Important

1. Due Date: **Thursday, October 5th**
2. This homework is graded out of 100 points.
3. This is an Individual Assignment. You may collaborate with other students in this class. Collaboration means talking through problems, assisting with debugging, explaining a concept, etc. Students may only collaborate with fellow students currently taking CS 1301, the TA's and the lecturer. You should not exchange code or write code for others. For individual assignments, each student must turn in a unique program. Your submission must not be substantially similar to another student's submission. Collaboration at a reasonable level will not result in substantially similar code.
4. For Help:
    - TA Helpdesk (Schedule posted on class website.)
    - Email TA's or use Piazza Forums Notes:
    - How to Think Like a Computer Scientists [http://openbookproject.net/thinkcs/python/english3e/]
    - CS 1301 Python Debugging Guide [http://www.cc.gatech.edu/classes/AY2016/cs1301_spring/CS-1301-Debugging-Guide/index.html]
5. Don't forget to include the required collaboration statement (outlined on the syllabus). Failing to include the Collaboration Statement will result in no credit.
6. Do not wait until the last minute to do this assignment in case you run into problems.
7. Comment or delete all your function calls. When your code is run, all it should do is build without any errors. Having function calls or extraneous code outside the scope of functions will result in a lower grade. (import statements and global variables are okay to be outside the scope of a function)
8. **Read the entire specifications document before starting this assignment.**
9. **IF YOUR CODE CANNOT RUN BECAUSE OF AN ERROR, IT IS A 0%.**
10. **IF YOUR DELIVERABLES ARE NOT NAMED EXACTLY AS STATED IN THE INSTRUCTIONS IT IS A 0%.**

Introduction

The goal of this homework is for you to showcase your knowledge about how to properly use tuples and modules. For this assignment you will need to implement 6 functions. The functions you are asked to code will use tuples and modules. Refer to the rubric to see how points will be rewarded for each function. You have been given `HW5.py` to fill out with instructions in the docstrings. However, below you will find more detailed information to complete your assignment. Read it thoroughly before you begin. You have until **Thursday, October 5th** to complete this assignment. Don't forget to include your name and your collaboration statement. Re-download your submission from T-Square after you submit it to make sure that your code runs successfully.

Function name: **complex_calculator**
Parameters: a string, a list
Return value: An int, a float, or None

Description: Write a function that takes two arguments. The first argument will be a name (string). The second argument will be a list of numbers. The name will decide what function from the math module will be used. Refer to the table below to decide which name will correspond to which math function. If the name passed into the function is not one in the table, return None. The list of numbers contains the arguments to be used for the math function. However, some of these math functions take only 1 parameter and others take 2. Assume that if the math function needs one parameter, the list will be of length one, and if the math function needs two parameters the list will be of length two. Some math functions might give a long decimal number. Please round the numbers to 3 decimal points if needed.

Name	**Function**
"ceil"	math.ceil
"fabs"	math.fabs
"gcd"	math.gcd
"pow"	math.pow
"sin"	math.sin

Notes:

- Assume that if a math function is called, the numbers in the list will work for that math function. This is not a case sensitive function (i.e. "ceil" is the same as "CEIL", "Ceil", or "cEil", etc.), you must account for this.
- Look in the python library to see which functions take 1 parameter and which take 2. You may find that reading https://docs.python.org/3/library/math.html will be helpful.

Test Cases:

```
>>> test1 = complex_calculator("pow", [2, 4])
>>> print(test1)
16.0

>>> test2 = complex_calculator("code", [1, 2])
>>> print(test2)
None

>>> test3 = complex_calculator("Gcd", [24, 8])
>>> print(test3)
8
```

Function name: **help_recruit**
Parameters: A string
Return value: A tuple of length 3

Description: Write a function that takes in a string that holds the information of a recruit. You may assume that the string will always be formatted in this way: "NAME: SPORT, RANK". Return a tuple in the format (SPORT (string), RANK (int), NAME (string)). Please note there is a space before SPORT. The tuple that is returned must not have any spaces within the strings.

Test Cases:

```
>>> test1 = help_recruit("Joseph Macrina: Football, 1")
>>> print(test1)
('Football', 1, 'Joseph Macrina')

>>> test2 = help_recruit("Chris Eubanks: Tennis, 8")
>>> print(test2)
('Tennis', 8, 'Chris Eubanks')
>>> test3 = help_recruit("Ashley Askin: Volleyball, 6")
>>> print(test3)
('Volleyball', 6, 'Ashley Askin')
```

Function name: **recruiting_profile**
Parameters: A list of strings, sport_name
Return value: A list of tuples

Description: Write a function that takes in a list of strings. The list can be empty, in which case return an empty list. Each string contains information about a person. You may assume that the string will always be formatted in this way: "NAME: SPORT, RANK". Call the **help_recruit** function to convert each string to a tuple. Find all the tuples that have sport_name and add them to a list sorted by their rank. However, the tuples in this list should be formatted as (RANK(int), SPORT(string), NAME(string)). No one will have the same rank. Return the list.

Notes:

- If you have a function that returns a tuple of length three another way you can call the function and store the returned values is:
    - `a, b, c = myfunc()`
- When sorting the tuples, you may use a built-in function.

