Introduction to Computer Programming

2.2 User input & nested conditional statements

University of BRISTOL

`input`

A function that dynamically accepts input from a user while a program is running.

Takes one *argument*,

The argument is a string, which is the prompt displayed to the user.

The program then accepts typed input from the user.

The program outputs the typed input as string data.

In [32]:
```python
name = input("Enter your name: ")

print(name, type(name))
```

```
Enter your name: Hemma
Hemma <class 'str'>
```

In [3]:
```python
print(input("Enter your name: ")) # type response when prompted
```

```
Enter your name: Hemma
Hemma
```

This is a quick and easy way to add dynamic input to your program.

Input - a word of warning!

The data input by the user, regardless of type, is output by the `input` function as a string.
Numbers entered must be *cast* as a numerical data type to use as numerical values.
Example: `+` will join string data, unless re-cast.

In [33]:
```python
a = input("Enter a number ")
b = input("Enter another number ")
```

```
Enter a number 4
Enter another number 5
```

In [34]:
```python
print(a + b)
```

```
45
```

In [35]:
```python
print(int(a) + int(b))
```

```
9
```

Example

The volume control on a keyboard has a 'volume up' and 'volume down' button.

We will represent these buttons as inputs from a user:

- u : increase volume by 1
- d : decrease volume by 1

Write a program that shifts the current volume up or down depending on the input from the user.

In [ ]:

Nested conditional statements

Conditional statements can be nested
(a conditional statement within a conditional statement).

This allows more complex decision making in a program.

Example

Write a program that checks a number, `x` , and:

- prints `"positive"` if the number is positive
- prints `"negative"` if the number is negative

In [ ]:   1

If the number is positive the program should *also* print:

- `"square"` if `x` is a square number
  (a number of the form $x = n^2$ where n is an integer)
- `"not square"` otherwise

In [ ]:   1

Example

How could we edit the volume control program in the previous example to prevent the volume from ever going outside of the range 0 to 10?

Hint: There are multiple ways to achieve this using techniques from week 1/2

In [ ]:   1

Summary

- `input` :
  - accepts typed input from the user.
  - outputs the typed input as string data!
- Conditional statements can be nested, increasing the number of paths the program can take.

Questions?