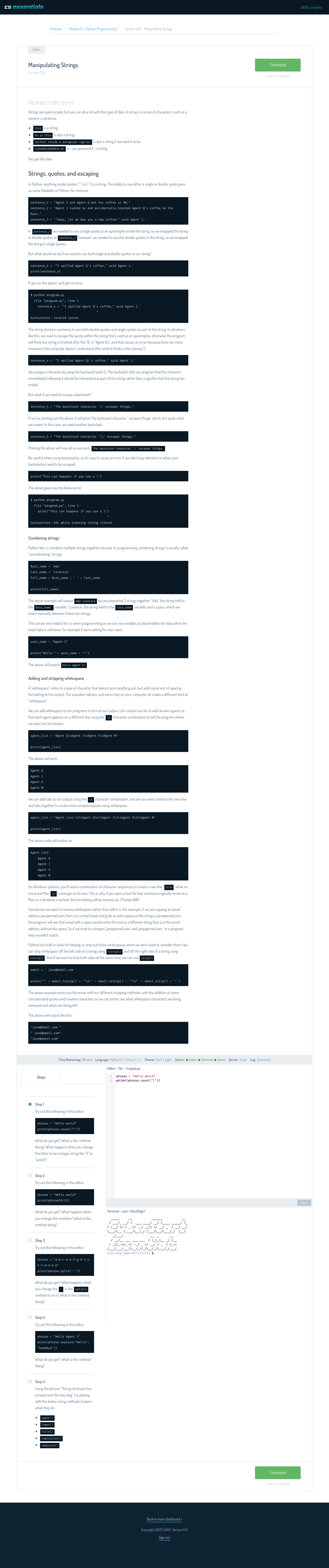

Editor

Manipulating Strings

Section 21.8

Completed

< back to module

Skip straight to editor/terminal

Strings are quite simple, but you can do a lot with this type of data. A string is a series of characters, such as a word or a sentence.

- `this` is a string.
- `So is this` is also a string.
- `<p>Text inside a paragraph tag</p>` is also a string if we want it to be.
- `lu3hd6h3309&%34-0+` is – you guessed it – a string.

You get the idea.

Strings, quotes, and escaping

In Python, anything inside quotes (" ") or (' ') is a string. The ability to use either a single or double quote gives us some flexibility in Python, for instance:

```
sentence_1 = "Agent S and Agent Q met for coffee at HQ."
sentence_2 = "Agent J rushed by and accidentally knocked Agent Q's coffee on the floor."
sentence_3 = '"Oops, let me buy you a new coffee!" said Agent J.'
```

In `sentence_2` we needed to use a single quote as an apostrophe inside the string, so we wrapped the string in double quotes. In `sentence_3` however, we needed to use the double quotes in the string, so we wrapped the string in single quotes.

But what would we do if we need to use *both* single and double quotes in our string?

```
sentence_4 = '"I spilled Agent Q's coffee," said Agent J.'
print(sentence_4)
```

If we run the above, we'll get an error:

```
$ python program.py
  File "program.py", line 1
    sentence_4 = '"I spilled Agent Q's coffee," said Agent J.'
                                    ^
SyntaxError: invalid syntax
```

The string stored in sentence_4 uses both double quotes and single quotes as part of the string. In situations like this, we need to escape the quote within the string that's used as an apostrophe, otherwise the program will think our string is finished after the "Q" in "Agent Q's", and that causes an error because there are more characters the computer doesn't understand after what it thinks is the closing (').

```
sentence_4 = '"I spilled Agent Q\'s coffee," said Agent J.'
```

We escape a character by using the backward slash (). The backslash tells our program that the character immediately following it should be interpreted as part of the string rather than a signifier that the string has ended.

But what if we need to escape a backslash?

```
sentence_5 = "The backslash character '\' escapes things."
```

If we try printing out the above, it will print The backslash character '' escapes things. which isn't quite what we meant. In this case, we need another backslash.

```
sentence_5 = "The backslash character '\\' escapes things."
```

Printing the above will now do as we want, `The backslash character '\' escapes things.`

Be careful when using backslashes, as it's easy to cause an error if you don't pay attention to when your backslashes need to be escaped:

```
print("This can happens if you use a \")
```

The above gives you the below error.

```
$ python program.py
  File "program.py", line 1
    print("This can happens if you use a \")
                                           ^
SyntaxError: EOL while scanning string literal
```

Combining strings

Python lets us combine multiple strings together into one. In programming, combining strings is usually called "concatenating" strings.

```
first_name = 'Ada'
last_name = 'Lovelace'
full_name = first_name + ' ' + last_name

print(full_name)
```

The above example will output `Ada Lovelace` by concatenating 3 strings together: "Ada", the string held in the `first_name` variable; "Lovelace", the string held in the `last_name` variable; and a space, which we insert manually between these two strings.

