Using IDLE

IDLE is a particular Integrated Development Environment (IDE) that is typically included in a Python installation. An IDE will enable you to use a text editor to create programs (in the case of Python, we refer to the program as a **script**) as well as execute the program. The IDLE editor understands the syntax of the Python programming language and can help you to debug your program by identifying code that is not valid and must be corrected.

When you first start IDLE, it will display a window that is the Python interpreter. You can type Python commands directly into this environment, and Python will interpret and execute them.

For all of the assignments in this course, you must create a **script**. To create a script, select "New window" from the "File" menu. This will open a new script file. You can also use the "Open" or "Recent files" options under the "File" menu to open files that you have previously created.

You can enter your script into the window that opens when you use the "New window" option. Before executing your script, you must save it. The first time you attempt to save a new script, you will need to give it a name. All Python scripts must have the suffix of `.py` on the filename.

To execute a script, you can select the "Run Module" option under the "Run" menu, or optionally press F5 on your keyboard.

Assignment

Using the IDLE development environment, create a Python **script** named `tryme4.py`. (Note: an alternative to IDLE is to use a free account on the pythonanywhere website: https://www.pythonanywhere.com/)

IDLE has both an interactive mode and a script mode. You must use the **script** mode to develop your script. Your script must use meaningful variable names and have comments that describe what is happening in your script. Comments may describe the assignment of a value to a variable, a computation and the assignment of the result to a variable, or the display of the result.

Write a function in this file called `nine_lines` that uses the function `three_lines` (provided below) to print a total of nine lines.

Now add a function named `clear_screen` that uses a combination of the functions `nine_lines`, `three_lines`, and `new_line` (provided below) to print a total of twenty-five lines. The last line of your program should call first `nine_lines` and then the `clear_screen` function.

The function `three_lines` and `new_line` are defined below so that you can see nested function calls. Also, to make counting “blank” lines visually easier, the `print` command inside `new_line` will print a dot at the beginning of the line:

```
def new_line():
    print('.')
def three_lines():
    new_line()
    new_line()
    new_line()
```

Submit your Python script file in the posting of your assignment. Your Python script should be either a `.txt` file or a `.py` file.

You must execute your script and paste the output produced into a document that you will submit along with your Python script.

It is very helpful if you print a placeholder between the printing of 9 lines and the printing of 25 lines. It will make your output easier to read for your peer assessors. A placeholder can be output such as “Printing nine lines” or “Calling clearScreen()”.

The following items will be used in the grading rubric for this assignment. Make sure that you have addressed each item in your assignment.

- Does the assignment implement `new_line`, `three_lines`, `nine_lines`, and `clear_screen` functions, as well as a main section of the program which calls the functions?
- Does the assignment demonstrate the use of **nested** function calls?
- Does the assignment produce the appropriate output when executed? The output should be recorded in a text file, a Microsoft Word document, or an RTF-formatted document by copying the output from the Python script into the document. The successful script will print out 9 "." lines first and then 25 "." lines.
- Does the program code include comments where appropriate?