Homework 5 - Lists, Tuples, and Modules

CS 1301 - Intro to Computing - Fall 2021

Important

- Due Date: **Tuesday, October 5th, 11:59 PM**.
- This is an individual assignment. High-level collaboration is encouraged, **but your submission must be uniquely yours.**
- Resources:
  - TA Helpdesk
  - Email TA's or use class Piazza
  - How to Think Like a Computer Scientist
  - CS 1301 YouTube Channel
- Comment out or delete all function calls. Only import statements, global variables, and comments are okay to be outside of your functions.
- **Read the entire document before starting this assignment.**

The goal of this homework is for you to enhance your understanding of lists, tuples, and modules. The homework will consist of 5 functions for you to implement. You have been given the `HW05.py` skeleton file to fill out. However, below you will find more detailed information to complete your assignment. Read it thoroughly before you begin.

Hidden Test Cases: In an effort to encourage debugging and writing robust code, we will be including hidden test cases on Gradescope for some functions. You will not be able to see the input or output to these cases. Below is an example output from a failed hidden test case:

```
Test failed: False is not true
```

Written by Alex King (aking1@gatech.edu) & Nikhila Alavala (nalavala6@gatech.edu)

Helpful Information to Know

Modules

Modules allow you to use code from another source. This could include other code that you've written in a different Python file, or external code written by other programmers. To use a module, you must **import** it into your file first. There are multiple ways to do this. Let's look at some ways to import and use the `pi` constant and `sqrt` function from Python's built-in **math** module.

Note: By convention, `import` statements should be at the top of your file, **not in a function.**

- Importing the module itself:

```
import math

val = math.pi * math.sqrt(4)
```

- Importing specific item(s) from a module:

```
from math import pi, sqrt

val = pi * sqrt(4)
```

- Importing a module with a different name:

```
import math as m

val = m.pi * m.sqrt(4)
```

Dinner With Friends

Function Name: dinnerParty()
Parameters: list of names and availabilities (`list`), day (`str`)
Returns: list of friends (`list`)
Description: You and your friends want to have dinner on a given day this week to enjoy the new fall weather but you need help figuring out which of your friends are available on that day. Given a list of tuples where each tuple containts a person's name and a list of the days they are available to get dinner, return a list that contains the names of the people that can get dinner on the specific day passed into the function.

```
>>> availabilities = [("Amy", ["Monday", "Tuesday", "Thursday", "Friday"]),
                      ("Tommy", ["Monday", "Wednesday", "Thursday"]),
                      ("Sarah", ["Wednesday", "Thursday", "Friday"])]
>>> dinnerParty(availabilities, "Thursday")
['Amy', 'Tommy', 'Sarah']
```

```
>>> availabilities = [("Larry", ["Sunday", "Monday", "Tuesday", "Friday"]),
                      ("Bob", ["Monday", "Wednesday", "Thursday"]),
                      ("Sally", ["Thursday"])]
>>> dinnerParty(availabilities, "Saturday")
[]
```

Fall Weather

Function Name: whatShouldIWear()
Parameters: list of temperatures (`list`), list of recommendations (`list`)
Returns: list of tuples (`list`)
Description: It's finally Fall in Georgia and some of your out-of-state friends need help deciding what to wear. You've been given two lists, one list containing temperature values and another list containing clothing recommendations. Write a function that returns a list of tuples in which each tuple contains the temperature and the corresponding clothing recommendation.

Note: Each temp value in `temps` list corresponds to the clothing recommendations at the same position in the `recs` list i.e. the first clothing recommendation corresponds to the first temperature and so forth.

```
>>> temps = [47, 64, 72]
>>> recs = ["It's sweater season baby", "Only people from Georgia need a jacket",
"Shorts. How is it not summer?!"]
>>> whatShouldIWear(temps, recs)
[(47, "It's sweater season baby"), (64, 'Only people from Georgia need a jacket'),
(72, 'Shorts. How is it not summer?!')]
```

```
>>> temps = [45, 56, 62]
>>> recs = ["It's sweater season baby", "Only people from Georgia need a jacket",
"It's perfect weather, dress normally"]
>>> whatShouldIWear(temps, recs)
[(45, "It's sweater season baby"), (56, 'Only people from Georgia need a jacket'),
(62, "It's perfect weather, dress normally")]
```

Cheap Meals

Function Name: cheapMeals()
Parameters: list of strings (`list`) and budget (`float`)
Returns: tuple containing menu items (`tuple`)
Description: You love Moe's but you spend too much money on it weekly, so you've decided to budget. Given a budget and a list of all the items on the menu with their corresponding prices, write a function that returns a tuple containing all the menu items that are less than or equal to your budget in **ascending** order of price.

