Working with Strings

Read the following code.

```
sPart1 = "The bunny"
sPart2 = "hops over the fence."
print(sPart1)
print(sPart2)
```

The result is

```
The bunny
hops over the fence.
```

These are two separate string variables that live in different locations in the memory.

Modify the program to create a third string that forms the full sentence into a single variable.

Attempt the solution on your own. Compare your answers to the solution provided on the next page.

Solution

The solution for this problem is:

```
sPart1 = "The bunny"
sPart2 = "hops over the fence."
print(sPart1)
print(sPart2)

sSentence = sPart1 + " " + sPart2
print(sSentence)
```

Run the module. The result is:

```
The bunny
hops the fence.
The bunny hops over the fence.
```

Try out this solution without the

+ " " +

part and find out why it was added.

Storing Strings in Lists

The following code has turned every word in the previous sentence into a list of words.

```
sWords = ["the", "bunny", "hops", “over”, "the", "fence"]
print(sWords)
```

Run the module. The result is:

```
['the', 'bunny', 'hops', 'over', 'the', 'fence']
```

To get a nice prinout we’ll replace print(sWords) with print(*sWords). Run the module to get the result:

```
the bunny hops over the fence
```

Let’s add lines of code that change this so that the sentence is capitalized.

The function .capitalize() ran on any string, returns a clone of the string with the first letter capitalized.

For example:

print(sWords[0].capitalize())

will print the word “The”, instead of “the”, which lives at position 0, i.e. sWords[0].

Change the code as follows.

```
sWords = ["the", "bunny", "hops", "over", "the", "fence"]
print(*sWords)

print(sWords[0].capitalize())
print(*sWords)
```

Run the module. The result is still:

```
the bunny hops over the fence
```

Why has nothing changed? It is because we have produced the capitalized cloned string, but we did not store it back into the list, we have only printed it.

This is how we store it to make a difference.

Change the code as follows.

```
sWords = ["the", "bunny", "hops", "over", "the", "fence"]
print(*sWords)

sWords[0] = sWords[0].capitalize()
print(*sWords)
```

Run the module. The result is now:

```
The bunny hops over the fence
```

Let us add one more word to the list but make that word simply a dot. This means instead of 6 elements in the list, we want to have 7 elements.

Change the code as follows.

```
sWords = ["the", "bunny", "hops", "over", "the", "fence"]
print(*sWords)

sWords[0] = sWords[0].capitalize()
print(*sWords)

sWords.append(".")
print(*sWords)
```

Run the module. The result is now:

```
The bunny hops over the fence .
```

Notice that Python places a blank between each two words, the dot is no exception.

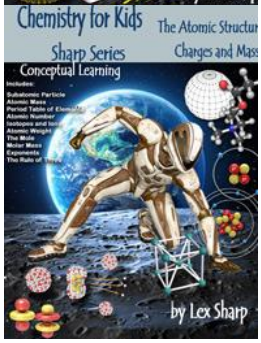
Next, the goal is to fix the sentence, so it does not have a blank before the dot. It cannot be fixed with the code we learned so far. Next time we learn about for loops to help us fix this problem.

Books in Sharp Series

Find out more about **Workbooks 1 to 4** in this series at:
https://sharpseries.ca/scratch/w.html.

Find out about **Color Your Way to Math** Volumes 1 to 3 at:
https://sharpseries.ca/cyw/cyw2m.html.

Find out about **Early Math Concepts** at:
https://sharpseries.ca/em/v1.html.



Find out about **Chemistry for Kids** at:
https://sharpseries.ca/chem/v1.html.