****

DTSC 575 CodeGrade Assignments

Here you will find the description for the programming assignments. Remember, this is a learning experience, and you are encouraged to utilize other forms of resources to complete these assignments.

All assignments below will be accounted towards your final grade.

There are multiple ways to complete some of these assignments. As long as they are outputting results as intended, that’s fine. If sys.argv is not used to test CodeGrade cases, points will be deducted at the end of the semester.

General rule for CodeGrade:

* If importing a certain package is not necessary, do NOT import it. If you do so, CodeGrade will throw an error message.
  + For example, you do NOT need to packages such as:
    - “import matplotlib.pyplot as plt”
    - “import seaborn as sns”
    - “import sklearn”
    - and etc.
  + If you do, even if the rest of your code is correct, CodeGrade will mark it as wrong.
* CodeGrade is case-sensitive to .py file submissions.
  + For example, submitting “Change.py” instead of “change.py” will throw an error.
* For assignments that use csv files, you do NOT need to upload the .csv to CodeGrade because it’s pre-loaded.

# **General Python**

# Strings

## **countVowels.py**

* Create a program called countVowels.py that has a function that takes in a string then prints the number of unique vowels in the string (regardless of it being upper or lower case).
* For example:
  + The argument “swEet” should print 1
  + The argument “AaaaeeE” should print 2

## **capCount.py**

* Create a program called capCount.py that has a function that takes in a string and prints the number of capital letters in the first line, then prints the sum of their indices in the second line.
* The string “hEllo, World” would should look like this:

2  
8

## **shortest.py**

* Create a program, shortest.py, that has a function that takes in a string argument and prints a sentence indicating the shortest word in that string. If there is more than one word print only the first. Your print statement should read:

“The shortest word is x”

Where x = the shortest word. The word should be all uppercase.

## **palindrome.py**

* Create a program, palindrome.py, that has a function that takes one string argument and prints a sentence indicating if the text is a palindrome. The function should consider only the alphanumeric characters in the string, and not depend on capitalization, punctuation, or whitespace.

If your string is a palindrome it should print:

It's a palindrome!

If it is not a palindrome, it should print:

It's not a palindrome!

\*Note: please copy and paste these versions into your code because the type of apostrophe can differ between IDEs, command line, etc.

# Dictionary

## **luke.py**

* Do NOT use elif statement or nested if-else statement for this assignment. You should only need one single if-else statement at most (points will be deducted if more than one single if-else statement is used or the dictionary is changed).
* Create a program, luke.py, using the following dictionary:

relations = {'Darth Vader':'father', 'Leia':'sister', 'Han':'brother in law', 'R2D2':'droid', 'Rey':'Padawan', 'Tatooine':'homeworld'}

* The program will take one argument, corresponding to one of the relations’ keys. The program will print out the statement:

“Luke, I am your x”

Where x = the relationship.

* For example:
  + If the argument is Leia, it should print “Luke, I am your sister”
  + If the key is ‘Darth Vader’ it should instead print “No, I am your father”

## **grades.py**

* Use this exact dictionary to complete this assignment (Do NOT alter this dictionary - points will be deducted if dictionary is changed):

grades = {'Biology':80, 'Physics':88, 'Chemistry':98, 'Math':89, 'English':79, 'Music':67, 'History':68, 'Art':53, 'Economics':95, 'Psychology':88}

* Create a program, grades.py, that takes an argument (subject) and prints the average score excluding that subject, to two decimals.
* For example:
  + If the argument is Biology, it should print 80.56
  + If the argument is Chemistry, it should print 78.56

## **gpacalc.py**

* Use this exact dictionary to complete this assignment (Do NOT alter this dictionary - points will be deducted if dictionary is changed):

gpa\_dict = {'A':4.0, 'A-':3.66, 'B+':3.33, 'B':3.0, 'B-':2.66, 'C+':2.33, 'C':2.0, 'C-':1.66, 'D+':1.33, 'D':1.00, 'D-':.66, 'F':0.00}

* Create a program, gpacalc.py, that takes four letter grade arguments and prints out the corresponding GPA, to two decimals. Your program should work both in arguments are upper-case and lower-case.
* Your program should print in the form:

“My GPA is x”

Where x = GPA calculation

# List & Sets

## **inrange.py**

* Create a program inrange.py that has a function that takes one integer argument. The function will print a list of all values between 3000 and 5000 that is divisible by:

