**COSC 1336, Lab 8 Instructions, More about Strings, string operations**

Learning a new programming language can be like learning to play a musical instrument, you have to practice! A good way to practice the piano is to play scales. A good way to practice python is to “play” with the various methods, functions, command and operators.

**Part 1: (10 points: play)** Play with python, just play. Turn to chapter 8, “More about Strings”(ch9 if 2ed) and experiment in the shell. Look at each page of the “More About Strings” chapter and try various things you can do with strings. Go page by page, and, **in the python shell**, type in statements and see what happens. Play like a kitten with a ball of string!

Items to try: print(), loops, index[], exceptions, len(), +, +=, =, ==. ‘string’, slicing, [i:j:k], string in string, isalnum, isalpha, isdigit, islower, isspace, isupper, lower, lstrip, rstrip, strip, upper, endswith, find, replace, startswith, \*, split, join. Many, but not all, of these items were demonstrated in class.

Allow your interaction with the shell accumulate **to over two hundred lines.** Don’t worry about python error messages; just keep playing in the shell. Try things that interest you. Attempt most or all of the topics in the chapter on strings.

When done, save the shell output to: **DDHH\_L8\_Lastname\_shell.py.** (In the upper left corner of the shell, see: File -> Save) This is not an executable python script. Instead, it is a record of your interactive experimentation. I grade this based on your experiments with the items underlined above.

**Part 2: (10 points: validate password)** Secure websites require a login with a username and password. To insure a “strong” password, rules are enforced to prohibit a simple password like: “pwd”. Write a Python function that validates a password. If the password is valid, return True, else return False. The IPO is:

* Input: one string
* Processing: test all these rules:
  + At least 8 characters long
  + No spaces or whitespace (No space, tab, newline)
  + At least 1 digit (0 to 9)
  + At least 1 punctuation character, such as: ~!@#$%^&\*()\_+-=:”<>?;’,./|\
  + At least 2 uppercase letters (A .. Z)
  + At least 2 lowercase letters (a .. z)
* Output: True if passes all the tests, False otherwise

**Part 3: (10 points: test)** Create a loop that prompts for and allows the user to enter a password. After getting the password, test it on the function written in Part 2. If it passes, accept the password. If it fails, keep looping until the password is valid. Keep count of how many times the password entered was not valid. Display: “Valid” / “Invalid” and the number of attempts taken.

**Part 4: (5 points: extra credit)** If the user fails to enter a valid password after 3 tries, generate a valid password for user. Use random.randint(), chr(), ord(), etc. to generate random characters as needed. Assemble these random characters into a correct and valid password that passes the validation test. Assign this password to the user. In addition to the output described in part 3, also output the valid, computer-generated password.

Save parts 2, 3 and optionally 4 to the file: **DDHH\_L8\_Lastname\_password.py**

**Summary: Submit the following files for this lab:**

**DDHH\_L8\_Lastname\_shell.py** (10 points)

**DDHH\_L8\_Lastname\_password.py** (part 2, 3 and extra credit, 20 points