Note: In the given list, each menu item is written in the format `"name of item - price"`

```
>>> menu = ["Burrito Bowl - $2.99", "Tacos - $0.99", "Quesadilla - $6.99"]
>>> budget = 5.00
>>> cheapMeals(menu, budget)
('Tacos', 'Burrito Bowl')
```

```
>>> menu = ["Nachos - $4.99", "Stacks - $8.50", "Burrito Bowl - $6.99"]
>>> budget = 6.00
>>> cheapMeals(menu, budget)
('Nachos',)
```

Mean Girls

Function Name: wednesdays()
Parameters: list of tuples with dates and holidays (`list`)
Returns: list of holidays (`list`)
Description: As you already know, on Wednesdays we wear pink! Help Regina and her crew figure out which fall holidays they should remember to wear pink. Write a function that takes in a list of tuples containing dates and holidays, and return a list of all the holidays that occur on a Wednesday. You may assume all the inputs are valid dates.

Each tuple in the list is given in the following format:

```
(holiday, day, month, year)
```

Note: Use the **weekday()** function from the **calendar module** to check the number of the day of the week.

```
>>> holidays = [("Autumn Equinox", 22, 9, 2021),
                ("Johnny Appleseed Day", 26, 9, 1774),
                ("Labor Day", 6, 9, 1882),
                ("Yom Kippur", 15, 9, 2021)]
>>> wednesdays(holidays)
['Autumn Equinox', 'Labor Day', 'Yom Kippur']
```

```
>>> holidays = [("Chocolate Milk Shake Day", 12, 9, 2018),
                ("Elephant Appreciation Day", 22, 9, 1996),
                ("World Beard Day", 4, 9, 2019)]
>>> wednesdays(holidays)
['Chocolate Milk Shake Day', 'World Beard Day']
```

Starbucks Drinks

Function Name: starbucksFanatic()
Parameters: list of starbucks menu items (`list`)
Returns: list of tuples containing menu item name and price (`list`)
Description: You are a huge fan of Starbucks seasonal items, and you're so excited to try all of the new Fall drinks and snacks this year! Write a function that takes in a list of Starbucks menu items and return a list of tuples where each tuple contains the menu item name and its

price respectively. At the end of your returned list, also include the sum of all menu items. Round your sum to two decimal places.

We have provided a Python file called `starbucks.py` that contains a function called menu(). This function will take in a starbucks menu item and return the price of that item as a float. If the menu item is not found, the menu function will return 0. **Do not add menu items with a price of zero to your final returned list of tuples.**

```
>>> fallMenuItems = ['pumpkin spice latte', 'apple crisp macchiato',
                     'iced apple crisp macchiato', 'pumpkin cream cold brew',
                     'croissant']
>>> starbucksFanatic(fallMenuItems)
[('pumpkin spice latte', 4.78), ('apple crisp macchiato', 5.23),
 ('iced apple crisp macchiato', 4.33), ('pumpkin cream cold brew', 6.18), 20.52]
```

```
>>> fallMenuItems = ['double-smoked bacon, cheddar & egg sandwich',
                     'iced pumpkin spice latte', 'pumpkin cream cold brew']
>>> starbucksFanatic(fallMenuItems)
[('double-smoked bacon, cheddar & egg sandwich', 5.89),
 ('iced pumpkin spice latte', 5.5),
 ('pumpkin cream cold brew', 6.18), 17.57]
```

Grading Rubric

Function	Points
dinnerParty()	20
whatShouldIWear()	20
cheapMeals()	20
wednesdays()	20
starbucksFanatic()	20
Total	**100**

Provided

The `HW05.py` skeleton file has been provided to you. This is the file you will edit and implement. All instructions for what the functions should do are in this skeleton and this document.

Submission Process

For this homework, we will be using Gradescope for submissions and automatic grading. When you submit your `HW05.py` file to the appropriate assignment on Gradescope, the autograder will run automatically. The grade you see on Gradescope will be the grade you get, unless your grading TA sees signs of you trying to defeat the system in your code. You can re-submit this assignment an unlimited number of times until the deadline; just click the "Re-submit" button at the lower right-hand corner of Gradescope. You do not need to submit your `HW05.py` on Canvas.