And now for something quite similar . . .

Strings

- A String is a special kind of list.
- In some ways a String is like a list ...
- In some ways a String is not like a list ...
- Similarities and Differences are highlighted in this presentation ...

Python Strings

8.1 Introduction to Strings

8.2 String Operations

Python Strings

8.1 Introduction to Strings

8.2 String Operations

Introduction to Strings

Strings are a collection of characters contained within quote marks.

The following are examples of strings...

```
"Anne was here"
"Anne was here on Friday 31st October 2008"
"9396633"
"A"
"7"
```

Strings are composed of characters

Strings are a collection of characters contained within quote marks.

Each character has a numeric code and belongs to a set of characters known as the *Unicode character set*.

The Unicode character set

The Unicode character set includes the characters A - Z, the numbers 0 - 9, punctuation marks, the space character, as well as many other characters...

http://www.unicode.org/standard/WhatIsUnicode.html

http://www.unicode.org/charts/

The Unicode character set includes the ASCII code character set.

http://www.annedawson.net/ASCII.htm

ASCII = American Standard Code for Information Interchange

Strings are contained within quote marks

Strings are a collection of characters contained within quote marks.

Strings can be contained within single, double or triple quote marks...

An example program using strings

Strings can be contained within single, double or triple quote marks...

```
'Anne was here'
"9396633"
'''Anne was here
   on Saturday
   30th October 2004'''
```

http://www.annedawson.net/python3programs.html 08-01.py

Printing double quote marks within a string

If you want to print a double quote mark (") within a string, contain the string in single quote marks (')...

```
print ('Here is a double quote ", and "more" ')
```

http://www.annedawson.net/python3programs.html 08-02.py

Printing a string which contains an apostrophe

If you want to print an apostrophe (or a single quote mark) within a string, contain the string in double quotes...

```
print ("This is Anne's spam")
```

http://www.annedawson.net/python3programs.html 08-02.py

Python Strings

8.1 Introduction to Strings

8.2 String Operations

Numbers can be treated as characters

Within a string, (i.e. contained in quote marks), a number is treated as a character.

You cannot perform regular arithmetic on numbers stored as characters...

Repeating (multiplying) strings using *

You cannot perform regular arithmetic on numbers stored as characters, but...

you can "multiply" (repeat) strings:

`"3" * 4` results in `"3333"`

`http://www.annedawson.net/python3programs.html` `08-03.py`

Strings can be joined together using the + symbol

Joining strings together using the + symbol is a process known as *concatenation*.

```
"Anne " + "was " + ("here " * 3)
```

results in `"Anne was here here here "`

http://www.annedawson.net/python3programs.html 08-04.py

Python's + operator

http://www.annedawson.net/Python_Plus_Operator.html

Indexing strings using the [] operator

Any element (character) of a string can be accessed by indexing.

```
s1 = "Anne Dawson"
print (s1[0], s1[5])
```

prints `A D`

http://www.annedawson.net/python3programs.html 08-05.py

Slicing strings using the [] operator

Any substring of a string can be obtained by using the [] operator.

```
s1 = "Anne Dawson"
print (s1[0:1],s1[5:7])
print (s1[6:9])
```

prints

```
A Da
aws
```

http://www.annedawson.net/python3programs.html

08-06.py

Finding the length of a string using `len`

The length of any string can be determined using the `len` method.

```
s1 = "Anne"
s2 = "Dawson"
s3 = ""
print (len(s1), end=" ")
print (len(s2), end=" ")
print (len(s3))
```

prints `4 6 0`

http://www.annedawson.net/python3programs.html 08-07.py

String formatting

String formatting was first introduced in Unit 5 Section 3:

http://www.annedawson.net/Python3_Repetition_StringFormatting.htm

and for convenience, is included here...

