EMAT10007 – Introduction to Computer Programming

Exercises – Week 1. Introduction to Python

1. Familiarise yourself with PyCharm.

   (a) Open PyCharm and complete the following set-up tasks:

   - Select "Create New Project"
   - Give your project an appropriate name. "Week1" will do for now. Check that the location selected is an appropriate place in your user space.
   - Next, drop-down the "Project Interpreter" section, to reveal two check-boxes. Make sure to select "New environment using `Virtualenv`".
   - Under "Base interpreter", ensure that the version reflects `Python3.X`.
   - Then click "Create". You may be asked a few extra set-up questions related to your preferences, such as colour-schemes.
   - Locate the python console (usually bottom left) and enter the following:
     `Hello world`
     then press enter. What is the output? What happens if, instead, you enter:
     `"Hello world"`
     including quotation marks?
   - The python interpreter *interprets* input line-by-line, and so expects properly structured input every time you press return (enter). When we added quotation marks, it recognised the input as a string, and simply prints the string to the console.

   (b) A few of PyCharm's features to note:

   - Most IDEs like PyCharm provide many features to help you when writing programs.
   - Some of the most useful ones are:
     - Syntax highlighting: PyCharm will colour different parts of your code to help you easily identify things like `keywords`, `variables`, `types` (strings, numbers, and Boolean values are coloured differently).
     - Inline errors/warnings: if you see red or grey underlines in your code, hover over the area to see what the problem is, and possibly a hint on how to fix it.
     - Documentation: you can easily access the documentation for a function by selecting that function, and then pressing `ctrl + Q`. This will bring up a floating box detailing the definition of the function, including the parameters expected.
   - There are many more features that you will discover during the course, so explore the menus, and search the internet if you're wondering what useful features you might be missing out on!

2. **Python as a calculator.** The Python console by itself can be used as a calculator: you can input operations, assign values to variables, and store the results of operations for use in additional calculations.
   In the interpreter, do `import math` to extend the range of mathematical functions that are available. Do `help(math)` to find what has been added.

   (a) Variables

- The `math` module we just imported contains variables of its own that are intended to be used, but never modified, also known as "constants". Enter the following into the console to see the values of the two constants provided by the `math` module:
  - `math.pi`
  - `math.e`

  (It is useful to know that these exist, but don't worry too much about them for now)
- Other variables are assigned values and (may) change during the execution of your program. In many languages, to assign a value to a variable, we just use a single "equals" sign: `=`. Assignment looks like this:
  ```
  x = 10
  y = 5
  ```
- Variable names should start with a letter, and may contain letters or numbers. Be aware that some `keywords` are reserved by the Python language and cannot be used as variable names. Try the following:
  ```
  True = 1
  ```
  What happens here, and why? Now try:
  ```
  true = 1
  ```
  Note: Python is case-sensitive, so be careful when naming and using your variables! For a full list of keywords reserved by Python, enter the following:
  ```
  import keyword
  keyword.kwlist
  ```

(b) Numbers and operators

- Create two variables called "A" and "B", and assign a value of your choice to each of these variables.
- To calculate the addition of A and B, simply enter: `A+B`.
- Now enter `A*(A+B)`.
- What happens if you enter `A*A+B` instead? Python follows the same ordering of mathematical operations as any other calculator.
- Set `A = 10` and `B = 3`. Now enter `A/B` to calculate the division of A and B. What happens if you enter `A//B` instead?
- Now try `A%B` (`%` is the modulus operator, and calculated the remainder of a division).
- **(Optional)** Try the same operations above but assign A to be a random number between 1 and 10. You will need to use two functions for this:
  `random.randint()` - you will need to `import random` to access this function, and include `random.` prior to the function name, to let Python know that the `randint()` function is part of the `random` module.
  Refer to the python documentations to learn about the parameters for `randint()`.

3. Text (otherwise known as strings)

(a) Strings

- Create two variables called "A" and "B" - if continuing from the last exercise in the console, simply reassigning values to A and B will work, because Python is *dynamically typed* (if you're interested, you can research this term later).

- Assign `A` to "Hello" and `B` to "world". The quotation marks indicate that these are strings.
- Join these variables together by *adding* `A` and `B`:
  `A + B`
- Notice how the string is missing a space between "Hello" and "world". Combine `A` and `B`, with a new space in the middle, and assign this new string to a new variable `C`.
- Print the length of this new string by entering `len(C)`, where `len()` is a built-in function in Python which simply returns the *length* of something. What kinds of variables does `len()` work on? Have a look in the docs.

