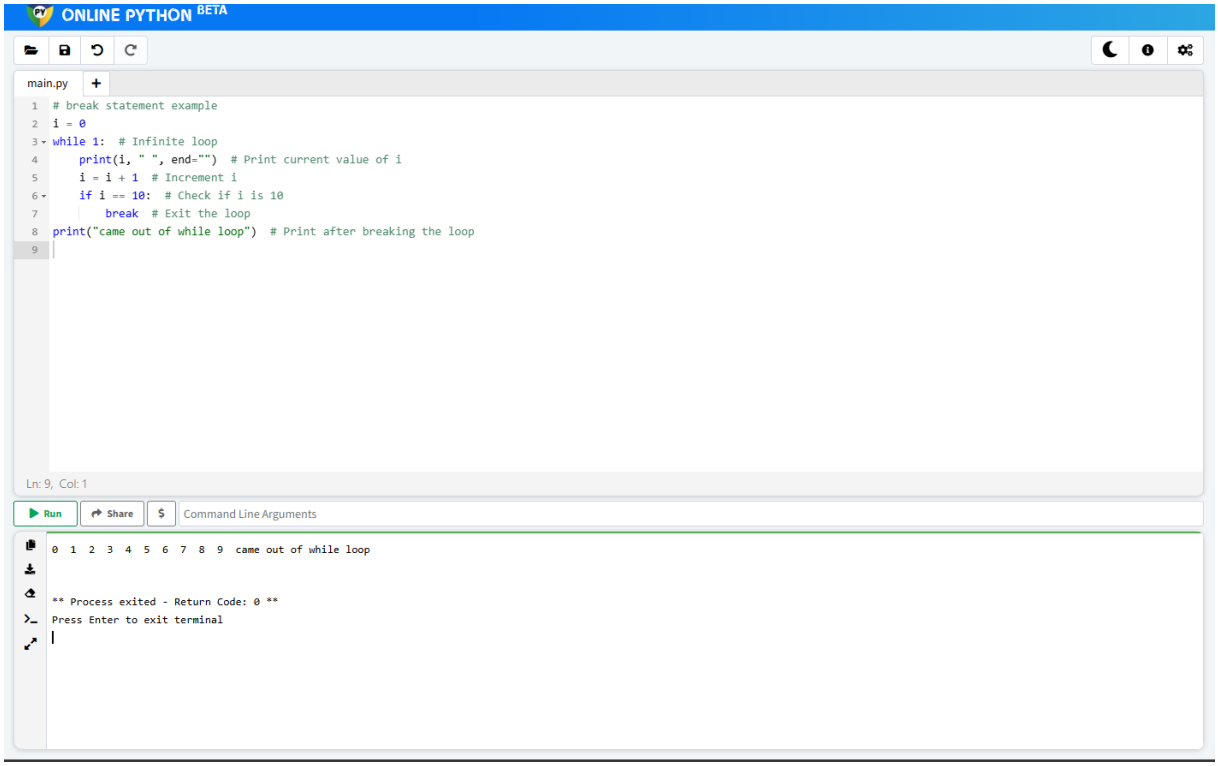
Example 1 : break statement with for loop



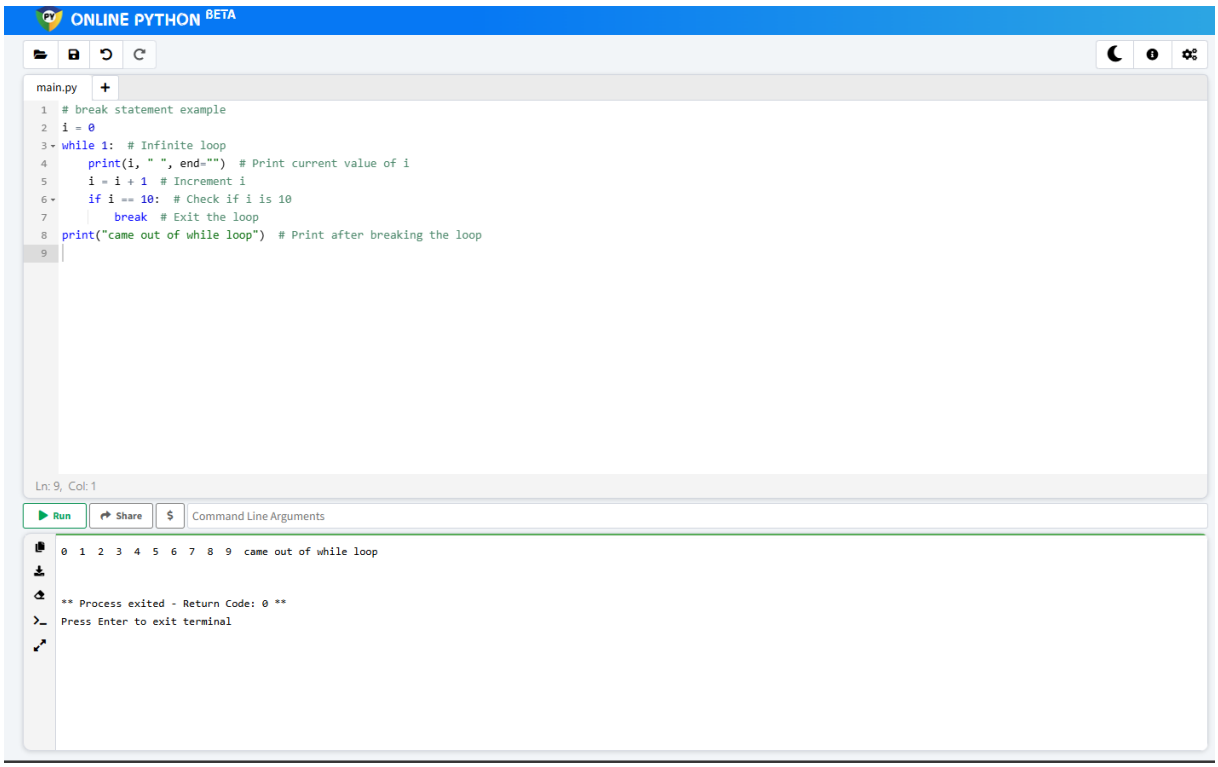
Example 2 : Breaking out of a loop early



