For Loops in Python

February 17, 2021

Administrative Notes

“For” loops

Allow us to do some things with some or all the elements of a list

Two cases to consider:

- You want to do something with each element in the list, exactly once
  - Called a “for each” loop
- You want to do something with some of the elements in the list, but not necessarily all
  - Called a “for i” loop

For each loops - doing the same thing with each element in a list

Start with a list:

grocery_list = [“Milk”, “Eggs”, “Cereal”, “Coffee”, “Apples”, “Strawberries”, “Broccoli”, “Cucumber”, “Tomatoes”, “Green Onions”]

We want to print the list, one item on a line, so that we can send someone else to the store.

Use a “for each” loop.

Remember that reserved word “in”? We’ll use it here.

Example

```
for item in grocery_list:

    print(item)
```

"item in grocery_list" is a boolean conditional. The colon ends the conditional. This continues to execute as long as there are more items.

Indent!!! Just like with if-else. White space is important in Python

What this means:

- "Item" is a new variable; not used in the program
- Python automatically creates this to be the same type as the elements of the list
  - What if not all elements are the same type? We'll get to that in a minute
- "Item" is given the value of each element in the list, one at a time, and the code is executed for each value of "item"

“For i” loops

Used when you may not want to iterate over every item in the object

Syntax:

```
for iterator in range(a, b, c): # see the next slide

    #do something
```

“Iterator” is just a variable that tracks where you are in the list or other object. It is most often an integer, although it doesn’t absolutely have to be. “I” is often used.

The value of “iterator” does NOT have to be pre-set.

Lists with different element types

```
mixed_list = [“eggs”, 12, “milk”, 128.0]
for item in mixed_list:
        print(item)
```

- Python automatically changes the type of item to match the value of each element

```
#here’s a test

for item in mixed_list:
        if item == str(item):
                print(item)
```

- This test decides if a value of item is a string, or another type. The == will only be true if “item” is already a string.

“For i” loops

- Called this because “i” is often used as the index variable. But you don’t have to use “i”
- Used to loop through an object - e.g., a list - and optionally skip some elements

Home on the range()

range() is a function that takes 3 parameters:

1. A starting integer
2. An ending integer
3. A hop size

Range will give you back all the integers from (1) to (2) hopping by (3) each time. Also, we have to wrap range in list() to see all the numbers at once.

Let’s try it!

Note: doesn’t have to be an integer, but it’s really confusing if it’s not.

Not all the arguments are needed

If you give one argument it’s the end of the range.

- What are the assumed start and hop size?

If you give two it’s the start and end of the range.

- What is the assumed hop size?

For loops

Here's how a "for i" loop iterates through the items of a list

```
lizards = [“gecko”, “iguana”, “komodo dragon”, “chameleon”]
for i in range(len(lizards)):
    print(lizards[i])
```

Modifying your list

You can modify the contents of a list you're iterating through, while you're iterating through it.

An example:

```
grade_list = [98, 92.5, 123, 199.8]
for i in range(len(grade_list)):
        grade_list[i] *= grade_list[i]
print(grade_list)
```

Let's try it!

Let's try to take a list of integers and increase each integer by one.

First we'll try it with a "for each" loop. What happens?

Next we'll try it with a "for i" loop. Any better?

Some questions:

1. How can we print the even numbers between two integers x and y with a for loop?
2. How can we use range to go backwards through a list?
3. Print all the elements of a list with their index!

range(len(integer_list)-1,0, -1)

For loops and strings

A string is zero or more characters treated as a single entity

Sounds kind of like a list, where each element of the list is one character long.

- You can treat it that way

```
word = “supercalifragilisticexpialidocious”:
for i in word:
    print(i)
```

Or:

```
word = “supercalifragilisticexpialidocious”:
for i in range(len(word)):
    print(word[i])
```

Some Examples

The “states” list from Monday

```
states = ["Alabama","Alaska","Arizona","Arkansas","California","Colorado",
"Connecticut","Delaware","Florida","Georgia","Hawaii","Idaho","Illinois",
  "Indiana","Iowa","Kansas","Kentucky","Louisiana","Maine","Maryland",
  "Massachusetts","Michigan","Minnesota","Mississippi","Missouri","Montana",
  "Nebraska","Nevada","New Hampshire","New Jersey","New Mexico","New York",
  "North Carolina","North Dakota","Ohio","Oklahoma","Oregon","Pennsylvania",
  "Rhode Island","South Carolina","South Dakota","Tennessee","Texas","Utah",
  "Vermont","Virginia","Washington","West Virginia","Wisconsin","Wyoming"]
```

- Write a “for” loop that prints out each state that ends in “a”
- Write a “for” loop that prints out each state that ends in a vowel. The hard way; the medium way; and the easy way