Printing strings and numbers

We can print string values and number values from the same print statement by separating them with commas:

```
d = 10
c = 75
print ('Total is: ', d, 'dollars and', c, ' cents ')
```

```
>>>
 Total is:  10 dollars and 75  cents
```

http://www.annedawson.net/python3programs.html 05-12.py

Numbers can be printed from within a single string by using a special method known as *string formatting*.

String Formatting and %

In string formatting, we use the % symbol.

The % operator can also be used for a different purpose as the *modulus operator* (finding the remainder after an integer division).

The % symbol is said to be *overloaded*.

String Formatting and %

An overloaded operator behaves differently depending on the context.

In the following example we see the % operator being used to specify how a string should be printed (i.e. string formatting).

Printing integers within a string

```
x = 20
y = 75
print ('The sum of %d and %d is %d' % (x, y, x + y))
```

```
>>>
The sum of 20 and 75 is 95
```

The three %d formatting codes represent where the decimal integers shown in brackets after the % symbol should be printed.

http://www.annedawson.net/python3programs.html 05-13.py

Printing floats within a string

```
x = 20.512
y = 15.269
print ('The sum of %f and %f is %f' % (x, y, x + y))
```

```
>>>
The sum of 20.512000 and 15.269000 is 35.781000
```

The three %f represent where the float values shown in brackets after the % symbol should be printed.

http://www.annedawson.net/python3programs.html 05-13.py

Specifying the number of decimal places

```
x = 20.512
y = 15.269
print ('The sum of %0.2f and %0.2f is %0.2f' % (x, y, x + y))
```

```
>>>
The sum of 20.51 and 15.27 is 35.78
```

The three %0.2f represent where the float values (to 2 decimal places) shown in brackets after the % symbol should be printed.

http://www.annedawson.net/python3programs.html 05-13.py

String formatting codes

`%d` and `%f` are only two of a set of formatting codes available for string formatting.

%c char single character

%e (%E) float or double exponential format

%o int unsigned octal value

%s string

%u int unsigned decimal

%x (%X) int unsigned hex value

Reference:
https://docs.python.org/3/library/stdtypes.html#printf-style-string-formatting

String formatting code %s

%s is the formatting code which represents a string.

```
print ('Python is a %s language.' % 'great')
>>>
Python is a great language.
```

The %s represents where the string value (shown after the % symbol) should be printed.

http://www.annedawson.net/python3programs.html 08-08.py

String formatting using `.format()`

There is a newer way to format strings in Python which uses the `.format()` method.

This method is explained here:

goo.gl/52yx3q

Use of the `.format()` method is optional on the CSCI120 course.

String Formatting video:

https://youtu.be/XqkANFm0H70

Finding a string within a string

The string method `find` is used to locate the position of one string within another.

```
s1 = 'spamandeggs'
x = s1.find('and')
print (x)
```

The output:

```
>>>
4
```

http://www.annedawson.net/python3programs.html 08-09.py

Replacing text within a string

The string method `replace` is used to replace text within a string with new text.

```
s1 = 'spam and eggs'
s2 = s1.replace('and','without')
print (s2)
```

http://www.annedawson.net/python3programs.html

08-10.py

The output:

```
>>>
spam without eggs
```

String Methods video:

https://youtu.be/oEw8A2Gt7QE

Escape Sequences

An escape sequence is a backslash (`\`) followed by one or more characters, which is inserted into a string to perform a special task. E.g. `\n` generates a new line and `\t` generates a tab *within* the string. . .

