Functions – Little Factories

Let's learn about functions. We will be using them regularly in our course.

Example of a function: print()

```
In [1]: print("print() is a function!")
```

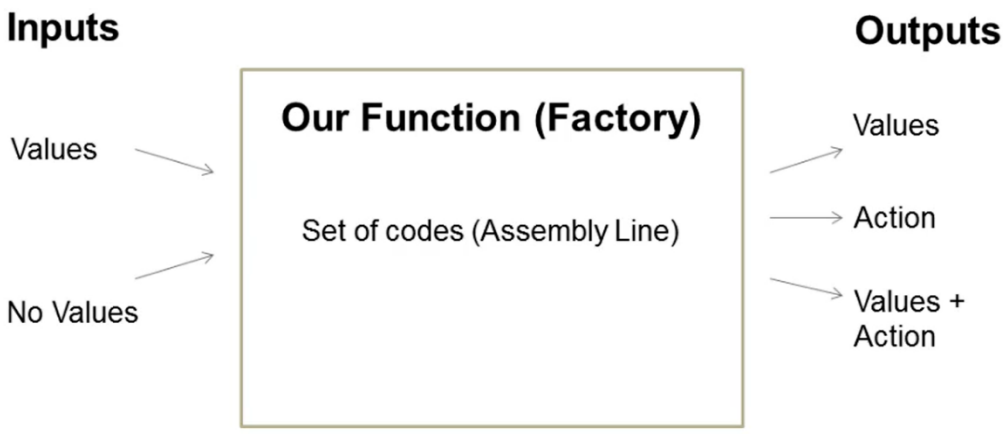
```
print() is a function!
```

What are functions?

A function is a set of code that takes in inputs and releases outputs (like factories!)

Inputs are: 1) Values, or 2) there might be no inputs

Outputs are: 1) Values, 2) Actions, or both 3) Values + Actions



What does a function look like?

A function looks like a word followed by parentheses.

Types of functions

There are 2 types of functions, user-defined or in-built

In-built functions

In-built functions are made by the creators of Python

E.g. print(), range(), len()

User-defined functions

User-defined functions are made by us. You can name it any way you want.

E.g. lucas1(), donkeykong(), square(), greet(), get_data()

These are just random names. We will show you how to code them in the next lecture.

Inputs and outputs

Let's analyse some examples:

Case study 1: print(input)

Inputs: A text to be printed (this is a value)

Outputs: Prints a text (this is an action)

```
In [1]: print("print() is a function!")
        print() is a function!
```

Case study 2: len(input)

Inputs: A text or a list of items (this is a value)

Outputs: If the input is a text, the output is the number of characters of that text. If the input is a list, the output is the number of items in the list (this is a value)

```
In [12]: len("Input!!!")
Out[12]: 8
```

Why use functions

- Convenience: It is easier for us to use a function like len() instead of writing 20 lines of code every time we want to check the length of a list.
- Neat: It is easy to read the code and makes the code modular[1].
- Easy for us/others to read and use: We can create functions for others to use. Likewise, the creators of Python or other coders can create many standardised functions for us to use.

[1] Modular code refers to code that is designed such that each part of the code can exist in a standalone manner.

E.g. The print(something1) function only relies on the input "something1" and will not break even if you mess up your code elsewhere.

Building and calling functions

Before we can use a user-defined function, we need to build the function.

E.g. of a user-defined function that adds 2 values together

```
# "sum_two_values" is the name of the function
# "something1" and "something2" are 2 inputs
# "def" is shorthand for define.
# "def" is a standardised code to tell Python that we are creating a user-defined function

def sum_two_values(something1, something2):

    # This is the factory assembly line that converts inputs to outputs
    result = something1 + something2

    return result # This is the output. It returns a value.
```

In the next lecture, we will show you how to do this.

Note: In-built functions do not need to be built. You can just use it directly.