What is Python

Python is a general-purpose, dynamically typed, high-level, compiled and interpreted, garbage-collected, and object-oriented programming language that supports procedural, object-oriented, and functional programming.

In the late 1980s, [**Guido van Rossum**](https://en.wikipedia.org/wiki/Guido_van_Rossum) dreamed of developing Python. The first version of **Python 0.9.0 was released in 1991**. Since its release, Python started gaining popularity. According to reports, Python is now the most popular programming language among developers because of its high demands in the tech realm.

## Features of Python:

* **Easy to use and Read -** Python's syntax is clear and easy to read, making it an ideal language for both beginners and experienced programmers. This simplicity can lead to faster development and reduce the chances of errors.
* **Dynamically Typed** - The data types of variables are determined during run-time. We do not need to specify the data type of a variable during writing codes.
* **High-level** - High-level language means human readable code.
* **Compiled and Interpreted** - Python code first gets compiled into bytecode, and then interpreted line by line. When we download the Python in our system form [org](https://www.python.org/) we download the default implement of Python known as CPython. CPython is considered to be Complied and Interpreted both.
* **Garbage Collected** - Memory allocation and de-allocation are automatically managed. Programmers do not specifically need to manage the memory.
* **Purely Object-Oriented** - It refers to everything as an object, including numbers and strings.
* **Cross-platform Compatibility** - Python can be easily installed on Windows, macOS, and various Linux distributions, allowing developers to create software that runs across different operating systems.
* **Rich Standard Library** - Python comes with several standard libraries that provide ready-to-use modules and functions for various tasks, ranging from **web development** and **data manipulation** to **machine learning** and **networking**.
* **Open Source** - Python is an open-source, cost-free programming language. It is utilized in several sectors and disciplines as a result.

## Java vs. Python

* In Python, it is easy to learn and write code. While in Java, it requires more code to perform certain tasks.
* Python is dynamically typed, meaning we do not need to declare the variable Whereas Java is statistically typed, meaning we need to declare the variable type.
* Python is suitable for various domains such as Data Science, Machine Learning, Web development, and more. Whereas Java is suitable for web development, mobile app development (Android), and more.

## Python Basic Syntax

There is no use of curly braces or semicolons in Python programming language. It is an English-like language. But Python uses indentation to define a block of code. Indentation is nothing but adding whitespace before the statement when it is needed.

Python is a case-sensitive language, which means that uppercase and lowercase letters are treated differently. For example, 'name' and 'Name' are two different variables in Python.

In Python, comments can be added using the '#' symbol. Any text written after the '#' symbol is considered a comment and is ignored by the interpreter. This trick is useful for adding notes to the code or temporarily disabling a code block. It also helps in understanding the code better by some other developers.

['If'](https://www.javatpoint.com/python-if-else), 'otherwise', ['for'](https://www.javatpoint.com/python-for-loop), ['while'](https://www.javatpoint.com/python-while-loop), 'try', 'except', and 'finally' are a few reserved keywords in Python that cannot be used as variable names. These terms are used in the language for particular reasons and have fixed meanings. If you use these keywords, your code may include errors, or the interpreter may reject them as potential new Variables.

Why learn Python?

Python provides many useful features to the programmer. These features make it the most popular and widely used language. We have listed below few-essential features of Python.