This can be very helpful for us when programming as we can use variables as placeholders for data when the exact data is unknown, for example if we're asking for user input.

```
user_name = "Agent S"

print("Hello " + user_name + "!")
```

The above will output `Hello Agent S!`

Adding and stripping whitespace

A "whitespace" refers to a type of character that doesn't print anything out, but adds some sort of spacing formatting to the output. The spacebar, tab key, and return key on your computer all create a different kind of "whitespace".

We can add whitespace to our programs to format our output. Let's output our list of well-known agents so that each agent appears on a different line using the `\n` character combination to tell the program where we want our line breaks.

```
agent_list = "Agent Q\nAgent J\nAgent S\nAgent M"

print(agent_list)
```

The above will print:

```
Agent Q
Agent J
Agent S
Agent M
```

We can add tabs to our output using the `\t` character combination, and we can even combine the new line and tabs together to create more complex layouts using whitespace.

```
agent_list = "Agent List:\n\tAgent Q\n\tAgent J\n\tAgent S\n\tAgent M"

print(agent_list)
```

The above code will output as:

```
Agent List:
    Agent Q
    Agent J
    Agent S
    Agent M
```

On Windows systems, you'll need a combination of character sequences to create a new line, `\r\n`, while on Linux and Mac `\n` is enough on its own. This is why if you open a text file that someone originally wrote on a Mac on a Windows machine, the formatting will be messed up. (Thanks Bill!)

Sometimes we want to remove whitespace rather than add it in. For example, if we are copying an email address jane@email.com from our contact book and grab an extra space so the string is 'jane@email.com ', the program will see that email with a space tacked onto the end as a different string than just the email address without the space. So if we tried to compare 'jane@email.com' and 'jane@email.com ' in a program, they wouldn't match.

Python has built-in tools for helping us strip out these whitespaces when we don't want to consider them. We can strip whitespace off the left side of a string using `lstrip()` and off the right side of a string using `rstrip()`. And if we want to strip both sides at the same time, we can use `strip()`.

```
email = ' jane@email.com '

print('"' + email.lstrip() + '"\n"' + email.rstrip() + '"\n"' + email.strip() + '"')
```

The above example prints out the email with our different stripping methods, with the addition of some concatenated quotes and newline characters so we can better see what whitespace characters are being removed and which are being left.

The above will output like this:

```
"jane@email.com "
" jane@email.com"
"jane@email.com"
```

Time Remaining: 59 mins **Language:** Python 2 | Python 3 | C **Theme:** Dark | Light **Status:** ● Editor ● Terminal ● Server **Server:** Reset **Log:** Download

Steps

● **Step 1**

Try out the following in the editor:

```
phrase = "Hello world"
print(phrase.count("l"))
```

What do you get? What is this method doing? What happens when you change the letter to be a longer string like "ll" or "world"?

○ **Step 2**

Try out the following in the editor:

```
phrase = "Hello world"
print(phrase[0:5])
```

What do you get? What happens when you change the numbers? What is this method doing?

○ **Step 3**

Try out the following in the editor:

```
phrase = "a-b-c-d-e-f-g-h-i-j-k-l-m-n-o-p"
print(phrase.split('-'))
```

What do you get? What happens when you change the `-` in the `split()` method to an h? What is this method doing?

○ **Step 4**

Try out the following in the editor:

```
phrase = "Hello Agent J"
print(phrase.replace("Hello", "Goodbye"))
```

What do you get? What is this method doing?

○ **Step 5**

Using the phrase "The quick brown fox jumped over the lazy dog", try playing with the below string methods to learn what they do.

- `upper()`
- `lower()`
- `title()`
- `capitalize()`
- `swapcase()`

Editor – File: ~/output.py

```
phrase = "Hello world"
print(phrase.count("l"))
```

Save

Terminal – user: v1UzJ3bglJ

```
  _____     __           ______          __
 / ___/_ __/ /  ___ ____/ __/ /____ ____/ /_
/ /__/ // / _ \/ -_) __/\ \/ __/ _ `/ __/ __/
\___/\_, /_.__/\__/_/ /___/\__/\_,_/_/  \__/
   _/___/              __  _      __
  / __/__ ___ ___ ___  / /_(_)__ _/ /__
 / _/(_-<(_-</ -_) _ \/ __/ / _ `/ (_-<
/___/___/___/\__/_//_/\__/_/\_,_/_/___/
v1UzJ3bglJ@da7097132af1:~$
```

Completed

< back to module

Copyright 2020 SANS. Version 1.8.1