Homework 8 - APIs

CS 1301 - Intro to Computing - Spring 2020

Important

- Due Date: **Tuesday, March 24th, 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.**

Responsible Computing: Some functions in this assignment will have a 'Responsible Computing' badge next to them. These functions are written to help you think about how you can use your skills as a programmer to analyze problems related to *sustainability*, such as ethics, health, and the environment.

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
```

API Keys: In order to prevent people from spamming/abusing their services, some APIs require you to register for keys so that you can use them. You will need to register for an API key for Spoonacular, the API used for the first three problems. Register for a key at this link (it's free): https://spoonacular.com/food-api/console#Plan. Once you create an account, click "Profile" on the left menu, and then click the "Show/Hide API Key" button. Use this key in each of your requests to the Spoonacular API. You do not need a key for the Bloowatch API.

Important Note: API data changes over time. Don't be alarmed if the outputs you're seeing in the SHELL are different than the test cases in this PDF. For debugging, you may have to rely on manually looking through the API data and checking if your outputs match the requirements of the function.

Written by Arushi Gupta (arushigupta@gatech.edu), Arvin Poddar (apoddar32@gatech.edu), & Anthony Zheng (azheng47@gatech.edu).

API Documentation: This homework uses APIs for each problem. In order to use an API properly, looking through the documentation is essential. Here is the documentation for the two APIs in this homework:

- **Spoonacular:** https://spoonacular.com/food-api/docs
- **Bloowatch:** http://www.bloowatch.org/developers/api

Cook Fast

Function Name: cookFast()
Parameters: dish name (`str`), amount of minutes (`int`)
Returns: list of foods (`list`)
Description: You're late for class, and need to make a meal as fast as possible. Using the Spoonacular API, write a function that lets you look up recipes for a particular dish. Return a list of recipes that can be prepared in less than a given amount of minutes. Make sure that each recipe prepares at least 4 servings of food.

```
>>> print (cookFast("pasta", 30))
['Spaghetti Carbonara', 'Copycat Panera Macaroni and Cheese']
```

```
>>> print (cookFast("sandwich", 35))
['Croque Monsieur Ham and Cheese Sandwich', 'Turkey Salad Sandwich', 'Hot Turke
y Sandwich', 'Grilled Chicken Sandwich with Cheese']
```

Nutrients

Function Name: nutrients()
Parameters: nutrient name (`str`), minimum amount (`int`), maximum amount (`int`)
Returns: list of foods (`list`)
Description: To make sure you're getting essential micronutrients (like calcium, sodium, or iron) from your food, you want to use the Spoonacular API to find foods that have specific amounts of nutrients. Given a specific micronutrient (`str`), a minimum amount of the nutrient in milligrams (`int`), and a maximum amount of the nutrient in milligrams (`int`), return a list of foods that contain the nutrient within (inclusive) the acceptable range of milligrams. You can assume that the given nutrient is valid, and that the minimum will always be below the maximum.

```
>>> print (nutrients("Calcium", 1500, 1510))
['Sautéed Brussels Sprouts', 'Multigrain Maple Mini Muffins', 'Confiture De Lai
t - Dulce De Leche', 'Croute Savoyarde', 'Beautiful Butterfly Cake', 'Pepperoni
 Pizza', 'Ooey Gooey Blackberry Butter Bars', 'Chocolate Marshmallow Peanut But
ter Brownies', 'Lobster Macaroni and Cheese', 'Baked Lasagne']
```

```
>>> print (nutrients("VitaminB12", 200, 240))
['Escarole - Stuffed Seared Trout', 'Whole Fish Roasted With Fennel & Olives',
'Smoked Trout Recipe', 'Smoked Barbecue Baby Back Ribs']
```

Grocery Time

Function Name: groceryTime()
Parameters: recipeID (`int`), list of allergies (`list` of `str`)
Returns: dictionary (`dict`)
Description: You're about to go to the grocery store in order to pick up ingredients for a new dish you will cook! Given a list of allergies and the recipeID, find the ingredients you can eat (ie: not in your list of allergies) and return a dictionary where each key is the aisle the ingredient is found, and the value is a list of tuples. For each tuple, the first element is the ingredient (`str`) and the second element as the amount (use US measures, not metric) needed for the recipe (`float`).

