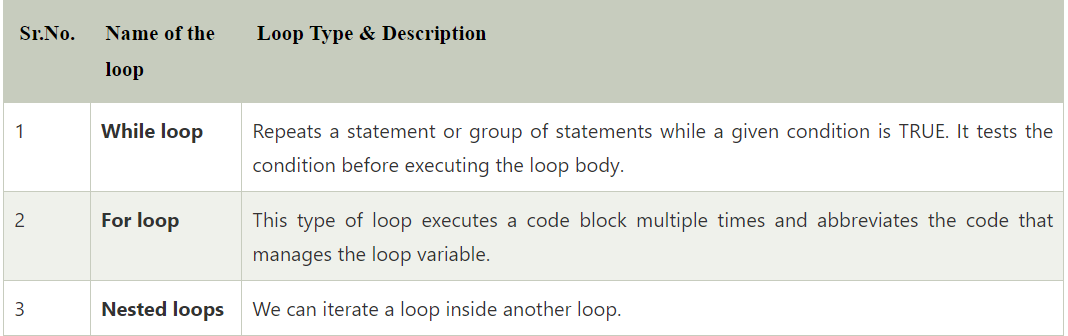
**Day 3**

Python Loops

Python programming language provides two types of Python loops checking time. In this article, we will look at Python loops and understand their working with the help of example – **For loop** and **While loop** to handle looping requirements. Loops in Python provides three ways for executing the loops.

While all the ways provide similar basic functionality, they differ in their syntax and condition-checking time. In this article, we will look at Python loops and understand their working with the help of examples.



Python for loop

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

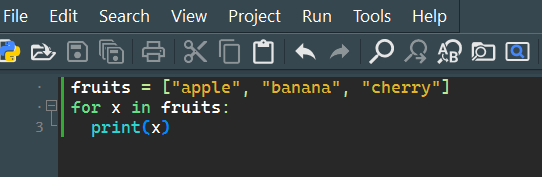
This is less like the for keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages.

With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.

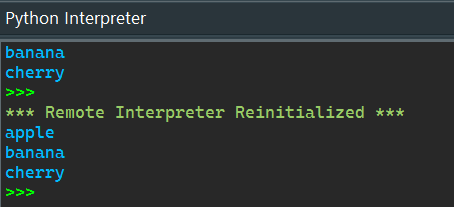
Example of Python for Loop

Print each fruit in a fruit list:

**Code**



**Output:**

****

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

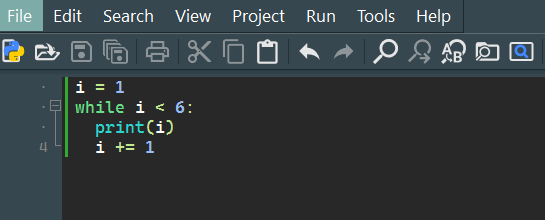
In this article, we are discussing while loops in Python. The Python while loop iteration of a code block is executed as long as the given Condition, i.e., conditional\_expression, is true.

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

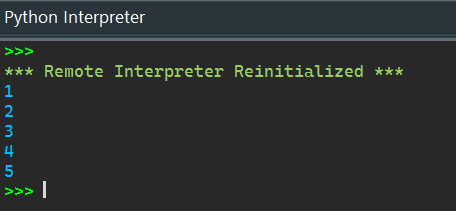
Example

**Print i as long as i is less than 6:**

**Code**

****

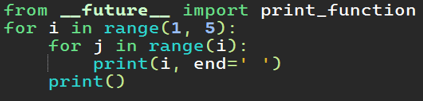
**Output:**

****

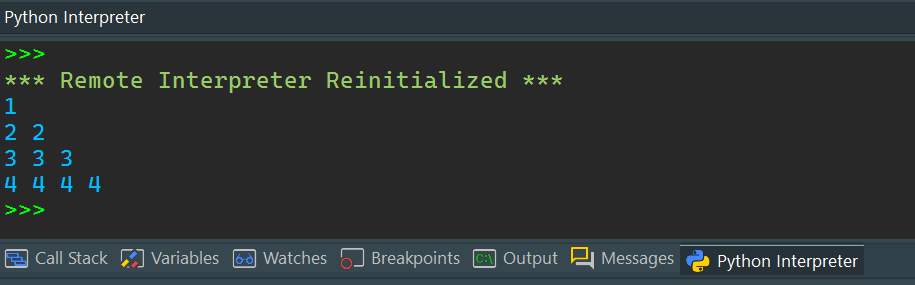
Nested Loops

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.

**Code**



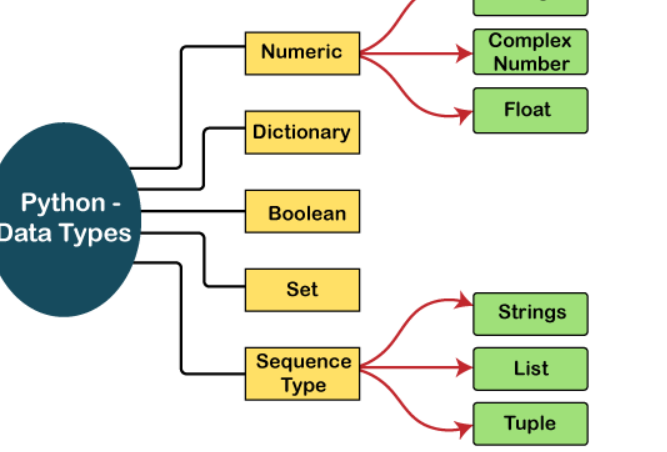
**Output:**



Data Types

Python Data Types

Every value has a datatype, and variables can hold values. Python is a powerfully composed language; consequently, we don't have to characterize the sort of variable while announcing it. The interpreter binds the value implicitly to its type.