(b) Booleans

- Create two variables called "A" and "B", and assign some numeric values to each of them.
- Print the results of:
  `A < B`
  `A > B`
  `A == B` *(Note: Two equals signs)*
  You can also use `<=` for $\leq$ and `>=` for $\geq$ comparisons.
- What happens if you write `not` in front of one of the lines above?
- You can also set A and B to be Boolean truth values themselves, such as: `A = True` and `B = False`.
- Now try some logical (Boolean) operations such as:
  `A and B`
  `A or B`
  `A and not B`
- These will be important for conditional statements, as we will see in the next section.

Note: This next part is dependent on how much time remains in the first session, and we are not expecting you to complete the entire sheet unless time allows.

4. Download the code examples on blackboard, and either open them in PyCharm, or copy and paste the code into a blank `.py` python file. Try to run the programs from within PyCharm. These are:

   - `SumFirstTenIntegers.py`
   - `HowManySquares.py`
   - `HowOldAreYou.py`
   - `NumberGuessing.py`

5. If you haven't already done so, it would be advisable to create a Python file within your project going forward so you can start saving your work, and so that you can write larger programs and edit them easily.
   Provided you already have a project open, right-click the project folder you created, select `New ¬ Python file` and then give it a name, i.e. "ConditionalStatements" for the upcoming section.

6. Conditional Statements

   (a) If-then-else

   - We are now going to look at how Boolean statements can be used to control the flow of the program. To do so, we shall use `if` and `else` statements. We can combine these statements to check whether certain *conditions* evaluate to `True` or `False`, hence "conditional statements".
   - Open the `HowOldAreYou.py` file and take a look at the use of conditional statements. Run the file, and follow the instructions in the console. Pay attention to what is happening with the if-statements, and which parts of the code produce output.
   - The code is made up of `if-elif-else`, where `elif` is Python's shorthand for writing `else if`, which is a condition that is only checked if the previous condition evaluates to `False`. If all conditions do not evaluate to `True`, then the `else` statement is interpreted.
   - **Note:** Python relies on using the correct indentation levels, and so the final `print()` statement is always interpreted, whereas because the others are indented, these are only evaluated if the above conditions are satisfied.

7. Loops

   (a) For

   - For-loops are typically used when you know how many times you need to repeat something, before ending the loop. You specify a limit to the number of loops you wish to run the code for, and then the loop will automatically stop.
   - Open and run the `SumFirstTenIntegers.py`, reading the comments as additional exercises.
   - Notice that in this file, the loop is written in the following format:
     `for value in range()`
     Why do we use the value 11 here? What are the values of the range?

   (b) While

   - While-loops are used when the number of times the program needs to loop is not always known beforehand. The while-loop contains a condition to check at the beginning of each loop, and will continue to loop over the contained code until the condition fails.
   - Open and run the `HowManySquares.py`, reading the comments as additional exercises.

- In this example, we don't (usually) know when the cumulative sum will exceed $1,000,000$. Therefore, we use a `while` loop until the condition no longer evaluates to `True`, and the sum exceeds this limit.

Checklist

- Met your TA - they will be mentoring you throughout the course, and will be there to answer question and provide feedback along the way.
- Setup PyCharm to use for this unit. If you are downloading this on your own computer, you will also require a Python 3 installation.
- Practice with using the Python interpreter as a useful calculator with access to some powerful abilities provided in "modules". You'll learn more about these during this course.
- Check that you understand the basics: variables, different types of variables (numbers (integers, floats), Booleans, strings), the different built-in operators, and how these work with both numbers and strings.
  Do not worry if you haven't completed all the exercises! We will be going over them again next week, with an emphasis on writing your own code during the exercises.

Additional resource

- You may also be interested in this tutorial on learning to use PyCharm - `https://www.lynda.com/Python-tutorials/Learning-Python-PyCharm/590828-2.html`. You will need to "Sign-in with your organization portal", by entering the `bristol.ac.uk` which will allow you to use Bristol's single sign-in portal.
- Of course, there are many online tutorials for both Python and PyCharm, so if you're wondering what more you can do with either of these, please do search online or ask your TAs for pointers.