Homework 5 - Lists, Tuples, and Modules

CS 1301 - Intro to Computing - Fall 2022

Important

- Due Date: **Tuesday, October 4th, 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 Josh Tabb (jtabb6@gatech.edu) & Harshith Lanka (hlanka3@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)
```

Intramurals Team Picker

Function Name: teamPicker()
Parameters: sportInterests (`list`), intramural (`str`)
Returns: team (`list`)
Description: You have decided that you want to play in one intramural sport this semester but want to be in a team with your friends. Given the intramural (`str`) you want to play and your friends' sportInterests (`list`), a list of tuples with each tuple containing your friends' names and a list of their intramural interests, write a function that returns a list of friends who would be interested in playing with you. If none of your friends want to play your intramural sport, the functions should return `"Free agent :/"`.

The list `sportInterests` will be in the format stated below:

```
[(friend1, [list of friend1's sports interests]),
 (friend2, [list of friend2's sports interests])...]
```

Note: The returned list should be in **alphabetical order.**

```
>>> sportInterests = [('Timothy', ['Basketball', 'Tennis', 'Badminton']),
                      ('Mo', ['Tennis', 'Ping Pong', 'Dodgeball']),
                      ('Ashley', ['Sand Volleyball', 'Volleyball'])]
>>> intramural = "Tennis"
>>> teamPicker(sportInterests, intramural)
['Mo', 'Timothy']
```

```
>>> sportInterests = [('Kyle', ['Soccer', 'Badminton']), ('Omar', ['Soccer'])]
>>> intramural = "Badminton"
>>> teamPicker(sportInterests, intramural)
['Kyle']
```

Fall Vacation Planner

Function Name: vacationPlanner()
Parameters: trips (`list`), costs (`list`)
Returns: vacationSpots (`tuple`)
Description: You and your friends are trying to decide where to go for Fall Break. You have been given two lists: One list containing trips and the other containing each trip's respective cost. Write a function that returns a tuple with the cheapest trip(s) and a list of tuples (each tuple containing a trip and its cost) in **alphabetic order**. If more than one of the trips are the cheapest, have the other trips also be in the returned tuple also in **alphabetic order**. If the passed in lists are empty, the returned tuple should have an empty string and an empty list.

Note: Each trip `(str)` value in the `trips` list corresponds to the cost `(int)` at the same index in the `costs` list i.e. the first trip corresponds to the first cost and so on, thus you can assume both the lists will be of the same length.

```
>>> trips = ['China', 'England', 'Australia', 'Japan', 'India']
>>> costs = [1200, 800, 1950, 1730, 1500]
>>> vacationPlanner(trips, costs)
('England', [('Australia', 1950), ('China', 1200), ('England', 800),('India', 1500),
             ('Japan', 1730)])
```

```
>>> trips = ['NYC', 'DC', 'Miami', 'Hollywood']
>>> costs = [150, 150, 190, 460]
>>> vacationPlanner(trips, costs)
('DC', 'NYC', [('DC', 150), ('Hollywood', 460), ('Miami', 190), ('NYC', 150)])
```

Fast Food Orders

Function Name: fastFood()
Parameters: foods (`list`), costs (`list`)
Returns: friendsOwed (`list`)
Description: You and your friends are hungry late one night, and decide to order some fast food, but you're the only one with their wallet with you. Write a function that calculates how much each of your friends owe you and return it in the form of a list of tuples, where each tuple contains a name of a friend, and how much that friend owes you. In order to calculate this, you are given 2 lists: `foods` and `costs`. The `foods` list is a list of tuples containing a certain food and the friend that wants that food. The `costs` list is a list of tuples that each contain a specific food and its cost.

The returned list should be in alphabetical order (according to the names of your friends).

```
>>> foods = [('Chris', 'Burger'), ('Josh', 'Ice cream'), ('Parul', 'K-dog'),
             ('Paige', 'Fries')]
>>> costs = [('Ice cream', 1.0), ('Fries', 0.5), ('Burger', 3.0),
             ('K-dog', 5.0)]
>>> fastFood(foods, costs)
[('Chris', 3.0), ('Josh', 1.0), ('Paige', 0.5), ('Parul', 5.0)]
```

```
>>> foods = [('Priyam', 'Frosty'), ('Audrey', 'Bread'),
             ('Priyam', 'Onion rings'), ('Jane', 'Chicken nuggets')]
>>> costs = [('Bread', 2.0), ('Onion rings', 1.0), ('Frosty', 0.75),
             ('Chicken nuggets', 4.0)]
>>> fastFood(foods, costs)
[('Audrey', 2.0), ('Jane', 4.0), ('Priyam', 1.75)]
```

Review Session

Function Name: reviewSession()
Parameters: rooms (`list`), dates (`list`)
Returns: roomsAvailable (`list`)
Description: The next exam review session is coming up and you have to be in charge of booking a room! However, availability on campus is very limited and you can only book review sessions on either Thursdays or Fridays. Write a function that takes in two lists as parameters: one containing the names of the available rooms and another containing tuples of their available days in the format: `(month, day)`. Using the calendar module, your function should return a list containing the name of the rooms which will be available on either a Thursday or Friday. **The returned list should be in alphabetical order, and should contain no repeated elements.**

Note: Use the **weekday()** function from the **calendar module** to check the number of the day of the week. The function takes in three values `year, month, day`, year should equal to 2022.

```
>>> rooms = ['CULC 144', 'CULC 152', 'CCB 016', 'Skiles 246']
>>> dates = [(9,30), (10,1), (10,22), (10,21)]
>>> reviewSession(rooms, dates)
['CULC 144', 'Skiles 246']
```

```
>>> rooms = ['CULC 300', 'PG 100', 'Smith 112']
>>> dates = [(11,3), (11,3), (12,7)]
>>> reviewSession(rooms, dates)
['CULC 300', 'PG 100']
```

Esports Bracket

Function Name: esportsBracket()
Parameters: tourneyList (`list`)
Returns: tourneyWinner (`str`)
Description: You and your friend are watching a Top 4 in a tournament, and want to predict the winner of the tournament. You are given a list containing 2 sublists. Each of these sublists contains two team names who will face off. The winners of each of these 2 sublists will then play each other to be the winner of the tournament. Use the given **`esportsMatch.py`** file to predict the winners of each match, and thus ultimately predict the winner of the tournament.

We have provided you with a file called **`esportsMatch.py`** that contains a function called `matchHistory()`. This function will take in 2 team names and return the name of the team that is favored to win. If the team is not found, the function returns `None`. **You should not submit `esportsMatch.py` on Gradescope.**

```
>>> tourneyList = [['DRX', 'Optic'], ['Paper Rex', 'Fnatic']]
>>> esportsBracket(tourneyList)
'Paper Rex'
```

```
>>> tourneyList = [['Mang0', 'Hbox'], ['Plup', 'Leffen']]
>>> esportsBracket(tourneyList)
'Mang0'
```

Grading Rubric

Function	Points
teamPicker()	20
vacationPlanner()	20
fastFood()	20
reviewSession()	20
esportsBracket()	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 “Resubmit” button at the lower right-hand corner of Gradescope. You do not need to submit your `HW05.py` on Canvas.