Physics Without Frontiers

The Abdus Salam International Centre for Theoretical Physics

Practical Programming
in Python

Inspired by 'Practical Programming' by Paul Gries, Jennifer Campbell, Jason Montojo

Lecture 8: Summary & Exercises
Lists

Storing collections of data, Mutability, Using Lists

> *"Should array indices start at 0 or 1? My compromise of 0.5 was rejected without, I thought, proper consideration."*
>
> – S. Kelly-Bootle

Lecture 8: Summary

In this lecture you learned the following:

- Lists are used to keep track of zero or more objects. The objects in a list are called items or elements. Each item has a position in the list called an index and that position ranges from zero to one less than the length of the list.
- Lists can contain any type of data, including other lists.
- Lists are mutable, which means that their contents can be modified.
- Slicing is used to create new lists that have the same values or a subset of the values of the originals.
- When two variables refer to the same object, they are called aliases.

Lecture 8: Exercises

When writing code, only use Python concepts that have been introduced in the lectures already.

Exercise 1:

Variable `kingdoms` refers to the list

```
['Bacteria', 'Protozoa', 'Chromista', 'Plantae', 'Fungi', 'Animalia']
```

Using `kingdoms` and either slicing or indexing with positive indices, write expressions that produce the following:

- The first item of `kingdoms`
- The last item of `kingdoms`
- The list `['Bacteria', 'Protozoa', 'Chromista']`
- The list `['Chromista', 'Plantae', 'Fungi']`
- The list `['Fungi', 'Animalia']`
- The empty list

Exercise 2:

Repeat the previous exercise using negative indices.

Exercise 3:

Variable `appointments` refers to the list

```
['9:00', '10:30', '14:00', '15:00', '15:30']
```

An appointment is scheduled for 16:30, so `'16:30'` needs to be added to the list.

- Using list method `append`, add `'16:30'` to the end of the list that `appointments` refers to.
- Instead of using `append`, use the `+` operator to add `'16:30'` to the end of the list that `appointments` refers to.
- You used two approaches to add `'16:30'` to the list. Which approach modified the list and which approach created a new list?

Exercise 4:

Variable `ids` refers to the list

```
[4353, 2314, 2956, 3382, 9362, 3900]
```

Using list methods, do the following:

- Remove `3382` from the list.
- Get the index of `9362`.
- Insert `4499` in the list after `9362`.
- Extend the list by adding `[5566, 1830]` to it.
- Reverse the list.
- Sort the list.

Exercise 5:

In this exercise, you'll create a list and then answer questions about that list.

- Assign a list that contains the atomic numbers of the six alkaline earth metals – beryllium (4), magnesium (12), calcium (20), strontium (38), barium (56), and radium (88) – to a variable called `alkaline_earth_metals`.
- Which index contains radium's atomic number? Write the answer in two ways, one using a positive index and one using a negative index.
- Which function tells you how many items there are in `alkaline_earth_metals`?
- Write code that returns the highest atomic number in `alkaline_earth_metals`. (Hint: use one of the built-in functions operating on lists.)

Exercise 6:

In this exercise, you'll create a list and then answer questions about that list.

- Create a list of temperatures in degrees Celsius with the values `25.2`, `16.8`, `31.4`, `23.9`, `28`, `22.5`, and `19.6`, and assign it to a variable called `temps`.
- Using one of the list methods, sort `temps` in ascending order.
- Using slicing, create two new lists, `cool_temps` and `warm_temps`, which contain the temperatures below and above 20 degrees Celsius, respectively.
- Using list arithmetic, recombine `cool_temps` and `warm_temps` into a new list called `temps_in_celsius`.

Exercise 7:

Complete the examples in the docstring and then write the body of the following function:

```
def same_first_last(L):
    """
    Return True iff the first item of L is the same as the last.

    Precondition:

        len(L) >= 2

    Examples:

        >>> same_first_last([3, 4, 2, 8, 3])
        True
        >>> same_first_last(['apple', 'banana', 'pear'])

        >>> same_first_last([4.0, 4.5])

    """
```

Exercise 8:

Complete the examples in the docstring and then write the body of the following function:

```
def is_longer(L1, L2):
    """
    Return True iff L1 is longer than L2.

    Examples:

        >>> is_longer([1, 2, 3], [4, 5])
        True
        >>> is_longer(['abcdef'], ['ab', 'cd', 'ef'])

        >>> is_longer(['a', 'b', 'c'], [1, 2, 3])

    """
```

Exercise 9:

Draw a memory model showing the effect of the following statements:

```
values = [0, 1, 2]
values[1] = values
```

Exercise 10:

Variable `units` refers to the nested list

```
[[ 'km', 'miles', 'league'], ['kg', 'pound', 'stone']]
```

Using units and either slicing or indexing with positive indices, write expressions that produce the following:

- The first item of `units` (the first inner list)
- The last item of `units` (the last inner list)
- The string `'km'`
- The string `'kg'`
- The list `['miles', 'league']`
- The list `['kg', 'pound']`

Exercise 11:

Repeat the previous exercise using negative indices.