

Session 2

Introduction to Programming - Continued

- ✓ **Structure of a Program**
- ✓ **Variables and Types**
- ✓ **Keywords**

Structure of a Program

In the previous session we compared a program to a recipe, a series of steps that must be followed in order. Just like a recipe a computer program is a series of steps that are followed in order, from top to bottom. Our programs are read by a computer in the same way, from top to bottom. In a computer program we call these lines **instructions** or **statements**. Most programming languages have a special character that defines the end of a statement, in C based languages it is a semi-colon. See below for an example of a statement.

```
this could be a statement;  
this would be another statement;
```

Programming languages have very strict rules defining their structures, this makes it easier for the computer to read and understand the program. A clear structure also means that other programmers can easily pick up and understand other people's code. These strict rules can often trip up new programmers, things will 'look right' but not work as expected. Stick with it and team up with a more experienced programmer who has already been tripped up by these common situations.



Exercise

Let's get started with your first program!

In this course we will be using the Python programming language. Follow the instructions on this website for getting a Python interpreter installed on your system. <https://>

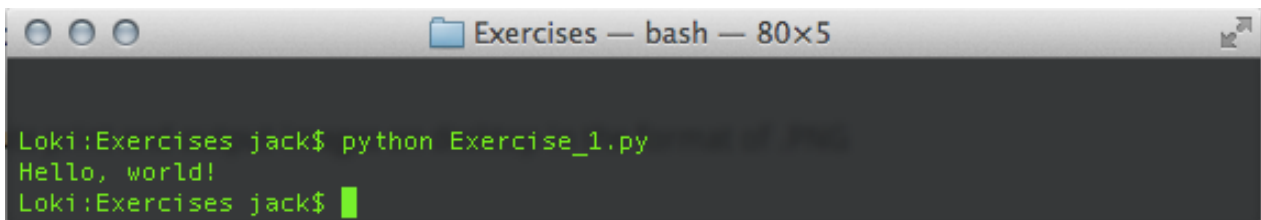
www.python.org/about/gettingstarted/, there is plenty of documentation and help available at the Python website so if you are having trouble getting things running you should check the website's help sections <https://docs.python.org/2/index.html>.

As well as getting Python installed you should get a text editor that is designed to help you code. For this course we recommend something simple such as Notepad++ (<http://www.notepad-plus-plus.org/>) or Sublime Text (<http://www.sublimetext.com/>). From this point on we will assume that you have a working text editor and are able to run Python programs from the command line.

Ok, your first program should always be a hello world program. In Python this is extremely simple, you probably will remember it from the previous chapter.

```
print "Hello, world!"
```

Put that into a file and save it as Exercise_1.py then run it, you should see the following happen.

A terminal window titled "Exercises — bash — 80x5" is shown. The prompt is "Loki:Exercises jack\$". The user enters "python Exercise_1.py" and the terminal outputs "Hello, world!". The prompt returns to "Loki:Exercises jack\$".

```
Loki:Exercises jack$ python Exercise_1.py
Hello, world!
Loki:Exercises jack$
```

Congratulations! That's your first program done, if you keep following the exercises you'll be writing mobile applications, computer games and more in no time!

Variables and Types

As previously discussed a program is a collection of instructions, during the running of a program it's very likely that the programmer will want to store data in between instructions. For example the program might want to store the name of the person who is running it; we store these bits of information in **variables**. Variables are temporary stores of information, they exist for the life time of the program. What we mean by this is that a variable has to be loaded each time the program is run, it is not stored on your

computer when the program ends. Variables can be stored by writing them to something more permanent such as a text file or a database. Variables are given names to describe them, this means that the program can access the contents of a variable by using the variable name. Think of it like a label on a tin, you find a tin that says soup for example and inside that tin is the actual soup, the type of soup could be completely different but it comes under the soup variable name. Let's take a look at some example variables -

```
name = "Jack"  
name = "George"
```

In the above example the variable name is 'name' and the contents of that variable is 'Jack' or 'George'.



Note

A variable can only contain one value, the variable 'name' can only contain 'Jack' or 'George' but not both at the same time.

We have established that variables are containers of data, the next thing to learn is that data in computer programs all have **types**. A type describes what the data is, such as text or a number or a decimal number. The type of a variable is more important in some languages than others. In Python the type of a variable is generally not something to worry about. Let's look at some different variables that contain different types of data.

```
name    = "Jack"    // string  
height = 182       // integer  
weight = 78.5      // float
```

The previous example shows three different types. A **string**, an **integer** and a **float**. These are the three main data types that are common amongst all languages. Different languages will also define additional types that will be more complicated than the ones described here.



Exercise

In this exercise you are going to write a program that contains variables. You will fill these

variables from the within the program and then print them out to the user. Enter the following program into a file and save it as `Exercise_2.py`, run the program to see the output.

```
name = "Jack"  
height = 182  
weight = 78.5  
  
print "My name is {0} and I am {1}cm tall and weigh {2}  
kg.".format(name, height, weight)
```



Note

Take note that the print statement should be written on one line, there is no line break after the `{2}`. Change the variables to describe your self and people that you know. See how you can only have one value at a time in each variable.

You may have encountered variables in mathematics, for example x and y are frequently used to represent unknown numbers. Variables in programming can be used in combination with mathematical operators to produce new values. The basic arithmetic operators (addition, subtraction, multiplication and division) are supported in all programming languages. Some examples can be seen below, see if you can calculate what the program will output before running it.

```
x = 10  
y = 5  
  
print x + y    # addition  
print x - y    # subtraction  
print x * y    # multiplication  
print x / y    # division
```

We have talked about some data types that you will likely recognise but there is another very common data type that will be used throughout. The **boolean** data type represents a **true** or **false** value. We will frequently use boolean values to compare two variables, if they match then it is true, if they do not match then it is false. As well as comparing values to see if they match you can use boolean operators to compare two values to see if one is greater than or less than the other, these are called **relational operators**. An

example of boolean comparison is shown below.



Note

In the comparison it is comparing the value inside the variable and not the variable name itself.

```
x = 10  
y = 5
```

```
print x is x  
print x is y
```

Boolean conditionals are the same as you will have seen in mathematics. The operators are the greater than (>) sign, less than (<) sign, equals sign (==) and the not equals sign (!=). Take note that the equality conditional is a double equals sign, the single equals sign is used to assign values to variables. The greater than and less than operators can be combined with an equals sign to make them greater than or equal to (>=) and less than or equal to (<=). These symbols are summarised in the table below.

Less than	<
Less than or equal to	<=
Greater than	>
Greater than or equal to	>=
Equal to	==
Not equal to	!=

Keywords

Keywords are special words in a programming language that form the structure of a program. Key`words cannot be used as variable names, for example **if** is a commonly used keyword in programming languages but **x** is not, so you could have a variable called **x** but not **if**.

In our example programs we have already seen a common keyword - **print** is a very useful keyword that writes the string after the keyword out to the command line. Print

will be one of the most used keywords in this course.

Keywords can be used to control the flow of our program through the boolean logic operators **if** and **else**. **if** is a common keyword that you will see in most languages, it allows certain parts of code to be run if a certain condition is true; the optional **else** is the second part of an if-else block and it defines a piece of code that will be run if the **if** condition is false. This might seem complicated at first but remember that if-else blocks simply run on true or false and switch the flow of the program based on that. The next session will focus on the if-else conditional.