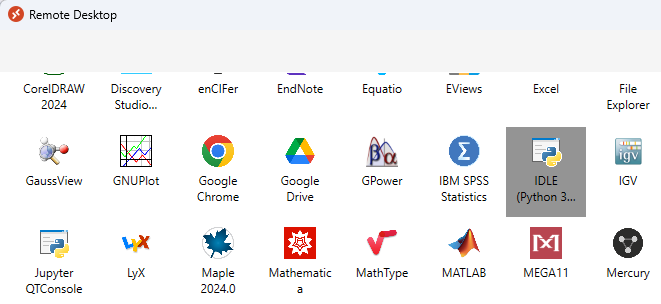
Algorithms & Data Structures

Week 1 -Variables & Expressions

# PART I - Using the interpreter command line

## Getting started

The first thing you need to do is open a Python shell. For simplicity and to ensure the experience is the same for all, you need to use Azure Virtual Desktop Services (See [Virtual desktop service (VDS) - IT Services, University of York](https://www.york.ac.uk/it-services/tools/vds/) for more information). Open the remote desktop app and double click on the IDLE (Python 3) icon (see Figure 1a). After you enter your UoY credentials, a new window (IDLE shell) will open as shown in Figure 1b (Note, program version may vary). You will use this shell to type your code for the time being. Toward the end of the exercises, you will be saving your code in a file.

 A screenshot of a computer

Description automatically generated

1. (b)

Figure 1

## First lines of code

A Python shell could be considered as a big calculator where you can do calculations using some mathematical expression. The basic element of an expression is a value. In Figure 2, the first line of code represents a value, which is a number and has value 1. You can also use known operators such as addition, multiplication, etc. Such combinations of operators and operands is known as an expression. Once you have typed an expression and pressed the return key, the expression is evaluated, and the result is printed in blue on the line below.

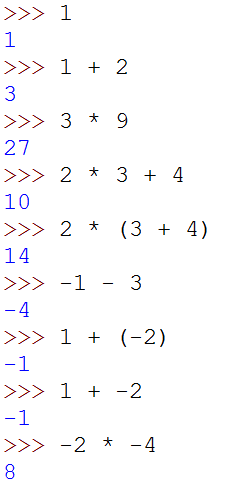


Figure 2

As you know mathematical operators have rules of precedence, it is the same here. The use of parenthesis can be used to counteract the rules of precedence between operators as shown in Figure 2, lines of code 4 and 5.

When considering the division operator // something strange happens as shown in Figure 3. 3//2 gives 1. This is called the **“integer division**” operator. What are the results of -3/2, -3/-2 and -3.0/-2?

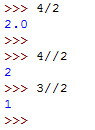


Figure 3

Even old calculators (e.g. when I was still at school) had a way to memorise values that can be reused later. You have the same principle in Python. Such objects are called variables. In Figure 4 you declare a variable x and assign the value 2 to it. The = symbol is called the assignment operator, x is the name of the variable, and 2 is the value assigned to x. Every time you use x it is the same (almost) as using the value 2. You can use x in any expression, you can have more than one variable (here x and y); you can also use several variables in the same expression as shown in Figure 4. The sixth line of code shows how you can change the value in a variable. From now on, x has the value 5 in memory and does not remember having the value 2.

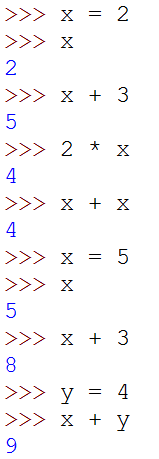


Figure 4

Using x and y as names for variables is allowed, however it is not very meaningful. In Figure 5 the variable names are very explicit and therefore it is not difficult to understand what the aim of the program is. You are strongly encouraged to do the same thing. You MUST read the [PEP0008 Style Guide for Python Code](http://www.python.org/dev/peps/pep-0008/).

