**Strings**

Another useful data type is the **string**. A **string** can contain letters, numbers, and symbols.

name = "Ryan" age = "19" food = "cheese"

1. In the above example, we create a variable name and set it to the string value "Ryan".
2. We also set age to "19" and food to "cheese".

Strings need to be within quotes.

Do you remember how to declare and assign variables in Python? If not, refresh your memory [here](https://www.codecademy.com/courses/learn-python/lessons/python-syntax/exercises/welcome)!

There is no difference between using single quotes ' and double quotes ". However, sometimes it is helpful to use one or the other. If we want to include an apostrophe in our string, it would be smart to use double quotes to create the string like "I'm a string". If we want to use quotes in the string, we might want to create the string with single quote like 'The man screamed "I love Python!" so that everyone could hear.'

**String methods**

Great work! Now that we know how to store strings, let's see how we can change them using **string methods**.

**String methods** let you perform specific tasks for strings.

We'll focus on four string methods:

1. len()
2. lower()
3. upper()
4. str()

Let's start with len(), which gets the length (the number of characters) of a string!

# lower()

Well done!

You can use the lower() method to get rid of all the capitalization in your strings. You call lower() like so:

"Ryan".lower()

which will return "ryan".

# Dot Notation

Let's take a closer look at why you use len(string) and str(object), but dot notation (such as "String".upper()) for the rest.

lion = "roar" len(lion) lion.upper()

Methods that use dot notation only work with strings.

On the other hand, len() and str() can work on other data types.

# String Formatting with %, Part 1

When you want to print a variable with a string, there is a better method than concatenating strings together.

name = "Mike" print "Hello %s" % (name)

The % operator after the string is used to combine a string with variables. The %operator will replace the %s in the string with the string variable that comes after it.

If you'd like to print a variable that is an integer, you can "pad" it with zeros using %02d. The 0 means "pad with zeros", the 2 means to pad to 2 characters wide, and the d means the number is a signed integer (can be positive or negative).

day = 6 print "03 - %s - 2019" % (day) # 03 - 6 - 2019 print "03 - %02d - 2019" % (day) # 03 - 06 - 2019

As of Python 2.6 there is a new way to do dynamic string formatting. There is now a string method called .format() which is more advanced than using %. The reason we don't focus on this method at Codecademy (at least for now) is that it will not work in older versions of Python and thus its use is discouraged in some settings. Here's a quick explanation.

.format() allows you to set values and then replace those values in the string. It uses curley braces {some\_name} with a variable name inside. Then the method sets those variables as arguments. It also supports many more features than using %. It's easiest to explain using examples:

"I am a {type}".format(type="string") my\_name = "Michael" "Hello, my name is {name}".format(name=my\_name)

There are many other great uses and you can check out the Python documentation for more [here](http://docs.python.org/2/library/stdtypes.html#str.format), [here](http://docs.python.org/3/library/string.html#string-formatting) and [here](http://www.python.org/dev/peps/pep-3101/).

<https://docs.python.org/2/library/stdtypes.html#str.format>

<https://docs.python.org/3/library/string.html#string-formatting>

<https://www.python.org/dev/peps/pep-3101/>

# String Formatting with %, Part 2

Remember, we used the % operator to replace the %s placeholders with the variables in parentheses.

name = "Mike" print "Hello %s" % (name)

You need the same number of %s terms in a string as the number of variables in parentheses:

print "The %s who %s %s!" % ("Knights", "say", "Ni") # This will print "The Knights who say Ni!"

# String Formatting with %, Part 2

Remember, we used the % operator to replace the %s placeholders with the variables in parentheses.

name = "Mike" print "Hello %s" % (name)

You need the same number of %s terms in a string as the number of variables in parentheses:

print "The %s who %s %s!" % ("Knights", "say", "Ni") # This will print "The Knights who say Ni!"

Remember, the syntax is:

print "%s" % (string\_variable)

Are you stuck with a spinning gear and it feels like your code is frozen? That's because the code is asking you three questions! Click inside the console window, type in your name, and then press Enter (or Return).

The \ character on line 5 is a *continuation marker*. It simply tells Python that line 5 continues onto line 6.

# And Now, For Something Completely Familiar

Great job! You've learned a lot in this unit, including:

Three ways to create strings

'Alpha' "Bravo" str(3)

String methods

len("Charlie") "Delta".upper() "Echo".lower()

Printing a string

print "Foxtrot"

Advanced printing techniques

g = "Golf" h = "Hotel" print "%s, %s" % (g, h)

The % operator after a string is used to combine a string with variables.

quarter = 0.25 print "A quarter is %f" % (quarter)

The % operator will replace a %fin the string with the floating point number variable that comes after it.