Introduction

When doing data science, you need a way to organize your data so you can work with it efficiently. Python has many data structures available for holding your data, such as lists, sets, dictionaries, and tuples. In this tutorial, you will learn how to work with Python lists.

Motivation

In the Petal to the Metal competition, your goal is to classify the species of a flower based only on its image. (This is a common task in computer vision, and it is called image classification.) Towards this goal, say you organize the names of the flower species in the data.

One way to do this is by organizing the names in a Python string.

flowers = "pink primrose,hard-leaved pocket orchid,canterbury bells,sweet pea,english marigold,tiger lily,moon orchid,bird of paradise,monkshood,globe thistle"

print(type(flowers))

print(flowers)

<class 'str'>

pink primrose,hard-leaved pocket orchid,canterbury bells,sweet pea,english marigold,tiger lily,moon orchid,bird of paradise,monkshood,globe thistle

Even better is to represent the same data in a Python list. To create a list, you need to use square brackets ([, ]) and separate each item with a comma. Every item in the list is a Python string, so each is enclosed in quotation marks.

flowers\_list = ["pink primrose", "hard-leaved pocket orchid", "canterbury bells", "sweet pea", "english marigold", "tiger lily", "moon orchid", "bird of paradise", "monkshood", "globe thistle"]

print(type(flowers\_list))

print(flowers\_list)

<class 'list'>

['pink primrose', 'hard-leaved pocket orchid', 'canterbury bells', 'sweet pea', 'english marigold', 'tiger lily', 'moon orchid', 'bird of paradise', 'monkshood', 'globe thistle']

At first glance, it doesn't look too different, whether you represent the information in a Python string or list. But as you will see, there are a lot of tasks that you can more easily do with a list. For instance, a list will make it easier to:

get an item at a specified position (first, second, third, etc),

check the number of items, and

add and remove items.

Lists

Length

We can count the number of entries in any list with len(), which is short for "length". You need only supply the name of the list in the parentheses.

# The list has ten entries

print(len(flowers\_list))

10

Indexing

We can refer to any item in the list according to its position in the list (first, second, third, etc). This is called indexing.

Note that Python uses zero-based indexing, which means that:

to pull the first entry in the list, you use 0,

to pull the second entry in the list, you use 1, and

to pull the final entry in the list, you use one less than the length of the list.

print("First entry:", flowers\_list[0])

print("Second entry:", flowers\_list[1])

# The list has length ten, so we refer to final entry with 9

print("Last entry:", flowers\_list[9])

First entry: pink primrose

Second entry: hard-leaved pocket orchid

Last entry: globe thistle

Side Note: You may have noticed that in the code cell above, we use a single print() to print multiple items (both a Python string (like "First entry:") and a value from the list (like flowers\_list[0]). To print multiple things in Python with a single command, we need only separate them with a comma.

Slicing

You can also pull a segment of a list (for instance, the first three entries or the last two entries). This is called slicing. For instance:

to pull the first x entries, you use [:x], and

to pull the last y entries, you use [-y:].

print("First three entries:", flowers\_list[:3])

print("Final two entries:", flowers\_list[-2:])

First three entries: ['pink primrose', 'hard-leaved pocket orchid', 'canterbury bells']

Final two entries: ['monkshood', 'globe thistle']

As you can see above, when we slice a list, it returns a new, shortened list.

Removing items

Remove an item from a list with .remove(), and put the item you would like to remove in parentheses.

flowers\_list.remove("globe thistle")

print(flowers\_list)

['pink primrose', 'hard-leaved pocket orchid', 'canterbury bells', 'sweet pea', 'english marigold', 'tiger lily', 'moon orchid', 'bird of paradise', 'monkshood']

Adding items

Add an item to a list with .append(), and put the item you would like to add in parentheses.

flowers\_list.append("snapdragon")

print(flowers\_list)

['pink primrose', 'hard-leaved pocket orchid', 'canterbury bells', 'sweet pea', 'english marigold', 'tiger lily', 'moon orchid', 'bird of paradise', 'monkshood', 'snapdragon']

Lists are not just for strings

So far, we have only worked with lists where each item in the list is a string. But lists can have items with any data type, including booleans, integers, and floats.

As an example, consider hardcover book sales in the first week of April 2000 in a retail store.

hardcover\_sales = [139, 128, 172, 139, 191, 168, 170]

Here, hardcover\_sales is a list of integers. Similar to when working with strings, you can still do things like get the length, pull individual entries, and extend the list.

print("Length of the list:", len(hardcover\_sales))

print("Entry at index 2:", hardcover\_sales[2])

Length of the list: 7

Entry at index 2: 172

You can also get the minimum with min() and the maximum with max().

print("Minimum:", min(hardcover\_sales))

print("Maximum:", max(hardcover\_sales))

Minimum: 128

Maximum: 191

To add every item in the list, use sum().

print("Total books sold in one week:", sum(hardcover\_sales))

Total books sold in one week: 1107

We can also do similar calculations with slices of the list. In the next code cell, we take the sum from the first five days (sum(hardcover\_sales[:5])), and then divide by five to get the average number of books sold in the first five days.

print("Average books sold in first five days:", sum(hardcover\_sales[:5])/5)

Average books sold in first five days: 153.8

Your turn

Now it's your turn to practice creating and modifying lists.

Have questions or comments? Visit the course discussion forum to chat with other learners.