FIT1008 Introduction to Computer Science
Practical Session 1

Semester 2, 2014

Objectives of this practical session

- For you to get to know each other and your demonstrator.
- To introduce you to some of the software you will be using in this unit.
- To make you aware of the requirements for documenting your code.

Pracs in FIT1008

Please read the "*FIT1008 PracGuide*" document on Moodle which provides more details of how pracs are organised for this unit.
As may not have had any lectures as yet, this prac does not require any preparation. However, in general you need to be aware that the pracs in this unit are longer and more complicated than those you have seen in previous units. If you do not make a start on your program before your class, you will soon realise that you **do not have enough time** to complete it. You also need to be aware that pracs are always marked during the lab. Do not leave the lab class before your demonstrator has marked your work and explained your mark to you!
For the first few pracs you will work alone but soon you will be required to work in pairs. Although you should not share code, you are welcome to discuss the principles behind the prac with other students in your lab class to help each other understand them, and it is a good idea to get to know your classmates.

Task 1

Using one of the following IDEs, *idle3*, *PyCharm* or *Eclipse*, run the following four Python modules.[1]

[1] copies are available on Moodle.

hello.py

```
1 name = input('Enter␣name␣(max␣60␣chars):␣')
2 print('Hello␣' + name + '.␣Welcome')
```

convert_temp.py

```
1 temp_C = int(input('Enter␣temperature␣in␣Celsius␣'))
2 temp_F = int(9*temp_C/5 + 32)
3 print('Temperature␣in␣Fahrenheit␣is␣' + str(temp_F))
```

leap_year.py

```
year = int(input('Enter year: '))

if ((year % 4 == 0) and (year % 100 != 0))
            or (year % 400 == 0):
    print(year, 'is a leap year')
else:
    print(year, 'is NOT a leap year')
```

construct_list.py

```
size = int(input("Enter number of values: "))
the_list = []

for i in range(size):
    the_list.append(int(input("Value: ")))

print(the_list)
```

Task 2

We will use *Pydoc* to generate documentation for your Python modules. The Python documentation is written, in terms of *docstrings*, alongside the code itself and not in a separate file. For this course, we expect you to note the following features in your code:[2]

[2] Read the sample code we give you, and follow its example, and you'll be all right.

A description of each class and function.

@author	the author(s) of a file, a class, a function, etc. If you write a whole module, no need to repeat this one on all the subparts of the module
@since	the date you wrote the file, function, class, etc
@modified	last time the file, function, class etc was modified
@param	not relevant for modules, but mandatory for functions
@param	where you can have more than one @param descriptor (see below)
@pre	or @precondition: eventually required for all your functions
@post	or @postcondition: eventually required for all your functions
@complexity	eventually required for all your functions

(i) For each of the Python modules in Task 1, add documentation using docstrings.

(ii) Using *Pydoc* generate HTML files for each of the modules.