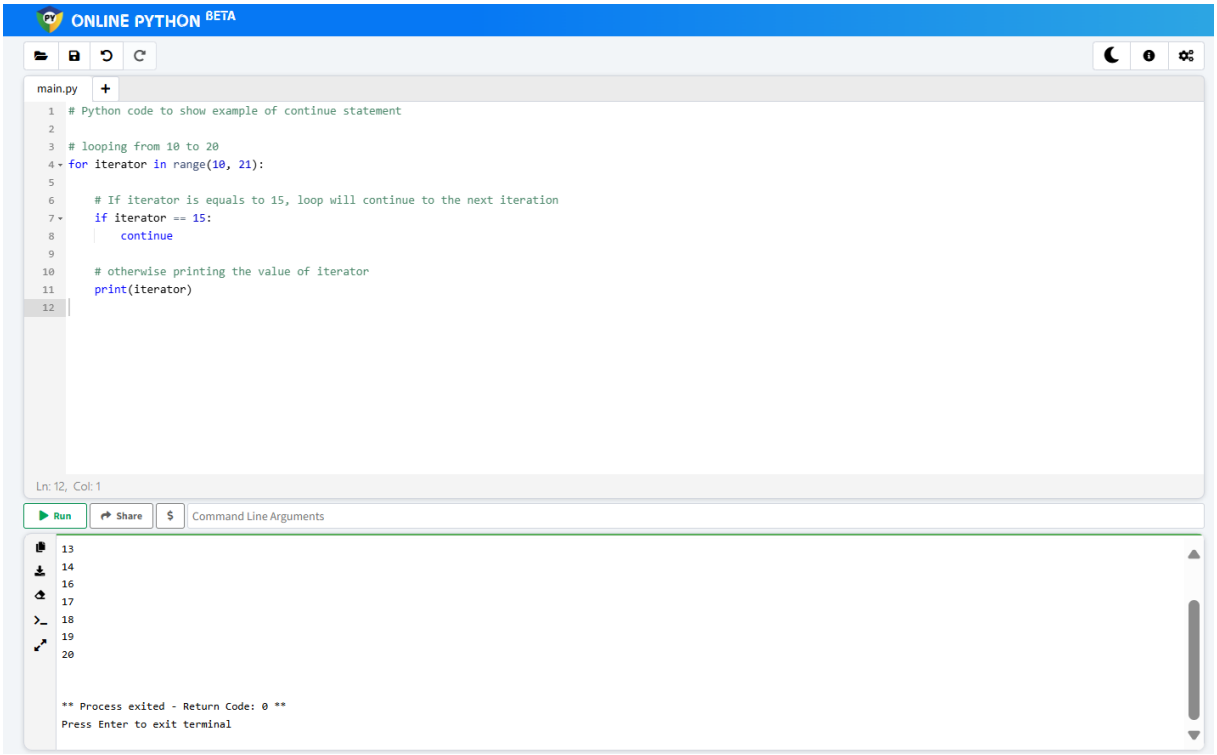
Example 3: break statement with while loop



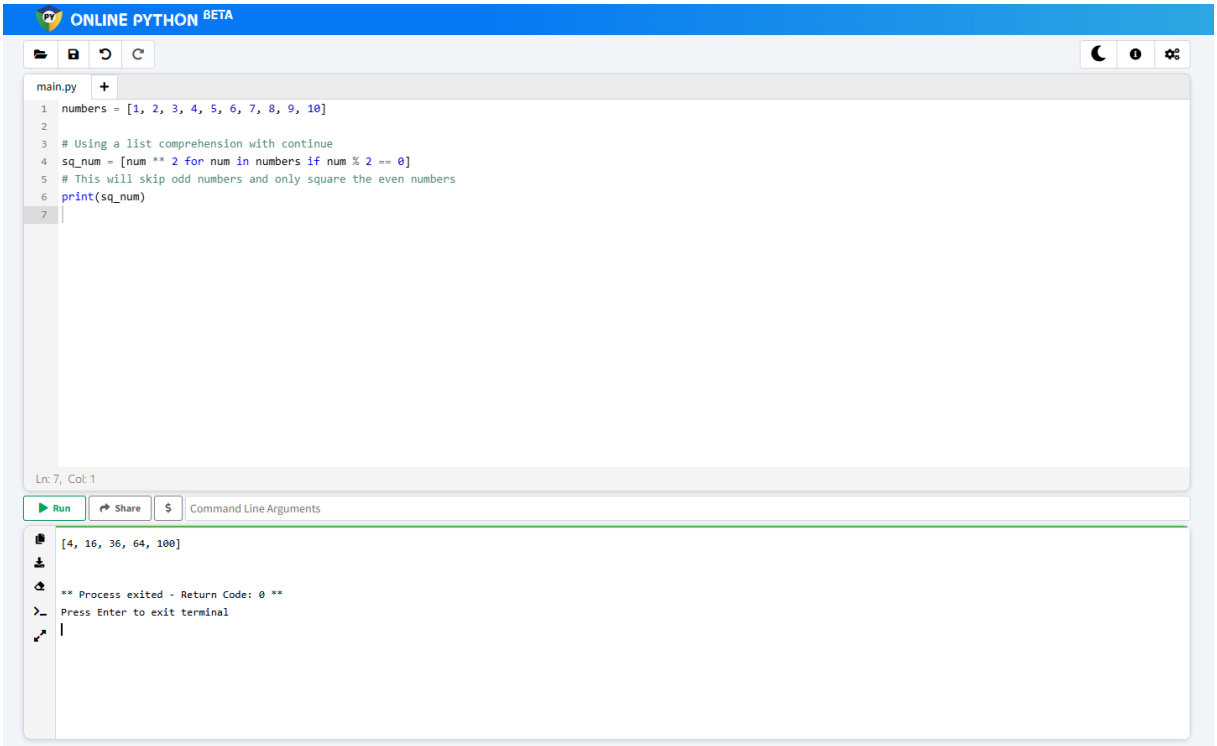
Example 4 : break statement with nested loops



Python continue Statement



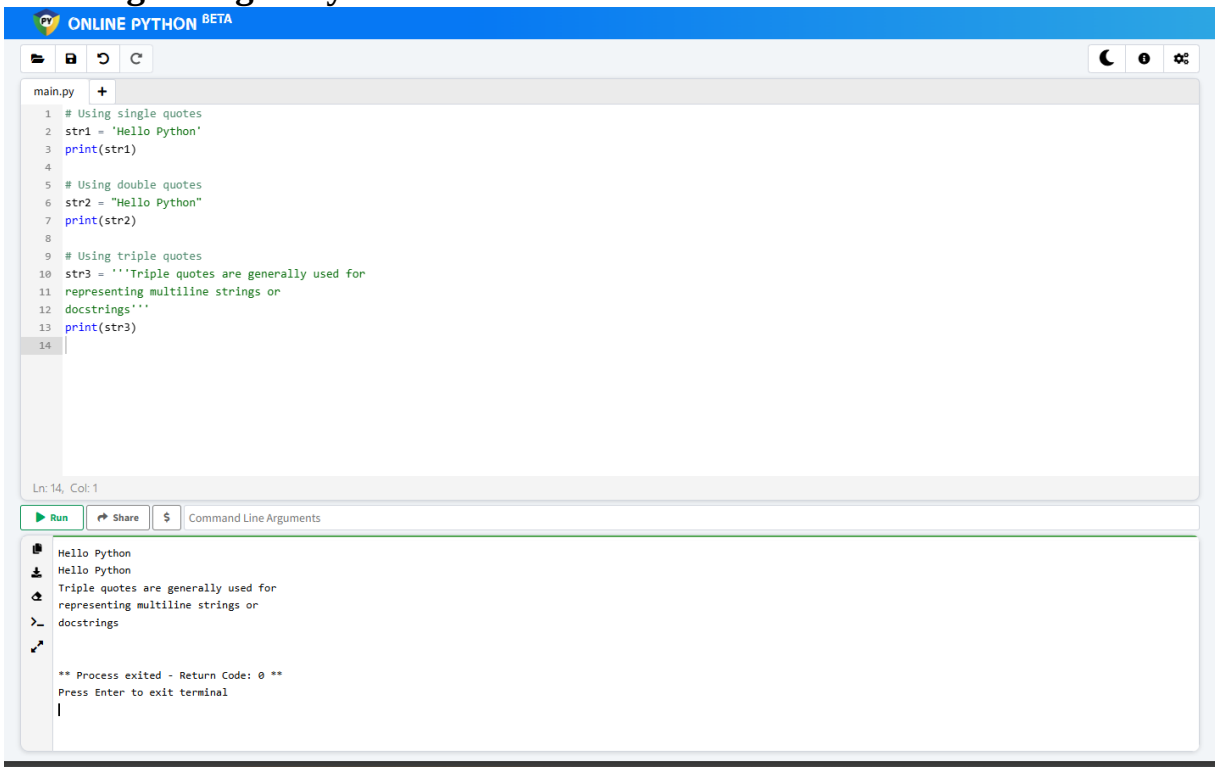
Python Continue statement in list comprehension



