1-2_Strings

September 15, 2015

1 Strings

There are operations that can be done with strings.

```
In [1]: firstName = "Johan"
```

```
In [2]: lastName = "Gambolputty"
```

When concatenating strings, we must explicitly use the concatenation operator `+`. Computers don't understand context.

```
In [3]: fullName = firstName + lastName
```

```
In [4]: print fullName
```

```
JohanGambolputty
```

```
In [5]: fullName = firstName + " " + lastName
```

```
In [6]: print fullName
```

```
Johan Gambolputty
```

You can also think of strings as a sequence of smaller strings or characters. We can access a piece of that sequence using `[]`.

```
In [7]: fullName[1]
```

```
Out[7]: 'o'
```

Gotcha - Python (and many other langauges) start counting from 0.

```
In [8]: fullName[0]
```

```
Out[8]: 'J'
```

```
In [9]: fullName[4]
```

```
Out[9]: 'n'
```

One more gotcha - in Python, if you want a range (or "slice") of a sequence, you get everything *before* the second index:

```
In [10]: fullName[0:4]
```

```
Out[10]: 'Joha'
```

```
In [11]: fullName[0:5]
```

```
Out[11]: 'Johan'
```

You can see some of the logic for this when we consider implicit indices.

```
In [12]: fullName[:5]
```

```
Out[12]: 'Johan'
```

```
In [13]: fullName[5:]
```

```
Out[13]: ' Gambolputty'
```

1.1 String Methods

There are other operations defined on string data. IPython lets you do tab-completion after a dot ('.') to see what an object (i.e., a defined variable) has to offer. Try it now!

```
In [14]: str.
```

```
      File "<ipython-input-14-58171b034407>", line 1
    str.
        ^
SyntaxError: invalid syntax
```

Let's look at the upper method. What does it do? Lets take a look at the documentation. IPython lets us do this with a question mark ('?') before *or* after an object (again, a defined variable).

```
In [15]: str.upper?
```

So we can use it to upper-caseify a string.

```
In [16]: fullName.upper()
```

```
Out[16]: 'JOHAN GAMBOLPUTTY'
```

You have to use the parenthesis at the end because upper is a method of the string class.
For what its worth, you don't need to have a variable to use the upper() method, you could use it on the string itself.

```
In [17]: "Johann Gambolputty".upper()
```

```
Out[17]: 'JOHANN GAMBOLPUTTY'
```

What do you think should happen when you take upper of an int? What about a string representation of an int?

1.1.1 More methods

Try seeing what the following string methods do: * `split` * `join` * `replace` * `strip`