Cloud, APIs and Alerts > Introduction Python

Lists in Python

Introduction to Data Types

Now that we have learnt to write different types of Python programs, let us learn about the fundamentals of, types of data allowed in Python. This is extremely useful for any programmer as a good knowledge of data structures helps you make the choice of the right data format to solve your programming issues.

All programming languages allow us to store different types of data and then use the data to get various results based on the kind of programs we write. Most programming languages vary between each other based on the type of data they allow programmers to store and what functionality they provide to manipulate this data. Python is no exception to this. Python provides you with various data types to store different types of data and ways to manipulate this data. The major data types in Python are:

1. Numbers
2. Strings
3. Lists
4. Tuples
5. Dictionary

We have used numbers and strings in some of the programs that we have already written. These are the basic data types as they are similar to the data we use in our day to day lives and are supported by most programming languages. Lists, tuples and dictionaries are compound data types, which can be created by a combination of basic data types as well as compound data types.

Introduction to lists

In this topic, we will learn more about lists. You can imagine a list to be similar to a bucket. When you want to group various items together due to some similar properties, you would put all of them in one bucket. Below are a few examples of lists:

```
countries = ['India', 'USA', 'Australia', 'South Africa', 'Germany']
flowers = ['rose', 'tulip', 'orchid', 'sunflower']
colors = ['green', 'red', 'blue', 'black', 'yellow']
```


As you can see, we have created 3 lists. The lists are created based on similarity such as first one is a list of countries, second one is types of flowers and the third one is colors.

The syntax for lists is that they are surrounded by square brackets and all items in the list are separated by a comma.

Indexes in Python

An important point to note about Python is the indexes in lists are counted from the number 0 and not 1. What do I mean by this? In the list of countries above, there are 5 items. If I ask you what is the position of South Africa in the list, what will your answer be? Do you think it is 4 or do you think it is 3?

The correct answer to the question is 3. In real life, we always start counting from number 1 and hence the answer would be 4. But in most programming languages, we start counting from number 0 and hence the position of South Africa is 3.

The word position is referred to as index in programming and hence we shall use that from now onwards.

Accessing data from a list

If we want to access the value South Africa, then the syntax will be:

```
countries[3]
```

What if we want to print all the country names present in the list countries? What do you think the program would look like?

```
countries = ['India', 'USA', 'Australia', 'South Africa', 'Germany']
```

```
print(countries[0])
print(countries[1])
print(countries[2])
print(countries[3])
print(countries[4])
```

If you wrote a program as shown above, then you are correct. We will need to write the program as shown above to print all the country names present in the countries list. This was easy since there are only 5 country names in our list. However, imagine we had written the names of all the countries

present in the world, which would be 196 country names in total. How many print statements would we need to write in order to print all those names in that case? A whopping 196 print statements right! This would be really tiring and a waste of your precious time. Hence to access lists in a more easier way, Python provides us the with loops. We can use the `for` loop to go through each item in a list and carry out any action we want to, on the item. Lets see how we can use a `for` loop to print all the country names:

```
countries = ['India', 'USA', 'Australia', 'South Africa', 'Germany']

for country in countries:
    print country
```

Look at how short the program became! We reduced the number of lines of code from 6 to just 3 lines. The above `for` loop goes through each item in the countries list and stores it in the country variable and then prints it.

Updating data in a list

Updating the data in a list is quite simple if you know how to access the data. Let us say, we want to replace the country name at index 1 i.e. USA to Canada, we can do so by writing the syntax as shown below:

```
print("Country name at index 1 before updating is "countries[1])
countries[1] = "Canada"
print("Country name at index 1 after updating is "countries[1])
```

Adding data in a list

We can add data to a list in 2 ways, first one is by using the append function and second is by using the insert function. The `append` function always adds the data to the end of the list. Here is the syntax for `append`:

```
countries = ['India', 'USA', 'Australia', 'South Africa', 'Germany']
print("Countries before appending data ", countries)
countries.append("Mexico")
```



```
print("Countries after appending data ", countries)
```

The insert function adds the data at the specified index. For the insert function we need to specify the data and the index where the data should be insert in the list. Note that if the list index already contains some data, insert does not delete the older data, it only shifts the index of the older data by one. Also, if the index specified is more than the indexes present in the list then it will add the data to the end of the list. Here is the syntax of `insert`:

```
countries = ['India', 'USA', 'Australia', 'South Africa', 'Germany']
print("Countries before inserting data ", countries)
countries.insert(1, "Mexico")
print("Countries after inserting data ", countries)

print("Adding to an index which is greater that the total indexes in the list")
countries.insert(20, "Japan")
print("Countries after inserting data ",countries)
```

Deleting data from a list

We can delete the data from a list in 2 ways. First one is by using the del function and second is by using the pop function.

The `del` function allows us to delete the data at a specified index. The syntax for `del` is as shown below:

```
countries = ['India', 'USA', 'Australia', 'South Africa', 'Germany']
print("Countries before deleting data at index 1 ", countries)
del countries[1]
print("Countries after deleting data at index 1 ", countries)
```

The `pop` function allows us to delete the last data present in the list. The output of the pop function is the value of data deleted. The syntax of the pop function is shown below:

```
countries = ['India', 'USA', 'Australia', 'South Africa', 'Germany']
print("Countries before popping data ", countries)
deleted_data = countries.pop()
print("Countries after popping data ", countries)
print("The deleted data value was ", deleted_data)
```

- **NOTE:** * If there are no items to delete in the list then the pop function throws an error `IndexError: pop from empty list`. This means you are trying to delete data from a list that does not contain any data.

More built-in functions

Python provides some more built-in functions that we can use to do common tasks on lists. Some of them are

1. **len(list_name)**
2. **max(list_name)**
3. **min(list_name)**

Let us use these above functions in a program to understand how they work

```
exam_scores = [78, 90, 65, 83, 71]

print("The total number of exam scores is ", len(exam_scores))
print("The lowest exam score is ", min(exam_scores))
print("The maximum exam score is ", max(exam_scores))
```

Write the above program and see what output you get. In this program, we have created a list of exam scores that contains the marks obtained in 5 subjects. The `len` function will print the total count of items present in the exam_scores list. The `min` function prints the item with the least value in the list. In this case it will print the marks of the subject with the lowest score i.e. 65. The `max` function will print the marks of the subject with the highest score i.e. 90.