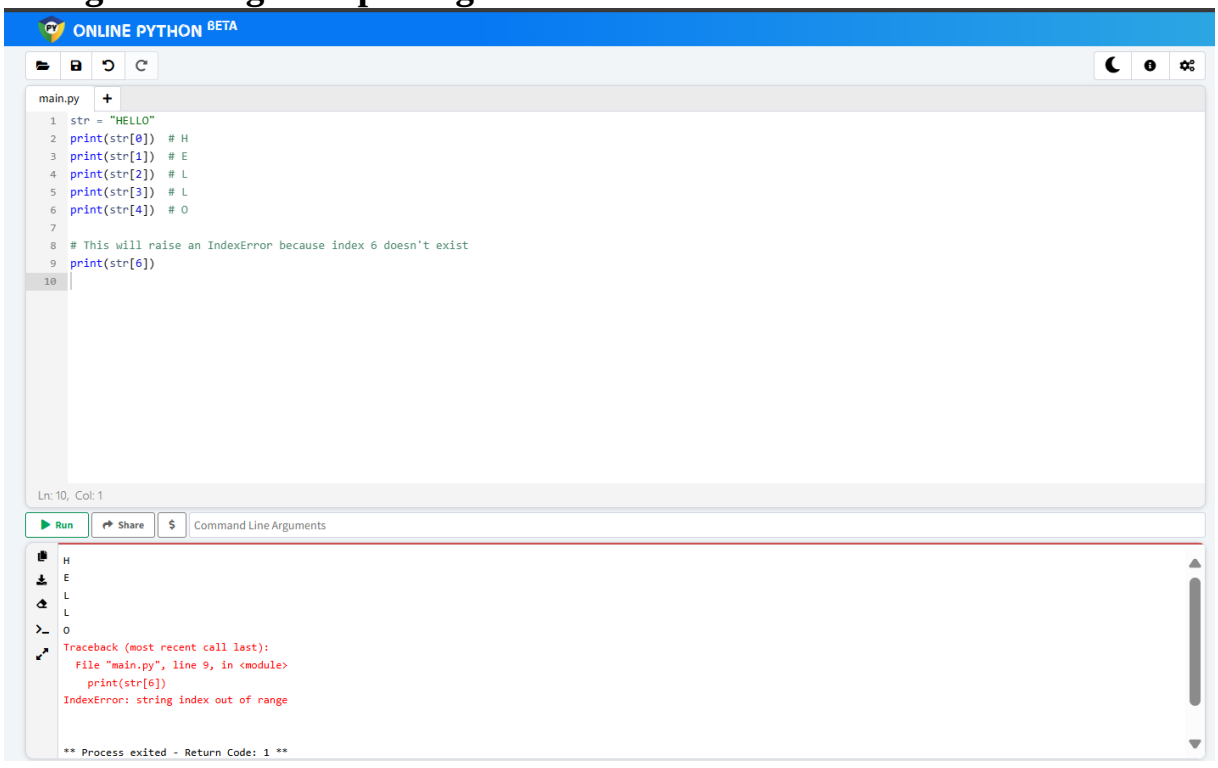
Python String

Python string is the collection of the characters surrounded by single quotes, double quotes, or triple quotes. The computer does not understand the characters; internally, it stores manipulated character as the combination of the 0's and 1's.

Creating String in Python



Strings indexing and splitting



ONLINE PYTHON BETA

main.py

```python
# Given String
str = "JAVATPOINT"

# Start from 0th index to the end
print(str[0:])        # Output: JAVATPOINT

# Start from 1st index to 4th index (index 5 is excluded)
print(str[1:5])       # Output: AVAT

# Start from 2nd index to 3rd index (index 4 is excluded)
print(str[2:4])       # Output: VA

# Start from 0th index to 2nd index (index 3 is excluded)
print(str[:3])        # Output: JAV

# Start from 4th index to 6th index (index 7 is excluded)
print(str[4:7])       # Output: TPO

```

Ln: 18, Col: 1

Run | Share | $ | Command Line Arguments

```
JAVATPOINT
AVAT
VA
JAV
TPO


** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

ONLINE PYTHON BETA

main.py

```python
str = 'JAVATPOINT'

# Accessing elements using negative indexing
print(str[-1])        # Output: T  (Last character)
print(str[-3])        # Output: I  (Third from the end)

# Slicing using negative indices
print(str[-2:])       # Output: NT  (Last two characters)
print(str[-4:-1])     # Output: OIN  (From the 4th-last to the 2nd-last, excluding the last character)
print(str[-7:-2])     # Output: VATPO  (From the 7th-last to the 3rd-last, excluding the 2nd-last)