* **Easy to use and Learn:** Python has a simple and easy-to-understand syntax, unlike traditional languages like C, C++, Java, etc., making it easy for beginners to learn.
* **Expressive Language:** It allows programmers to express complex concepts in just a few lines of code or reduces Developer's Time.
* **Interpreted Language:** Python does not require compilation, allowing rapid development and testing. It uses Interpreter instead of Compiler.
* [**Object-Oriented Language**](https://www.javatpoint.com/python-oops-concepts): It supports object-oriented programming, making writing reusable and modular code easy.
* **Open-Source** **Language:** Python is open-source and free to use, distribute and modify.
* **Extensible:** Python can be extended with modules written in C, C++, or other languages.
* **Learn Standard Library:** Python's standard library contains many modules and functions that can be used for various tasks, such as [string manipulation](https://www.javatpoint.com/string-manipulation-in-python), [web programming](https://www.javatpoint.com/django-tutorial), and more.
* **GUI Programming Support:** Python provides several GUI frameworks, such as [Tkinter](https://www.javatpoint.com/python-tkinter) and [PyQt](https://www.javatpoint.com/pyqt-library-in-python), allowing developers to create desktop applications easily.
* **Integrated:** Python can easily integrate with other languages and technologies, such as C/C++, Java, and . NET.
* **Embeddable:** Python code can be embedded into other applications as a scripting language.
* **Dynamic Memory Allocation:** Python automatically manages memory allocation, making it easier for developers to write complex programs without worrying about memory management.
* **Wide Range of Libraries and Frameworks:** Python has a vast collection of libraries and frameworks, such as [NumPy](https://www.javatpoint.com/numpy-tutorial), [Pandas](https://www.javatpoint.com/python-pandas), [Django](https://www.javatpoint.com/django-tutorial), and [Flask](https://www.javatpoint.com/flask-tutorial), that can be used to solve a wide range of problems.
* **Versatility:** Python is a universal language in various domains such as web development, [machine learning](https://www.javatpoint.com/machine-learning), [data analysis](https://www.javatpoint.com/python-data-analytics), scientific computing, and more.
* **Large Community:** Python has a vast and active community of developers contributing to its development and offering support. This makes it easy for beginners to get help and learn from experienced developers.
* **Career Opportunities:** Python is a highly popular language in the job market. Learning Python can open up several career opportunities in [data science](https://www.javatpoint.com/data-science), [artificial intelligence](https://www.javatpoint.com/artificial-intelligence-ai), web development, and more.
* **High Demand:** With the growing demand for automation and digital transformation, the need for Python developers is rising. Many industries seek skilled Python developers to help build their digital infrastructure.
* **Increased Productivity:** Python has a simple syntax and powerful libraries that can help developers write code faster and more efficiently. This can increase productivity and save time for developers and organizations.
* **Big Data and Machine Learning:** Python has become the go-to language for big data and machine learning. Python has become popular among data scientists and machine learning engineers with libraries like [NumPy](https://www.javatpoint.com/numpy-tutorial), [Pandas](https://www.javatpoint.com/python-pandas), [Scikit-learn](https://www.javatpoint.com/what-is-sklearn-in-python), [TensorFlow](https://www.javatpoint.com/tensorflow), and more.

## Where is Python used?

Python is a general-purpose, popular programming language, and it is used in almost every technical field. The various areas of Python use are given below.

