Teaching London Computing

Programming for GCSE
Topic 8.1: Functions

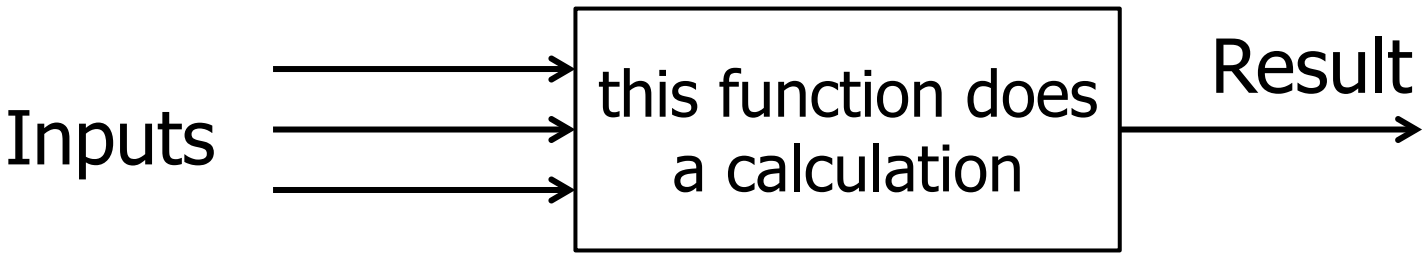
William Marsh
School of Electronic Engineering and Computer Science
Queen Mary University of London

Outline

- Functions
  - What's the big idea?
  - Syntax
- Variables in functions: scope
  - Name clashes
- Functions that make changes
  - Global variables

What's a Function?

- A part of a program with
  - A name
  - Parameters
  - Result



BIG IDEA

- This IS A **BIG** IDEA

- Building blocks of a program
  - Big programs cannot be made in one piece
  - Use 'blocks' from another programmer (*library*)

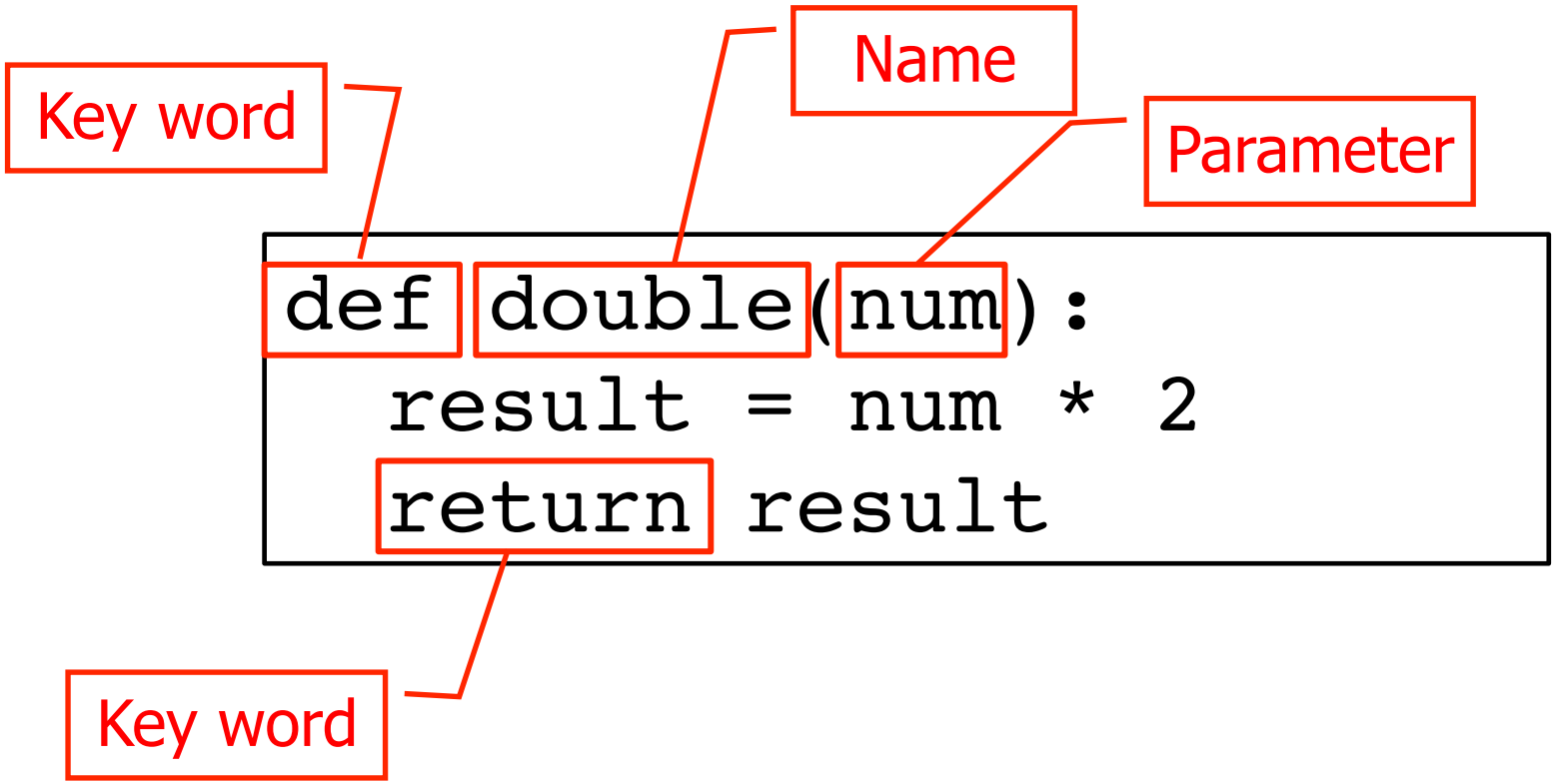
- Naming parts of a program
  - Name the function → behaviour

