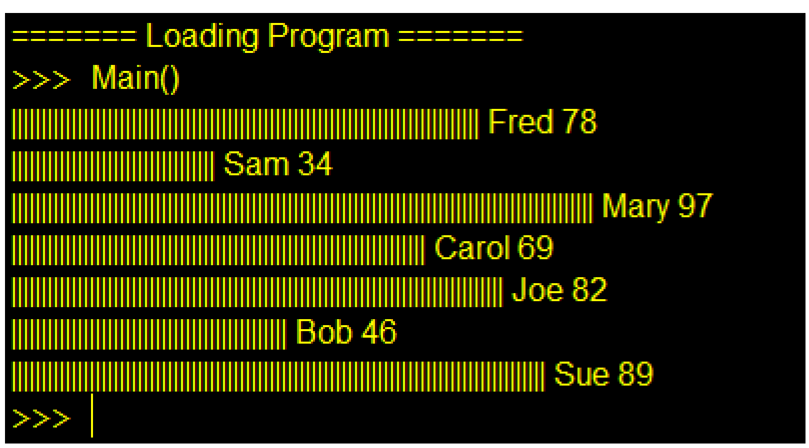
CMPSCI 119
LAB #1 – Bar Graphs
Professor William T. Verts

The goal of this first Python programming assignment is to successfully create, enter, debug, and run a simple program in the JES environment.

If you have not done so already, you must first install the JES environment on your own personal computer from the download site (`https://github.com/gatech-csl/jes/releases`). Please set up a special folder on your desktop or in your documents folder to hold all your Python projects. I also recommend that you buy a flash drive (SanDisk recommended) and put a folder for your Python projects there as well – the flash drive can then be used for backup and for bringing code in for us to help you debug.

Background

In this assignment you are going to write a program to generate a set of text-based bar graphs for students who have taken an exam. For example, student Fred received a 78 on his exam, so the line of output for Fred will be 78 copies of the | character (the vertical bar), followed by his name and his score. Sam only got a 34 on the exam, so his line of output will be 34 copies of the | character, followed by his name and his score. This process repeats for every student in the class. When your program runs, the expected output from JES will look like the following image:



Setting up the Assignment

In the JES environment, type in the following program code framework, and save it in your Python folder with `Lab1.py` as the filename. For now, leave the gray areas blank; you will write your own code there later.

```
def Process (Name, Score):
    S = ""
    Your code goes here
    return S

def Main():
    Names = ["Fred","Sam","Mary","Carol","Joe","Bob","Sue"]
    Scores = [78,34,97,69,82,46,89]
    Your code goes here
    return
```

At the top of your program, write a comment containing *your own name* and the lab number. For example, I would put in a comment as the first line of text:

```
# William T. Verts - Lab #1
```

If you load this program and try to run it by typing **`Main()`** at the Python **`>>>`** command prompt, it should do nothing, but neither should it contain any errors. Try it.

Task #1 - Process

Notice that the `Process` function has two parameters, `Name` and `Score`. Those parameters will contain the name and the score for a particular student. For example, if the function is called as `Process("Fred",78)` then `Name` will contain the string `"Fred"` and `Score` will contain the integer 78.

The `Process` function also initializes a string `S` to the empty string, and returns `S` as the value of the function.

Finish the function by writing code in the gray area so that calling `Process` from the `>>>` command line as `Process("Fred",78)` returns as the value of `S` the following string:

|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| Fred 78

This function requires the use of a loop to build up the string – for this assignment I want you to use a `while`-loop (not a `for`-loop).

The `Process` function <u>must not</u> contain any `print` statements. It returns its result as the value of the function; printing happens elsewhere.

When you finish writing the `Process` function, save your file and then click the Load Program button. Fix any syntax errors. Every time you change the program you must re-save the source code (File-Save, or Ctrl `S` on Windows, or ⌘`S` on Mac), then click the Load Program button. Be very careful about indentation (please use exactly four spaces per level throughout the program) and capitalization (for example, variables `frog`, `FROG`, and `Frog` are all different). Places where errors can creep in are through missing or extra quotes, parentheses, colons, commas, etc., along with incorrect indentation levels.

Test your function in the interactive area of Python by typing `Process("Fred",78)` and `Process("Sam",34)`, and make certain it works before proceeding.

Task #2 - Main

Now we need to complete the main program, which is to call the `Process` function for all associated values from the lists `Names` and `Scores`, and in each case print out the result returned from `Process`. You may assume that `Names` and `Scores` contain the same number of elements, but your program must not make any assumptions about what that number is (i.e., there are currently 7 entries in both lists, but you cannot use the constant 7 as the length – your program MUST be adaptable to any number of students).

This function also requires the use of a loop – for this assignment I want you to again use a `while`-loop (not a `for`-loop).

Notice that `"Fred"` is the first item in `Names`, and 78 is the first item in `Scores`. These values are what are first passed to the `Process` function. The result of that call are printed by `Main`. Then, in the second call to `Process`, your main program must use the second values from `Names` and `Scores` (`"Sam"` and 34), in the third call to `Process` it uses the third values from `Names` and `Scores` (`"Mary"` and 97), and so on for as many items as there are in the lists.

If everything works correctly when you type `Main()` followed by Enter↵, you will see the expected output shown on the first page of this assignment.

If the output does not contain the correct information, debug your program and run it again. Repeat the process until the resulting output does contain the correct information.

Finishing Up

When you are finished and everything runs correctly, go to the class site and click on the link for submitting lab assignments. In JES select all the text, copy it to the clipboard, in the Web page paste the text into the program area of the submission form, fill in your name and ID number and the lab number in the appropriate slots, and then submit the assignment for grading.

The graders will score your program by running it to see if it correctly generates the appropriate graphs. You will also be graded on efficiency and completeness, and the graders will be explicitly looking to see if your program will work for an arbitrary number of students (number of entries in `Names` and `Scores`).