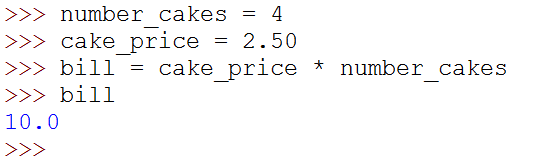


Figure 5

## Your first error (and probably not the last one)

What happens if You misspell the name of a variable? You will get an error as you are asking the interpreter to use something it does not know. The interpreter will promptly show you its discontentment in red as shown in Figure 6.

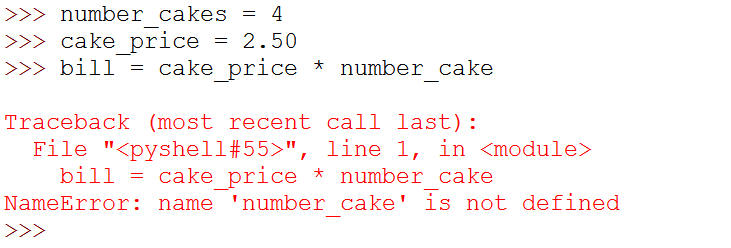


Figure 6

It is very important for you to recognise the type of error and try to fix it. A Python interpreter tries its best to help you in this task. Here it points to the line containing an error <bill = ….> followed by the type of error <NameError:…>.

In your case it tells us that number\_cake is not defined. Indeed, you misspelt the name of the variable forgetting the ‘s’ at the end. When you encounter an error that you are not familiar with, I encourage you to search for an explanation online. If it proves unsuccessful, ask your tutor.

## Another data type

So far you have seen two types of numbers, does Python have any other type of values? The answer to that question is yes, Python does have quite a few more types. The next one you are going to see represents a series of characters to form words or sentences for example.

Values of this type are called strings. Figure 7 shows a string representing a sentence. Python has two ways of representing a string, using single quotes or double quotes.

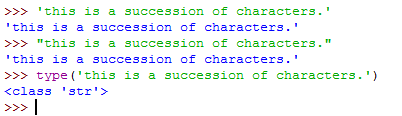


Figure 7

In the same way as numbers, strings can be assigned to variables (Figure 8) and you can use the function print to print the string value to the console. For example, the statement print(words) displays the content of the variable words to the console.

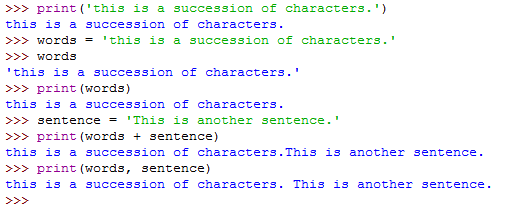


Figure 8

To concatenate two strings, e.g. build one string from two substrings, you can use the addition operator +. For example:

longString = words+sentence

We can also print several strings in the same statement by separating them with a comma. Note that by doing so a space will be added between the two strings.

## More errors

You should be careful when declaring strings that you are using the same delimiter at the start and end of the string (first error in Figure 9).

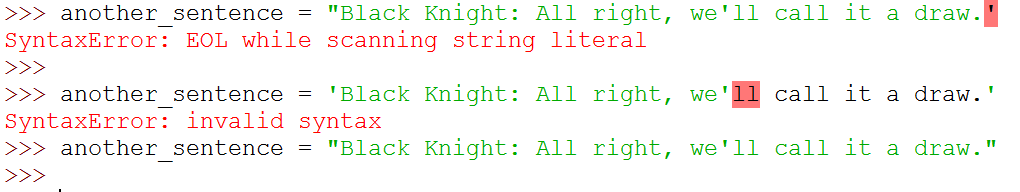


Figure 9

What about the second error? What can You do if You want to use single quote or double quotes in a string? Look on the web to find the answer.

## First attempt at a small program

Let’s try to create a small program that computes a bill for a cake shop. You would like to get the number of cakes, the price of a single cake and then print the total price of the bill. Figure 10 shows such a small program where the number of cakes is four.

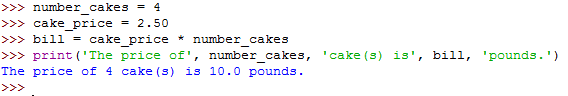


Figure 10

Now if you want to compute a bill for seven cakes, you need to rewrite the code as shown in Figure 11. As you can see the only change between the two scripts is the assignment of the value 7 to the variable number\_cakes.

