Homework 2 - Conditionals

CS 1301 - Intro to Computing - Spring 2021

Important

- Due Date: **Tuesday, September 7th, 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 practice and understand how to write functions that implement conditionals. The homework will consist of 5 functions for you to implement. You have been given `HW02.py` skeleton file to fill out. Please read this PDF thoroughly as you will find more detailed information to complete your assignment.

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 Carolyn Yuan (cyuan78@gatech.edu) and Assata Quinichett (aquinichett@gatech.edu)

Helpful Information To Know

Print vs Return

Two concepts that may be difficult for beginner programmers to differentiate between are the print function and the return statement. While it may appear that they do the same thing, it is important to note that they are quite different. The purpose of the print function is to display information to the user. You cannot save what you print. The return statement, on the other hand, is part of a function definition. All functions have a return value, whether you explicitly write a return statement or not. Functions that do not explicitly have a return statement always return the value None. The return statement is useful because it allows you to give a value to a function. This allows you to either save it for later, use it in a more complex expression, or print it for human consumption.

For example, let's say we have the following two functions below in a file called `file.py`:

```
def printFunc():
    print(2)

def returnFunc():
    return 2
```

This is what would happen if we ran the file and typed the following into the Python shell:

```
>>> a = printFunc()
2
>>> print(a)
None
```

Notice that although the number '2' is printed to the screen, the variable we assigned the function call to, a, has the value None (`NoneType`).

```
>>> b = returnFunc()
>>> print(b)
2
```

When we call returnFunc() and assign it to a variable, nothing is printed to the screen because there are no print statements inside the function. However the variable, b, now holds the value 2 (`int`).

Intramurals

Function Name: intramuralGames()
Parameters: gameName (`str`), numFriends (`int`)
Returns: None (`NoneType`)
Description: After the first week at Georgia Tech, you and your group of 7 other friends decide to play an intramural sport! Everyone is interested in `"spike ball"`, `"volleyball"`, and `"ultimate frisbee"`, so you decide to each vote for one. Write a function that takes in the name of the sport and the number of friends that want to play that sport.

If less than 25% of your friend group (including you) want to play this game, print out a string in the following format:

```
"Let’s choose something else."
```

If 25% or more but less than 75% of your friend group (including you) want to play the game, print out a string in the following format:

```
"We will try out {gameName} for a little bit!"
```

If 75% or more of your friend group (including you) want to play the game, print out a string in the following format:

```
"Let’s register for {gameName}!!!"
```

```
>>> intramuralGames("spike ball", 6)
Let's register for spike ball!!!
```

```
>>> intramuralGames("volleyball", 5)
We will try out volleyball for a little bit!
```

Atlantic Station

Function Name: goShopping()
Parameters: item (`str`), quantity (`int`), budget (`float`)
Returns: moneyLeft (`float`) or error message (`str`)
Description: The past two weeks have been stressful! This weekend, you finally have some free time so you and your friends decide to go shopping for new outfits at the H&M in Atlantic Station. Using the table below, write a function that will help you decide whether or not to purhcase the items you like. If the total cost exceeds your budget, return "Not enough money!". Otherwise, return how much money you will have left after your purchase.

Note: Assume the item for goShopping will be a type of clothing from the table below.

Product	Price
`"Shorts"`	13.00
`"T-Shirts and Tanks"`	9.75
`"Swimwear"`	20.00
`"Sunglasses"`	7.50
`"Slides"`	15.50

```
>>> goShopping("Shorts", 2, 30.00)
4.0
```

```
>>> goShopping("T-Shirts and Tanks", 5, 25.00)
'Not enough money!'
```

Dinner Time

Function Name: getDinner()
Parameters: budget (`float`), diningOption (`str`)
Returns: restaurantName (`str`)
Description: After a long day of shopping, you and your friends are hungry so you decide to get dinner. You can only consider restaurants that have prices within your budget, and places that have your friends preferred dining option. Write a function that takes in a budget and dining option, and determine which restaurant you'll be getting dinner from.

Note: Assume the budget will always be less than $40.00.
Note: The strings representing the possible dining options are `"Takeout"`, `"Delivery"`, `"Indoor"`, and `"Outdoor"`.

