# SOFTWARE 1 PRACTICAL

## ELEMENTARY PROGRAMMING

Week 1 – Exercise 01b

## How to create a module?

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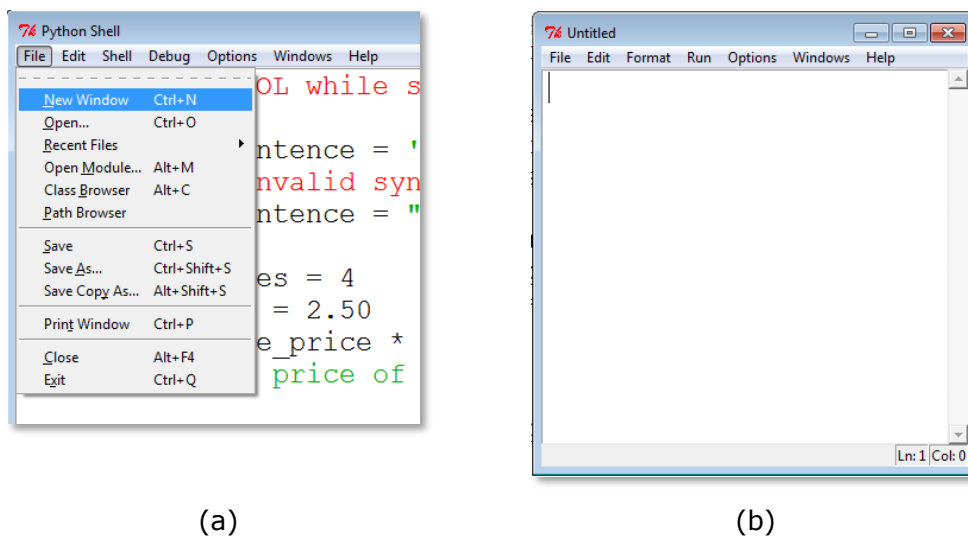


(a)                                    (b)

**Figure 1**

From the Python shell select the file menu and click on the <New Window> item (Figure 1a). A new window will open that is different from the Python shell (Figure 1b). This is the place where we will be writing our code.

Rewrite the code in the window as shown in Figure 2. Note that some part of the code is highlighted with different colours. Before we can run the code, we need to save the file.
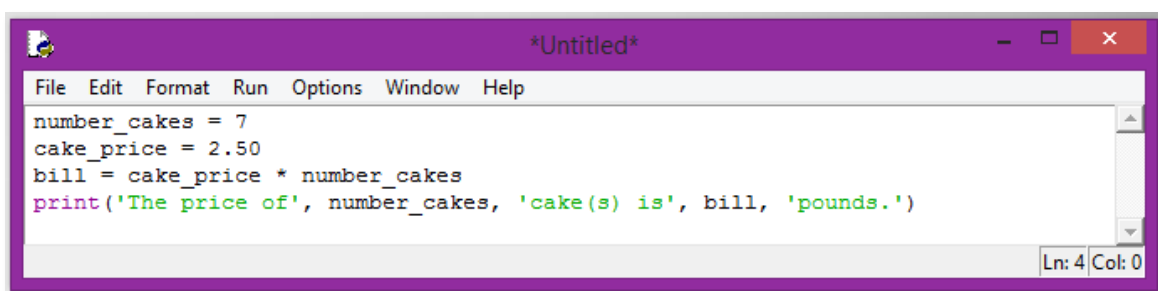


```
number_cakes = 7
cake_price = 2.50
bill = cake_price * number_cakes
print('The price of', number_cakes, 'cake(s) is', bill, 'pounds.')
```

**Figure 2**

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 **<u>manually</u>** add the `.py` extension to keep the colour highlighting.

Trouble shooting: 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 we 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 3). You must save your changes before running the program in order to run the latest version. Python will remind you if you have forgotten.
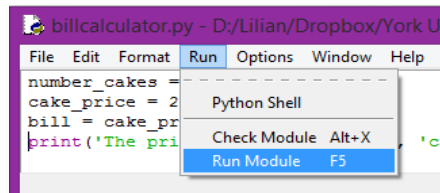


**Figure 3**

The result of the execution will appear in the Python shell as shown in Figure 4.
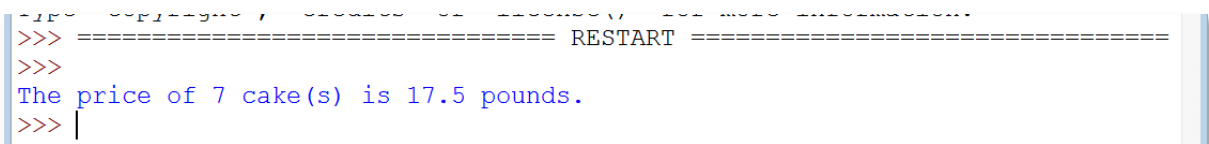


