Python While Loops

In coding, loops are designed to execute a specified code block repeatedly. We'll learn how to construct a while loop in Python, the syntax of a while loop, loop controls like break and continue, and other exercises in this tutorial.

Introduction of Python While Loop

In this article, we are discussing while loops in Python. The Python while loop iteration of a code block is executed as long as the given Condition, i.e., conditional\_expression, is true.

If we don't know how many times we'll execute the iteration ahead of time, we can write an indefinite loop.

**Syntax of Python While Loop**

Now, here we discuss the syntax of the Python while loop. The syntax is given below -

1. Statement
2. **while** Condition:
3. Statement

The given condition, i.e., conditional\_expression, is evaluated initially in the Python while loop. Then, if the conditional expression gives a boolean value True, the while loop statements are executed. The conditional expression is verified again when the complete code block is executed. This procedure repeatedly occurs until the conditional expression returns the boolean value False.

* The statements of the Python while loop are dictated by indentation.
* The code block begins when a statement is indented & ends with the very first unindented statement.
* Any non-zero number in Python is interpreted as boolean True. False is interpreted as None and 0.

Example

Now we give some examples of while Loop in Python. The examples are given in below -

**Program code 1:**

Now we give code examples of while loops in Python for printing numbers from 1 to 10. The code is given below -

1. i=1
2. **while** i<=10:
3. print(i, end=' ')
4. i+=1

**Output:**

Now we compile the above code in python, and after successful compilation, we run it. Then the output is given below -

1 2 3 4 5 6 7 8 9 10

**Program Code 2:**

Now we give code examples of while loops in Python for Printing those numbers divisible by either 5 or 7 within 1 to 50 using a while loop. The code is given below -

1. i=1
2. **while** i<51:
3. **if** i%5 == 0 or i%7==0 :
4. print(i, end=' ')
5. i+=1

**Output:**

Now we compile the above code in python, and after successful compilation, we run it. Then the output is given below -

5 7 10 14 15 20 21 25 28 30 35 40 42 45 49 50

**Program Code:**

Now we give code examples of while loops in Python, the sum of squares of the first 15 natural numbers using a while loop. The code is given below -

AD

1. # Python program example to show the use of **while** loop
3. num = 15
5. # initializing summation and a counter **for** iteration
6. summation = 0
7. c = 1
9. **while** c <= num: # specifying the condition of the loop
10. # begining the code block
11. summation = c\*\*2 + summation
12. c = c + 1    # incrementing the counter
14. # print the **final** sum
15. print("The sum of squares is", summation)

**Output:**

Now we compile the above code in python, and after successful compilation, we run it. Then the output is given below -

The sum of squares is 1240

Provided that our counter parameter i gives boolean true for the condition, i less than or equal to num, the loop repeatedly executes the code block i number of times.

Next is a crucial point (which is mostly forgotten). We have to increment the counter parameter's value in the loop's statements. If we don't, our while loop will execute itself indefinitely (a never-ending loop).

Finally, we print the result using the print statement.

AD

AD

Exercises of Python While Loop

Prime Numbers and Python While Loop

Using a while loop, we will construct a Python program to verify if the given integer is a prime number or not.

**Program Code:**

Now we give code examples of while loops in Python for a number is Prime number or not. The code is given below -

1. num = [34, 12, 54, 23, 75, 34, 11]
3. def prime\_number(number):
4. condition = 0
5. iteration = 2
6. **while** iteration <= number / 2:
7. **if** number % iteration == 0:
8. condition = 1
9. **break**
10. iteration = iteration + 1
12. **if** condition == 0:
13. print(f"{number} is a PRIME number")
14. **else**:
15. print(f"{number} is not a PRIME number")
16. **for** i in num:
17. prime\_number(i)

**Output:**

Now we compile the above code in python, and after successful compilation, we run it. Then the output is given below -

34 is not a PRIME number

12 is not a PRIME number

54 is not a PRIME number

23 is a PRIME number

75 is not a PRIME number

34 is not a PRIME number

11 is a PRIME number

2. Armstrong and Python While Loop

We will construct a Python program using a while loop to verify whether the given integer is an Armstrong number.

**Program Code:**

Now we give code examples of while loops in Python for a number is Armstrong number or not. The code is given below -

1. n = **int**(input())
2. n1=str(n)
3. l=len(n1)
4. temp=n
5. s=0
6. **while** n!=0:
7. r=n%10
8. s=s+(r\*\*1)
9. n=n//10
10. **if** s==temp:
11. print("It is an Armstrong number")
12. **else**:
13. print("It is not an Armstrong number ")

**Output:**

Now we compile the above code in python, and after successful compilation, we run it. Then the output is given below -

342

It is not an Armstrong number

Multiplication Table using While Loop

In this example, we will use the while loop for printing the multiplication table of a given number.