* **Data Science:** Data Science is a vast field, and Python is an important language for this field because of its simplicity, ease of use, and availability of powerful data analysis and visualization libraries like [NumPy](https://www.javatpoint.com/numpy-tutorial), [Pandas](https://www.javatpoint.com/python-pandas), and [Matplotlib](https://www.javatpoint.com/matplotlib).
* **Desktop Applications:** [PyQt](https://www.javatpoint.com/pyqt-library-in-python) and [Tkinter](https://www.javatpoint.com/python-tkinter) are useful libraries that can be used in GUI - Graphical User Interface-based Desktop Applications. There are better languages for this field, but it can be used with other languages for making Applications.
* **Console-based Applications:** Python is also commonly used to create command-line or console-based applications because of its ease of use and support for advanced features such as input/output redirection and piping.
* **Software Development:** Python is considered one of the best software-making languages. Python is easily compatible with both from Small Scale to Large Scale software.
* [**Artificial Intelligence**](https://www.javatpoint.com/artificial-intelligence-ai): AI is an emerging Technology, and Python is a perfect language for artificial intelligence and machine learning because of the availability of powerful libraries such as [TensorFlow](https://www.javatpoint.com/tensorflow), [Keras](https://www.javatpoint.com/keras), and [PyTorch](https://www.javatpoint.com/pytorch).
* **Web Applications:** Python is commonly used in web development on the backend with frameworks like [Django](https://www.javatpoint.com/django-tutorial) and [Flask](https://www.javatpoint.com/flask-tutorial) and on the front end with tools like [JavaScript](https://www.javatpoint.com/javascript-tutorial) [HTML](https://www.javatpoint.com/html-tutorial) and [CSS](https://www.javatpoint.com/css-tutorial).
* **Enterprise Applications:** Python can be used to develop large-scale enterprise applications with features such as distributed computing, networking, and parallel processing.
* **3D CAD Applications:** Python can be used for 3D computer-aided design (CAD) applications through libraries such as Blender.
* **Machine Learning:** Python is widely used for machine learning due to its simplicity, ease of use, and availability of powerful machine learning libraries.
* **Computer Vision or Image Processing Applications:** Python can be used for computer vision and image processing applications through powerful libraries such as [OpenCV](https://www.javatpoint.com/opencv) and Scikit-image.
* **Speech Recognition:** Python can be used for speech recognition applications through libraries such as [SpeechRecognition](https://www.javatpoint.com/speech-recognition-python) and [PyAudio](https://www.javatpoint.com/python-audio-modules).
* **Scientific computing:** Libraries like [NumPy](https://www.javatpoint.com/numpy-tutorial), [SciPy](https://www.javatpoint.com/python-scipy), and [Pandas](https://www.javatpoint.com/python-pandas) provide advanced numerical computing capabilities for tasks like data analysis, machine learning, and more.
* **Education:** Python's easy-to-learn syntax and availability of many resources make it an ideal language for teaching programming to beginners.
* **Testing:** Python is used for writing automated tests, providing frameworks like unit tests and pytest that help write test cases and generate reports.
* **Gaming:** Python has libraries like [Pygame](https://www.javatpoint.com/pygame), which provide a platform for developing games using Python.
* **IoT:** Python is used in IoT for developing scripts and applications for devices like [Raspberry Pi](https://www.javatpoint.com/what-is-raspberry-pi), [Arduino](https://www.javatpoint.com/arduino), and others.
* **Networking:** Python is used in networking for developing scripts and applications for network automation, monitoring, and management.
* [**DevOps**](https://www.javatpoint.com/devops): Python is widely used in DevOps for automation and scripting of infrastructure management, configuration management, and deployment processes.
* **Finance:** Python has libraries like [Pandas](https://www.javatpoint.com/python-pandas), [Scikit-learn](https://www.javatpoint.com/what-is-sklearn-in-python), and [Statsmodels](https://www.javatpoint.com/how-to-install-statsmodels-in-python) for financial modeling and analysis.
* **Audio and Music:** Python has libraries like Pyaudio, which is used for audio processing, synthesis, and analysis, and Music21, which is used for music analysis and generation.
* **Writing scripts:** Python is used for writing utility scripts to automate tasks like file operations, [web scraping](https://www.javatpoint.com/web-scraping-using-python), and [data processing](https://www.javatpoint.com/pandas-data-processing).

Python Popular Frameworks and Libraries

Python has wide range of libraries and frameworks widely used in various fields such as machine learning, artificial intelligence, web applications, etc. We define some popular frameworks and libraries of Python as follows.