# Reversing the given string
print(str[::-1])      # Output: TNIOPTAVAJ

# Attempt to access an index that is out of range (will cause an error)
print(str[-12])       # This will raise an IndexError because the string length is only 10

```

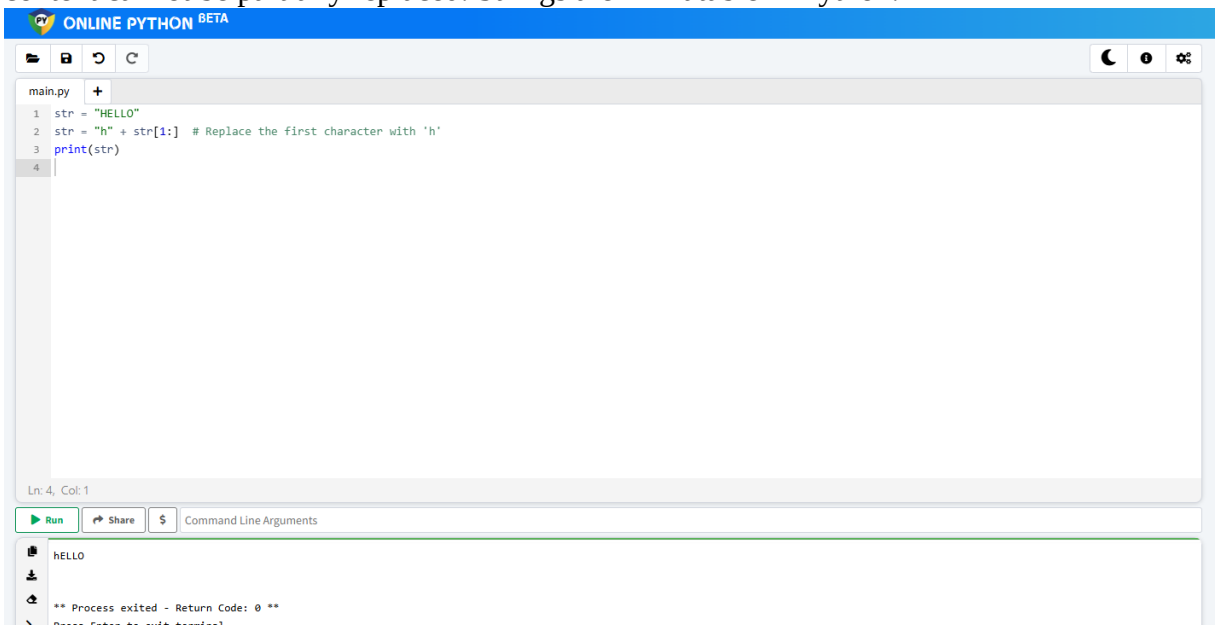
Ln: 17, Col: 1

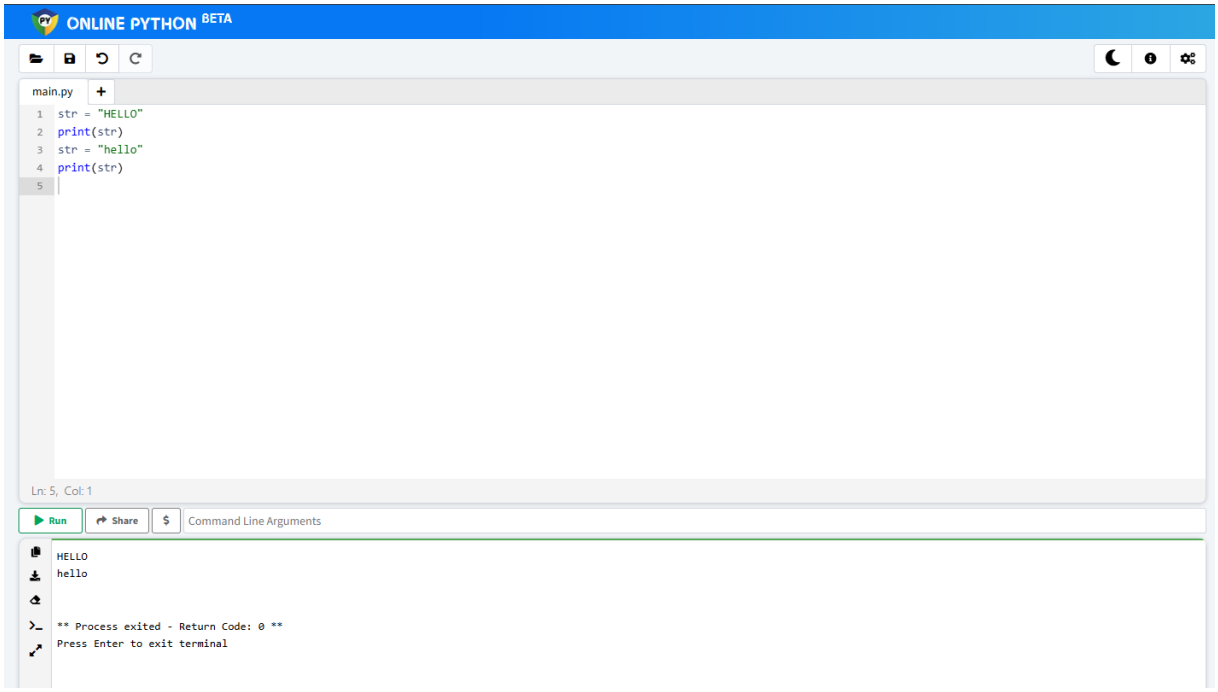
Run | Share | $ | Command Line Arguments