A string containing escape sequences

The following example shows a string with two escape sequences:

```
s = 'one\ntwo\tthree'
print (s)
```

The output:

```
>>>
one
two	three
```

http://www.annedawson.net/python3programs.html

08-11.py

An escape sequence counts as *one* character

The following example shows that each escape sequence counts as one character:

```
s = 'one\ntwo\tthree'
print (s)
print (len(s))
```

The output:

```
>>>
one
two     three
13
```

http://www.annedawson.net/python3programs.html

08-12.py

Escape Sequences

https://docs.python.org/2.0/ref/strings.html

Escape Sequences video: https://youtu.be/8SGzhngNTzM

String iteration and membership

The following example shows that you can iterate over strings in loops using `for` statements and test for membership with the `in` expression operator:

```
s = 'Anne was here'
for c in s:
    print (c, end=" ")
print ('w' in s, end=" ")
print (' ' in s, end=" ")
print ('x' in s)
```

http://www.annedawson.net/python3programs.html

08-13.py

The output:

```
>>>
A n n e   w a s   h e r e True True False
```

Unicode characters within a string

The following example shows how to insert special characters into a string. Run the example program and make sure to read the comments.

```
# For explanation go to:
# http://www.network-theory.co.uk/docs/pytut/tut_17.html
# For character charts go to:
# http://www.unicode.org/charts/
# http://www.unicode.org/charts/PDF/U2580.pdf (Block Elements)

# \u2588 is a Full Block which can be used to build up a black square
str1 = u'Hello\u2588out there' # solid black block within text
print (str1)
```

http://www.annedawson.net/python3programs.html

08-14.py

String Methods

The example programs `08-09.py` and `08-10.py` show the use of the `find` and `replace` string methods.

`find` and `replace` are just two of the many built-in string methods available in Python. . .

String Methods

The following web page from the Python organisation lists all the available string methods to date:

`https://docs.python.org/3/library/stdtypes.html#string-methods`

String Methods video: https://youtu.be/oEw8A2Gt7QE

Strings

- Most real-world Python programs contain strings.
- Strings allow you to collect characters, so that you can treat them as a group.
- Strings have left-to-right positional ordering, with index capability.
- Unlike lists, strings are *immutable* which means that they cannot be changed. But new string objects can be created from existing string objects.
- Strings are homogeneous in that they consist only of characters.

Strings

- Most real-world Python programs contain strings.
- Strings allow you to collect characters, so that you can treat them as a group.
- Strings have left-to-right positional ordering, with index capability.
- Unlike lists, strings are *immutable* which means that they cannot be changed. But new string objects can be created from existing string objects.
- Strings are homogeneous in that they consist only of characters.

Strings

- Most real-world Python programs contain strings.
- Strings allow you to collect characters, so that you can treat them as a group.
- Strings have left-to-right positional ordering, with index capability.
- Unlike lists, strings are *immutable* which means that they cannot be changed. But new string objects can be created from existing string objects.
- Strings are homogeneous in that they consist only of characters.

Strings

- Most real-world Python programs contain strings.
- Strings allow you to collect characters, so that you can treat them as a group.
- Strings have left-to-right positional ordering, with index capability.
- Unlike lists, strings are *immutable* which means that they cannot be changed. But new string objects can be created from existing string objects.
- Strings are homogeneous in that they consist only of characters.

Strings

- Most real-world Python programs contain strings.
- Strings allow you to collect characters, so that you can treat them as a group.
- Strings have left-to-right positional ordering, with index capability.
- Unlike lists, strings are *immutable* which means that they cannot be changed. But new string objects can be created from existing string objects.
- Strings are homogeneous in that they consist only of characters.

Strings

- Most real-world Python programs contain strings.
- Strings allow you to collect characters, so that you can treat them as a group.
- Strings have left-to-right positional ordering, with index capability.
- Unlike lists, strings are *immutable* which means that they cannot be changed. But new string objects can be created from existing string objects.
- Strings are homogeneous in that they consist only of characters.

This Presentation uses the following program files:

```
08-01.py
08-02.py
08-03.py
08-04.py
08-05.py
08-06.py
08-07.py
05-12.py
05-13.py
08-08.py
08-09.py
08-10.py
08-11.py
08-12.py
08-13.py
08-14.py

See all programs at:
http://annedawson.net/python3programs.html
```

End of Python_Strings.ppt

Last updated: Tuesday 2nd January 2018, 7:43 PT, AD