Python, Day 3: Strings

Andrew Bydlon

January 8, 2019

What are strings?

Strings are one of the data types we discussed previously.

A **string** is a collection of characters enclosed in quotations. There are 2 equivalent methods to declare strings.

Example (Declaring strings)

```
> > > string1 = "Hi Class!"
> > > string2 = 'Hi Class!'
> > > string1 == string2
True
> > > EmptyString = ""
```

Why choose ’ ’ vs " "?

Notice the following interesting use case for each of the quotation types:

Example

```
> > > "It’s time for class!"
> > > ’It’s time for class!’
SyntaxError: invalid syntax
> > > "And then Andrew said" Welcome to class!"
SyntaxError: invalid syntax
> > > ’And then Andrew said "Welcome to class!"’
```

We will be able to resolve this issue with a command, but it can still be quicker to follow this setup.

Some functions for strings: len

We can count the number of characters in the string by using len:

Example

```
> > > len("It's time for class!")
20
> > > len('And then Andrew said "Welcome to class!"')
40
```

A quick note: we can use \ to continue writing code on one line:

Example

```
> > > (3 + 3) * \
...         "My Story is "
'My Story is My Story is My Story is My Story is My Story is My Story
is '
```

Getting around special characters: Escape Sequences

Some characters, such as "", ”, \, etc, are special characters to Python. To use them in strings, we need to input code:

1. Newline, or Enter: \n
2. Tab: \t
3. Backslash: \\
4. Single Quote: \’
5. Double Quote: \"

Tale of Two Strings

Example

> > > To2C = "\t It was the best of times, \n it was the worst of times, \n it was the age of wisdom, \n it was the age of foolishness, \n it was the epoch of belief, \n it was the epoch of incredulity, \n it was the season of Light, \n it was the season of Darkness, \n it was the spring of hope, \n it was the winter of despair, ..."

Example

> > > BeatlesLyrics = "And then Paul uttered, \"Don\'t let me down!\" \\END"

Getting rid of those pesky spaces

Another operation we can apply to strings is +.

This performs **string concatenation**, by which the 2 strings are made into 1 by picking up the first where the second begins.

Example

```
> > > SportsString = "My favorite sport is " + "Basketball!" + \
...         " Or maybe raquetball..."
> > > print(SportsString)
My favorite sport is Basketball! Or maybe raquetball...
```

Correcting our old errors

We can fix our issues in spacing from the previous 2 class periods by converting our numbers to a string with the str() function.

Example

```
> > > Moneys = 12385.12
> > > print("Your money at time t is $" + str(Moneys)+"!")
Your money at time t is $12385.12!
```

The repetition operator

As we have seen before, another common operation is *. We can use this to make the string repeat itself a certain number of times.

Example

```
> > > "We got the " + "spam"*10
'We got the spamspamspamspamspamspamspamspamspamspam'
```

Note that it follows the same order of operations as with numbers.

Finding characters in a string

We can use the in function to test whether a string of characters lies within another.

Example

```
> > > MyString = "Once upon a time there was a woman named..."
> > > "nce" in MyString
True
> > > "Once U" in MyString  # Note case sensitivity
False
> > > "Once U" not in MyString  # Combining with not operator
True
> > > "Once u" in MyString  # Note case sensitivity
True
```

Character ranges in a string

You can also extract from a string the characters in a certain range, or a specific character, using [ ].

Note that it starts at 0 and goes to len(str)-1.

Example

```
> > > MyString = "Once upon a time there was a woman named..."
> > > len(MyString)
43
> > > MyString[1]
n
> > > MyString[3:14]
'e upon a ti'
```

Examples Continued

Example

```
> > > MyString[-15:-1]  # You can use negative numbers!
' woman named..'      # Note that -1 is the 2nd to last character.

> > > MyString[-15:0]  # And negative and positive numbers...
' '           # ...don't play well together, yielding the empty string.

> > > MyString[6:]  # You can also leave out one side.
'pon a time there was a woman named...'

> > > MyString[-15:]
' woman named...'
```

Some tricks with print

You can specify the way print behaves. By default, it ends with a newline command, and puts the separator to a space:

Example

```
> > > print("Hi.", "I'm Andrew", end="")
...         print(" Bydlon.")
Hi. I'm Andrew Bydlon. # No new line after andrew


> > > print("Hi. ", "I'm Andrew", " Bydlon.", sep="")
Hi.I'm Andrew Bydlon. # The usual spaces are removed.
 > > > print("Hi. ", "I'm Andrew", "Bydlon.", sep="$")
Hi. $I'm Andrew$Bydlon. # The usual spaces are $.
```

Comparing strings

We can use the commands that we use for numbers to compare strings as well. They compare strings using the **dictionary order**.

Example

```
> > > "Hi" > "Hello"
True
> > > "Z" > "a"
False # Capital letters appear before lower case.
> > > "Hello" != 'Jello'
True
```

Formatting a string

Strings are well formatted by default. Of course, sometimes you want to change it.

Example

```
> > > format("Test","10s")
'Test      '
> > > format("Test","3s")  # It increases width dynamically
'Test'
> > > format("Test", ">10s")  # You can also right align
'      Test'
```

Assignment 4

Write a program which

1. Takes a string from the user.
2. Tells them how long the string is.
3. Asks the user for a excerpt.
4. Tells them if the excerpt occurs within the original string.
5. Asks them for an integer N less than the string's length.
6. Prints the first N and last N characters of the string, and separates them by a character of your choice.

Be careful about numbering in step 6!

Submit the .py file when you finish.