FIT1045 Algorithmic Problem Solving – Workshop 3.

Objectives

The **objectives of this workshop** are:

- To get familiar with Python and debugging code.

- To implement and manipulate the list and table data structures in Python.

- To implement iteration and selection in Python.

Useful Material

Iteration and Selection: Documentation on control statements is available at `https://docs.python.org/3/tutorial/controlflow.html`. A chapter on flow control is available at `https://automatetheboringstuff.com/chapter2/` (A link to this book is available at `https://wiki.python.org/moin/BeginnersGuide/NonProgrammers`).

Lists and Tuples: Documentation on lists is available at `https://docs.python.org/3/tutorial/datastructures.html`.

Task 0: Prelab

Use the Python interactive shell to experiment with the creation and manipulation of lists. A good introduction would be to complete the exercises at `http://www.diveintopython.net/native_data_types/lists.html`.

Task 1: Using IDLE's debugger

IDLE provides a simple debugger that you can use to step through your code *(without PythonTutor's restricted number of steps)*.
You will be using this debugger to step through a simple loop
In order to to this you will click on the **debug -> debugger** option from IDLE's main menu and a window will open as below:
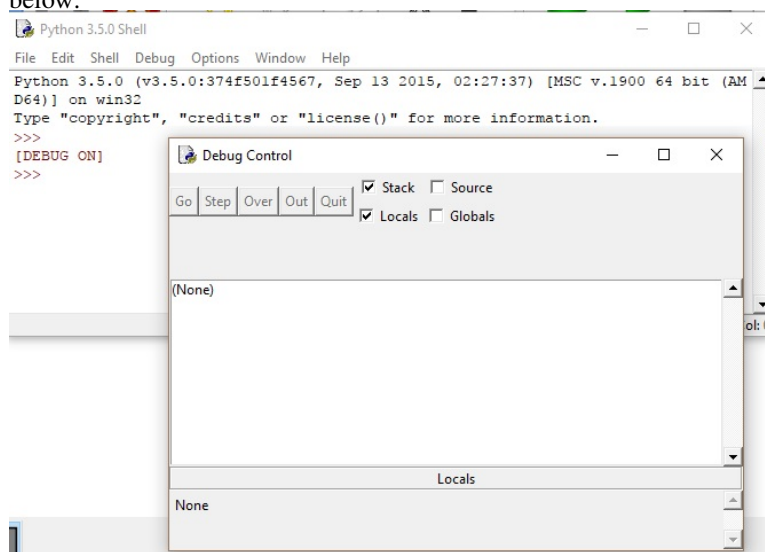
This has a button called step which allows you to move through the code one line at a time and values appear down the bottom

Note: this is much easier to do when your code is saved as a file otherwise you'll be clicking the step button a lot
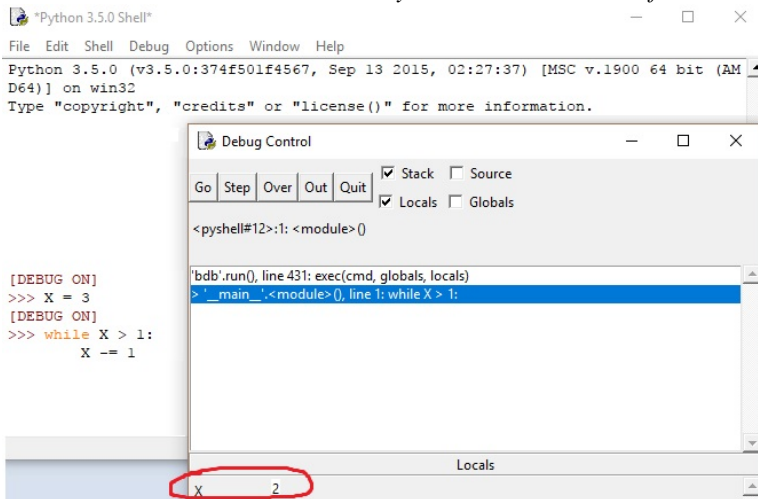
Write up a python program that implements the following algorithm.

