ENGG1003 - Thursday Week 5

Functions and Arrays

Sarah Johnson

University of Newcastle

26 March, 2021

The Story So Far

- Course summary:
  - Variables and data types
  - Arrays (via numpy)
  - Plotting (via mathplotlib)
  - Flow control
    - `if`
    - `while`
    - `for`
  - Functions
- Today: Arrays and functions together

Example 1 - Access an Array Without Modification

- Write a function which takes as input an array x and calculates it's minimum value, maximum value and its average value

Arrays and Functions

- Let's re-write our code from Monday so that our ball height function works with arrays:

```
1 include numpy as np
2
3 def ball_height(v0, t, g):
4     return v0*t - 0.5*g*t**2
5
6 v0 = 5
7 t = np.linspace(0,10,100)
8 y = ball_height(v0, t, 9.81)
```

Arrays and Functions

- What actually changed? Went from this:

```
1 def ball_height(v0, t, g):
2     return v0*t - 0.5*g*t**2
3
4 v0 = 5
5 t = 0.6
6 y = ball_height(v0, t, 9.81)
```

- to this:

```
1 include numpy as np
2 def ball_height(v0, t, g):
3     return v0*t - 0.5*g*t**2
4
5 v0 = 5
6 t = np.linspace(0,10,100)
7 y = ball_height(v0, t, 9.81)
```

Arrays and Functions

- Python is hiding a lot from us here
- Advantage: code easy to read / easy to write
- Disadvantage: it's very easy to write code which is not doing what you think it is doing
  - have a look through the Discord chat for issues that arise when arrays are confused with a single variable (scalar)
- Technically: a pointer to the array is being passed to the function not a copy of the array.
  - Pointer = the *memory address* of the array's start
- In practice: this means that any modifications to the array remain after the function is finished.

Key Points

- Because arrays are passed via a pointer the function gets *the actual array*
- Modifying the array in the function modifies the original variable
- You don't *need* a return value
  - In a technically incorrect way: all the array's elements are "returned"

Example 2

- Lets write and test this live...
- Write a function which:
  - zeros any negative values in an array x
  - creates a new array y which keeps the positive values in x and has zeros in place of the negative values in x

When should functions be used?

- When should functions be used?
- Well, what do they achieve?
  - *Much* easier to solve problems when they're broken down into sub-tasks
  - Reduce code line count and complexity (if they are called multiple times)
  - Allows code re-use between projects
  - *Much* easier to perform project management between multiple programmers
  - Bugs in a function are easier to fix than a bug in code which has been copy+pasted multiple times
  - ...the list goes on

When should functions be used?

- What about in an ENGG1003 context?
  - Vague rule of thumb? More 10-20 lines or so in your code - consider breaking up into functions
  - Break a big problem into multiple sub-problems
    - Implement each as their own function
    - Yes, even if they are only called once
    - Do what you feel is most "readable"
    - Your opinion here will change with experience

More Information

- Further Reading: Section 4.1 of the course textbook
- More Practice: All the exercises in Section 4.3 of the course textbook