Important

1. Due Date: **Thursday, September 7th**
2. This homework is graded out of 100 points.
3. This is an Individual Assignment. You may collaborate with other students in this class. Collaboration means talking through problems, assisting with debugging, explaining a concept, etc. Students may only collaborate with fellow students currently taking CS 1301, the TA's and the lecturer. You should not exchange code or write code for others. For individual assignments, each student must turn in a unique program. Your submission must not be substantially similar to another student's submission. Collaboration at a reasonable level will not result in substantially similar code.
4. For Help:
   - TA Helpdesk (Schedule posted on class website.)
   - Email TA's or use Piazza Forums Notes:
   - How to Think Like a Computer Scientists [http://openbookproject.net/thinkcs/python/english3e/]
   - CS 1301 Python Debugging Guide [http://www.cc.gatech.edu/classes/AY2016/cs1301_spring/CS-1301-Debugging-Guide/index.html]
5. Don't forget to include the required collaboration statement (outlined on the syllabus). Failing to include the Collaboration Statement will result in no credit.
6. Do not wait until the last minute to do this assignment in case you run into problems.
7. **Read the entire specifications document before starting this assignment.**
8. **IF YOUR CODE CANNOT RUN BECAUSE OF AN ERROR, IT IS A 0.**

Introduction

The goal of this homework is for you to learn the difference between print and return as well as give you practice with conditionals. Part 1 consists of 4 simple functions and 1 function which uses those 4 functions. Part 2 consists of 2 functions that will test your knowledge of conditionals. **You have been given HW2.py** to fill out with instructions in the docstrings. However, below you will find **more detailed information** to complete your assignment. Read it thoroughly before you begin. You have until **Thursday, September 7th** to complete this assignment.

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 **do not** do the same thing. The print function as its name states, is a function just like round or any other pre-defined and user defined functions. The **only purpose** for 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 statement, whether you explicitly write one 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 **assign a value** to a function, thus allowing you to either save it for later, use it in a more complex expression, or print it for human consumption.

Important note to remember: print is for the user whereas return is for the program.

Part 1: Market Place

You've just moved into your dorm room and now you need to do some shopping. Using the table below, write the following functions to help you at the grocery store. NOTE: the *list price* value is the usual price of the item; the *current price* value is how much the store is selling the item for right now.

Item name	# in stock	List price	Current price
"avocado"	OUT OF STOCK	$1.50	$1
"toothpaste"	5	$2.75	$2.75
"popcorn"	10	$1	$1
"bottled water"	8	$5.50	$4
"phone charger"	1	$15	$12

Function name (1): **`is_in_stock`**
Parameters: **`item` (str), `quantity` (int)**
Return value: **`True or False` (bool) or `None` (NoneType)**
Description:
Write a function that determines whether or not the parameter ***`item`*** is in stock. If so, you also need to check that there are enough of that item in stock to fulfill your order (specified by the parameter ***`quantity`***). Return `True` if these conditions are met, `False` otherwise. Return None if the item is not in the table.

Test Cases (not an exhaustive list:

```
is_in_stock("avocado", 3) → False
is_in_stock("bottled water", 1) → True
is_in_stock("popcorn", 100) → False
is_in_stock("potato chips", 5) → None
```

Function name (2): **`can_afford`**
Parameters: **`item` (str), `quantity` (int), `wallet` (int)**
Return value: **`True or False` (bool) or `None` (NoneType)**
Description:
Write a function that determines whether or not you can afford the parameter ***`item`***, given the number of that item you would like to buy (specified by parameter ***`quantity`***), the amount of money you have (specified by parameter ***`wallet`***), and the item's *current price* value in the table. You do not need to consider whether there are enough of the item in stock for this function. Return `True` if you can afford this item, `False` otherwise. Return None if the item is not in the table.

