EMAT10007 – Introduction to Computer Programming

Exercises – Week 2. Types, Variables, Strings, Lists, Loops

1. **(Re-)familiarise yourself with PyCharm.** If you haven't already done so, please follow along with the demonstration for setting up a PyCharm project, or refer to last week's exercise sheet for step-by-step instructions.

   - Check that you know how to create a project, choosing to create a new virtual environment and selecting **Python 3** as the language version.

2. **PyCharm's help menu.** PyCharm includes several very useful resources listed under the `Help` menu. Important ones we wish to point out here are:

   - `Help ¬ Getting Started` and `Help ¬ Demos and Screencasts` - will take you to videos introducing you to PyCharm, covering the basics of using the IDE and its many features, as well as a brief look at different programming-related tasks you may wish to conduct in PyCharm. You may find some of these helpful.
   - `Help ¬ Productivity Guide` is very useful for identifying features you *have* and *have not* used (as well as when you last used them). Clicking on an item will provide details about a feature, including the shortcut keys to access the feature and a visual illustration of it in use. Try finding out more about `code completion` which we mentioned briefly last week.

---

3. **Variables.** You may have attempted variants of these exercises last week, so work through them at your own pace and make sure you are comfortable with the different types of variables. You should use the Python console for these exercises - typically found in the bottom-left corner of PyCharm.

   (a) General information

   - Variables are simply "labelled" pieces of data in a program that tend to change during the execution of your program.
     Variables that do not change are referred to as "constants". These might be used to represent approximations to numbers like π or e.
   - To assign a value to a variable, we just use a single "equals" sign: `=`. This is called "variable assignment" and looks like this:
     ```
     A = 3
     B = 5
     ```
   - Once a variable exists (i.e. that it has been assigned some value), it can be reassigned afterwards by simply reusing the name during an assignment. Try reassigning the above variables like so:
     ```
     A = 10
     B = 7
     ```
     If you type `A` and `B` now, then you'll see these new values, or you can check on the right-hand side. It should say something like `A = {int} 10` and `B = {int} 7`, indicating that both variables are integers. What happens if you set one of the

variables to equal `5.0`? Does the type inside of the brackets ({*type*}) change, and if so, why?

- You can even assign variables to be the values of other variables: `B = A`
  but what happens if we now change `A`? Does `B` also change?
- Variable names should start with a letter (though they may start with and underscore, _) and may contain letters, numbers, and underscores.
  Note: 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`
  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 *integer* values to each of them.
- 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?
- Try setting `C = A/B`, and then `C = A//B`, paying attention to `C`'s type changing on the right.
- 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 a function 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()`.

(c) Booleans

- Create two variables `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, this is a* comparison*, not an assignment.)*
  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.

(d) Text (otherwise known as 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`
  This is called "string concatenation".
- 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, or type `help(len)` in the Python console.
- What other operations can we perform on strings? Try the following:
  `A*3`
  `B.capitalize()`
  `A.lower()`
  `A.upper()`
  `C.title()`
  You will need to pay close attention to the changes in capitalisation of the different strings. For more information, or additional functions that can be used on strings, click here.

Demonstration: For the following exercises, while you *can* work in the Python console, we will be saving our progress in Python files so that we can make changes more easily. Please follow the brief demonstration on opening/creating/saving files in PyCharm.

4. If you didn't do so last week, download the code examples on Blackboard under "Lecture 1", 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. **Printing output when running a program.** When we write programs in the editor in the second half of these exercises, instead of directly printing the values of variables in the Python Console by simply typing in the variable name, such as:
   `A`
   `B`
   etc. we tend to use the `print()` function to produce **output**.
   To learn more about the `print()` function, try typing `help(print)` in the Python console.

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 sum exceeds this limit, and therefore the condition no longer evaluates to `True`.

8. Open Exercises

   (a) FizzBuzz

   - In FizzBuzz, we count from 1 to n, replacing any multiple of 3 with the word "Fizz" and any multiple of 5 with the word "Buzz" As follows:
     "1, 2, Fizz, 4, Buzz, Fizz, 7, 8, Fizz, Buzz, 11, Fizz, 13, 14, FizzBuzz, ...".
   - Create two variables called "Mult3" and "Mult5".
   - Set the values of the variables "Mult3" and "Mult5" to be strings of "Fizz" and "Buzz", respectively.
   - Create an additional variable "Limit", which will be the number we count up to.
   - At the beginning of your program, after you have assigned `Mult3` and `Mult5` their values, you will need to ask the user to **input** a value for `Limit`. This can be done using the `input()` function, which waits for the user to input some *string* when you run your program, before continuing.
     Hint: you will need to **convert** the input string created by `input()` into an integer, using `int()`.
   - The computer should say each number from 1 to `Limit`, replacing each multiple of 3 with the word "Fizz" and each multiple of 5 with "Buzz".
     To do so, you will need to use a **loop**.
   - You will also need to use the `%` operator, which returns the remainder of a division e.g. $4 \bmod 3 = 1$ and $15 \bmod 3 = 0$, indicating that 15 is a multiple of 3. You will need to check whether each number is either a multiple of 3, a multiple of 5, or *both.*
     Hint: Start with the basic loop, printing out each number, and then work on replacing it with "Fizz", "Buzz", or "FizzBuzz", in stages.

   (b) Number guessing game

   - Create two variables called "GuessedNumber" and "ChosenNumber".
   - Assign a random number to "ChosenNumber" between 1 and 10, using `random.randint()`, and ask the user to guess the number, assigning the user input to the variable "GuessedNumber".
     Hint: You will need to import `random`.
   - Make the program print out "Correct!" *if* the user has guessed the number correctly. *Otherwise*, print out "Incorrect!".
   - Now modify the program to repeatedly ask the user to guess a number until they guess correctly. Which kind of loop is most appropriate for this program?
   - Modify the program to provide additional information when the guess is incorrect e.g. "You guessed higher/lower than the chosen number."

9. **(Optional)** Brain teasers

(a) Multiples of 3 and 5:

If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23. Find the sum of all the multiples of 3 or 5 below 1000.

(b) Largest palindrome product:

A palindromic number reads the same both ways. The largest palindrome made from the product of two 2-digit numbers is $9009 = 91 \times 99$. Find the largest palindrome made from the product of two 3-digit numbers.

(c) Check out Project Euler for more brain teasers: https://projecteuler.net. (These are excellent to test your knowledge when learning any new programming language.)

Checklist

- You have recapped the basics - the different types of variables, variable assignment and operations on numbers, Booleans and strings.
- Learnt about conditional statements, and how these are used to control the flow of the program.
- Understand how loops can be used to iteratively repeat some parts of the program, allowing you to reduce the amount of code you need to write while increasing the amount of calculations your program can compute.