San José State University
Computer Science Department
CS152, Programming Paradigms, Springs 2022

Homework #1

Objective:

This homework's objective is to practice procedural programming in Python programming language.

Details:

Solutions to all exercises for this homework assignment should be implemented in Python. Submit solution to each exercise in a separate .py file. Make sure to include the following comments into each .py file (replace with your information):

```
# CS152 Spring 2022
# Student name
# SJSU student ID
# Homework assignment 1
# Exercise X
```

Name each .py file with the exercise number. For example, exercise1.py, exercise2.py, etc.

Exercise 1:

Michael is searching for a job and is applying to various companies. He has some restrictions though. He loves the west coast of the US (the states of California, Oregon, Washington) and will accept any position that pays over $80,000/year in those states. In order to accept a job in another state, he wants an offer of at least least $100,000 a year. He also is not willing to move to some states: Arizona, New Mexico, and Texas. It is too hot for his liking in those states. He is not willing to accept any offer in those three states.

Michael wants to automate the process of selecting companies to which he will apply. Your task is to help Michael by implementing a function named ***select_company(state, salary)***, which accepts two input parameters: state and salary the position pays. This function will return either true or false, indicating whether Michael should apply to this company or not. Use conditionals to implement the logic flow for whether Michael should send in an application for a given position.

An example of a using *select_company()* function: *select_company("California", 85000)*
And the expected output should be: *True*

An example of a using *select_company()* function: *select_company("New Mexico", 160000)*
And the expected output should be: *False*

An example of a using *select_company()* function: *select_company("Colorado", 110000)*
And the expected output should be: *True*

Exercise 2:
Implement a function to compute cumulative sum of the values in numeric list. Name your function ***cum_sum(lst, limit)***, with two input parameters: ***lst*** is the list of numeric values, ***limit*** indicates up to which element in the list the sum is computed. Parameter limit should be optional and, if not specified, then the entire list should be used. The result should be returned from the function.

An example of a using *cum_sum()* function: *cum_sum(range(10))*
And the expected output should be: *45*

An example of a using *cum_sum()* function: *cum_sum(range(10), 4)*
And the expected output should be: *6*

Exercise 3:
Implement a function that computes factorial without utilizing recursion. Use a loop to compute factorial of a given number. Name your function ***factorial(n)***, where n is the input parameter indicating a number for which to compute the factorial. Make sure to account for negative numbers in your input. Return -1 for any invalid input. The result should be returned from the function.

An example of a using *factorial()* function: *factorial(5)*
And the expected output should be: 120

An example of a using *factorial()* function: *factorial(-5)*
And the expected output should be: -1

Exercise 4:
This exercise is an extension of the previous exercise. Write a function named ***multiple_factorials(n)***, where ***n*** is the maximum number for which the factorial is computed. A factorial should also be computed for each value less than n. The function does not return any value but instead print the result of each factorial computation on a separate line, starting from the largest number and going down to the output for factorial for 1. For any invalid input n the

function should print “ERROR: invalid input”. In this exercise you should utilize the function you wrote for the previous exercise.

An example of a using *factorial()* function: *multiple_factorials(10)*
And the expected output should be:
3628800
362880
40320
5040
720
120
24
6
2
1

An example of a using *factorial()* function: *multiple_factorials(0)*
And the expected output should be:
ERROR: invalid input

Exercise 5:
This exercise will provide you with some practice with object oriented programming in Python. In this exercise you will work with a list of *Student* objects. Class *Student* is a child of class *Person*. Class *Person* has the following attributes:
firstname
lastname
age

Class *Person* also has a method named *can_consume_alcohol()*, which returns *True* if the student is of legal drinking age (21 years old), and *False* otherwise. Class Person also has two getters, for the first name and the last name.

Class *Student* inherits from class *Person* and has the following additional attributes:
gpa
status

Class Student has a getter named *get_name()*, which returns first and last name of the student concatenated together.

Create a list of the following Student objects:
Mike Smith, 21 yo, 3.7 gpa, Senior status
Larry Mushroom, 19 yo, 2.1 gpa, Sophomore status

Marry Wolf, 22 yo, 3.2 gpa, Senior status
Tommy Tree, 20 yo, 3.5 gpa, Sophomore status
Laura Tall, 21 yo, 3.1 gpa, Junior status
Amy Paris, 18 yo, 3.9 gpa, Freshman status

Part1:
Iterate over the list of students and calculate how many are legal to drink alcohol. Output the computed value in the following format:
Out of 6 students 3 are legal to consume alcohol.

Part2:
Tabulate and output how many students of each status are present in the list. Make sure to sort the output by the status. Most convenient way to implement this is with a dictionary. Iterate over the list and collect counts of each status representation in a dictionary, then sort this dictionary and print out its entries. Your output should look like this:
Senior 2
Sophomore 2
Junior 1
Freshman 1

Part3:
Sort your list of students by the gpa (in descending order) and output the students in the new sorted list in the following format:
Amy Paris (3.9)
Mike Smith (3.7)
Tommy Tree (3.5)
Marry Wolf (3.2)
Laura Tall (3.1)
Larry Mushroom (2.1)

Submission:

Compress all the homework .py files into a single compressed file named “Assignment1”, with the appropriate file extension. Make sure submit by 11:59pm on the due date listed in Canvas. Submit your solution via Canvas.

If you have any questions, message me or the grader or both:
Yulia.Newton@sjsu.edu
madhujitaranjit.ambaskar@sjsu.edu

Grading:

I will return the grades as fast as we can grade this homework. Normally it should not take more than a few weeks.

A total of 25 points are possible for this homework assignment.