Test Cases (not an exhaustive list:

```
can_afford("avocado", 500, 5) → False
can_afford("phone charger", 10, 500) → True
can_afford("mouthwash", 1, 20) → None
```

Function name (3): **`is_on_sale`**
Parameters: **`item` (`str`)**
Return value: **`True or False` (`bool`) `or None` (`NoneType`)**
Description:
Write a function that determines whether or not the parameter ***`item`*** is on sale, according to the *list price* and *current price* values for the item in the table. You do not need to consider whether the item is in stock for this function. Return True if it is on sale, False otherwise. Return None if the item is not in the table.

Test Cases (not an exhaustive list:

```
isOnSale("phone charger") → True
isOnSale("toothpaste") → False
isOnSale("chocolate bar") → None
```

Function name (4): **`is_cheaper`**
Parameters: **`item1` (`str`), `item2` (`str`)**
Return value: **`0 (int) or 1 (int) or -1 (int) or None (NoneType)`**
Description:
Write a function that determines whether ***item1*** is cheaper than ***item2***, based on their *current prices*. If ***item1*** is cheaper than ***item2***, return 1. If ***item1*** is more expensive than ***item2***, return -1. If they are equally-priced, return 0**.** If either parameter ***item1*** or ***item2*** is invalid (i.e. is not included in the table), return None.
Notes:

- You do not need to consider whether the items are in stock for this function.
- Item names will always be strings; you are not responsible to account for different spellings or capitalizations of the items listed above.
- Review the test cases below to confirm edge cases.

Test Cases (not an exhaustive list:

```
is_cheaper("popcorn", "bottled water") → 1
is_cheaper("avocado", "popcorn") → 0
is_cheaper("toothpaste", "floss") → None
```

Part 2: Concert Listing

You come across a concert listing with some of your favorite artists! There are prices for single tickets and group tickets (in packs of 10), and you want to decide how you can attend as many concerts as possible with your given budget. The following functions help you achieve that goal.

	Taylor Swift	Adele	Zac Brown Band
Single Tickets	$275	$152	$25
Group Tickets(10)	$3000	$1500	$200

Function name (1): **`is_single_cheaper`**
Parameters: **artistName(str)**
Return value: **boolean**
Description:
Write a function that takes in the name of one of the artists from the chart. Decide whether or not it would be cheaper to buy a single ticket, or to get the group ticket price and divide it amongst 10 people. If it's cheaper for single tickets, then return True, and if not, then return False. If the artist is not valid, return None.

Test Cases (not an exhaustive list):
```
is_single_cheaper("Taylor Swift") → True
is_single_cheaper("Chainsmokers") → None
```

Function name (2): **`best_price`**
Parameters: **artistName(str)**
Return value: **representing the best price the user would pay to attend the artist's concert (int)**
Description: Write a function that takes in the name of one of the artists from the chart. Using the is_single_cheaper function, determine whether a single ticket or group ticket would be the cheapest, and then return the price of the ticket as an integer. If the group option ends up being the cheapest, do not return the full price for the group, but the price once it is divided by 10 people. If the artist is not valid, return None.

Test Cases (not an exhaustive list):
```
best_price("Taylor Swift") → 275
best_price("Avicii") → None
```

Function name (3): **all_three**
Parameters: **None**
Return value: **representing how much it would cost to attend all three concerts (int)**
Description: Write a function that uses the `best_price` function to determine the best prices of each of the concerts, sums them all up, and returns the total cost.

Function name (4): **`cheapest_concert`**
Parameters: **None**
Return value: **representing the name of the artist with the cheapest concert (str)**
Description: Write a function that uses the best_price function to determine the best prices of each of the concerts, and then returns the name of the artist with the cheapest concert. **You may not use any built in Python functions.**

Function name (5): **`add_two`**
Parameters: **artist1 (str), artist2 (str)**
Return value: **representing the cost to go to both concerts (int)**
Description: Write a function that takes in two artists from the table above, and calculates how much it would cost to attend both concerts based on their best prices. Return the total cost.

