

# Control Abstraction

# What is abstraction?

The goal of abstraction is to hide details.

For example, if we want to load an image, the real logic might look like this:

- Open file
- Read bytes
- Construct Image object
- Return object

This is painful, it's much easier if we can just say:

- Load image

That, in a nutshell, is abstraction.

# What does it let us do?

Good abstractions have many benefits, they:

- Let us reuse bits of code
- Make it easier to modify
- Make our code easier to understand
- Make it easier to extend

# Control Abstraction

This is about being able to refer to a complex sequence of steps in a simple way. Functions are the key tool you will use to do this.

The benefits of this are:


- It is easy to understand what some code is doing (see load image example)
- If we want to change the behaviour we just change it in one place
- We can easily use the behaviour in many places, or share it with others.

Good functions should:

- Do one thing only
- Have a descriptive name
- Should not have side effects


# Functions

Good



```
def product(items):  
    total = 1  
    for i in items:  
        total *= i  
    return total
```

Bad



```
def f(items):  
    total = 1  
    for i in items:  
        count += 1  
        total *= i  
    total *= count  
    return total
```