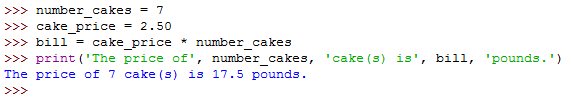


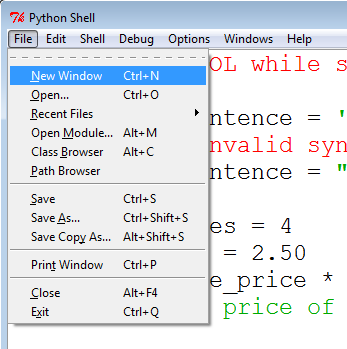
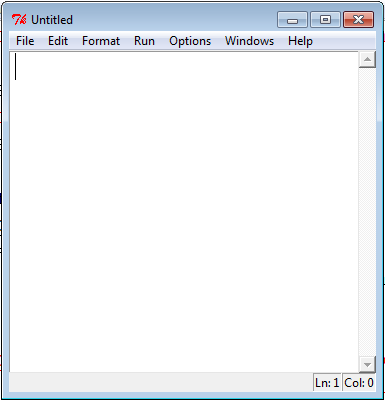
Figure 11

And if You want to do that again You will have to retype everything one more time. This is clearly not efficient. What can you do to change that? This is the subject of Part II.

# PART II - Create a module

## How to create a module?

To solve the problem of rewriting the same code over and over, you will use a file to store the lines of codes. In Python it is known as a Module. You will talk a little bit more about modules later during the year. It is an important concept, and you will need more time to discuss it in depth. So, for the moment just consider a module as a place to store your code so it can be reused/run multiple times.



(a)

(b)

Figure 12

From the Python shell select the file menu and click on the <New Window> item (Figure 12a). A new window will open that is different from the Python shell (Figure 12b). This is the place where you will be writing our code.

Rewrite the code in the window as shown in Figure 13. Note that some part of the code is highlighted with different colours. Before you can run the code,you need to save the file.

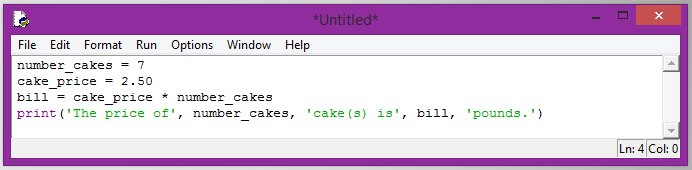


Figure 13

Select save as from the menu and type the name of the module, it must start with a letter and by convention contains only lowercase letters, numbers, and underscores. You must **manually** add the .py extension to keep the colour highlighting.

**Troubleshooting:** if your code appears in black only, this means you did not add the extension .py in your filename. Repeat the **“save as”** operation with the correct extension to solve the problem.

## Running my first program

Once you have written our code, the next step is to run it. From the module window containing our code, select Run from the menu and click on the <Run Module> item. Alternatively press the function key F5 (Figure 14). You must save your changes before running the program to run the latest version. Python will remind you if you have forgotten.

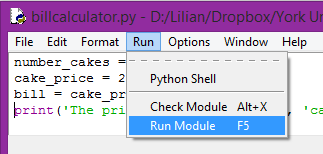


Figure 14

The result of the execution will appear in the Python shell as shown in Figure 15.