```
T
I
NT
OIN
ATPOI
TNIOPTAVAJ
Traceback (most recent call last):
  File "main.py", line 16, in <module>
    print(str[-12])       # This will raise an IndexError because the string length is only 10
IndexError: string index out of range
```

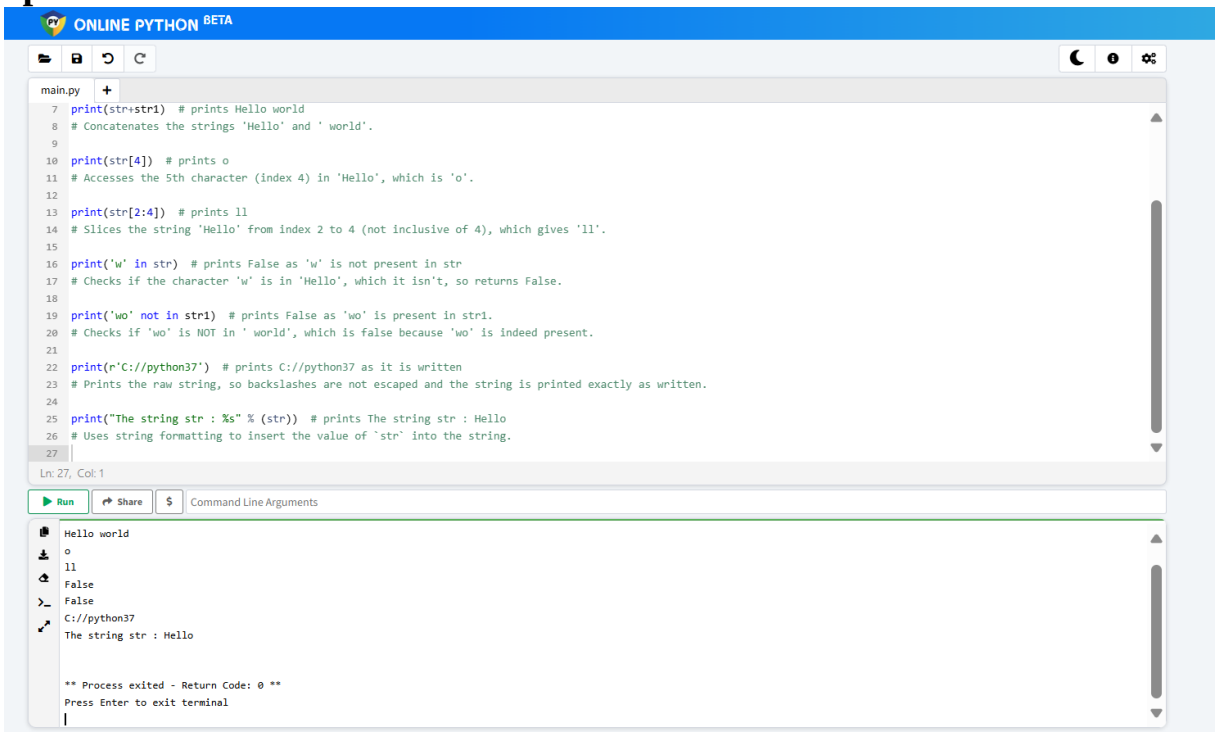
Reassigning Strings

Updating the content of the strings is as easy as assigning it to a new string. The string object doesn't support item assignment i.e., A string can only be replaced with new string since its content cannot be partially replaced. Strings are immutable in Python.



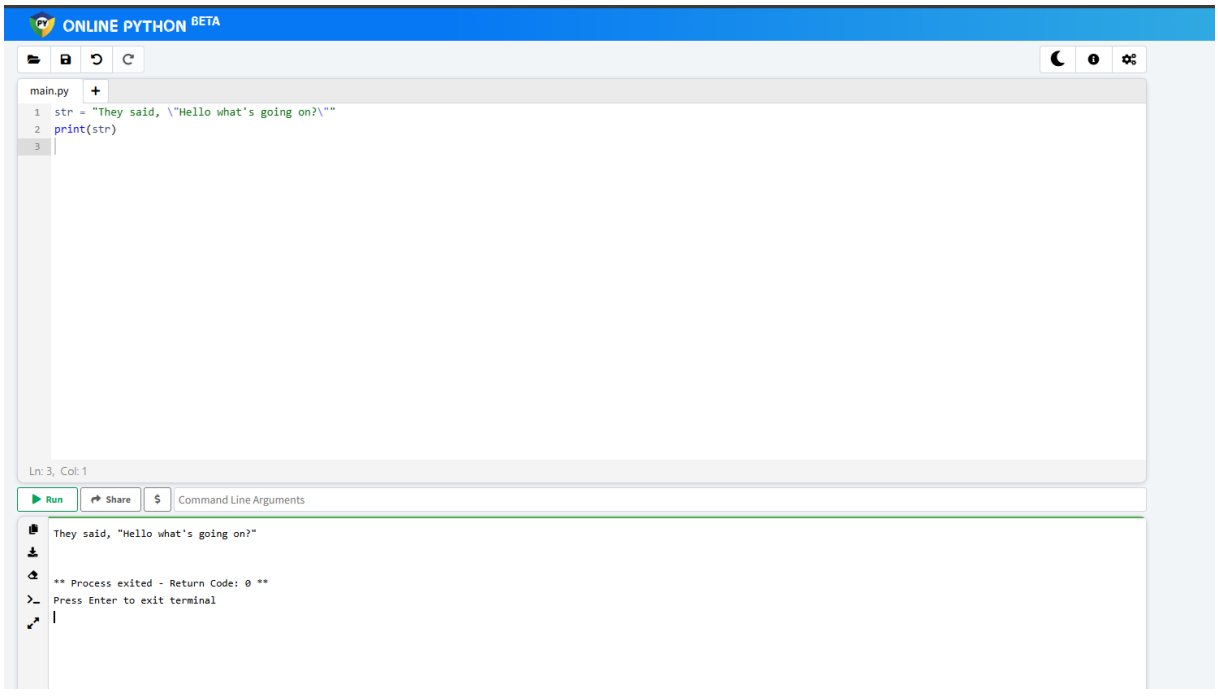


