Programming Basics

Motivation

This tutorial aims to get you up to speed and ready to learn a programming language. It covers the basic concepts and why you might want to program. It is not a technical document. Footnotes are for extra information, if they are confusing please ignore them.

Why Programming?

Programming is necessary because computers are after all not magic. They, like everything else, need to be set up properly. Programming is not even unique to computers, think about programming a concert or a ceremony. It is really just a fancy way of saying 'setting up'.

There's a lot of gory detail behind the scenes, just like when you drive a car: you steer the wheel and hundreds of levers, sprockets and springs do their thing and turn the wheels. Programming is exactly the same, so that you can just type in 1+2 and out comes 3.

Languages

You have probably heard about programming languages, and they're also pretty simple. Spoken languages all ultimately perform the same task: communication between people. Well, a programming language is your way of communicating with the computer.

Jargon

You will hear *a lot* of jargon when you learn to program. It is unavoidable, but ultimately only a slight nuisance. Most words you can quickly search as and when needed. Below are a few words that you will hear *all the time*.

Syntax

Syntax is a fun one that you will hear all the time for two reasons.

1. It sounds cool, I mean 'do you even syntax, bro?'
2. It is pretty important - it means '*how* you write something' So when someone says "this is the syntax for addition in Java", they are really saying "this is how you would add two numbers in Java".

Functions (Methods)

Functions (also called methods[1]) do things with variables. Often these involves calculations. *Calling* a function means running it with some input (arguments) if it needs any. A function *declaration* states what the function does, and a function *definition* what the function should *return* and its *parameters*.

Parameters vs. Arguments

This seems to trip a lot of people up. Parameters are what you put in the function declaration/definition. This is what the function expects as inputs. Arguments are what you give the function. In order for your function to work when called, the arguments should match the type (see next section) of the parameters.

Variables

Variables store data. Just like how a strongbox stores your most treasured (or forbidden) items. However, not all variables can store the same things. What they can store depends on their type.

Type

This can get technical very quickly—there is a whole theory for this: type theory. But really its simple. The data we store in our variables come in different flavours. Types categorise data based on how the computer handles the data.

Numbers have a few different types. Can you guess a few? Well, yes you've got primes, odds, evens etc. but a computer doesn't need to handle them differently. But it does with integers (whole numbers) and floats/doubles (decimal numbers).

A good analogy for this is food. There are five different types of food, called groups: grains, protein, dairy, vegetables and fruit. Why do we categorise them? "*Foods are grouped together because they provide similar amounts of the key nutrients of that food group.*" (EatForHealth.gov)

And types are grouped based on how the computer stores and handles them.

The basic types you will need to know are: integers, floats, strings, characters, and arrays. There are a lot more, but they will often be explained when you come across them.

- Integers (whole numbers) - $0, 1, 2, \ldots$
- Floats (decimal numbers) - $1.234, 1.235, \ldots$
- Characters (single) - 'a' or '1' or '@'
- Arrays (lists of common type) - $\{1, 2, 3, 4\}$

Therefore '1' is **not** the same as the integer 1 which is **not** the same as the float 1.000 which is **not** the same as the array $\{1\}$.

Files

Files are beautifully simple. There are only two kinds of file, a file and a *directory*. A directory is a file that can hold other files. A file is always at its core just a bunch of numbers. File extensions even, have no meaning to the computer, they're just a convention that make our lives easier. So when you're operating system 'knows' to open a `.pdf` in a pdf reader it just has some special program for that. If you remove the `.pdf`, the file will still open in a pdf reader (unless it is being

too smart).

We write our programs in files, just like you write your diary on paper.

Mathematics

Your maths skills need not be particularly extensive to program. I'm assuming you know how to add, subtract, multiply and divide. There are more advanced mathematical operators you will come across, but again they can be explored as and when needed.

One area you will need to know (and might not) is boolean logic. Just as we can say $1 + 2 = 3$, we can say that $1 + 2 = 4$ is FALSE. This can be captured in boolean logic. There are only two values: **True** and **False**.

We can do interesting things with this idea, for instance if I say:

$$1 + 2 = 3 \wedge 1 + 4 = 4 \tag{1}$$

this statement is clearly false because only half of it is true. This is AND (commonly written & or $\wedge$). You can think of it like multiplication. It takes two numbers and gives an answer, only here the answer is: True or False.

Two true statements ANDED is clearly a true statement:

"1 + 2 = 3 AND 1 + 4 = 5"

And two false statements ANDED is clearly a false statement: "1 + 4 = 3 AND 2 + 4 = 5"

So AND works like:

`&`	**True**	**False**
True	True	False
False	False	True

We also have OR (commonly written | or v), two statements ORed together is true as long as at least one of them is true. This can be confusing because in English when we say or, we usually mean that only one of them is true.

Hence, "1 + 2 = 3 OR 1 + 4 = 5" is true, "1 + 2 = 3 OR 1 + 2 = 5" is also true, and "1 + 4 = 3 OR 2 + 4 = 5" is false.

So OR works like:

`|`	**True**	**False**
True	True	True
False	True	False

There are more boolean operators, just like there are more mathematical operators, but we need not be concerned with them for now.

---

1. there is a slight technical difference between the two, methods are functions used by a class in OOP↩