STRING OPERATIONS

The Python Bakery

What are the operations?

- Addition
- Comparisons
- Membership
- Subscripting
- Indexing

Strings have a number of operators that can be used on them, just like integers or booleans.
In fact, they have some special operators we haven't seen before.
We will learn about the five kinds of operators shown here.
Pay particular attention to membership, subscripting, and indexing, which are brand new.

Addition

```
print("Hello " + "World")
```

Like numbers, you can add two strings together. This puts them side by side in a single new string. This is sometimes called "Concatenation".

Comparing Strings

```
print("Dog" == "Dog")
print("Dog" != "Cat")
print("Aardvark" < "Zoo")
print("Orange" <= "Apple")
```

You can test if two strings are equal, not equal, or even compare them using less than and greater than. This measures which ones come first in the alphabet.

Membership in Strings

```
print("house" in "boathouse")
print("cow" in "cowabunga")
print("y" not in "axes")
print("xe" not in "axes")
```

There's another test you can check with Strings: using the "in" operator.
This simply checks whether the first string appears in the second.
You can also use the "not in" operator to test the opposite.

Subscripting

```
message = "Hello world"
print(message[0])
```

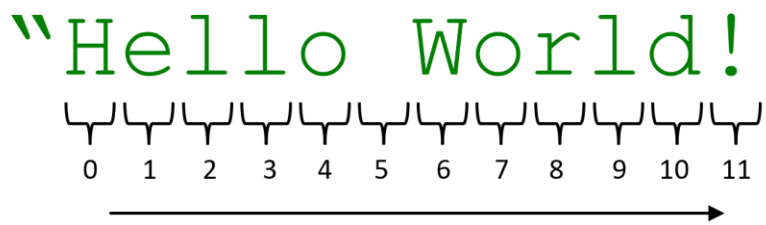
Subscripting is one of the more powerful and more complex features of strings. When we subscript a string, we extract one or more characters from the string.

Subscript Syntax

```
book_title = "Harry Potter"
print(book_title[2])
```

We can subscript a string value in a variable by using square brackets.
Notice the key components: On the left is the name of the variable or the string literal value.
Next we have an opening square bracket.
Then, we have a number, which is called the index.
Finally, we end with a closing square bracket.

Starting at 0



Here's a weird thing: computers start counting at 0, not 1.
So if you want the first character from the string, you need to write 0 instead of 1.
There are boring math reasons why this happens, but ultimately it ends up being more convenient, once you get used to it.

Subscripting Multiple Characters

```
message = "Hello world"
print(message[2:5])
```

It wouldn't be too useful to only grab out one character at a time.
So you can actually grab out more than one by using the subscript range syntax:
Inside the square brackets, you put a pair of numbers separated by the colon.

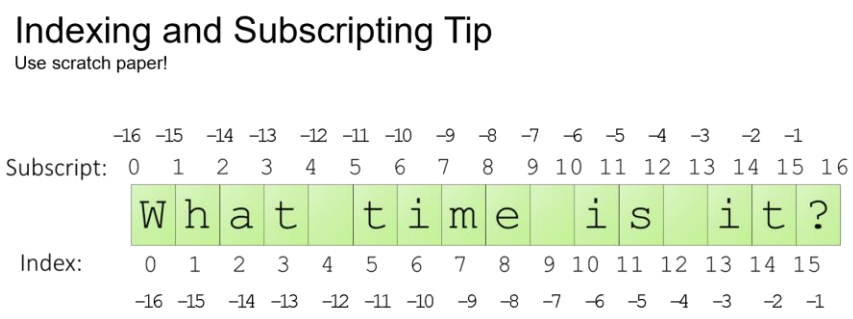
Negative Subscript Indices

```
word = "hamster"

print(word[-1])
print(word[1:-1])
print(word[-3:])
print(word[:3])
```

If you use negative numbers as subscript indexes, you can work from the back of the list.
If you use -1, then you get the last character.
You can combine positive and negative in range.
To go from the start or the end, simply leave the number missing.

Figuring out Subscripts



Figuring out subscripts can be quite tricky. If you try to stare and count the index by hand, you are almost certainly going to make a mistake.

Instead, take out a piece of paper and draw boxes around each character of the string. Then, number the boxes.

Put indexes directly BELOW the boxes, and subscripts directly BETWEEN the boxes.

You can then quickly check what character a given index corresponds to, or what range of characters are sandwiched between two subscripts.