Homework 2 - Conditionals

CS 1301 - Intro to Computing

Important

- Due Date: **Tuesday, September 1st, 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. 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
```

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:

```python
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`).

Skill Level

Function Name: skillLevel()
Parameters: passRate (`int`)
Returns: "Beginner" or "Moderate" or "Advanced" (`str`)
Description: The semester just started and your professor wants to know how well the class already understands Python by administering a diagnostic exam to all students. Write a function that takes in the percentage of students that passed the exam and returns the skill level of the class. If 25% or less pass, the skill level is "Beginner." If more than 25% and 75% or less pass, the skill level is "Moderate". If more than 75% pass, the skill level is "Advanced".

```
>>> skillLevel(13)
'Beginner'
```

```
>>> skillLevel(78)
'Advanced'
```

Shopping Fun

Function Name: bookStore()
Parameters: item (`str`), walletAmount (`float`), quantity (`int`)
Returns: moneyLeftOver (`float`)
Description: While you wait for your exam results, you decide to head over to the Barnes and Noble Bookstore to pick up some Georgia Tech swag for you and your close friends. A **Shirt** costs $15.50, a **Lanyard** costs $4.25, a **Sweatshirt** costs $25.00, and a **Mug** costs $10.50. Write a function that takes a bookstore item and checks if you can buy that item based on the amount of money in your wallet and the quantity you want to purchase. If you can buy the items requested, return a float that represents how much money is left in your wallet. If you cannot buy the items, return “Not enough money!”. **Round your answer to 2 decimal places.**

```
>>> bookStore("Shirt", 350.48, 8)
226.48
```

```
>>> bookStore("Lanyard", 200.0, 70)
'Not enough money!'
```

Dinner Plans

Function Name: dinnerPlans()
Parameters: distance (`int`), hungerLevel (`str`)
Returns: transportMode (`str`)
Description: After some shopping fun, you and a couple friends want to go out for dinner and are deciding whether you should walk or take an Uber. Write a function that takes in the distance from the restaurant and the hunger level of the group and returns whether you choose to Uber or Walk. There are four hunger levels: “Not Hungry”, “Slightly Hungry”, “Hungry”, “Very Hungry”. Use the table below to make your decision.

distance	hungerLevel	decision
<= 7	Not Hungry	Walk
<= 5	Slightly Hungry	Walk
<= 3	Hungry	Walk
<= 1	Very Hungry	Walk
> 7	Not Hungry	Uber
> 5	Slightly Hungry	Uber
> 3	Hungry	Uber
> 1	Very Hungry	Uber

```
>>> dinnerPlans(4, "Slightly Hungry")
'Walk'
```

```
>>> dinnerPlans(6, "Very Hungry")
'Uber'
```

Weekend Trip

Function Name: weekendTrip()
Parameters: distance (`float`), speed (`float`), freeTime (`float`)
Returns: transportMode (`str`)
Description: Over the weekend, you decide to explore the city and visit Atlanta's various attractions. However, you first want to determine the best way to get there based on how much free time (hours) you have. If your travel time is 20% or less of your free time, return the mode of transportation based on the speed (miles per hour) using the table below. If the travel time takes more than 20% of your free time, return "Going to this destination would take too much time." **Distance will be given in miles and the speed will always be 2.5 mph or greater.**

speed	transportMode
2.5 <= speed <= 15	walking
15 < speed <= 20	biking
20 < speed	driving

```
>>> weekendTrip(7.0, 46.66, 1.0)
'driving'
```

```
>>> weekendTrip(10.0, 5.0, 8.0)
'Going to this destination would take too much time.'
```

Friends

Function Name: textFriends()
Parameters: distance (`float`), speed (`float`), freeTime (`float`), numSnacks (`int`), numFriends (`int`)
Returns: textMsg (`str`)
Description: After you have determined whether you can go to the attraction, you decide to text your friends and tell them how many snacks you should get for the trip and how it should be split amongst you all. If the trip will take 20% or less of your free time, return a text message of this format:

```
'If each of us gets __ snack(s), there will be __ left. I will be _____, who else
is doing the same?'
```

If the trip takes up too much of your free time, return "Going to this destination would take too much time." **Hint: You must call weekendTrip() in this function.**

```
>>> textFriends(25.0, 65.0, 2.5, 17, 3)
'If each of us gets 5 snack(s), there will be 2 left. I will be driving, who else
is doing the same?'
```

```
>>> textFriends(1.5, 2.5, 3.0, 13, 7)
'If each of us gets 1 snack(s), there will be 6 left. I will be walking, who else
is doing the same?'
```

Grading Rubric

Function	Points
skillLevel()	15
bookStore()	20
dinnerPlans()	20
weekendTrip()	20
textFriends()	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 "Resubmit" button at the lower right-hand corner of Gradescope. You do not need to submit your `HW02.py` on Canvas.