

Lists

Remix CS 2019-20

What are Lists?

- Lists are a way of storing sequences of data in your programs.
- Variables only hold 1 item, but an array can have as many items as you want added to in sequence and then removed as you wish.
- The operations that add and remove items from lists are called **Methods**.

List Syntax

Lists are denoted using square brackets

| Code | Output |
|---|------------------------------------|
| <pre>1 my_list = [1, "hello", 3.2, True] 2 3 print(my_list)</pre> | <pre>[1, 'hello', 3.2, True]</pre> |

Items in list are separated by commas

Note that an empty list (a list with no items in it) can be created by using square brackets with nothing in them such as:

`x = []`

Indexing Lists

```
Code
1 my_list = [1, 'a', 9.2, "List-Item"]
```

0th Item 1st Item 2nd Item 3rd Item

- An item in a list can be referenced by its index.
- Indexing for lists starts at zero (the first item is index 0) and then continues for every item in the list.

Accessing Items in a List by their Index

The index is surrounded by square brackets

The name of the list comes first followed by the index

| Code | Output |
|-------------------------|--------|
| 1 my_list = [1,2,3,4,5] | 1 |
| 2 | 3 |
| 3 item1 = my_list[0] | |
| 4 print(item1) | |
| 5 | |
| 6 item2 = my_list[2] | |
| 7 print(item2) | |

Negative Indices in Lists

| Code | Output |
|---|----------------|
| <pre>1 my_list = ['a', 'b', 'c', 'd'] 2 3 item1 = my_list[-1] 4 print(item1) 5 item2 = my_list[-2] 6 print(item2)</pre> | <pre>d c</pre> |

Querying the item in a list at the index of -1 will give you the last element of the list, and the querying the item in a list at the index of -2 will give you the second to last element, etc, etc.

Finding the Index of an Element in a List

The index method returns the first instance of the element if the element is in the list multiple times

The index method returns the index of the element in parentheses

| Code | Output |
|--|--------|
| <pre>1 a = [1, 2, 17, 2] 2 3 index = a.index(2) 4 print(index)</pre> | 1 |

Adding Elements to Lists

Note that the item is added to the end of the list

Code

```
1 my_list = [1, 'a']
2 print(my_list)
3
4 my_list.append(9.2)
5 my_list.append('List-Item')
6
7 print(my_list)
```

Output

```
[1, 'a']
[1, 'a', 9.2, 'List-Item']
```

The append method adds the item in parentheses to the list

Removing Elements from Lists

If the same 2 elements in a list have the same value and you call `remove` or `index` on one of the elements it will use the first occurrence of the element

| Code | Output |
|---|---|
| <pre>1 my_list = [1, 'a', 9.2, "List-Item"] 2 print(my_list) 3 4 my_list.remove(9.2) 5 print(my_list)</pre> | <pre>[1, 'a', 9.2, 'List-Item'] [1, 'a', 'List-Item']</pre> |

The `remove` method removes the item in parentheses from the list

The item is removed from wherever it is located in the list

Removing Elements from Lists by Index


| Code | Output |
|---|----------------------------|
| <pre>1 my_list = ['a', 'b', 'c', 'd'] 2 3 my_list.pop(2) 4 print(my_list)</pre> | <pre>['a', 'b', 'd']</pre> |

The pop method removes the item at the index specified in parentheses

'c', the item at index 2 (the third element) is removed from the list

Slicing Lists

Slicing allows you to take a section of a list rather than the whole list or a specific element

| Code | Output |
|---|-----------------------|
| <pre>1 my_list = ['a', 'b', 'c', 'd'] 2 3 subset = my_list[1:3] 4 print(subset)</pre>  | <pre>['b', 'c']</pre> |

To slice a list use the name of the list followed by square brackets containing the starting index of the slice, then a semicolon, and then the index you want to stop the slice to continue up to but not include.

More about Slicing

| Code | Output |
|--|-----------------------|
| <pre>1 my_list = ['a', 'b', 'c', 'd'] 2 3 subset = my_list[-3: -1] 4 print(subset)</pre> | <pre>['b', 'c']</pre> |

Slicing can also be done with negative indices, where the slice starts at the first index and continues up to, but not including the second index

More about Slicing

| Code | Output |
|--|-----------------------|
| <pre>1 my_list = ['a', 'b', 'c', 'd'] 2 3 subset = my_list[2:] 4 print(subset)</pre> | <pre>['c', 'd']</pre> |

Omitting the second index means the slice will start at the first index and continue to the end of the list.

More about Slicing

| Code | Output |
|---|-----------------------|
| <pre>1 my_list = ['a', 'b', 'c', 'd'] 2 3 subset = my_list[-2:] 4 print(subset)</pre> | <pre>['c', 'd']</pre> |

Similarly using a negative index and omitting the second index means the slice will start at the whatever the index is from the end and continue to the end of the list.

More about Slicing

| Code | Output |
|--|-----------------------|
| <pre>1 my_list = ['a', 'b', 'c', 'd'] 2 3 subset = my_list[:2] 4 print(subset)</pre> | <pre>['a', 'b']</pre> |

Omitting the first index means the slice will start at the beginning of the list and continue to up but not including the second (and only) index.

More about Slicing

| Code | Output |
|---|-----------------------|
| <pre>1 my_list = ['a', 'b', 'c', 'd'] 2 3 subset = my_list[:-2] 4 print(subset)</pre> | <pre>['a', 'b']</pre> |

Similarly omitting the first index and using a negative index as the second means the slice will go up to, but not include whatever the index is from the end

Practice Exercises

Using the list [1, 2, 8, -3, 12, -21, 34, 17, 2] ...

1. Print whether the first and last elements are the same.
2. Print the last 3 elements of the list.
3. Print the slice of the list [-21, 34, 17] two different ways.
4. Remove the first element from the list.
5. Repeat exercise #1.
6. Add a string to the end of the list.
7. Remove the number 12 from the list.
8. Find the index of the number 8.

The “in” Keyword

- There are many cases in which it is useful to check if a list contains a certain element.
- This can be done with the “in” keyword
- The “in” keyword creates a boolean that is True if the element is in the list and False if it is not

“In” Keyword Syntax

The in keyword can be used to create a boolean

| Code | Output |
|--|-----------------------|
| <pre>1 a = [1, 2, 8, -3, 12, -21, 34, 17, 2] 2 3 result = 8 in a 4 print(result) 5 6 if 6 in a: 7 print(True) 8 else: 9 print(False)</pre> | <pre>True False</pre> |

The in keyword can be used as part of the boolean in a conditional statement

Nested Lists

Lists can be placed within lists

Code

```
1 my_list = [1, [2, 3, 4], [], [5, [6, 7]]]
2 print(my_list)
3
4 item = my_list[3][1][0]
5 print(item)
6
7 my_list[3].append(8)
8 print(my_list)
```

After accessing a nested list you can then call methods on it specifically

Output

```
[1, [2, 3, 4], [], [5, [6, 7]]]
6
[1, [2, 3, 4], [], [5, [6, 7], 8]]
```

The first index accesses the third element of the outer list, which is a list and can then be accessed with indexing to retrieve another list, which can be indexed to access the element with the value 6

Practice Exercises

Using the list ['a' , [], ['c' , 'd'], ['e' , ['f']]] ...

1. Check if the third element of the outermost list contains 'c'.
2. Add the number 27 to the empty list.
3. Access and print the element 'e' using indexing.
4. Add 'g' to the nested list containing 'f'.