```
>>> recipeID = 1003464
>>> allergies = ["blueberries", "flour", "nutmeg", "rhubarb"]
>>> print (groceryTime(recipeID, allergies))
{'Spices and Seasonings': [('salt', 0.333)], 'Produce': [('lemon juice', 1.0)],
 'Refrigerated': [('pie dough round', 2.0)], 'Baking': [('granulated sugar', 0.
75), ('quick cooking tapioca', 2.0)], 'Milk, Eggs, Other Dairy': [('egg white',
 1.0), ('unsalted butter', 0.5)]}
```

```
>>> recipeID = 716429
>>> allergies = ["butter", "cloves", "white wine", "cheese"]
>>> print (groceryTime(recipeID, allergies))
{'Pasta and Rice': [('pasta', 6.0), ('whole wheat bread crumbs', 0.25)], 'Spice
s and Seasonings': [('red pepper flakes', 2.0), ('salt and pepper', 2.0)], 'Pro
duce': [('cauliflower florets', 2.0), ('garlic', 5.0), ('scallions', 3.0)], 'Oi
l, Vinegar, Salad Dressing': [('extra virgin olive oil', 1.0)]}
```

Animal Population Responsible Computing

Function Name: animalPopulation()
Parameters: maximum population (`int`)
Returns: list of tuples (`list`)
Description: You received a job offer from the World Wildlife Fund (WWF) as a junior software developer. The WWF is an international organization working in the field of wilderness preservation. They have released information about vulnerable/endangered species. In an effort to organize your data, you want to find animals whose populations are below a given maximum number. Your job is to use the Bloowatch API and return a list of tuples where each tuple is the (animal_name (`str`), population (`int`)). If the population for an animal is a range of val-

ues, use the lower bound of the range for the population.
Note: If the population for an animal is unknown, do not include this animal in the returned list.
Hint: The `.replace()` function may be useful for this problem.

```
>>> maximum = 10000
>>> print (animalPopulation(maximum))
[("Bengal Tiger", 2500), ("Giant Panda", 2000), ("Snow Leopard", 3500), ("Black
 Rhino", 5000)]
```

```
>>> maximum = 3500
>>> print (animalPopulation(maximum))
[("Bengal Tiger", 2500), ("Giant Panda", 2000)]
```

Encylopedia Responsible Computing

Function Name: encyclopedia()
Parameters: endangerment level (`str`)
Returns: a dictionary of animals (`dict`)
Description: In order to spread awareness of endangered animals, you want to write an encyclopedia. Using the Bloowatch API, write a function that takes in a certain endangerment level (`str`) and returns a dictionary, where the keys are all the animals with that endangerment level, and the values are a short description of each animal. For the description, use the first sentence from the original description the API provides for each animal. If the endangerment level is not valid (something other than "Vulnerable" "Endangered", or "Critically Endangered"), return `None`.

```
>>> print (encyclopedia("Vulnerable"))
{"Giant Panda": "Giant panda bear is native to south central China and lives ma
inly high in the mountains in cool and wet bamboo forests which is their primar
y source of food.", "Snow Leopard": "The snow leopard is a large cat native to
alpine and subalpine ranges of eastern Afghanistan, Mongolia and western China.
", "Polar Bear": "The polar bear is a large bear classified as a marine mammal
because it spends most of its live on the sea ice of the Arctic Ocean."}
```

```
>>> print (encyclopedia("Critically Endangered"))
{'Gorilla': 'Gorillas are apes that live in the forests of central Sub-
Saharan Africa.', 'Orangutan': "Orangutans are the world's largest tree-
climbing mammals and are closely related to humans.", 'Black Rhino': 'Black rhi
no is a species of rhinoceros, native to eastern and southern Africa.'}
```

Grading Rubric

Function	Points
cookFast()	15
nutrients()	20
groceryTime()	20
animalPopulation()	25
encyclopedia()	20
Total	**100**

Provided

The `HW08.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 `HW08.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 resubmit 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 `HW08.py` on Canvas.