Test Cases:

```
>>> recruits= recruiting_profile(["Jair Anderson: Football,4","TaQuon Marshall:
Football, 1", "Daniel Yun: Tennis, 3", "Gabby Benda: Volleyball, 7","Ashley
Askin: Volleyball, 2", "Sydney Wilson: Volleyball, 5", "Chris Eubanks: Tennis,
1", "Zach Matthews: Football, 11", "Andrew Li: Tennis, 2"], "Tennis")
>>>print(recruits)
```

```
[(1, 'Tennis', 'Chris Eubanks'), (2, 'Tennis', 'Andrew Li'), (3, 'Tennis',
'Daniel Yun')]

>>> recruit1=recruiting_profile(["Wade Bailey: Baseball, 3", "Josh Okogie:
Basketball, 1", "Tadric Jackson: Basketball, 4","Megan Young: Swimming, 7","Ben
Lammers: Basketball, 2","Jake Lee: Baseball, 8","Brad Oberg: Swimming, 8","Megan
Hansen: Swimming, 5","Joey Bart: Baseball, 5", "Matt Casillas: Swimming, 2"],
"Swimming")
>>>print(recruit1)
[(2, 'Swimming', 'Matt Casillas'), (5, 'Swimming', 'Megan Hansen'), (7,
'Swimming', 'Megan Young'), (8, 'Swimming', 'Brad Oberg')]

>>> recruits2= recruiting_profile(["Lauren Frerking: Volleyball, 11","James Clark:
Golf, 3", "Michael Pisciotta: Golf, 2" ], "Tennis")
>>> print(recruits2)
[]
```

Function name: **precious_pets**
Parameters: A string
Return value: A tuple of length 2

Description: Write a function that takes in a string containing information about multiple pets. You may assume that the string will always be formatted in this way: "Pet_Name, Pet_Excitement_Level + Pet_Name, Pet_Excitement_Level + etc.". Using the information given in the string, find the pet with the highest excitement level, and return the pet's name and the pet's excitement level in a tuple formatted as (Pet_Name, Pet_Excitement_Level).

Notes:

- An empty string will never be passed in.
- If two pets have the same excitement level, then return the pet that comes last in the string

Test Cases:

```
>>> test1 = precious_pets("Ruby, 10000 + Dawg, 2 + Fluffy, 9999")
>>> print(test1)
('Ruby', 10000)

>>> test2 = precious_pets("Henry, 300 + Bean, 500 + TaoTao, 900 + Hen,
900")>>> print(test2)
('Hen', 900)

>>> test3 = precious_pets("Joey, 6 + Lima, 25 + Tofu, 2 + Grump, 1")
>>> print(test3)
('Lima', 25)
```

Function name: **hungry_puppies**
Parameters: A list of tuples, a float, a float
Return value: A tuple

Description: Write a function that takes a list of tuples, a float representing the price per a pound of food, and a float representing the total amount of money that can be spent on dog food. The tuples in the list contain a puppy's name (string), the puppy's age (int), and how many pounds of food the puppy can eat in a day (float). If the puppy is younger than 2, then only add half of the pounds that the puppy needs to eat. Return a tuple in the following order with the following information: (Boolean value representing if the amount of dog food can be afforded or not, Floating point value representing the total pounds of dog food that the puppies need, a String representing the name of the dog who needs the most food after taking into account the halving of the pounds).

Notes:

- Assume all numbers in the tuples are positive. In the returned list, all numbers in the tuples should be floats.
- The tuples can be in any order
- If the list is empty, then return a tuple with True, 0, and an empty string
- If there is a tie for the dog who needs the most amount of food, then add the dog who appears first in the string.

Test Cases:

```
>>> test1 = hungry_puppies([(9.0, "Scott", 2), (5, "Blue", 14.0),
("Cindy", 1, 20.0)], 4.2, 500.0)
>>> print(test1)
(True, 33.0, 'Blue')

>>> test2 = hungry_puppies([], 2.2, 400.0)
>>> print(test2)
(True, 0, '')

>>> test3 = hungry_puppies([("Buster", 10.0, 4), (3, 12.3, "Git"), (4.1
"Arty", 1)], 3.6, 87.0)
>>> print(test3)
(False, 24.35, 'Git')
```

Function name: **test_release**
Parameters: a list of tuples
Return value: a list of tuples

Description: Write a function that takes in a list of tuples as a parameter and returns a tuple of each student's average in a list. Each tuple in the list will contain floating point values to represent exam grades and a string name. Return the average for each student in a tuple with the student's name first and the student's average second. Round the average to two decimal points. If there are less than three grades in the tuple, then do not include the student in the dictionary. If the list is empty, return an empty list.

Test Cases:

```
>>> test1 = test_release([(80.0, 91.0, 98.0, "Jan"), (100.0, 97.8, "Oscar",
78.0), ("Jim Jr.", 85.0, 85.0)])
>>> print(test1)
[('Jan', 89.67), ('Oscar', 91.93)]

>>> test2 = test_release([(80.0, 91.0, "Jan"), (61.0, 90.0, "Frank", 92.0)])
>>> print(test2)
[('Frank', 81.0)]

>>> test3 = test_release([(15.0, "Michael", 98.0, 100.0), (65.0, "Joey",
70.0, 90.0), ("Jamie", 80.0, 81.0, 80.0)])
>>> print(test3)
[('Michael', 71.0), ('Joey', 75.0), ('Jamie', 80.33)]
```

Grading Rubric

complex_calculator:	10 points
help_recruit:	10 points
recruiting_profile:	10 points
precious_pets:	20 points
hungry_puppies:	25 points
test_release:	25 points
Total	100/100 points

Provided

The following file(s) have been provided to you. There are several, but you will only edit one of them:

1. `HW5.py`
   This is the file you will edit and implement. All instructions for what the methods should do are in the docstrings.

2. `HW5_test.py`
   This is a file containing tests for you to use if you will like to debug and test your code. You are not required to edit, submit, or even use this file.

Deliverables

You must submit all of the following file(s). Please make sure the filename matches the filename(s) below. Be sure you receive the confirmation email from T-Square, and then download your uploaded files to a new folder and run them.

1. `HW5.py`
   If this file does not run (if it encounters an error while trying to run), you will get no credit on the assignment. If your submission is not named exactly like this, you will get no credit on the assignment.