* **Web development (Server-side)** - [**Django**](https://www.javatpoint.com/django-tutorial) [**Flask**](https://www.javatpoint.com/flask-tutorial), [**Pyramid**](https://www.javatpoint.com/pyramid-framework-in-python), **CherryPy**
* **GUIs based applications** - **[Tkinter](https://www.javatpoint.com/python-tkinter)**, **[PyGTK](https://www.javatpoint.com/pygtk-for-gui-programming)**, **[PyQt](https://www.javatpoint.com/pyqt-library-in-python)**, **PyJs**, etc.
* **Machine Learning** - **[TensorFlow](https://www.javatpoint.com/tensorflow)**, **[PyTorch](https://www.javatpoint.com/pytorch)**, **[Scikit-learn](https://www.javatpoint.com/what-is-sklearn-in-python)**, **[Matplotlib](https://www.javatpoint.com/matplotlib)**, **[Scipy](https://www.javatpoint.com/python-scipy)**, etc.
* **Mathematics** - **[NumPy](https://www.javatpoint.com/numpy-tutorial)**, [**Pandas**](https://www.javatpoint.com/python-pandas), etc.
* **BeautifulSoup:** a library for web scraping and parsing HTML and XML
* [**Requests**](https://www.javatpoint.com/python-requests-module-http-request): a library for making HTTP requests
* [**SQLAlchemy**](https://www.javatpoint.com/flask-sqlalchemy): a library for working with SQL databases
* [**Kivy**](https://www.javatpoint.com/kivy)**:** a framework for building multi-touch applications
* [**Pygame**](https://www.javatpoint.com/pygame)**:** a library for game development
* **Pytest:** a testing framework for Python Django
* [**REST framework**](https://www.javatpoint.com/create-rest-api-using-django-rest-framework): a toolkit for building RESTful APIs
* [**FastAPI**](https://www.javatpoint.com/fast-api-a-framework-to-create-apis): a modern, fast web framework for building APIs
* [**Streamlit**](https://www.javatpoint.com/deploy-a-machine-learning-model-using-streamlit-library): a library for building interactive web apps for machine learning and data science
* [**NLTK**](https://www.javatpoint.com/natural-language-toolkit): a library for natural language processing

## Python print() Function

print(object(s), sep=separator, end=end, file=file, flush=flush)

It takes one or more arguments separated by comma(,) and adds a 'newline' at the end by default.

* *object(s) - As many as you want data to display, will first converted into string and printed to the console.*
* *sep - Separates the objects by a separator passed, default value = " ".*
* *end - Ends a line with a newline character*
* *file - a file object with write method, default value = sys.stdout*

**Example:**

1. # Displaying a string
2. print("Hello, World!")
4. # Displaying multiple values
5. name = "Aman"
6. age = 21
7. print("Name:", name, "Age:", age)
9. # Printing variables and literals
10. x = 5
11. y = 7
12. print("x =", x, "y =", y, "Sum =", x + y)
14. # Printing with formatting
15. percentage = 85.75
16. print("Score: {:.2f}%".format(percentage))

**Output:**

Hello, World!

Name: Aman Age: 21

X = 5 y = 7 Sum = 12

Score: 85.75%

In this example, the print statement is used to print string, integer, and float values in a human readable format.

## Python Conditional Statements

Conditional statements help us to execute a particular block for a particular condition.

 Python provides if and else keywords to set up logical conditions. The **elif** keyword is also used as a conditional statement.

**Example code for if..else statement**

1. x = 10
2. y = 5
4. **if** x > y:
5. print("x is greater than y")
6. **else**:
7. print("y is greater than or equal to x")

**Output:**

x is greater than y

## Python Loops

Sometimes we may need to alter the flow of the program. The execution of a specific code may need to be repeated several times. For this purpose, the programming languages provide various loops capable of repeating some specific code several times. Consider the following tutorial to understand the statements in detail.

### **Python**[**For Loop**](https://www.javatpoint.com/python-for-loop)

1. fruits = ["apple", "banana", "cherry"]
2. **for** x **in** fruits:
3. **print**(x, end=" ")

**Output:**

apple banana cherry

1. # Code to find the sum of squares of each element of the list using for loop
3. # creating the list of numbers
4. numbers = [3, 5, 23, 6, 5, 1, 2, 9, 8]
6. # initializing a variable that will store the sum
7. sum\_ = 0
9. # using for loop to iterate over the list
10. **for** num **in** numbers:
12. sum\_ = sum\_ + num \*\* 2
14. **print**("The sum of squares is: ", sum\_)