Simple Function Example

Defining and calling a function

Definition of a Function

- A function is a NOT a complete program



Calling a Function – I

```
def double(num):
    result = num * 2
    return result
```

- Call the function

Function call

```
anum = int(input("A number:"))
anum = double(anum)
anum = double(anum)
print("Now doubled twice:", anum)
```

Calling a Function – II

```
def double(num):
    result = num * 2
    return result
```

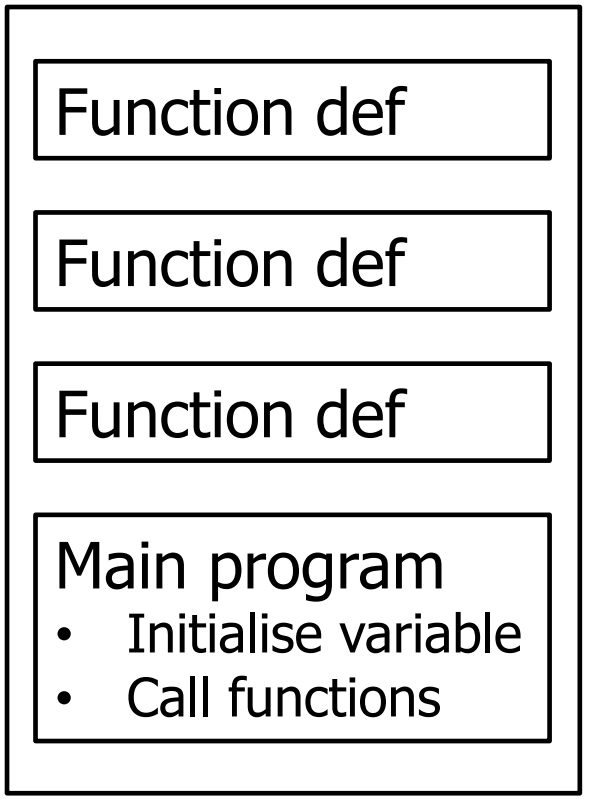
- Call the function

Function call

```
anum = int(input("A number:"))
anum = double(double(anum))
print("Now doubled twice:", anum)
```

Program Order

- Write the functions first
  - One function can call another (providing it is defined first)
  - Do not put one function inside another
- The 'main' program calls the functions



Words, Words ...

- You **define** (or declare) a function
- A function has **parameters**
- You **call** a function
- You **pass** a value to a function
- ... it **returns** a result
- The function creates a new scope
- Functions are also called
  - Procedures
  - Subroutines
  - Methods
  - ... and more

Example

- Create a function that is passed a name and prints the string "Hello XXXX"
  - Choose a suitable name
- Change the function to capitalise the name
  - Choose a new name

Example Solution

```
def greetMe(name):
  print("Hello", name)
```

```
def greetMeLoudly(name):
  print("Hello", name.upper())
```

Variables in Functions

The idea of 'scope'

Variable Scope

- Function create a 'box'

```
def double(num):
    result = num * 2
    return result
```