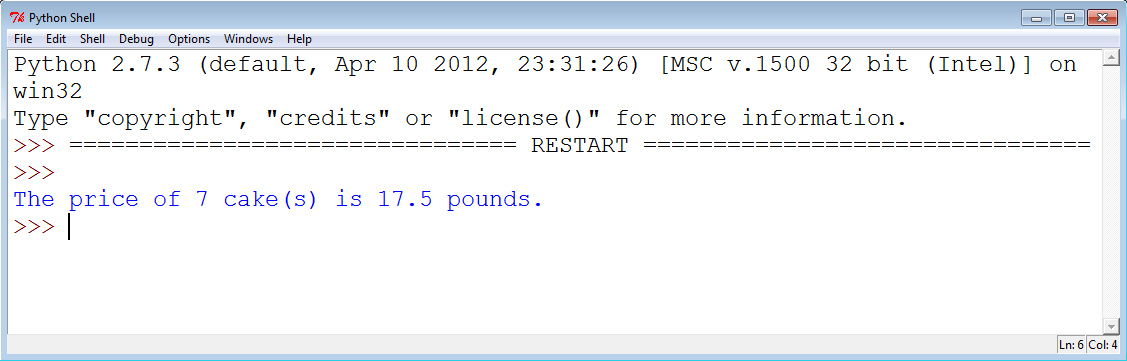


Figure 15

What if you want to compute the bill for nine cakes? In that case you just change the value 7 to 9 in the module, save the changes, and run the program again. Figure 16 shows the new result.

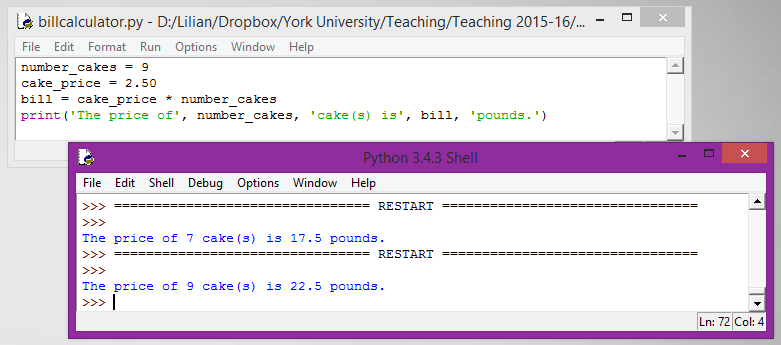


Figure 16

## User input

Using modules is an improvement, however every time You want to change the number of cakes, you need to change a value in my code, save the changes and run the program. This is not satisfactory. It would be much better if you could use a statement to ask users to enter new values without changing the code.

Luckily such a statement exists, it uses the function input(string). If you are looking at the first line of code in Figure 17, the statement uses the function input. How does it work?

input is the name of the function you want to use (same as for len). Between parentheses you have a string containing the message you want to display to the user; it is called a **parameter** of the **function** input.

On the left-hand side of the assignment is a variable named number\_cakes, It means that the **value returned** by the function input will be **assigned/stored** into the variable number\_cakes. The value entered by the user can be used via the variable number\_cakes.

The returned value of the input function is always string (even if you enter 3.0) via the keyboard. For this reason you convert (cast) the returned value into an integer using the cast operator int(…).

Figure 17 shows the values returned by the program when run twice (without changing the code), and where the user entered two different inputs (3 and 5).

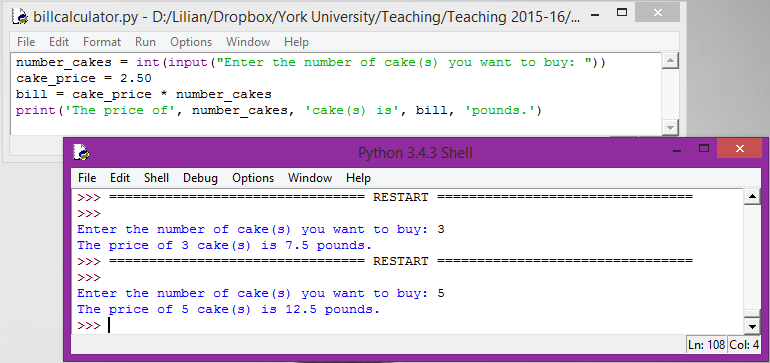


Figure 17

This is an important step; you now have a single program that can return different values depending on user input (user being human or machine) without any change to the code.

## Next step

Create another program in a module named converter.py, that converts the weight of a person from stones and pounds to kilograms. The program requests two integer inputs from the user, one for the stones and one for the pounds, and prints the weight of the person in kilograms.