

MORE ABOUT STRINGS

DR IAN CORNELIUS

HELLO

- Learning Objectives:
 1. Understand the extra functionality of strings in Python
 2. Demonstrate the ability to use strings and their extra functions

PREVIOUSLY...

- Last week you were introduced to the string data type
- You learnt how to declare variable that consists of a string
 - i.e. using a double (") quote, single (') quote or three double (""") or three single ('''') quotes

MORE STRING STUFF! (1)

STRINGS ARE A SEQUENCE

- Strings in Python are considered to be an array of bytes that represent unicode characters
 - this is because Python does not have a character data type
- Therefore, each element of a string can be accessed by its index number

```
</> stringExample1 = "Hello 4061CEM"
```

```
</> stringExample1[1]
```

```
▶ stringExample1[1] = e
```

MORE STRING STUFF! (2)

STRING LENGTH

- As aforementioned, a string is a sequence of characters you can find out the length of a string
 - i.e. how many characters are in the string
- You can find out the length of a string by using the `len()` function

```
</> stringExample1 = "Hello 4061CEM"
```

```
</> len(stringExample1)  
len('Hello World')
```

```
▶ len(stringExample1) = 13  
len('Hello World') = 11
```

MORE STRING STUFF! (3)

FINDING A SUBSTRING

- You have been introduced to **membership** operators, and these can be used to check for substrings inside a string
- This is achieved using the **in** keyword
 - you can also check whether a character or phrase is not in the string itself
 - this is achieved using a combination of the **not** and **in** keywords (**not in**)

```
</> stringExample1 = "Hello 4061CEM"
```

```
</> "4061CEM" in stringExample1  
"4063" in stringExample1  
"4063" not in stringExample1
```

```
▶ "4061CEM" in stringExample1 = True  
"4063" in stringExample1 = False  
"4063" not in stringExample1 = True
```

MODIFYING STRINGS (1)

- Python consists of built-in methods that can be used to modify strings

UPPERCASE

- Strings can be converted to all uppercase styling using the `upper()` method
 - the method is called directly on the variable or string itself

```
</> stringExample1 = "hello 4061cem"
```

```
</> stringExample1.upper()  
"hello 4061cem".upper()
```

```
▶ stringExample1.upper() = HELLO 4061CEM  
"hello 4061cem".upper() = HELLO 4061CEM
```

MODIFYING STRINGS (2)

LOWERCASE

- Strings can be converted to all lowercase styling using the `lower()` method
 - the method is called directly on the variable or string itself

```
</> stringExample1 = "HELLO 4061CEM"
```

```
</> stringExample1.lower()  
"hElLo 4061CeM".lower()
```

```
▶ stringExample1.lower() = hello 4061cem  
"hElLo 4061CeM".lower() = hello 4061cem
```


MODIFYING STRINGS (3)

WHITE-SPACE REMOVAL

- Strings can be modified to remove white-space that may exist at the beginning or end of the string using the `strip()` method
 - the method is called directly on the variable or string itself

```
</> stringExample1 = " Hello 4061CEM "
```

```
</> stringExample1.strip()  
" hello 4061cem ".strip()
```

```
▶ stringExample1.strip() = Hello 4061CEM  
" Hello 4061CEM ".strip() = hello 4061cem
```

MODIFYING STRINGS (4)

REPLACING A SUBSTRING

- You can replace a substring in a string using the `replace()` keyword
 - the method is called directly on the variable or string itself

```
</> stringExample1 = "Hello 4061CEM"
```

```
</> stringExample1.replace("4061", "4059")  
"Hello 4061CEM".replace("4061", "4063")
```

```
▶ stringExample1.replace("4061", "4059") = Hello 4059CEM  
"Hello 4061CEM".replace("4061", "4063") = Hello 4063CEM
```

MODIFYING STRINGS (5)

MERGING STRINGS

- Strings can be merged/joined/concatenated using the `+` operator
- Unlike integers where it would sum the two variables, in a string it will join or concatenate the two variables together

```
</> stringExample1 = "4061"  
      stringExample2 = "CEM"
```

```
</> print("Welcome to " + stringExample1 + stringExample2)
```

```
▶ Welcome to 4061CEM print("Welcome to" + stringExample1 + stringExample2) = Welcome to 4061CEM
```

MODIFYING STRINGS (6)

MERGING STRINGS AND OTHER DATA TYPES I

- Merging strings together with a number cannot be achieved using the `+` operator
- However, it can be done using the `format()` method
 - the method will take passed arguments and format them into placeholders denoted by curly braces (“{}”)

```
</> intExample1 = 4061  
    stringExample1 = "CEM"
```

```
</> "{}{}".format(intExample1, stringExample1)
```

```
▶ "{}{}".format(intExample1, stringExample1) = 4061CEM
```

MODIFYING STRINGS (7)

MERGING STRINGS AND OTHER DATA TYPES II

- You can also position arguments into a placeholder by using an index number

```
</> intExample1 = 4061
    stringExample1 = "CEM"
    stringExample2 = "Programming and Algorithms 1"
```

```
</> "{}{}: {}".format(intExample1, stringExample2, stringExample1)
    "{0}{2}: {1}".format(intExample1, stringExample2, stringExample1)
```

```
▶ "{}{}: {}".format(intExample1, stringExample2, stringExample1) =
    4061Programming and Algorithms 1: CEM

    "{0}{2}: {1}".format(intExample1, stringExample2, stringExample1) =
    4061CEM: Programming and Algorithms 1
```

FORMATTING A STRING

- This uses the `f` character at the beginning of a string declaration
- Inside this string, variables can be used when enclosed by curly braces (`{}`)

```
</> name = "Ian Cornelius"  
    age = 33
```

```
</> f"Hello {name} it is nice to meet you!"  
    f"Hello {name} it is nice to meet you! Your age is: {age}."
```

```
▶ f"Hello {name} it is nice to meet you!" =  
    Hello Ian Cornelius it is nice to meet you!  
  
f"Hello {name} it is nice to meet you! Your age is: {age}." =  
    Hello Ian Cornelius it is nice to meet you! Your age is: 33.
```

ESCAPE CHARACTERS

- There are some characters that are considered illegal when being used in a string
 - i.e. strings created with a double quote (") will not allow another double quote inside it
- To use illegal characters in a string, you can escape them using the backslash symbol (\)

```
</> stringExample1 = "Hello 4061CEM, this is the "best" course."  
# This will throw an error
```

```
</> stringExample2 = "Hello 4061CEM, this is the \"best\" course."  
# This will not throw an error as the second set of double quotes have been escaped
```

```
</> stringExample3 = "Hello 4061CEM, this is the 'best' course."  
stringExample4 = 'Hello 4061CEM, this is the "best" course.'
```

GOODBYE

- Questions?
 - Post them in the **Community Page** on Aula
- Contact Details:
 - Dr Ian Cornelius, ab6459@coventry.ac.uk