- Variable 'result' is a 'local' variable
  - It only exists inside the box
- 'num' can be used like a variable
  - It is given a value in the call

Scope: Simple Version

- The variables used inside a function are **totally separate** from other variables
  - Appear when function is called
  - Disappear afterwards

- Name clash: confusing variables inside and outside a function
  - Use different names

Functions that Make Changes

Some more complex and less essential ideas

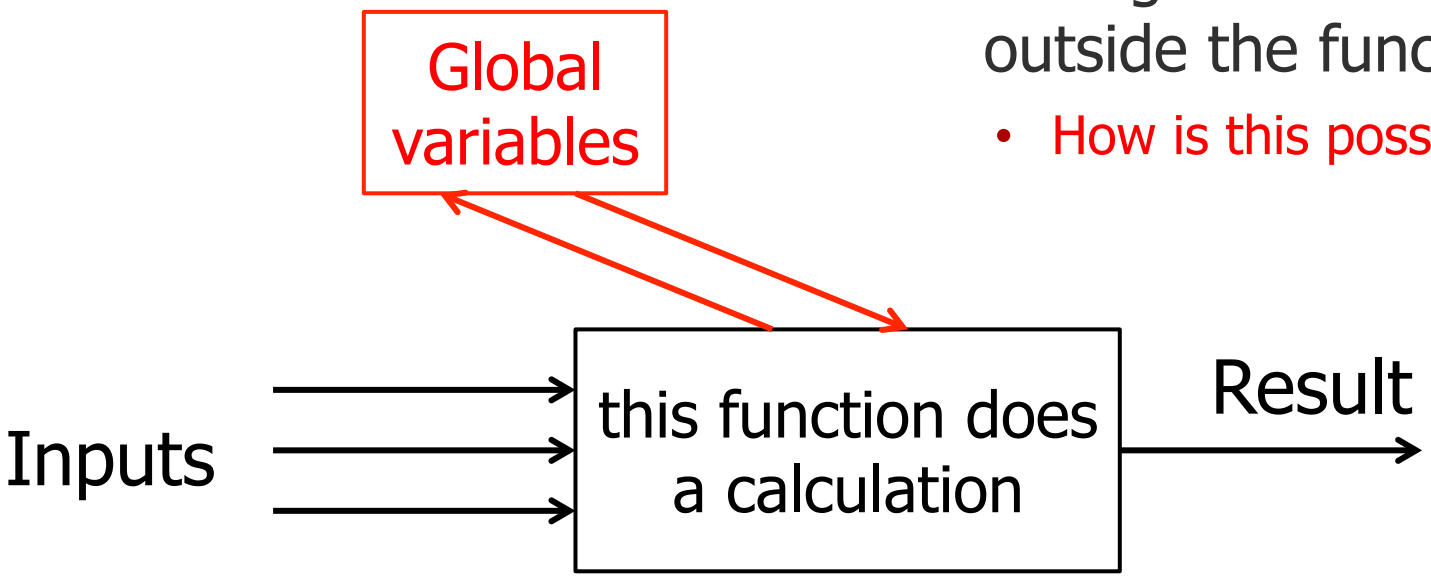
What is the Effect of a Function?

No effect

- Return a value
  - Nothing changes!

Effect

- Print something
  - File output too
- Change value of a variable outside the function
  - How is this possible?



Global Variables

- Local: inside a function
- Global: outside a function
  - Variable inside (local) and outside (global) not totally separate

```
def double() :
    global num
    num = num * 2

num = 10
double()
double()
print(num)
```

Using a List as a Parameter

- When a list is used as a parameter, you can change it

```
def addZero(mylist):
    mylist.append(0)

herList = [1,2,3]
addZero(herList)
print(herList)
```

```
>>>
[1, 2, 3, 0]
>>>
```

Parameters and Assignment

- There is a close parallel between

```
def myFunction(param) :
    ... statements

num = 10
myFunction(num)
```

```
num = 10
param = num
... statements
```

- Parameter passing is like assignment

Syllabus and Teaching Issues

Syllabus – Functions

- Writing functions is AS not GCSE (OCR)
  - … but lots of related ideas
- So why learn functions?
  - Using functions e.g. 'len'
  - Planning solutions: breaking down a problem in parts
  - … some students will teach themselves

Summary

- Programming is problem solving
- Problems are solved in steps
- Functions are for step-by-step programming

- Defining functions is not essential for GCSE
  - Using them is!