**Program Code:**

In this example, we will use the while loop for printing the multiplication table of a given number. The code is given below -

1. num = 21
2. counter = 1
3. # we will use a **while** loop **for** iterating 10 times **for** the multiplication table
4. print("The Multiplication Table of: ", num)
5. **while** counter <= 10: # specifying the condition
6. ans = num \* counter
7. print (num, 'x', counter, '=', ans)
8. counter += 1 # expression to increment the counter

**Output:**

Now we compile the above code in python, and after successful compilation, we run it. Then the output is given below -

The Multiplication Table of: 21

21 x 1 = 21

21 x 2 = 42

21 x 3 = 63

21 x 4 = 84

21 x 5 = 105

21 x 6 = 126

21 x 7 = 147

21 x 8 = 168

21 x 9 = 189

21 x 10 = 210

Python While Loop with List

**Program Code 1:**

Now we give code examples of while loops in Python for square every number of a list. The code is given below -

1. # Python program to square every number of a list
2. # initializing a list
3. list\_ = [3, 5, 1, 4, 6]
4. squares = []
5. # programing a **while** loop
6. **while** list\_: # until list is not empty **this** expression will give **boolean** True after that False
7. squares.append( (list\_.pop())\*\*2)
8. #  Print the squares of all numbers.
9. print( squares )

**Output:**

Now we compile the above code in python, and after successful compilation, we run it. Then the output is given below -

[36, 16, 1, 25, 9]

In the preceding example, we execute a while loop over a given list of integers that will repeatedly run if an element in the list is found.

**Program Code 2:**

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 -

1. list\_ = [3, 4, 8, 10, 34, 45, 67,80]        # Initialize the list
2. index = 0
3. **while** index < len(list\_):
4. element = list\_[index]
5. **if** element % 2 == 0:
6. print('It is an even number')       # Print **if** the number is even.
7. **else**:
8. print('It is an odd number')        # Print **if** the number is odd.
9. index += 1

**Output:**

Now we compile the above code in python, and after successful compilation, we run it. Then the output is given below -

It is an odd number

It is an even number

It is an even number

It is an even number

It is an even number

It is an odd number

It is an odd number

It is an even number

**Program Code 3:**

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 -

1. List\_= ['Priya', 'Neha', 'Cow', 'To']
2. index = 0
3. **while** index < len(List\_):
4. element = List\_[index]
5. print(len(element))
6. index += 1

**Output:**

Now we compile the above code in python, and after successful compilation, we run it. Then the output is given below -

5

4

3

2

Python While Loop Multiple Conditions

We must recruit logical operators to combine two or more expressions specifying conditions into a single while loop. This instructs Python on collectively analyzing all the given expressions of conditions.

We can construct a while loop with multiple conditions in this example. We have given two conditions and a and keyword, meaning the Loop will execute the statements until both conditions give Boolean True.

**Program Code:**

Now we give code examples of while loops in Python for multiple condition. The code is given below -

1. num1 = 17
2. num2 = -12
4. **while** num1 > 5 and num2 < -5 : # multiple conditions in a single **while** loop
5. num1 -= 2
6. num2 += 3
7. print( (num1, num2) )

**Output:**

Now we compile the above code in python, and after successful compilation, we run it. Then the output is given below -

(15, -9)

(13, -6)

(11, -3)

Let's look at another example of multiple conditions with an OR operator.

**Code**

1. num1 = 17
2. num2 = -12
4. **while** num1 > 5 or num2 < -5 :
5. num1 -= 2
6. num2 += 3
7. print( (num1, num2) )

**Output:**

Now we compile the above code in python, and after successful compilation, we run it. Then the output is given below -

(15, -9)

(13, -6)

(11, -3)

(9, 0)

(7, 3)

(5, 6)

We can also group multiple logical expressions in the while loop, as shown in this example.

**Code**

1. num1 = 9
2. num = 14
3. maximum\_value = 4
4. counter = 0
5. **while** (counter < num1 or counter < num2) and not counter >= maximum\_value: # grouping multiple conditions
6. print(f"Number of iterations: {counter}")
7. counter += 1

**Output:**

Now we compile the above code in python, and after successful compilation, we run it. Then the output is given below -

Number of iterations: 0

Number of iterations: 1

Number of iterations: 2

Number of iterations: 3

Single Statement While Loop

Similar to the if statement syntax, if our while clause consists of one statement, it may be written on the same line as the while keyword.

Here is the syntax and example of a one-line while clause -

1. # Python program to show how to create a single statement **while** loop
2. counter = 1
3. **while** counter: print('Python While Loops')

AD

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 of the Python interpreter to the beginning of the loop.

**Code**

1. # Python program to show how to use **continue** loop control
3. # Initiating the loop
4. **for** string in "While Loops":
5. **if** string == "o" or string == "i" or string == "e":
6. **continue**
7. print('Current Letter:', string)

Now we compile the above code in python, and after successful compilation, we run it. Then the output is given below -

**Output:**

Current Letter: W

Current Letter: h

Current Letter: l

Current Letter:

Current Letter: L

Current Letter: p

Current Letter: s

Break Statement

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

**Code**

1. # Python program to show how to use the **break** statement
3. # Initiating the loop
4. **for** string in "Python Loops":
5. **if** string == 'n':
6. **break**
7. print('Current Letter: ', string)

**Output:**

Now we compile the above code in python, and after successful compilation, we run it. Then the output is given below -

Current Letter: P

Current Letter: y

Current Letter: t

Current Letter: h

Current Letter: o

Pass Statement

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

**Code**

1. # Python program to show how to use the pass statement
2. **for** a string in "Python Loops":
3. pass
4. print( 'The Last Letter of given string is:', string)

Now we compile the above code in python, and after successful compilation, we run it. Then the output is given below -

**Output:**

The Last Letter of given string is: s