1. the integer argument
2. the integer argument + 7
3. the integer argument ^ 2

* For example, if the integer argument is 6, it should print:

[3276, 3744, 4212, 4680]

## **commonset.py**

* **IMPORTANT**
  1. Your autotest cases will NOT work if you don’t load your sys.argv like below. For this assignment, load in your variables like this:

set\_a = sys.argv[1:]

set\_b = ['apple', 'banana', 'mango', 'orange']

* 1. The above set\_a is a list-type variable which contains words.
* Create a program, commonset.py. Your program should:
  1. Find the common words between set\_a and set\_b.
  2. Print the output in a set format.
* Example 1 - If the set\_a is ['apple', 'banana', 'pear', 'grape'], the program should print:

{'banana', 'apple'}

* Example 2 - If the set\_a is ['mango', 'mango', 'mango', 'pear', 'grape'], the program should print:

{'mango'}

## **diffset.py**

* **IMPORTANT**
  1. Your autotest cases will NOT work if you don’t load your sys.argv like below. For this assignment, load in your variables like this:

set\_a = sys.argv[1:]

set\_b = ['apple', 'banana', 'mango', 'orange']

* 1. The above set\_a is a list-type variable which contains words.
* Create a program, diffset.py. Your program should:
  1. Find the words that exists in set\_a but are not in set\_b.
  2. Print the output in a set format.
* Example 1 - If the set\_a is ['apple', 'cherry'], the program should print:

{'cherry'}

* Example 2 - If the set\_a is ['apple', 'mango', 'pear', 'grape'], the program should print:

{'grape', 'pear'}

# Loops

## **comprehension.py**

* **IMPORTANT**
  + Your autotest cases will NOT work if you don’t load your sys.argv like below. For this assignment, load in your sys.argv like so:

my\_ints = sys.argv[1:]

* + The above my\_ints a list-type variable which contain numbers that are string-typed (ex. ['1', '2', '3', '4', '5'])
* Create a program, comprehension.py. Your program should:

1. Convert these string-type integers into integer-type.
2. If the number within the list is divisible by 3, multiply it by 10, then replace it.

* For example:
  + If your my\_ints = ['1', '2', '3', '4', '5'], your output should be [1, 2, 30, 4, 5]
  + If your my\_ints = ['3', '30', '1', '15', '10'], your output should be [30, 300, 1, 150, 10]

## **loopindex.py**

* **IMPORTANT**
  1. Your autotest cases will NOT work if you don’t load your sys.argv like below. For this assignment, load in your sys.argv like so:

loop\_list = sys.argv[1:]

* 1. The above loop\_list a list-type variable which contain numbers that are string-typed (ex. ['1', '2', '3', '4', '5'])
* Create a program, loopindex.py. Your program should:
  1. Convert these string-type integers into integer-type.
  2. For each of the numbers in the list, add its own index position.
* Example 1 - If the argument is ['5', '5', '5'], the program should print:

[5, 6, 7]

* Example 2 - If the argument is ['4', '11', '0', '0', '1'], the program should print:

[4, 12, 2, 3, 5]

# Combined

## **counter.py**

* Create a program, counter.py. You are given a single string argument. Print a dictionary where the keys are composed of each letter from a word and values are the sum of each letters’ appearances.
* Example 1 - If the word is “good”, the program should print:

{'g': 1, 'o': 2, 'd': 1}

* Example 2 - If the word is “onomatopoeia”, the program should print:

{'o': 4, 'n': 1, 'm': 1, 'a': 2, 't': 1, 'p': 1, 'e': 1, 'i': 1}

## **duckgoose.py**

* **IMPORTANT**
  1. Your autotest cases will NOT work if you don’t load your sys.argv like below. For this assignment, load in your sys.argv like so:

duck\_goose = sys.argv[1:]

* 1. The above duck\_goose a list-type variable which contains lower-cased words, either 'duck' or 'goose' (ex. [ 'duck', 'duck', 'goose']).
* Create a program, duckgoose.py which removes all the ‘goose’ within the list then print the remaining list.
* Example - If the duck\_goose is ['goose', 'duck', 'duck', 'goose', 'goose'], the program should print:

['duck', 'duck']

## **duplicates.py**

* **IMPORTANT**
  1. Your autotest cases will NOT work if you don’t load your sys.argv like below. For this assignment, load in your sys.argv like so:

duplicated\_words = sys.argv[1:]

