StringIO Objects and the io Module

Back in **Lecture 24 - Files** we opened files that exist outside of python, and streamed their contents into an in-memory file object. You can also create in-memory file-like objects within your program that Python treats the same way. Text data is stored in a StringIO object, while binary data would be stored in a BytesIO object. This object can then be used as input or output to most functions that would expect a standard file object.

Let's investigate StringIO objects. The best way to show this is by example:

In [1]:

```
import io
```

In [2]:

```
# Arbitrary String
message = 'This is just a normal string.'
```

In [3]:

```
# Use StringIO method to set as file object
f = io.StringIO(message)
```

Now we have an object *f* that we will be able to treat just like a file. For example:

In [4]:

```
f.read()
```

Out[4]:

```
'This is just a normal string.'
```

We can also write to it:

In [5]:

```
f.write(' Second line written to file like object')
```

Out[5]:

```
40
```

In [6]:

```
# Reset cursor just like you would a file
f.seek(0)
```

Out[6]:

```
0
```

In [7]:

```
# Read again
f.read()
```

Out[7]:

```
'This is just a normal string. Second line written to file li
ke object'
```

In [8]:

```
# Close the object when contents are no longer needed
f.close()
```

Great! Now you've seen how we can use StringIO to turn normal strings into in-memory file objects in our code. This kind of action has various use cases, especially in web scraping cases where you want to read some string you scraped as a file.

For more info on StringIO check out the documentation:
https://docs.python.org/3/library/io.html (https://docs.python.org/3/library/io.html)