Prices	Restaurant	Dining Option
<= $10.00	`"Chick Fil A"`	`"Takeout"`, `"Delivery"`
<= $10.00	`"Moe's"`	`"Indoor"`, `"Outdoor"`
> $10.00 and <= $20.00	`"Tin Drum"`	`"Indoor"`, `"Takeout"`
> $10.00 and <= $20.00	`"Umma's"`	`"Outdoor"`, `"Delivery"`
> $20.00 and <= $30.00	`"Yeah Burger"`	`"Indoor"`, `"Outdoor"`, `"Takeout"`
> $20.00 and <= $30.00	`"Flip Burger"`	`"Delivery"`
> $30.00 and <= $40.00	`"Two Urban Licks"`	`"Indoor"`
> $30.00 and <= $40.00	`"Gypsy Kitchen"`	`"Takeout"`, `"Delivery"`, `"Outdoor"`

```
>>> getDinner(15.24, "Indoor")
'Tin Drum'
```

```
>>> getDinner(32.75, "Outdoor")
'Gypsy Kitchen'
```

Backup Restaurant

Function Name: backupRestaurant()
Parameters: budget (`float`), diningOption (`str`), backup (`str`)
Returns: decision (`str`)
Description: Although you have chosen where to get dinner from, you're concerned about how much time it will take to get there, so you've chosen a backup restaurant option. According to Google Maps, it will take you 15 minutes to get to `"Chick Fil A"` , `"Umma's"` , `"Gypsy Kitchen"` and `"Flip Burger"` . For the other restaurants, it will take you 45 minutes to get there.

Write a function that takes in three parameters: your budget, dining option and backup restaurant name. With your budget and dining option, determine your preferred restaurant using the **getDinner()** function. Then, compare the time it takes to get to your preferred restaurant with the time it takes to get to your backup restaurant.

If it takes longer to get to your preferred restaurant, return:

```
"You'll have to get dinner at {backup restaurantName} instead."
```

Otherwise, return:

```
"Yay, you can get dinner at your first choice, {preferred restaurantName}."
```

Hint: You must call **getDinner()** in this function.
Note: Assume it will never take the same amount of time to get to both restaurants.

```
>>> backupRestaurant(32.75, "Outdoor", "Two Urban Licks")
'Yay, you can get dinner at your first choice, Gypsy Kitchen.'
```

```
>>> backupRestaurant(4.50, "Indoor", "Chick Fil A")
"You'll have to get dinner at Chick Fil A instead."
```

Ride Share

Function Name: rideShare()
Parameters: number of riders (`int`), miles(`int`), rideShareOptionaA (`str`), rideShareOptionaB (`str`)
Returns: rideShareOption (`str`)
Description: You and a group of friends are planning on sharing a ride using one of these 4 ridesharing options(Uber, Lyft, Taxi, or Hitch) and want to get the best deal. Given the size of the group and miles you want to travel, compare the prices of the two provided ride sharing options and return the ride sharing option that has the lowest price for the whole group.

The strings representing the four gym memberships are `"Uber"` , `"Lyft"` , `"Hitch"` , and `"Taxi"` .

Note: For **Uber** , there's a base cost of $5 and a cost per mile of $5.50. If more than 3 people join together, the base cost is waived.
Note: For **Lyft** there's a base cost of $10 and a cost per mile of $1.50. There are no group discounts.
Note: For **Hitch**, there's no base cost and a cost per mile cost of $7.50. If more than 2 people join together, they get a discount of $10 per rider from the total cost of the trip.
Note: For **Taxi**, there's no base cost and a cost per mile of $18. There are no group discounts.

Calculate the cost of each of the two ride sharing options the group wants to compare, and return whichever is cheaper. If they are equal, just return rideSharingOptionA.

```
>>> rideShare(2, 6, "Taxi", "Uber")
'Uber'
```

```
>>> rideShare(4, 2, "Lyft", "Hitch")
'Hitch'
```

Grading Rubric

Function	Points
intramuralGames()	15
goShopping()	20
getDinner()	20
backupRestaurant()	20
rideShare()	25
Total	**100**

Provided

The `HW02.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 `HW02.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 `HW02.py` on Canvas.