Statistical Programming with Python

Pepe García

2020-04-20

Statistical Programming with Python

Lists & Iteration

Plan for today

Lists

For loops

While loops

Lists

Lists are sequences of values

Constructing Lists

We construct lists with the brackets [] syntax:

```
[1, 2, 3, 4, 5]
["hello", "dolly"]
[]
[1, "hello", 2, "dolly", 3]
```

Constructing Lists

Accessing list elements

We use **square brackets** to access elements by their **index**.

indices in lists start by **0**, not 1.

```
words = ["hello", "dolly"]
words[0]
# "hello"
words[1]
# "dolly"
```

Accessing List elements

Operators on lists

As with strings, + and * operators work with lists too!

Operators on lists

Mutating lists

Lists are mutable values, and they provide functionality to add, delete, and update elements

Updating elements in the list

To update an element inside the list, we use a syntax similar to the one for declaring variables, but using the brackets and the index we refer to.

```
numbers = [1,2,4]
numbers[2] = 3
print(numbers)
```

Updating elements in the list

Appending elements to the list

To add a new element to the end of the list we use the append() method on it.

```
numbers = [1,2,3]
numbers.append(4)
print(numbers)
```

Appending elements to the list

Inserting elements in the list

There's an alternative way of adding new elements to the list, and it's using the insert() method on it:

```
words = ["hello","my","friends"]
words.insert(2, "dear")
print(words)
```

The difference between this and append is that with insert we can choose where to put it by using the target index

Inserting elements in the list

Removing elements from the list

In order to remove an element from a list, we should use the .pop() method, and pass the index of the element we want to remove

```
words = ["hello","my","friend"]
words.pop(1)
print(words)
```

Removing elements from the list

For loops

For loops are simpler than while loops. They iterate over elements in a list.

For loops

```
for <variable_name> in <list>:
    <body>
```

For loops

Iteration

Iteration is the act of repeating a process. In Python we express iteration with the **while** statement

While

1. Evaluate the condition
2. If the condition is False, exit while and go to next statement
3. If condition is true, execute body. Then go to step 1.

Stopping a loop

You can always stop a loop using the break keyword

```
while True:
    if im_bored_of_iterating:
        break
    else:
        print("i'm iterating!")
```

Infinite loops

Infinite loops are loops whose condition never evaluates to False. This means that they'll run forever (unless we break them)

Recap

Use lists to store collections of values

Use mutation operations on list to add, remove, or update elements in the list

Use for loops to iterate over elements in the list

Exercises

1. Create a function that returns a list of numbers from 0 to 500
2. Create a function that takes a list of numbers (you can use the one you created in the previous exercise) and returns the sum of all of them
3. Investigate the ~range()~ function. After you've used it, create a function that receives a number as parameter and prints all numbers from it to zero (using a for loop).
4. Create a function that takes a list of numbers and returns the maximum value among them
5. Create a function that takes a list of numbers and returns the minimum value among them

Recommended literature

https://www.py4e.com/html3/05-iterations

https://www.py4e.com/lessons/lists