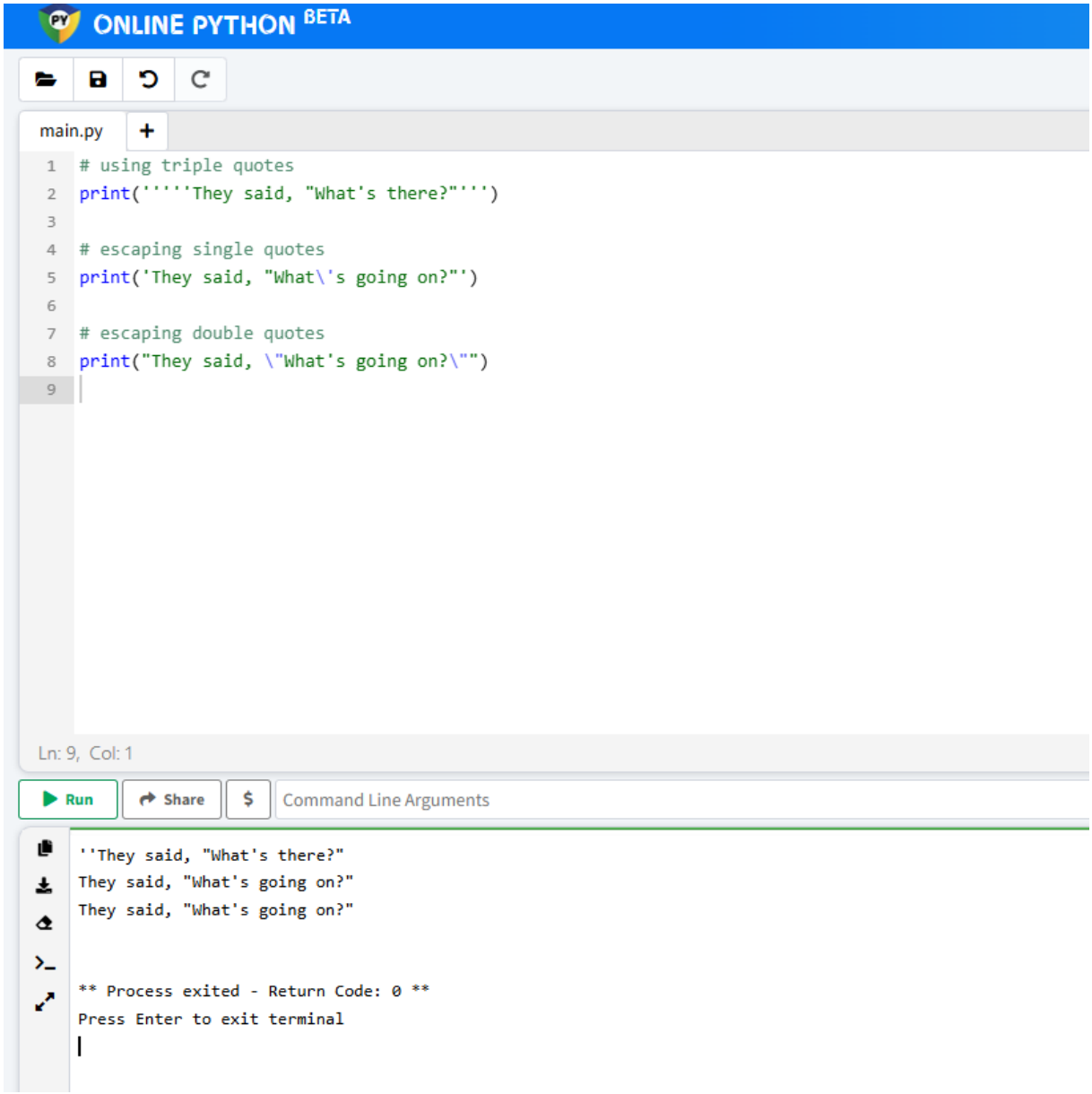
Consider the following example to understand the real use of Python operators.

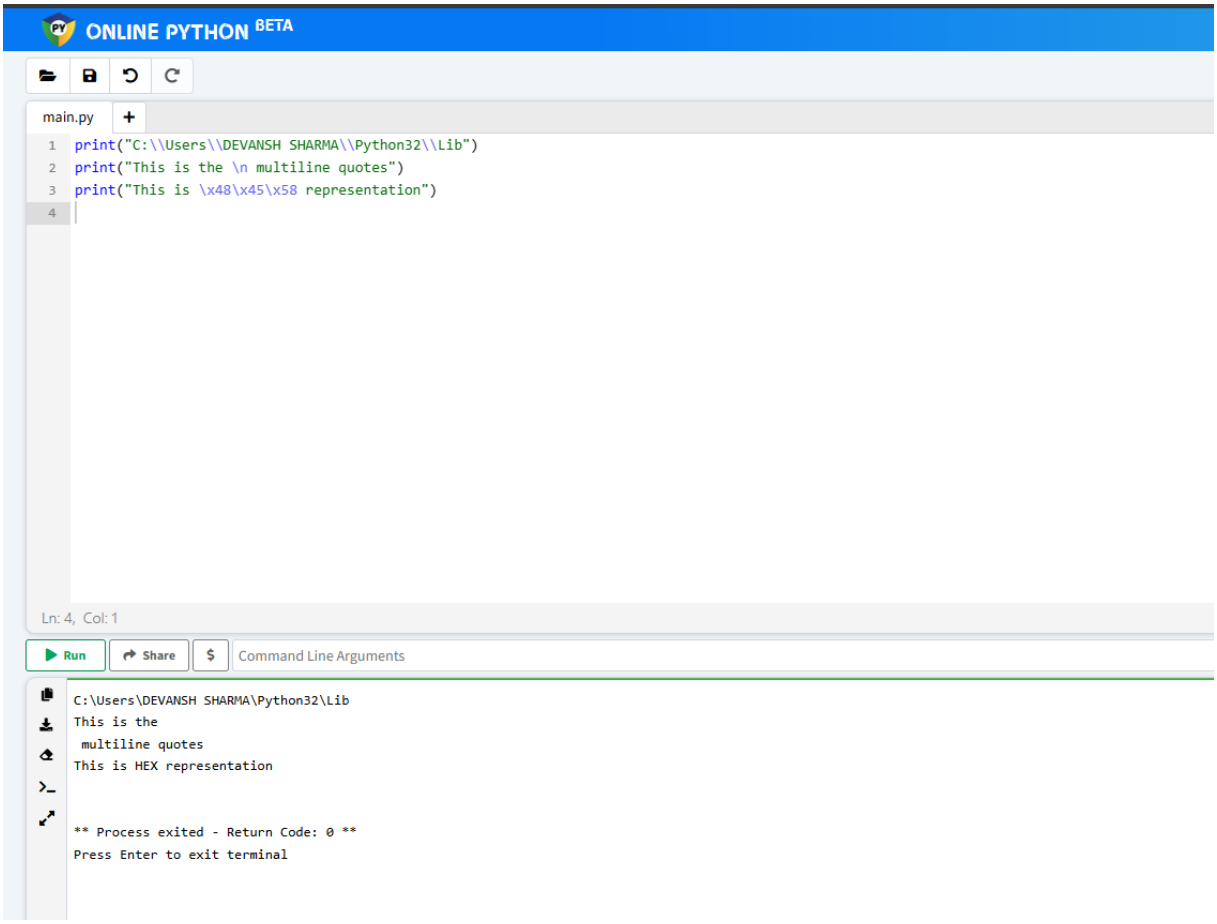


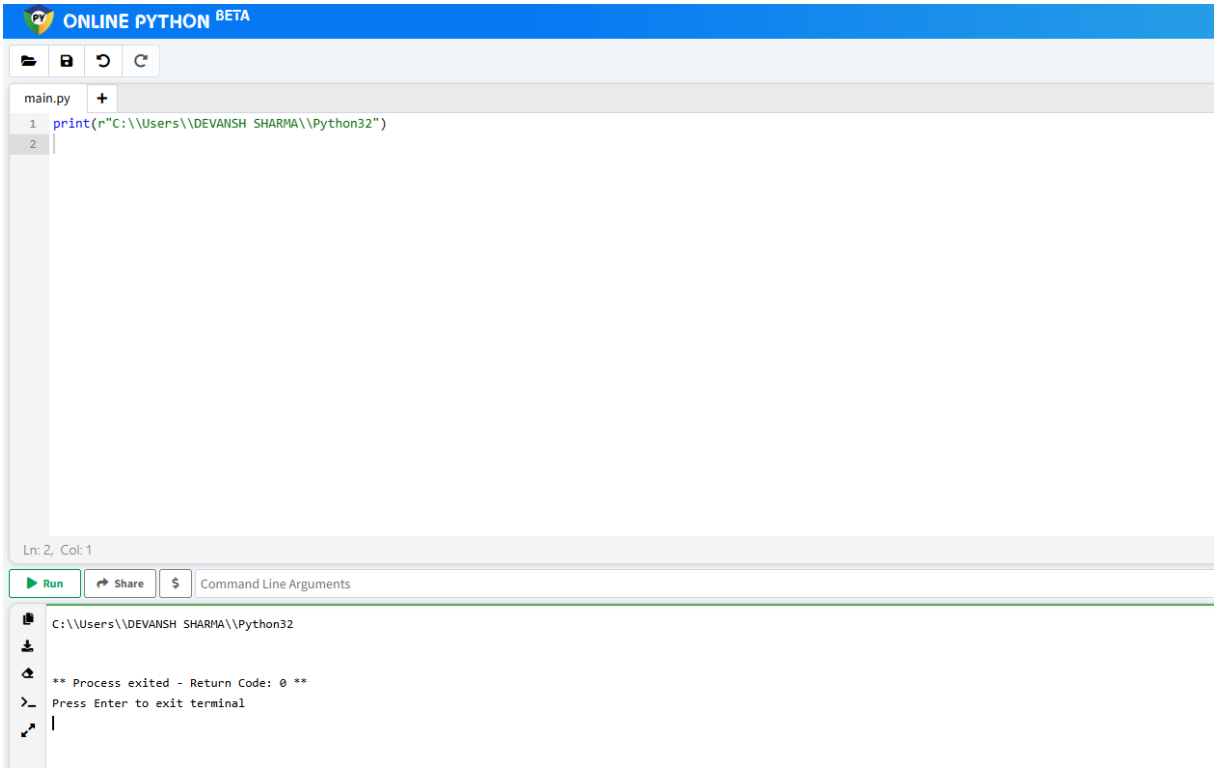
Python String Formatting
Escape Sequence

Let's suppose we need to write the text as - They said, "Hello what's going on?"- the given statement can be written in single quotes or double quotes but it will raise the SyntaxError as it contains both single and double-quotes.

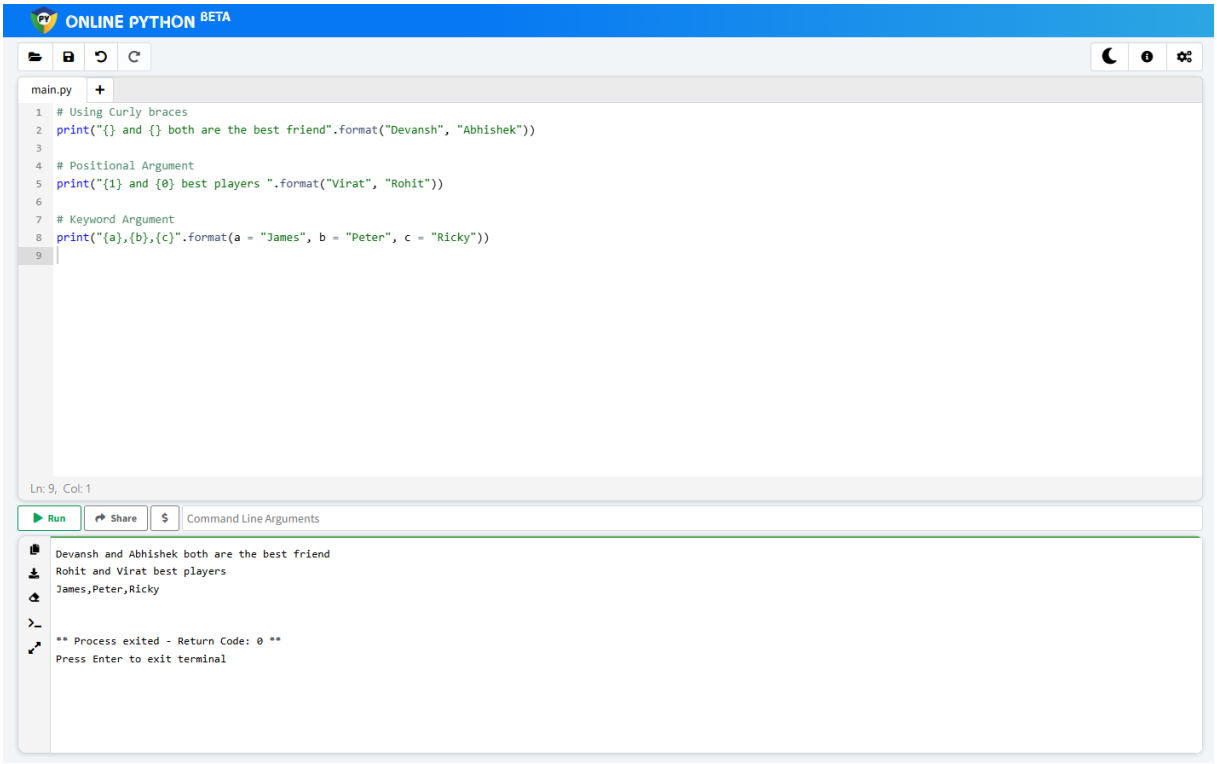



ONLINE PYTHON BETA
main.py
# using triple quotes
print('''''They said, "What's there?"''')