**Output:**

The sum of squares is: 774

The range() Function

Since the "range" capability shows up so habitually in for circles, we could erroneously accept the reach as a part of the punctuation of for circle. It's not: It is a built-in Python method that fulfills the requirement of providing a series for the for expression to run over by following a particular pattern (typically serial integers). Mainly, they can act straight on sequences, so counting is unnecessary. This is a typical novice construct if they originate from a language with distinct loop syntax:

**Code**

1. my\_list = [3, 5, 6, 8, 4]
2. **for** iter\_var **in** range( len( my\_list ) ):
3. my\_list.append(my\_list[iter\_var] + 2)
4. **print**( my\_list )

**Output:**

[3, 5, 6, 8, 4, 5, 7, 8, 10, 6]



## Break Statement in for Loop

The Python break statement is used to exit from the loop immediately after a certain condition is met.

Example:



## Continue Statement in for Loop

The function of the continue statement is to skip the current iteration of a loop and continue with the next one.

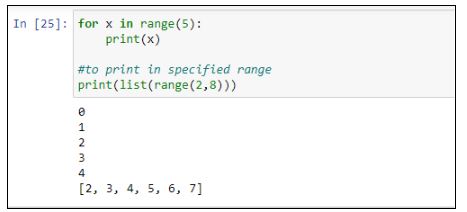
Example:



## The Range() Function

The range() function executes a group of statements for a specified number of times.

Example:



## Else in for Loop

Python enables an else clause at the end of a for loop. The else part is executed if the loop terminates naturally.

Example:



## Nested Loops

A loop inside another loop is called a nested loop. The inner loop will be performed once for each iteration of the outer loop.

Example:



## Access Index in for Loop

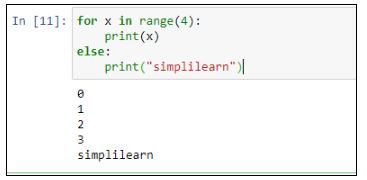
To iterate over the indices in a sequence, we use the enumerate() function.

Example:

## For loop Exercise

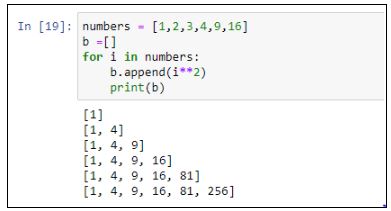
Question 1: Display a “simplilearn” message after successfully executing for loop

Solution:



Question 2: You are given a list of integer elements. Make a new list that will store squares of elements from the previous list.

Solution:



## for loop With else Output

Else in for loop can be executed if the for loop terminates naturally first and later else output can be executed.

### **Example:**

In [1] : colours = ['purple', 'blue', 'white', 'golden']

For x in colours:

print(x)

else:

printf("finished!")

## Introduction of Python While Loop

**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

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 -

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

## Exercises of Python While Loop

### **Prime Numbers and Python While Loop**

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

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)

**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')

## 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.

The pass keyword is used when a phrase is necessary syntactically to be placed but not to be executed.

**Code**

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

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

# **Python break statement**

The break is a keyword in python which is used to bring the program control out of the loop. The break statement breaks the loops one by one, i.e., in the case of nested loops, it breaks the inner loop first and then proceeds to outer loops. In other words, we can say that break is used to abort the current execution of the program and the control goes to the next line after the loop.

The break is commonly used in the cases where we need to break the loop for a given condition. The syntax of the break statement in Python is given below.

**Syntax:**

1. #loop statements
2. **break**;

## Example 1 : break statement with for loop

**Code**

1. # break statement example
2. my\_list = [1, 2, 3, 4]
3. count = 1
4. **for** item **in** my\_list:
5. **if** item == 4:
6. **print**("Item matched")
7. count += 1
8. **break**
9. **print**("Found at location", count)

**Output:**

Item matched

Found at location 2

In the above example, a list is iterated using a for loop. When the item is matched with value 4, the break statement is executed, and the loop terminates. Then the count is printed by locating the item.

