Recap

- From lecture 4
  - Use the `len()` function to calculate how many characters are in a string
  - Obtain a single character from a string
  - Slice strings
  - Concatenate strings
At the end of this lecture, students should be able to:

- use dot notation when using string functions with string instances
- use string functions: `upper()`, `lower()`, `strip()`, `find()`, `rfind()`
- use the inbuilt functions: `min()`, `max()`, `round()`, `abs()`
Every object type, as well as storing some data, has some defined functions which can be applied to that particular type of object.

Variables which reference an object are called instances, e.g., in the following code, greeting is a **string instance** and number is an **instance of type int**.

```
greeting = "Hello World"
number = 234
```

String instances have many functions which can be applied to them such as upper(), lower(), find(), strip(), isalpha(), isdigit(), rfind(), split() ... In this lecture we will look at a few of these functions.

In order to apply a function to an object we use **dot notation**, i.e., the variable name, followed by a dot, followed by the function name.

```
instance_name .function_name(...)  
```

Note that, functions use parentheses (round brackets) after the function name.
String functions – upper(), lower()

- The **upper() function** returns a new string object with all the characters converted to upper case. The **lower() function** returns a new string object with all the characters converted to lower case.

- For example,

```python
greeting = "Hello World"
greeting_lower = greeting.lower()
greeting_upper = greeting.upper()
print(greeting, greeting_lower, greeting_upper)
```

Hello World hello world HELLO WORLD

Notice that there are a total of three string objects.
The **find() function** is used to look for the position (index number) of the first occurrence (from the left) of some characters. If the characters are found, the find() function returns the index number, otherwise the function returns -1.

For example,

```python
greeting = "Hello World"
position1 = greeting.find(" ")
position2 = greeting.find("z")
position3 = greeting.find("orl")
print(position1, position2, position3)
```

```
5 -1 7
```
String functions – rfind()

- The **rfind() function** is used to look for the position of the last occurrence of some characters (i.e. the rightmost position). If the characters are found, the rfind() function returns the index number, otherwise the function returns -1.

- For example,

```python
greeting = "Hello World"
position1 = greeting.find("o")
position2 = greeting.rfind("o")
position3 = greeting.rfind("orl")
position4 = greeting.rfind("lro")
print(position1, position2, position3, position4)
```

```
4 7 7 -1
```
The **strip() function** removes all white space from the beginning and end of the string. It does not remove spaces from inside the string.

For example,

```python
letters1 = "    H e l l o o o o o o   "
letters2 = letters1.strip()
length1 = len(letters1)
length2 = len(letters2)
print(length1, length2, "***", letters1, "***", letters2, "***")
```

```
21 14 *** Hello oooo *** Hello oooo ***
```

Notice that there are two string objects
Exercise

Complete the following program so that it prints the initial from the first name followed by a full stop, a space and followed by the surname. Assume the full name is always two names separated by a single space.

```python
full_name = "Wystan Auden"

initialled_name = first_letter + ". " + last_name
print(initialled_name)
```

W. Auden
**Common Python built-in functions**

- **`min()`** is an **built-in function** which can be used to find the smallest number from a comma separated set of numbers and **`max()`** is the **built-in function** which can be used to find the largest number from a comma separated set of numbers, e.g.,

```
num1 = 32
num2 = 16
smallest = min(num1, num2)
print(smallest)
smallest = min(32.7, 56.4, 3, -1.1, 56.99, -1.2)
print(smallest)
```

```
largest = max(num1, num2)
print(largest)
largest = max(32.7, 56.4, 3, -1.1, 56.99, -1.2)
print(largest)
```
Common Python built-in functions

- The **built-in function**, `round()`, is used to round numbers to the closest whole number (or rounded to a number of digits after the decimal point), e.g.,

  ```python
  num1 = 32.657123
  num2 = 16.48926
  num3 = -16.48926
  print(round(num1))
  print(round(num2))
  print(round(num3))
  print()
  print(round(num1, 2))
  print(round(num2, 3))
  print(round(num3, 4))
  ```

  
  
  
  ```
  33
  16
  -16
  32.66
  16.489
  -16.4893
  ```
Common Python built-in functions

- Note that the **function, round()** with a single argument returns an int number and that rounding an int returns the int unchanged, e.g.,

```python
print("round(32.657123, 0): ", round(32.657123, 0))
print("round(16.48926, 0): ", round(16.48926, 0))
print("round(32.657123): ", round(32.657123))
print("round(16.48926): ", round(16.48926))
print("round(24.0, 0): ", round(24.0, 0))
print("round(24.0, 1): ", round(24.0, 1))
print("round(24, 0): ", round(24, 0))
print("round(24.0): ", round(24.0))
print("round(24): ", round(24))
```

```
round(32.657123, 0): 33.0
round(16.48926, 0): 16.0
round(32.657123): 33
round(16.48926): 16
round(24.0, 0): 24.0
round(24.0, 1): 24.0
round(24, 0): 24
round(24.0): 24
round(24): 24
```
sometimes the round() function seems to give an unexpected result e.g.,

```
num1 = 1.5
num2 = 2.5
num3 = 3.5
print(round(num1))
print(round(num2))  # surprising result
print(round(num3))
```

this problem happens because floating point numbers are stored in a finite space, e.g., 0.1 has an infinite number of digits when converted to base 2 but, when stored in the computer memory, float numbers are assigned 64 bits which means that some of the bits are cut off. Perhaps 2.5 is actually stored as 2.499999999999999999...99 which is rounded to 2 (not the expected 3).
The **built-in function**, `abs()`, is used to get the absolute value (the magnitude) of a number (int or float), e.g.,

```python
num1 = 32
num2 = -32
num3 = abs(num2 - num1)

print(abs(num1))
print(abs(num2))
print(num3)
```

```
32
32
64
```
Complete the following program so that it prints the total tax and the net pay rounded to a whole number. The first $14000 is not taxed. The next amount up to $38000 is taxed at 24% and the rest is taxed at 34%.

```python
salary = 54000
no_tax_boundary = 14000
rate1_boundary = 38000
rate1 = 0.24
rate2 = 0.34

print("=" * 32)
print("Total tax: $", total_tax, sep = "")
print()
print("Net pay: $", net_pay, sep = "")
print("=" * 32)
```
In Python:

- use dot notation when using string functions with string instances
- the string functions: `upper()`, `lower()`, `strip()`, `find()`, `rfind()` can be used with string instances
- the built-in functions: `min()`, `max()`, `round()` can be used
Examples of Python features used in this lecture

greeting = "Hello World"
position1 = greeting.find("o")
position2 = greeting.rfind("o")
position3 = words.find("Z")
position4 = words.rfind("o W")

greeting_lower = greeting.lower()
greeting_upper = greeting.upper()

smallest = min(32.7, 56.4, 3, -1.1, 56.99, -1.2)
largest = max(32.7, 56.4, 3, -1.1, 56.99, -1.2)

num1 = 32.657123
print(round(num1))
print(round(num1, 2))

num2 = abs(20 - num1)
print(num2)