# escaping single quotes
print('They said, "What\'s going on?"')
# escaping double quotes
print("They said, \"What's going on?\"")
Ln: 9, Col: 1
Run
Share
$
Command Line Arguments
''They said, "What's there?"
They said, "What's going on?"
They said, "What's going on?"
** Process exited - Return Code: 0 **
Press Enter to exit terminal

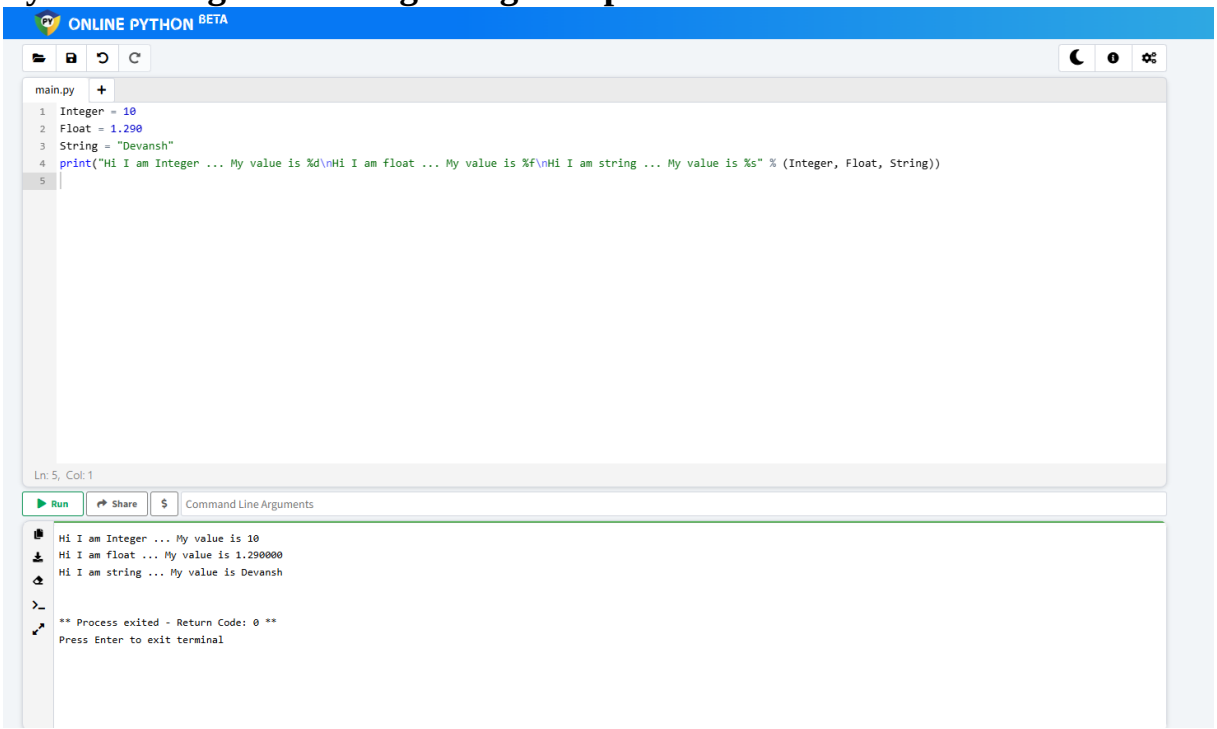

ONLINE PYTHON BETA
main.py
print("C:\\Users\\DEVANSH SHARMA\\Python32\\Lib")
print("This is the \n multiline quotes")
print("This is \x48\x45\x58 representation")
Ln: 4, Col: 1
Run
Share
Command Line Arguments
