Accelerated Programming 2

Python Programming Exercises

You should do these programming exercises in the level 2 laboratory (Boyd Orr 706). A tutor will be available throughout the laboratory sessions to give you any necessary help.

Sample solutions to all the exercises will be posted later at the Accelerated Programming 2 Moodle site[1]. *Attempt each exercise before consulting the sample solution.*

Some of the exercises are marked *optional*. Attempt the optional exercises if time permits. Attempt all the non-optional exercises.

The Level 2 Laboratory

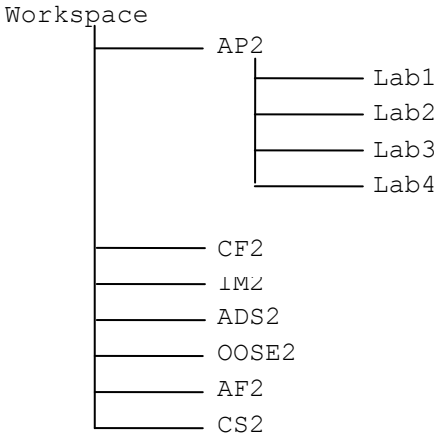
All computers in the level 2 laboratory use the Windows 7 operating system.

If you are unfamiliar with the Windows system, your tutor will give you a short introduction. If you are familiar with an older Windows system but not Windows 7, you should find it easy to adapt.

To log in, press CTRL+ALT+DELETE (depress all three keys simultaneously), then enter your login name and password. To log out, press CTRL+ALT+DELETE, then click on the Logout option.

You will use a system called IDLE for developing Python programs. Once you log in, you can launch IDLE simply by double-clicking the IDLE icon on the desktop. Exercise 1A will help you to become familiar with IDLE.

On the desktop you will also find a shortcut to a folder named `Workspace`. This is your own private folder on the file server, and you can access it from any computer in the laboratory. Inside it you will find a sub-folder for each of your Computing Science courses, including AP2. Inside the `AP2` folder you will create new folders `Lab1`, `Lab2`, `Lab3`, and `Lab4` for your AP2 lab exercises.



[1] Computing Science → Level 2 → Accelerated Programming 2

Exercises for Lab 1

Inside your `Workspace\AP2` folder create a new folder named `Lab1`. Then work through the following exercises.

1A. *(IDLE familiarization)*

Launch IDLE by double-clicking the icon on the desktop. This opens a window named Python Shell. In this window you will see a prompt "`>>>`", which invites you to enter a Python expression or statement.

(a) Enter the following expressions and statements:

```
m = 7
n = 3
m + n
m * n
m // n
m % n
m**2
greeting = 'hullo'
greeting
print greeting
print greeting + ' sailor'
```

At the end of each line enter RETURN. Observe the results. Note that, when you enter an expression, it is evaluated and its result is displayed; when you enter an assignment-statement, it is executed silently.

(b) Select the New-Window command (in the File menu). IDLE opens a new window in which you can compose and edit Python code. Complete the following function definition and type it into the window:

```
def cube (x):
    # Return the 3rd power of x.
    …
```

Notice that IDLE automatically indents the code following "`:`". IDLE also automatically highlights keywords, comments, etc., in distinctive colours.

Using the Save-As command (in the File menu), store your code in a file named `familiar.py` (in your `Lab1` folder).

Select the Run-Module command (in the Run menu). This causes IDLE to compile and run the code. If IDLE reports a syntax error, edit the code to correct the error, then select the Run-Module command again. Note that "running" a function definition does not call the function, but simply makes the function available to be called.

Go back to the Python Shell window. Test the function by entering expressions that call the function with suitable arguments, e.g., "`cube(2)`" and "`cube(1000)`". If the results are incorrect, edit the code to correct the error, then try again.

(c) Complete the following function and add it to the file `familiar.py`:

```
def is_odd (n):
    # Return True iff the integer n is odd.
    …
```

Save the file, then select the Run-Module command. Test the function by entering suitable expressions in the Python Shell window.

1B. *(Integer arithmetic)*

Write a program containing variables and functions useful for tax calculations. Assume also that tax is due on the amount by which a person's income exceeds the current *tax-free allowance* (£10,000), and is calculated at the current *tax rate* (20%). For example, an income of £8,000 should incur no tax, whilst an income of £12,000 should incur £400 in tax.

Assume that every sum of money is a whole number of pounds. When calculating the tax amount, any fraction of a pound should be discarded. For example, an income of £11,234 should incur £246 in tax (not £246.80).

Your program should contain the following variables and functions:

- `allowance` should contain the current tax-free allowance;
- `rate` should contain the current tax rate, i.e., the amount of tax payable per £100;
- `taxable(income)` should return `True` iff `income` exceeds the tax-free allowance;
- `tax(income)` should return the amount of tax payable on `income`.

Store your program in a file named `tax.py` (in your `Lab1` folder). Test your program thoroughly.

Test your program again with different values for `allowance` and `rate`.

(*Hint:* You might find it convenient to use an if-statement. You can find an example in the course notes, slide 2-15. If-statements will be covered fully in lecture 4.)

Submit a printout of `familiar.py` to your tutor.