Function name (6): **`can_afford_concerts`**
Parameters: **money(int)**
Return value: **None**
Description: Write a function that will be using some of the functions that you have written above. Based on the money passed in, determine if you can go to all three concerts, only two concerts, or only the cheapest concert. If you can go to all three, print "I can go to all three!", if you can only go to two of any combination, (Taylor Swift and Adele, Adele and Zac Brown Band, etc), then print "I can only go to two!", and if you can only go to one concert, print a statement in the format of "I can only go to one.". If there is not enough money for any of those options, then print out a statement that says "Dang it, I can't go to any concert.". Note: It's very important that you print out your answer EXACTLY as it's formatted in the instructions.

Test Cases (not an exhaustive list):
```
can_afford_concerts(5)
>> Dang it, I can't go to any concert.
can_afford_concerts(24)
>> I can only go to one.
```

Part 3: Miscellaneous

Function name: **`what_can_you_do`**
Parameters: **age(int)**
Return value: **None**
Description: Write a function that takes in the age of the user and prints out all the activities they are able to do based on the table below. If they can't do any of those activities, print out "Sorry, you're not old enough for any of these".

Age	**Activity**
18	vote
21	party
65	retire

Test Cases (not an exhaustive list):

```
what_can_you_do(5)
>> Sorry, you're not old enough for any of these.
what_can_you_do (68)
>> You can vote, party, and retire.
```

Function name: **`pass_or_fail`**
Parameters: current_grade (int), final_weight (float), final_score (int)
Return value: final letter grade A, B, C, D, or F (str)

Description: Write a function that will take your current grade in a class, the weight of the final exam as a decimal between 0 and 1, and the score you got on the final exam to determine what letter grade you'll receive using the following formula:

$$final\_grade = current\_grade * (1 - final\_weight) + final\_score * final\_weight$$

Use the following ranges for letter grades:

- A: 90-100
- B: 80-89.9999
- C: 70-79.9999
- D: 60-69.9999
- F: 0-59.9999

Test Cases (not an exhaustive list):

```
pass_or_fail(90, .15, 75)  <- B
pass_or_fail(60, .3, 100)  <- C
```

Grading Rubric

- `is_in_stock`: 10 points	
Passes some test cases:	3/10 Points
Passes all test cases:	10/10 Points
- `can_afford`: 10 points	
Passes some test cases:	3/10 Points
Passes all test cases:	10/10 Points
- `is_on_sale`: 10 points	
Passes some test cases:	3/10 Points
Passes all test cases:	10/10 Points
- `is_cheaper`: 10 points	
Passes some test cases:	3/10 Points
Passes all test cases:	10/10 Points
- `is_single_cheaper`: 10 points	
Passes some test cases:	3/10 Points
Passes all test cases:	10/10 Points
- `best_price`: 10 points	
Passes some test cases:	3/10 Points
Passes all test cases:	10/10 Points
- **all_three:** 5 points	
Correct Value is returned:	5 Points
- `cheapest_concert`: 10 points	
Correct Value is returned:	10 Points
- `add_two`: 5 points	
Passes some test cases:	2/5 Points
Passes all test cases:	5/5 Points
- `can_afford_concerts`: 10 points	
Passes some test cases:	3/10 Points
Passes all test cases:	10/10 Points
- `what_can_you_do`: 5 points	
Passes some test cases:	2/5 Points
Passes all test cases:	5/5 Points
- **pass_or_fail**: 5 points	
Passes some test cases:	2/5 Points
Passes all test cases:	5/5 Points

Provided

The following file(s) have been provided to you.

1. `HW2.py`
   This is the file you will edit and implement. All instructions for what the methods should do are in the docstrings.

Deliverables

You must submit all of the following file(s). Please make sure the filename matches the filename(s) below. Be sure you receive the confirmation email from T-Square, and then download your uploaded files to a new folder and run them.

1. `HW2.py`