**Lists**

A list type is a container that holds a number of values in sequence. Each value within the list is separated with a comma and is assigned a number known as its index (or position) in the sequence. The first index is 0.

*A list containing integers:*

**list1=[1, 0, 5, -5, 10]**

*To access an item in a list simple reference its index:*

**list1[0]** *Accessing the value in the 1st position of the list (index 0).*

*Print the value in index 4:*

**print(list1[4]) *Result:* 10**

*A list containing strings:*

**list2=['a ', 'apple', 'd', 'Hello, World!']**

*Print the item in index 3:*

**print(list2[3]) *Result:* Hello, World!**

*Print the first character (index 0) of the string in index 3.*

**print(list2[3][0]) *Result:* H**

*A list containing Booleans (a Boolean is a true or false value):*

**list3=[True, False, False, True]**

**print(list3[1]) *Result:* False**

*A list containing other lists:*

**list4=[[0,1,2,3],['a','b','apple'],[True,False]]**

*Explain the following:*

**print(list4[0][1], list4[2][1])**

*Explanation: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

**Slicing a List**

Slicing a list is to extracting a subsequence from a list.

*Recreate the first list we created above:*

**list1=[1, 0, 5, -5, 10]**

*Let’s extract and print the first 2 items from the list (above):*

**print(list1[0:2]) *Result:* [1, 0]**

*1st index Last index*

The above slicing statement states commence at index **0** and end at index **2**. **Wait a minute**, we did say the first 2 values, not 3 values, so why did we state **[0:2]**? Note the end index is one more that the actual end. The item in the last index position is not displayed.

As we began to extract from the beginning of the list, we do not actually have to specify index 0. We could have just state **print(list1[:2]).**

*Extract and print the first 4 items from the list:*

**print(list1[:4]) *Result:* [1, 0, 5, -5]**

*Extract and print the items in index 1 and 2:*

**print(list1[1:3]) *Result:* [0, 5]**

*Extract the entire list:*

**print(list1[:]) *Result:* [1, 0, 5, -5, 10]**

This is fun, let’s keep extracting. We know that if we want to commence the extraction from the start of the list, we do not need to specify index 0, recal the example we discussed earlier with the statement **print(list1[:2])**. This statement will extract and print the first 2 items in the list. With this in mind, what do you think the following statement will extract and print? **print(list1[2:])**.

*Try it:*

**print(list1[-2:]) *Result:* [-5, 10]**

*What about this:*

**print(list1[-3:]) *Result:* [5, -5, 10]**

*And finally:*

**print(list1[:-2]) *Result:* [1, 0, 5]**

Note: the end of the list is minus 1 (-1) not minus 0 (-0)

**Slicing a String**

Slicing a string is similar to slicing a list.

**quote="Don't Worry be Happy"**

*A closer look at the above string:*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **quote=** | **D** | **o** | **n** | **‘** | **t** |  | **W** | **o** | **r** | **r** | **y** |  | **b** | **e** |  | **H** | **a** | **p** | **p** | **y** |
| **Index:** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **12** | **14** | **15** | **16** | **17** | **18** | **19** |
|  |  |  |  |  |  |  |  |  |  |  |  | **-9** | **-8** | **-7** | **-6** | **-5** | **-4** | **-3** | **-2** | **-1** |

*Try the following:*

**print (quote[6]) *Result:* W**

**print (quote[6:8]) *Result:* Wo**

**print (quote[6:11]) *Result:* Worry**

**print (quote[:5]) *Result:* Don’t**

**print (quote[15:]) *Result:* Happy**

**print (quote[-5:]) *Result:* Happy**

**print ('S' + quote[7:11]) *Result:* Sorry**

*Can you identify what is happening with the following statements?*

**print (quote[::2]) *Result:* Dntwryb ap**

**print (quote[::3]) *Result:* D'wrbhp**

*What will happen when we run the following statement?*

**print ('L' + quote[7:11] + ' ' + quote[12:])**

Try it and explain the output?

Result: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Explanation: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Comments**

Comments are notes used to describe what the code does. There are two types of comments used in Python, single-line and multi-line comments.

* Single-line comments begin with the hash **#** character.

**# This is a comment.**

* Multi-line comments begin **‘‘‘** and end with **’’’**.

**‘‘‘ This is an example of**

**a multi-line comment that**

**spans multiple lines. ’’’**

The Python interpreter ignores all comments.

**The Return Statement**

The **return** statement is used to exit a function and recommence the point of execution where the function was called. The return statement can also return data. To understand the return statement, consider the following:

|  |  |
| --- | --- |
| **# returnNothing.py**  **def returnString():**  **a='Apple'**  **return**  **returnString()**  **b=returnString()**  **print(b)** | **# returnSomething.py**  **def returnString():**  **a='Apple'**  **return a**  **returnString()**  **b=returnString()**  **print(b)** |

Try it, what is the output:

|  |  |
| --- | --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

The return statement passes back data to the caller. A return statement with no arguments will return **None**.

What is the difference between the following programs?

|  |  |
| --- | --- |
| **def returnString():**  **a='Apple'**  **return a**  **returnString()**  **b=returnString()**  **print(b)** | **def returnString():**  **return 'Apple'**  **print(returnString())** |

*Explanation: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

Explain the difference between the following:

|  |  |
| --- | --- |
| **def returnString(num1, num2):**  **result = num1+num2**  **return result**  **print(returnString(10,20))** | **def returnString(num1, num2):**  **return num1+num2**  **print(returnString(10,20))** |

*Explanation: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*