****

Python Set

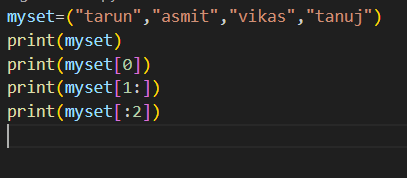
A Python set is the collection of the unordered items. Each element in the set must be unique, immutable, and the sets remove the duplicate elements. Sets are mutable which means we can modify it after its creation.

Unlike other collections in Python, there is no index attached to the elements of the set, i.e., we cannot directly access any element of the set by the index. However, we can print them all together, or we can get the list of elements by looping through the set.

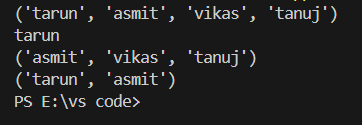
Creating a set

The set can be created by enclosing the comma-separated immutable items with the curly braces {}. Python also provides the set() method, which can be used to create the set by the passed sequence.

EXAMPLE:



OUTPUT:



Python String

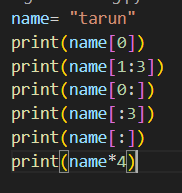
Till now, we have discussed numbers as the standard data-types in Python. In this section of the tutorial, we will discuss the most popular data type in Python, i.e., string.

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.

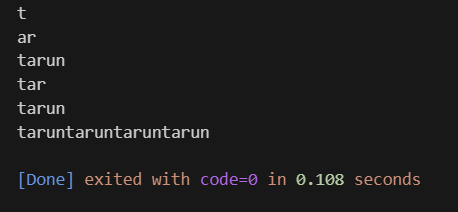
Each character is encoded in the ASCII or Unicode character. So we can say that Python strings are also called the collection of Unicode characters.

In Python, strings can be created by enclosing the character or the sequence of characters in the quotes. Python allows us to use single quotes, double quotes, or triple quotes to create the string.

EXAMPLE:



OUTPUT:



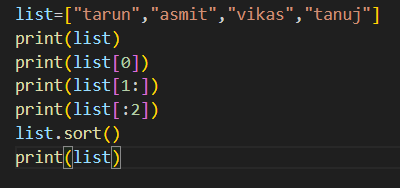
Python List

In Python, the sequence of various data types is stored in a list. A list is a collection of different kinds of values or items. Since Python lists are mutable, we can change their elements after forming. The comma (,) and the square brackets [enclose the List's items] serve as separators.

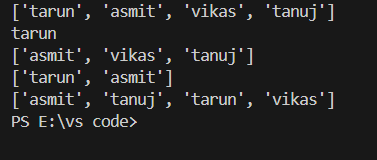
Although six Python data types can hold sequences, the List is the most common and reliable form. A list, a type of sequence data, is used to store the collection of data. Tuples and Strings are two similar data formats for sequences.

Lists written in Python are identical to dynamically scaled arrays defined in other languages, such as Array List in Java and Vector in C++. A list is a collection of items separated by commas and denoted by the symbol [].

*EXAMPLE:*



*OUTPUT:*

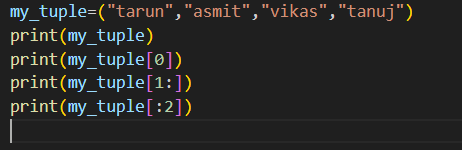


Python Tuples

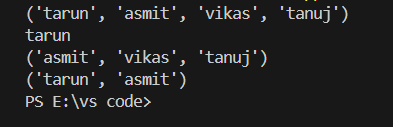
A comma-separated group of items is called a Python triple. The ordering, settled items, and reiterations of a tuple are to some degree like those of a rundown, but in contrast to a rundown, a tuple is unchanging.

The main difference between the two is that we cannot alter the components of a tuple once they have been assigned. On the other hand, we can edit the contents of a list.

**Example**



OUTPUT:



Python Dictionary

Dictionaries are a useful data structure for storing data in Python because they are capable of imitating real-world data arrangements where a certain value exists for a given key.

The data is stored as key-value pairs using a Python dictionary.

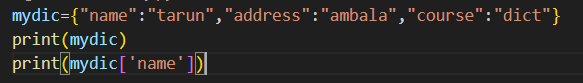
* This data structure is mutable
* The components of dictionary were made using keys and values.
* Keys must only have one component.
* Values can be of any type, including integer, list, and tuple.

A dictionary is, in other words, a group of key-value pairs, where the values can be any Python object. The keys, in contrast, are immutable Python objects, such as strings, tuples, or numbers. Dictionary entries are ordered as of Python version 3.7. In Python 3.6 and before, dictionaries are generally unordered.

Creating the Dictionary

Curly brackets are the simplest way to generate a Python dictionary, although there are other approaches as well. With many key-value pairs surrounded in curly brackets and a colon separating each key from its value, the dictionary can be built. (:). The following provides the syntax for defining the dictionary.

EXAMPLE:



OUTPUT:

