CS 251: Scripting and Version Control: Inlab

- Due: 8:10 PM 23/08
- Please write (only if true) the honor code. If you used any source (person or thing) explicitly state it. You can find the honor code on Piazza.

Overview

C++ is one of the most powerful languages, but probably is not the most user-friendly. The pipeline to run a C++ code is as you well know: write code, compile, and run the executable. C++ belongs to a programming language paradigm in which compilation is an integral part of the programming process.

For the now generation, there are other kinds of languages in which you write code and execute it without the hassle of compiling it. Well, this is to say that you don't have to manually compile the code (there is a longer subtext). Octave was one of the languages in this genre but it is tailored to scientific computing. You also used languages such as AWK, and JavaScript.

In this lab, we will get our hands dirty with `python`, one of the most versatile and easy to use scripting languages. Currently *the* language to learn, it would seem.

But programming is not the only story. Now that you are working in teams on a project you might find (for whatsoever reason) your code starts to malfunction and you have no idea why. You might want to see what changes you made or in the worst case revert to a previous version. A tool called `git` has been designed specifically for this task. Such a system is called a version control tool. It lets you maintain various versions of your project and compare across versions. There are other tools out there too such as subversion, mercurial, et. al. but git is probably the most popular one.

There are many other good programming practices and version control is one of them. Let's start the journey.

A. Git

If you've heard of GitHub or Bitbucket then you are probably familiar with how git works. Git is a great tool for version control. But it also comes handy while coding collaboratively. We will be using this lab to learn the fundamentals of git with the hope of using them in the coming labs.

1. Go to https://git.cse.iitb.ac.in/ and use your cse account to create a git account. (We will be using this so that it is convenient for you to create private repositories.) Take a moment to read about or explore GitHub, a widely used git platform in your leisure time.

2. Go to https://try.github.io and understand the basics of git and how to create a repository and adding files to it. Use your discretion if you want to emulate the tutorial on your local machine in parallel

3. Create a repository on git.cse with the name cs251_groupXX. Add all the three teammates to this repository. Each of the three team members should commit one lab each in order out of the three labs so far. Again, the magic sequence you use on Indian railways – lower, middle, and upper. (If you are a team with only two people, upper maps to lower). Commit the smallest possible file.

We will return to git.

B. Python

Python is a great programming language, and one of the many reasons for its success is a simple, easy-to-understand syntax. That being said, there are some things that can trip up beginners.

REPL stands for Read-Eval-Print-Loop. Put simply, it is a shell for a language similar to the Linux shell. Through this, you can type commands without having to save them in a file and then executing it. Type `python3` in your terminal to fire up the python REPL. This tutorial gives an introductory overview of the Python language. Follow it along to learn about the language and also about the Python REPL. Another thing which Python enforces is indentation. Blocks of code are distinguished by their indentation. [1]

For all the tasks below, document any interesting observations in `readme.txt`. After each sub-task, commit the work done until then. Do this religiously as you have to submit your entire git repo in the end. We will be working with `python3`.

0. Create a repository `taskB_inlab` for the purpose of the following tasks.

1. Write a Python file `task1.py` containing a function `function1` which takes two strings and a list as arguments. Its signature must look something like `function1(string1, string2, listVariable)`. Suppose `string1 = 'CS251'` and `string2 = 'Software'`, then the function should append the tuples `('C', 'S'), ('S', 'o'), ('2', 'f'), ('5', 't'), ('1', 'w')` to the passed list. Return a new list. This task should be done the 'Pythonic' way, i.e. using Python functions and without loops. In the same file, initialize a list containing 5 elements and pass the list along with strings of your choice to `function1`. Print the original list and the returned list

2. In a file named `task2.py`, create a function `function2` with the signature `function2(listVariable, n)`. The first argument passed to the function is a list of tuples. Make the function print a list with the n^{th} element from each tuple as its members. The function should also return the list after sorting it according to the n^{th} member of each tuple. As before, do it without loops and using only the in-built functions defined on lists. There are multiple ways to do this but read up on what the function `map` does and what lambda functions are and use them. This leads to a bigger picture in which **higher order functions** and **duck typing** feature. Enthusiastic people might find that interesting

3. You are familiar with what matrices are. You have been given a file matrix.txt which contains a matrix. The first row of the file contains two integers, the number of rows and the columns, respectively, in the matrix to follow. Each row of the file represents one row of the matrix with the columns being separated by spaces. Read in the first line of the file and initialize a 2D array containing 'None' elements. Read the matrix using the file handling operations Python supports and store it as a 2D array in the initialized matrix. Compute the transpose of the matrix and print it in the same format as that of the initial file. Do not forget the print the dimensions as well. Save the Python code in `task3.py` and the transpose in `transpose.txt`. Do not use any external libraries

Extra credits: You don't need to code anything for this part. Document your observations in the readme.txt.

1. Read up what tuples are in Python. Try appending elements to a tuple. Explain the observations

2. Set one of the elements of the tuple as a list. Try adding elements to this list in the tuple. What do you observe? Does it or does it not contradict the observation in the previous part?

Hopefully at this point you would have appreciated some awesomeness of Python! Relax with this XKCD.

[1] By the way, python has tons of Easter eggs hidden in it. For starters, type `import this` in the REPL.

Submission Guidelines

1. When you submit, please document individual percentages such as Student 1: 80%, Student 2:100%, Student 3:10%. In this example, the second student will get full marks (10/10) and the first student will receive 8/10.

2. Do include a readme.txt (telling me whatever you want to tell me). (The reflection essay is not required for inlab, but is required for outlab.) Do include group members (name, roll number), group number, honour code, citations etc.

3. The submission folder should contain the folder `taskB_inlab`, which is a git repository(contains `.git` folder) you created for task B. Also include `transpose.txt` in the same folder.

4. The folder and its compressed version should both be named `lab04_groupXY_inlab` for example folder should be named `lab04_group07_inlab` and the related `tar.gz` should be named `lab04_group07_inlab.tar.gz`

How We will Grade You - 30 + 10 Marks

1. Task A - 5 Marks
   (a) A.1 Creating a GIT CSE account - 2.5 Marks
   (b) A.3 Committing the previous labs to the repository - 2.5 Marks
2. Task B - 25 Marks
   (a) B.1 Solving it in the Pythonic way - 7 Marks
   (b) B.1 Solving it using loops - 3 Marks
   (c) B.2 - 8 Marks
      i. Printing the created list - 4 Marks
      ii. Returning the sorted list - 4 Marks
   (d) B.3 - 10 Marks
      i. Initializing the array with None elements - 2 Marks
      ii. Reading the file - 3 Marks
      iii. Storing the data read from the file as a 2D array - 2 Marks
      iv. Calculating the transpose - 3 Marks
3. Extra Credit
   (a) E.1 Correct explanation - 3 Marks
   (b) E.2 Observation+Explanation - 2+5 Marks