CSE/ISE 337 – Scripting Languages
Spring, 2023
Assignment 04 – Scripting Potpourri

Assigned: Wednesday, 04/12/2023
Due: **Friday, 05/05/2023, at 11:59 PM**
Total Problems: 4, Total Points: 60

Submission Instructions:

1. Collect and submit all of your scripts in a single zip archive named: `a04_<Last_name><First_name>.zip`
2. The zip file should contain the following:
   - p1.ps1, p2.py, p3.py, p4.zip
3. Make sure you include your name, netid, and Student ID number as comments at the top of each script file.
4. Make sure you use the specified file, function, and class names (and any other names specified in the instructions). Grading may be partially automated (by importing your code to run against our test cases) and the use of incorrect names will interfere.
5. This is an individual programming assignment. Any collaboration on coding will be considered a violation of academic honesty.

Problem 1 (15 points): `p1.ps1`
Sometimes the processes running on your computer escape from your benevolent control and so must be killed. Let's write a powershell script to do this work for us.

Task:
Your script should do the following:

1. Read a process name as a command-line argument
2. List all currently running instances of that process on the console.
   2a. If there is no currently running instance of the specified process, then print out the message, "No such process is running." to the console, and then terminate the script.
3. Kill all currently running instances of the specified process.
4. List all currently running processes on the console, so we can see the specified process is absent.

Parsing the arguments:
The script takes a single command-line argument. The argument is a string that names a process.

Script name:
Name your shell script: `p1.ps1`

Example invocation: `$ powershell p1.ps1`

Problem 02 (15): `p2.py`
In class, I built a small data-entry GUI application for building a collection of contact information. I would like you to extend this GUI application to add a view in which users can remove a person's contact information from the list. So, you will be completing the implementation of the "Remove Contact" view that I left unfinished.

Task:
Please do the following to complete the "Remove Contact" view:

1. The user should be able to search for contacts to be deleted by first name, last name, phone number or email address. Check the "Find Contact" view to see how I did this.
2. If the search returns multiple results, the user should be able to specify which of those results should be deleted.
   There are several ways to accomplish this, but as one option look up the Listbox widget available from Tkinter.
   https://www.tutorialspoint.com/python/tk_listbox.htm
3. The user should then be able to delete the selected contact (or contacts). It is removed from the contact list and will not show up on future searches.
4. Display the results to the user. Use the Messagebox widget to display the result of the deletion in a pop-up. The message should be: "<First_name> <Last Name> has been removed from the contact list." See the details on the MessageBox widget here:
   https://www.tutorialspoint.com/python3/tk_messagebox.htm

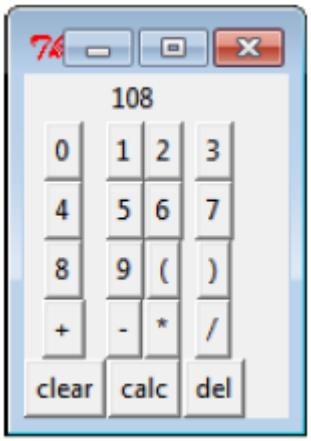
Parsing the arguments:
No arguments passed to this script.

Script name: `p2.py`

Example Invocation: python3 p2.py

Problem 03 (15): `p3.py`

Suppose we wanted to create a small calculator interface, similar to (but much simpler than) Calc.exe on Windows. Here is an example of what your calculator GUI might look like (It does not have to look like this).



Task:

Your calculator should do the following:

1. Your calculator should allow the user to enter digits. (Inputs are all integers).
2. Your calculator should support addition, subtraction, multiplication, and division.
3. Your calculator should support placing expressions in parentheses.
4. Your calculator should display the calculation entered by the user, and then the result of the calculation (Let Python do the math).
5. Your calculator should allow your user to clear the display and delete from entries to correct mistaken inputs.
6. Use the grid() function to layout the widgets in your calculator interface.

Parsing the arguments:

No arguments passed to this script.

Script name: `p3.py`

Example Invocation: python3 p3.py

Problem 04 (15): `p4.zip`
Build the Pizzeria project described in the "Try It Yourself" Exercises in Chapters 18 of *Python Crash Course, 2e.* The chapter is available on Blackboard.

Task:
You will complete the following exercises in building the Pizzeria Django project. Organize your project the way the Learning_Log project is organized.

A. Exercise 18-4 - Pizzeria:
1. Start a new project called pizzeria with an app called pizzas.
2. Define a model Pizza with a field called name, which will hold name values, such as Hawaiian and Meat Lovers.
3. Define a model called Topping with fields called pizza and name. The pizza field should be a foreign key to Pizza, and name should be able to hold values such as pineapple, Canadian bacon, and sausage.
4. Register both models with the admin site (Test your work by using the admin site to enter some pizza names and toppings and by using the Django shell to explore the data you entered.)
5. Set the username and password for the superuser account to: admin, admin

B. 18-6 - Pizzeria Home Page:
1. Add a home page to the Pizzeria project you started in Exercise 18-4.

C. 18-8 - Pizzeria Pages:
1. Add a page to the Pizzeria project from Exercise 18-6 that shows the names of available pizzas.
2. Then link each pizza name to a page displaying the pizza's toppings.
3. Make sure you use template inheritance to build your pages efficiently.

When your project is complete, compress the project directory into a zip archive, and include the entire archive in your submission.

Parsing the arguments:
No arguments passed to this script.

Script name: `p4.zip`

Example Invocation:
1. We will extract your zip.
2. Start your server (python manage.py runserver).
3. Open a browser to localhost:8000 (don't change the default address and port)
4. Interact with your web app.