### **Example 2 : Breaking out of a loop early**

**Code**

1. # break statement example
2. my\_str = "python"
3. **for** char **in** my\_str:
4. **if** char == 'o':
5. **break**
6. **print**(char)

**Output:**

p

y

t

h

When the character is found in the list of characters, break starts executing, and iterating stops immediately. Then the next line of the print statement is printed.

### **Example 3: break statement with while loop**

**Code**

1. # break statement example
2. i = 0;
3. **while** 1:
4. **print**(i," ",end=""),
5. i=i+1;
6. **if** i == 10:
7. **break**;
8. **print**("came out of while loop");

**Output:**

0 1 2 3 4 5 6 7 8 9 came out of while loop

It is the same as the above programs. The while loop is initialised to True, which is an infinite loop. When the value is 10 and the condition becomes true, the break statement will be executed and jump to the later print statement by terminating the while loop.

### **Example 4 : break statement with nested loops**

**Code**

1. # break statement example
2. n = 2
3. **while** True:
4. i = 1
5. **while** i <= 10:
6. **print**("%d X %d = %d\n" % (n, i, n \* i))
7. i += 1
8. choice = int(input("Do you want to continue printing the table? Press 0 for no: "))
9. **if** choice == 0:
10. **print**("Exiting the program...")
11. **break**
12. n += 1
13. **print**("Program finished successfully.")

**Output:**

2 X 1 = 2

2 X 2 = 4

2 X 3 = 6

2 X 4 = 8

2 X 5 = 10

2 X 6 = 12

2 X 7 = 14

2 X 8 = 16

2 X 9 = 18

2 X 10 = 20

Do you want to continue printing the table? Press 0 for no: 1

3 X 1 = 3

3 X 2 = 6

3 X 3 = 9

3 X 4 = 12

3 X 5 = 15

3 X 6 = 18

3 X 7 = 21

3 X 8 = 24

3 X 9 = 27

3 X 10 = 30

Do you want to continue printing the table? Press 0 for no: 0

Exiting the program...

Program finished successfully.

# **Python continue Statement**

Python continue keyword is used to skip the remaining statements of the current loop and go to the next iteration. In Python, loops repeat processes on their own in an efficient way. However, there might be occasions when we wish to leave the current loop entirely, skip iteration, or dismiss the condition controlling the loop.

We use Loop control statements in such cases. The continue keyword is a loop control statement that allows us to change the loop's control. Both Python while and Python for loops can leverage the continue statements.

**Syntax:**

1. **continue**

### **Python Continue Statements in for Loop**

Printing numbers from 10 to 20 except 15 can be done using continue statement and for loop. The following code is an example of the above scenario:

PlayNext

Mute

Current TimeÂ 0:03

/

DurationÂ 18:10

Loaded: 3.67%

Â

Fullscreen

**Code**

1. # Python code to show example of continue statement
3. # looping from 10 to 20
4. **for** iterator **in** range(10, 21):
6. # If iterator is equals to 15, loop will continue to the next iteration
7. **if** iterator == 15:
8. **continue**
9. # otherwise printing the value of iterator
10. **print**( iterator )

**Output:**

10

11

12

13

14

16

17

18

19

20

**Explanation:** We will execute a loop from 10 to 20 and test the condition that the iterator is equal to 15. If it equals 15, we'll employ the continue statement to skip to the following iteration displaying any output; otherwise, the loop will print the result.

### **Python Continue Statements in while Loop**

**Code**

1. # Creating a string
2. string = "JavaTpoint"
3. # initializing an iterator
4. iterator = 0
6. # starting a while loop
7. **while** iterator < len(string):
8. # if loop is at letter a it will skip the remaining code and go to next iteration
9. **if** string[iterator] == 'a':
10. **continue**
11. # otherwise it will print the letter
12. **print**(string[ iterator ])
13. iterator += 1

**Output:**

J

v

T

p

o

i

n

t

**Explanation:** We will take a string "Javatpoint" and print each letter of the string except "a". This time we will use Python while loop to do so. Until the value of the iterator is less than the string's length, the while loop will keep executing.