SENG265: Software Development Methods (Summer 2019)

Lab 05 – Introduction to Python

Week of June 10th

Authors: Nirav Galani*

*based on material provided by Prof. Mike Zastre

This week

- Writing, running and debugging python programs.

Running a python program

- Determine the version of python on your lab computer
  - $ python –version
- The lab computers are running version 2.7. We need version 3.
- $ setSENG265
- This will open a nested shell that will run python version 3.
- Check to version of python.

Running a python program

- Download hello1.py, hello2.py, hello3.py and hello4.py
- Run hello1.py using the python command
  - $ python hello1.py
- Now try to run hello1.py from the command line like an executable file.
  - $ ./hello1.py

Running a python program

- We get an message that permission denied.
- What are the permission for hello1.py?
- Change the permission for hello1.py to allow execution
  - $ chmod u+x hello1.py
- Now execute
  - $ ./hello1.py
- We still cannot execute this file.
- This is because the computer cannot find the path to python. We must add the bang path.

Running a python program

- Open hello2.py
- Notice that hello2.py is the same file as hello1.py, except that it has a bang path added at the top.
- Change the permissions of hello2.py to allow execution.
- Execute hello2.py
  - $ ./hello2.py
- Note that we can still run the file using the python command
  - $ python hello2.py

hello3.py & hello4.py

- Notice the bang path on both of these files.
- Run both files.
- Note that the outcome is the same.
- However, in hello4.py, all code is routed through a main functions.
- It is better to have fewer operations in global scope.
- Remember the way the main is defined and the incantation at the end that is calling the main.
- You may use the structure of hello4.py for the remaining exercises in this lab.

pythag01.py

- Create a program pythag01.py
  - It will calculate the hypotenuse of a right angle triangle, given the 2 sides at a right angle with each other
  - You will need → import math
  - Write a main function
    - **Declare and assign** 2 variables, each representing the 2 right angle sides of a right angle triangle.
    - Write the code to calculate the hypotenuse.
    - Print the answer. For example
      - "Right angle triangle with right angled sides of length 3 an 4 has a hypotenuse of 5"
  - Program should work for any lengths (hard coded) of the right angled sides
  - Note that everything is done in the main.

pythag02.py

- Create a program pythag02.py
  - It will also calculate the hypotenuse of a right angle triangle given the 2 sides at right angle with each other. You may hard code the 2 known sides.
  - Write a main function similar to pythag01.c with the below modification
  - Write a function named “hypotenuse(……..)” which does the calculation of the hypotenuse. Use parameters as required.
  - Print the answer using a **print statement in the main**. For example
    - “Right angle triangle with right angled sides of length 3 an 4 has a hypotenuse of 5”
  - Program should work for any lengths (hard coded) of the right angled sides

pythag03.py

- Create a program pythag03.py
- This will modify pythag01.c to include the following
  - Add → import sys
  - Lengths of the 2 right angle sides will be entered at the command line when running the executable file. An example of the commands is
    - $ python pythag03.py 3 4
    - This will execute the program, which will calculate the hypotenuse using the 2 numbers given in the command line and then print the answer to console.
  - If incorrect number for arguments are entered at the command line when running the executable file, the program should print an error message and terminate, For example
    - $ python pythat03.py
    - This should give an error message, something like – “You must enter atleast 2 numbers for the sides of the triangle.” -- and the program should then terminate.

text_processing.py

- Create a program test_processing.py that will use **one** string entered at the command line.
- This string will have words and commas.
- The program will print out all the words separated by commas.
- An example of running this program is:
  - $ python text_processing.py something,somewhere,is good,today
- The output of the above example would be

something

somewhere

is good

today

text_pipe.py

- Create a program text_pipe.py
- The input for this file will be a text file containing various lines of words separated by commas.
- This input file will be piped into the program.
- An example of such as file is provided – thesaurus.csv
- Command to run the program with thesaurus.csv as input would be
  - $ python text_pipe.py < thesaurus.csv

text_pipe.py

- The program will take content that is piped in, and identify all the unique words.
- The program will print these words to console in alphabetical order.
- The program will identify and print the shortest and the longest word.
- If there are multiple words of the same shortest or longest length, then print them all.

Git

- Place all you work into your course remote repository