C:\Users\DEVANSH SHARMA\Python32\Lib
This is the
multiline quotes
This is HEX representation
** Process exited - Return Code: 0 **
Press Enter to exit terminal


ONLINE PYTHON BETA
main.py
print(r"C:\\Users\\DEVANSH SHARMA\\Python32")
Ln: 2, Col: 1
Run
Share
Command Line Arguments
C:\\Users\\DEVANSH SHARMA\\Python32
** Process exited - Return Code: 0 **
Press Enter to exit terminal

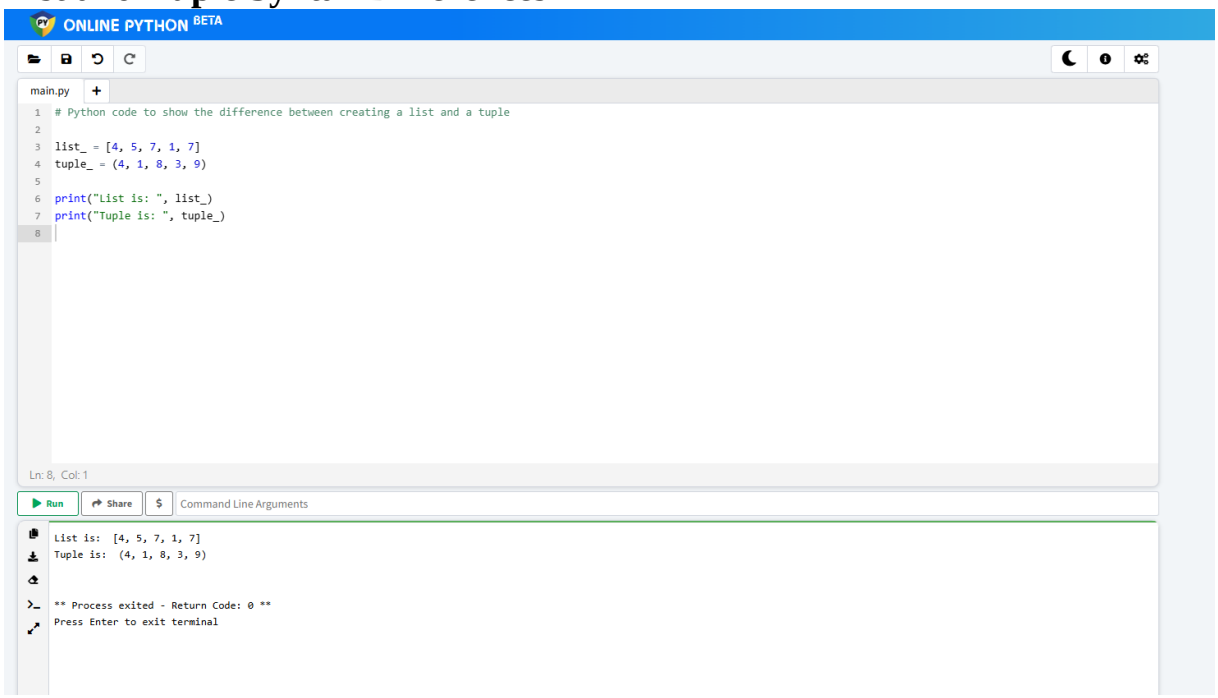
The format() method



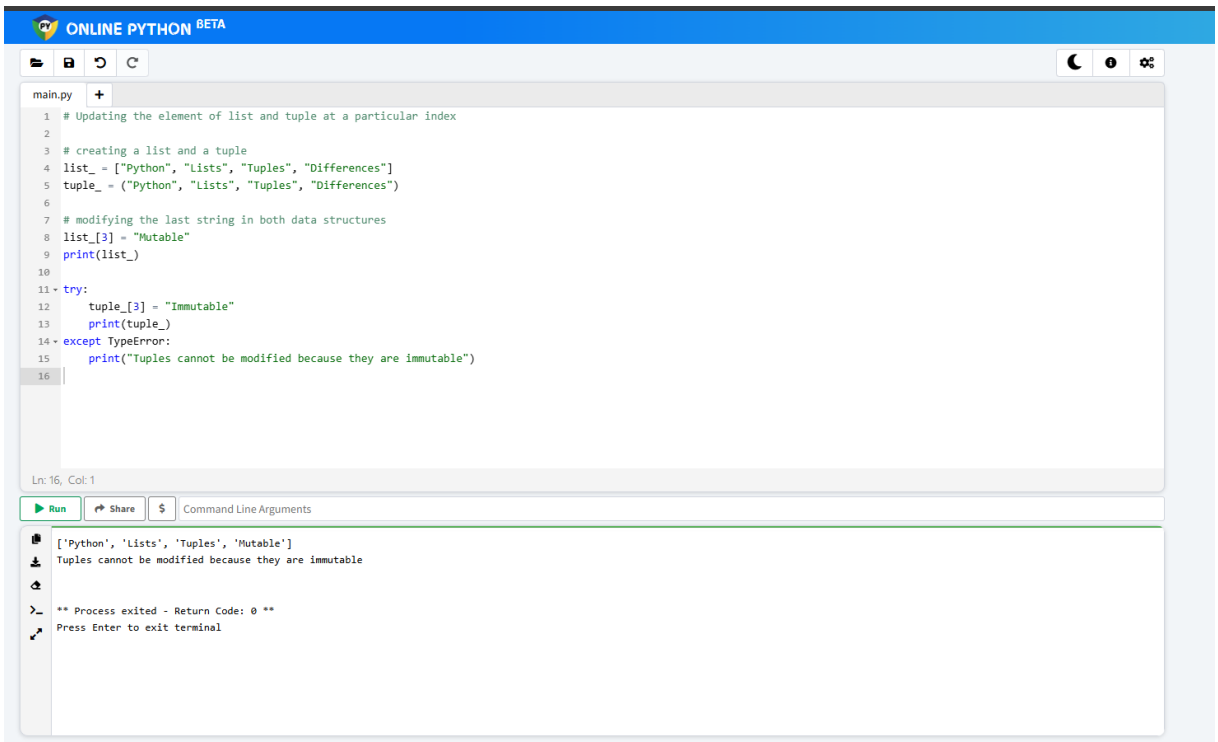
Python String Formatting Using % Operator



List and Tuple Syntax Differences



Mutable List vs. Immutable Tuple



Size Difference