**Lists**

A list is simply the way in which we store information in Python.

* Lists begin and end with square brackets [ ].
* Each item [must, be, separated, with, a, comma].
* It is good practice to insert a space, after, every, comma.

heights = [61, 70, 67, 64]

And here is a list.

Lists can contain numbers and strings.

ints\_and\_strings = [1, 2, 3, "four", "five"]

This string contains integers and strings.

Strings can even contain Booleans:

mixed\_list\_common = ["Mia", 27, False, 0.5]

**Empty Lists**

Lists do not have to contain anything, some maybe empty.

This maybe because we plan to fill it later with another input.

empty\_list = []

**List Methods**

In Python, for a specific data-type there is built-in functionality that we can use to create, manipulate and delete our data. We call this built-in functionality a method.

For lists, methods will follow the form of list\_name.method().

Some methods will require an input value that will go between the parenthesis of the method.

One example is the .append() method which allows us to add an element to the end of a list.

append\_example = [ 'This', 'is', 'an', 'example']

append\_example.append('list')

print(append\_example)

When we use the .append() method, a new element is always added to the end of the list.

garden = ["Tomatoes", "Grapes", "Cauliflower"]

# Append a new element

garden.append("Green Beans")

print(garden)

In the above example, Green Beans is added to the end of our list.

**Growing a List: Plus (+)**

We can add multiple items to a list. In this instance we can use the + sign, also known as concatenation.

items\_sold\_new = items\_sold + ["biscuit", "tart"]

print(items\_sold\_new)

Remember that if we want to add a single element using +, we have to put it into a list with brackets.

my\_list + [4]

**Accessing List Elements**

In Python, we call the location of an element in a list its index. Python lists are zero-index. This means that the first element in a list has index 0.

This list:

calls = ["Juan", "Zofia", "Amare", "Ezio", "Ananya"]

A picture containing text, screenshot, font, number

Description automatically generatedHas the following index positions:

It is possible to select a single element from a list by using square brackets( [] ) and the index of the list item. If we wanted to select the third element from the list, we’d use calls[2]:

print(calls[2])

This would output the name Amare who is at index position 2.

**Note:** When accessing elements of a list, you must use an int as the index. If you use a float, you will get an error. This can be especially tricky when using division. For example print(calls[4/2]) will result in an error, because 4/2 gets evaluated to the float 2.0.

To solve this problem, you can force the result of your division to be an int by using the int() function. int() takes a number and cuts off the decimal point. For example, int(5.9) and int(5.0)will both become 5. Therefore, calls[int(4/2)] will result in the same value as calls[2], whereas calls[4/2]will result in an error.

**Accessing List Elements: Negative Index**

What if we wanted the last element in the list? We can use the index -1to select the last item of a list, even when we don’t know how many elements are in the list.

pancake\_recipe = ["eggs", "flour", "butter", "milk", "sugar", "love"]

print(pancake\_recipe[-1])

Look at this list. In this instance our print command would print love to the terminal.

Here are the following index numbers for our list:A picture containing text, screenshot, font, number

Description automatically generated

**Modifying List Elements**

Sometimes we may want to replace elements within a list.

garden = ["Tomatoes", "Green Beans", "Cauliflower", "Grapes"]

garden[2] = "Strawberries"

Notice here that we would be replacing Cauliflower which as index position 2 (list starts from base 0) with Strawberries.

You can also use negative indices!

garden[-1] = "Raspberries"

print(garden)

This would replace the final element in the list.

**Shrinking a list: Remove**

We can remove elements in a list using the .remove() method.

Perhaps we have a shopping list:

shopping\_line = ["Cole", "Kip", "Chris", "Sylvana"]

We can remove Chris with the .remove() method.

shopping\_line.remove("Chris")

Duplicate Elements

IF we use .remove() on a list that has duplicate elements, only the first instance of the matching element is removed.

# Create a list

shopping\_line = ["Cole", "Kip", "Chris", "Sylvana", "Chris"]

# Remove a element

shopping\_line.remove("Chris")

print(shopping\_line)

This would output: [‘Cole’, ‘Kip’, ‘Sylvana’, ‘Chris’]

**Two-Dimensional (2D) Lists**

Lists can contain other lists!

heights = [["Noelle", 61], ["Ava", 70], ["Sam", 67], ["Mia", 64]]

Sometimes we can find that a two-dimensional list is a good structure for representing grids such as games like tic-tac-toe.

#A 2d list with three lists in each of the indices.

tic\_tac\_toe = [

["X","O","X"],

["O","X","O"],

["O","O","X"]

]