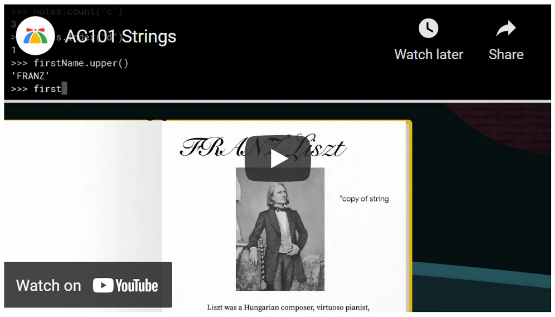
6.2. Strings and Lists

Throughout the first chapters of this book we have used strings to represent words or phrases that we wanted to print out. Our definition was simple: a string is simply some characters inside quotes. In this chapter we explore strings in much more detail.

Additionally, we explore lists, which are very much like strings but can hold different types.

6.2.1. Strings

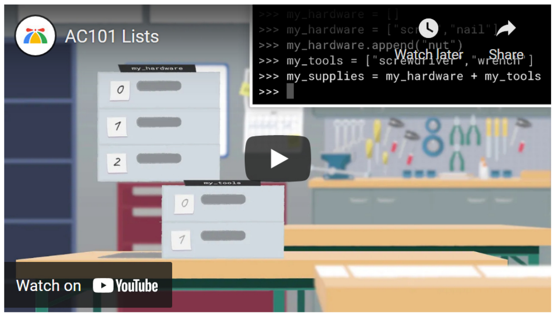


Activity: 1 -- Video: (stringintro)

Strings can be defined as sequential collections of characters. This means that the individual characters that make up a string are in a particular order from left to right.

A string that contains no characters, often referred to as the **empty string**, is still considered to be a string. It is simply a sequence of zero characters and is represented by '' or "" (two single or two double quotes with nothing in between).

6.2.2. Lists



Activity: 2 -- Video: (listintro)

A **list** is a sequential collection of Python data values, where each value is identified by an index. The values that make up a list are called its **elements**. Lists are similar to strings, which are ordered collections of characters, except that the elements of a list can have any type and for any one list, the items can be of different types.

There are several ways to create a new list. The simplest is to enclose the elements in square brackets (`[` and `]`).

```
[10, 20, 30, 40]
["spam", "bungee", "swallow"]
```

The first example is a list of four integers. The second is a list of three strings. As we said above, the elements of a list don't have to be the same type. The following list contains a string, a float, an integer, and another list.

```
["hello", 2.0, 5, [10, 20]]
```

Note

WP: Don't Mix Types!

You'll likely see us do this in the textbook to give you odd combinations, but when you create lists you should generally not mix types together. A list of just strings or just integers or just floats is generally easier to deal with.

6.2.3. Tuples

A **tuple**, like a list, is a sequence of items of any type. The printed representation of a tuple is a comma-separated sequence of values, enclosed in parentheses. In other words, the representation is just like lists, except with parentheses () instead of square brackets [].

One way to create a tuple is to write an expression, enclosed in parentheses, that consists of multiple other expressions, separated by commas.

```
julia = ("Julia", "Roberts", 1967, "Duplicity", 2009, "Actress", "Atlanta, Georgia")
```

The key difference between lists and tuples is that a tuple is immutable, meaning that its contents can't be changed after the tuple is created. We will examine the mutability of lists in detail in the chapter on Mutability.

To create a tuple with a single element (but you're probably not likely to do that too often), we have to include the final comma, because without the final comma, Python treats the `(5)` below as an integer in parentheses:

Save & Run 4/30/2021, 4:59:35 PM - 3 of 3 Show in CodeLens

```
t = (5,)
print(type(t))

x = (5)
print(type(x))

```

```
<class 'tuple'>
<class 'int'>
```

Activity: 3 -- ActiveCode (ac5_2_1)

Check your understanding

sequences-2-1: A list can contain only integer items.

[✓ A. False]

B. True

Check me Compare me

✓ Yes, unlike strings, lists can consist of any type of Python data.

Activity: 4 -- Multiple Choice (question5_2_1)

You have attempted 5 of 4 activities on this page

6.1. Introduction: Sequences">

oduction: Sequences">

6.3. Index Operator: Working with the Characters of a String">

✓ Completed. Well Done!

| Back to top