* 1. The above duplicated\_words is a list-type variable which contains a whole bunch of lower-cased words (ex. ['hello', 'world', 'welcome', 'hello', 'again']).
* Create a program, duplicates.py. It should:
  1. Remove all duplicate words from the list
  2. Then print it in descending order of alphabets (from Z to A).
* Example 1 - If the list is ['hello', 'world', 'welcome', 'hello', 'again'], the program should print:

['world', 'welcome', 'hello', 'again']

* Example 2 - If the list is ['apple', 'banana', 'apple', 'orange', 'pear', 'banana'], the program should print:

['pear', 'orange', 'banana', 'apple']

# **Numpy**

## **arrayargs.py**

* Create a program called arrayargs.py that has a function that takes four integer arguments. Those arguments should be put into an Numpy array.
* The function will have two print statements.
  + The first will print the type of the array you create (which should be <class ‘numpy.ndarray’>).
  + The second will print the multiplication of the four items in your array.
* Your output could look like this (it could differ in parts):

<class ‘numpy.ndarray’>

80

## **randdf.py**

## Create a program called randdf.py that has a function that takes two integer arguments and prints a Pandas dataframe. The two arguments will correspond to the number of rows and number of columns, respectively. The dataframe should be filled with random integers from 0 to 100.

## Set your numpy random seed to 56.

## Note: Use random from numpy. Otherwise, you might get a different result.

## **reallyrandom.py**

* Create a program called reallyrandom.py that has a function that takes in three arguments and prints one integer. Set your numpy random seed to 42.
  + The first argument should correspond to the size of a np.randint that has values from 0 to 10.
  + The second is an integer that you will multiply the randint by.
  + The third argument is a value you will index the result of the multiplication by.
* The program should not crash if the third value is larger than the first.
* You will print the integer that was indexed as ‘Your random value is x’ where x = the result of the indexing.

# **Pandas**

## **presidents.py**

* Use the president\_heights.csv file to complete the assignment. Create a program, presidents.py, that takes two arguments. These arguments will correspond to the start and stop of a slice, respectively. It will slice the heights column in the president\_heights.csv files.
* Read in the csv data like so:

import pandas as pd

df = pd.read\_csv(“president\_heights.csv”)

* Then print off the average height, rounded to two decimals, of the selected presidents in the following form:

“The average height of presidents number x to y is z”

Where:

* x = start of the slice
* y = end of the slice
* z = calculated average

Note: There would be 6 presidents if the “The average height of presidents number 4 to 10 is ...” (Think in terms of index slicing when it says 4 to 10)

# **Statsmodels**

**Note**: In the following statsmodels assignments, some information in the ‘Expected output’ portion uses regular expressions to allow for more flexibility in answers. Any expected output with curly braces (e.g., .{0,}71.73) is a regular expression, and you only need the non-curly braces portion to get credit (e.g., 71.73). Do not attempt to have the regular expression as your output.

## **fastfood.py**

* Use the fastfood.csv file to complete the following assignment. Create a file, fastfood.py, that loads the .csv file and runs a regression predicting calories from total\_fat, sat\_fat, cholesterol, and sodium, in that order. Add a constant using sm.add\_constant(data).
* Then, print the following to two decimals

print(model.mse\_total.round(2))

print(model.rsquared.round(2))

print(model.params.round(2))

print(model.pvalues.round(2))

## **german.py**

* Use the german\_credit\_data.csv file to complete the following assignment. Create a file, german.py, that loads the .csv file and runs a regression predicting credit amount from age and duration, in that order. Add a constant using sm.add\_constant(data).
* You will need to rename the column 'Credit amount' to 'Credit\_amount'.
* Then, print the parameters and R-squared to 2 decimals using

print(model.params.round(2))

print(model.rsquared.round(2))

## **sacramento.py**

* Use the sacramento.csv file to complete the following assignment. Create a file, sacramento.py, that loads the .csv file and runs a *logistic* regression. The regression should predict whether or not a house has 1 or more than one bathroom based on beds, sqft, and price, in that order.
* You will need to create a new variable from baths, and it should make it such that those observations of 1 bath correspond to a value of 0, and those with more than 1 bath correspond to a 1.
* Make sure to add a constant using sm.add\_constant(X)
* Your file should print the results in this way:

print(mod.params.round(2))

print(mod.pvalues.round(2))

print('The smallest p-value is for sqft')