Input: Two numbers X and Y representing numbers to multiply together
Output: An integer holding the result of $X * Y$

1   $Sum \leftarrow 0$
2   **while** $Y > 0$ **do**
3      $Sum \leftarrow Sum + X$
4      $Y \leftarrow Y - 1$
5   **return** Sum

Step through this code with IDLE's debugger and notice how the values change as you use the step command.

Note: there are many great IDEs (integrated development environments) which include debuggers that allow you to stop at a chosen line. We don't cover these in this unit, but if you want to invest a small amount of time into learning one we suggest pycharm (https://www.jetbrains.com/pycharm/)
Your demonstrator will be able to help you with using pycharm if you need

Task 2: Lists

Discuss as a group methods to iterate over a list.
Write a Python program that uses a while loop to ask the user to input 5 numbers. The program should then enter each of these numbers into a list.

For example:
Enter a Number: 1
Enter a Number: 9
Enter a Number: 5
Enter a Number: 10
Enter a number: 11
[1, 9, 5, 10, 11]

NOTE: The output ['1', '9', '5', '10', '11'] is a list of strings and is not the correct answer.

Modify your program such that it outputs the list containing the square of each entered number.

For example:
Enter a Number: 1
Enter a Number: 3
Enter a Number: 6
Enter a Number: 2
Enter a number: 3
[1, 9, 36, 4, 9]

Task 3: Assignment and Indexing

One approach to the problem in task 1 was to use the append function to add elements to the list. However, as we know the number of elements that can be entered, we could enter these numbers using assignment. Discuss with the person sitting next to how you can assign values to a specific index of an list.

Run the code given in 'list1.py' (available on Moodle) and discuss the result with the person sitting next to you. Determine what the problem is and as a pair ammend the program, without changing lines 5 or 6, so that the program gives a more sensible output. Once you have solved the problem check with you demonstrator.

Once you are comfortable with the notion of simple list assignment, modify the program you wrote in task 2 to use list assignment.

NOTE: Your demonstrators are there to help, if you cannot find a solution try discussing the problem with them, they might have some good ideas.

Task 4: Lists of Lists, Tables and References

Discuss how lists containing lists can be used to represent tables in Python.
Write a program in Python that asks user to input 5 lines of numbers. Your program should represent these lines as a table.

NOTE: Think carefully about which lists will have a fixed size and which lists will have a variable size.

For example:
Enter some numbers: 1 2 3 4 5
Enter some numbers: 5 6 7
Enter some numbers: 1 1 1 1
Enter some numbers: 1.5 6 7.5 2
Enter some numbers: 0
[[1, 2, 3, 4, 5], [5, 6, 7], [1, 1, 1, 1], [1.5, 6, 7.5, 2], [0]]

NOTE: As the input function returns a string, you will be required to split the string. The documentation for string splitting can be found here `https://docs.python.org/3/library/stdtypes.html#str.split`

Once you have some experience with tables as lists of lists, consider the following line of code:

```
new_table = [[1, 2, 3]]*5
```

Using the techniques you have learnt so far, work out what this code does. As a pair discuss what you think will happen when you change the first element of the first list to 0.

```
new_table[0][0] = 0
```

What happened? Did the table behave as expected? If not, what possible explanation could there be for the behaviour of the table? Discuss your findings with your demonstrator.
If you haven't already, you might light to step through the code with PythonTutor and IDLE's default debugger.

Task 5: Using Lists

Write a program in Python which simulates the rolling of 1000 dice, by generating a number from 1 to 6, and stores these numbers in a list. Calculate the arithmetic mean of this list.

NOTE: The arithmetic mean is given by,

$$\sum_{i=0}^{n-1} \frac{x_i}{n}$$

where x_i is the ith element of a list of size n.

Extension Question

Imagine you have a deck of cards, placed one by one in a row face down. You have the ability to flip a single card. However this triggers the card immediately to the left to flip also. If there is no card to the left then you ignore this rule. Your goal is to write a program that simulates this situation and finishes when all cards appear face up.

NOTE: Think carefully about how you should represent the cards.

ADVANCED NOTE: This problem has some interesting properties and as such has some interesting questions. Try changing the starting state of the game, mixing cards that are face up with face down. Is there any configuration of the starting state that makes the problem solvable/unsolvable?