**Figure 4**

What if I want to compute the bill for nine cakes? In that case I just change the value 7 to 9 in the module, save the changes, and run the program again. Figure 5 shows the new result.
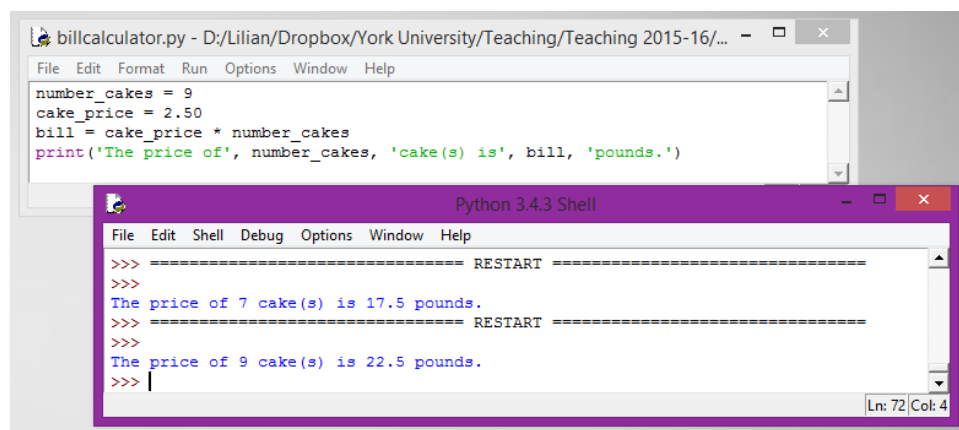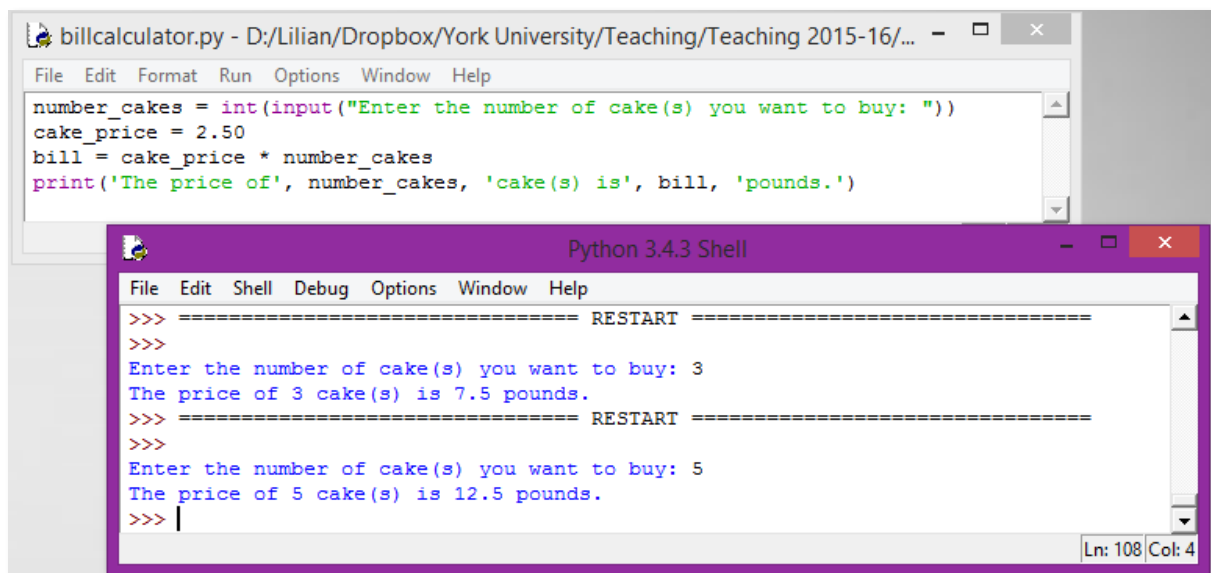


**Figure 5**

## User input

Using modules is an improvement, however every time I want to change the number of cakes, I 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 I 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 6, the statement uses the function `input`. How does it work?

- `input` is the name of the function we want to use (same as for `len`)

- Between parentheses we have a string containing the message we 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 we convert (cast) the returned value into an integer using the cast operator `int(...)`.

Figure 6 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 6**

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