## Single & Double Quotes

* Enclose strings. Either single or double quotes are fine.
* Use single quotes as enclosing quotes to eliminate the need of escaping double quotes in a string, and vice versa.

## Triple Quotes

* Enclose strings containing both single and double quotes such that no escaping is needed.
* Enclose multi-line strings.

Typically, either single or double quotes can be used interchangeably for creating a string. However, in certain situations, the type of quotes can matter, depending on the content of the string itself. If the string contains single quotes, it can be more convenient to wrap the entire string in double quotes. And if the string contains double quotes, you might use single quotes to wrap the string instead.

The reason for this is that each string starts and ends when a pair of quotes is encountered. If there is another matching quote within the string, that may end the string prematurely, causing errors.

*# Examples of valid strings*

print("I'm coding!")

print('The computer printed the string "Hello!"')

*# Example of invalid string*

print("A common phrase in programming is "Hello world"")

There is no difference unless you have words like (It’s, they’re, etc…)  
but you can always use '.

*# EXAMPLE*

*# the following two lines print Hello World! to the screen*

print('Hello World!')

print("Hello World")

*# the following two lines prints It's a sunny day*

print('It\'s a sunny day')

print("It's a sunny day")

One thing to consider is if moving into other languages, there is a distinction between " and '. I’m still new myself, but also wanted to develop good habits with my choice on this. I know I want to get into C# later on, and so I’m going to default to using " instead of '. Just something to think about for building a habit.

Single and Double Quotes

Basic Usage

The most common use of single and double quotes is to represent strings by enclosing a series of characters. As shown in the code below, we create these two strings using single and double quotes, respectively.

>>> quotes\_single = 'a\_string'  
>>> quotes\_double = "a\_string"  
>>> quotes\_single == quotes\_double  
True

As you notice, the strings created by using single and double quotes are the same. In other words, we can use single and double quotes interchangeably when we declare a string. However, it should be noted that we don’t want to mix them as it’s a syntactical error.

>>> "mixed quotes'  
 File "<stdin>", line 1  
 "mixed quotes'  
 ^  
SyntaxError: EOL while scanning string literal  
>>> 'mixed quotes"  
 File "<stdin>", line 1  
 'mixed quotes"  
 ^  
SyntaxError: EOL while scanning string literal

Escaping Behaviors

Like other programming languages, when a string contains special characters like quotes, we need to escape them. An example of failing to escape is shown below.

>>> 'It's a bad example.'  
 File "<stdin>", line 1  
 'It's a bad example.'  
 ^  
SyntaxError: invalid syntax

How can we fix this error? One is to escape the single quote by placing a backslash before it. The other is to use double quotes instead of single quotes as the enclosing quotes. Both ways are shown below.

>>> 'It\'s a good example.'  
"It's a good example."  
>>> "It's a good example."  
"It's a good example."

Similarly, if the string contains double quotes, we can use single quotes to represent the string such that we don’t have to escape the double quotes. An example is given below.

>>> 'She said, "Thank you!"'  
'She said, "Thank you!"'

However, if there are both single and double quotes in the string, it’s a syntactical error if you don’t escape the quotes that are the same as the enclosing ones used by the entire string.

>>> print('She said, "Thank you! It's mine."')  
 File "<stdin>", line 1  
 print('She said, "Thank you! It's mine."')  
 ^  
SyntaxError: invalid syntax  
>>> print('She said, "Thank you! It\'s mine."')  
She said, "Thank you! It's mine."

Triple Quotes

Enclosing Strings Containing Single and Double Quotes

As mentioned at the end of the above section, we need to escape single or double quotes depending on what enclosing quotes the string uses. Actually, we can use triple quotes (i.e., triplet of single quotes or triplet double quotes) to represent the strings containing both single and double quotes to eliminate the need of escaping any.

>>> print('''She said, "Thank you! It's mine."''')  
She said, "Thank you! It's mine."

It should be noted that **when a string starts or ends with a single or double quote and we want to use the triple quotes for the string, we need to use the ones that differ from the starting or ending one.** For example, for the string in the above code snippet, using triple double quotes would result in a syntactical error. In this case, we want to use the triple single quotes as above.

>>> print("""She said, "Thank you! It's mine."""")  
 File "<stdin>", line 1  
 print("""She said, "Thank you! It's mine."""")  
 ^  
SyntaxError: EOL while scanning string literal

Multi-line Strings

Another use case of the triple quotes is to represent a multi-line string. An example is given below. You can use either triple single or double quotes in this case.

>>> print("""Hello  
... World  
... !""")  
Hello  
World  
!

Although we can achieve the same effect by using the \n symbols to create multi-line strings as below, using the \n symbols makes the string harder to read. By contrast, using triple quotes can write the string in the way how it exactly looks like, and thus it has better readability.

>>> print('Hello\nWorld\n!')  
Hello  
World  
!

In addition, a useful application of the triple-quote enclosed strings is to specify some comments in a multi-line string, for example, as part of a function definition like below.

>>> def multiple\_line\_comment(a, b):  
... '''  
... a is a string # other additional description  
... b is a list of integers # other additional description  
... '''  
... pass  
...   
>>> print(multiple\_line\_comment.\_\_doc\_\_)  
  
 a is a string # other additional description  
 b is a list of integers # other additional description

We can clearly tell what are the comments for the function.