Assignment 7

Function Smorgasbord

The purpose of this assignment is twofold:

1. Practice the basics of writing functions in python.
2. Use an AI helper tool (link here) and provide feedback (link here).

This assignment is a little different to past assignments in that you'll be making several unrelated functions, rather than a single unified program.

Additionally, to get full credit, you need to use the AI Assistant Chatbot at least once while you complete this assignment, but we recommend using it for the whole assignment (even if you normally wouldn't) just so you can provide good feedback on how it did or didn't help you. Use it however you want to complete the assignment. Click here to access the AI Assistant Chatbot.

Additionally, we are doing a research project on how the AI Assistant Chatbot impacted you as you did your assignment. While it is not required and won't impact your grade either way, we STRONGLY encourage you to take the 2-3 minute survey after you complete the assignment. Click here to access the survey. It will help the IS department know how to best use AI in this and future courses.

Logical Flow:

You must create 9 different functions (most of them unrelated). I provide the function names below using snake_case, but feel free to use camelCase or PascalCase. So if the instructions say to make an `example_function` function, you could name it either `example_function`, `exampleFunction`, or `ExampleFunction`.

You'll be writing out 9 separate functions, and call each of them. Pay attention to the number of parameters and what the functions should return. Remember that when you write out the function you use def, and then the name of the function. When you "call" the function, you just use the name of the function followed by parentheses (and any arguments).

Remember to use the AI Assistant, linked above!

Function 1: welcome_message

- *purpose*:
  - print out a simple message with a person's name.
- *parameters*:
  - `name`:
    - a string containing a name.
- *logic and return values*
  - the function should take the name parameter and insert it into this message and print it:
    - `Hello <name>, welcome to IS 303!`
  - For example, if I provided "Jacob" as the argument, it should print:
    - `Hello Jacob, welcome to IS 303!`
  - The function should not return anything
- *calling the function*
  - call `welcome_message` with `"Diego"` as an argument, and then call it again with `"Mai"` as an argument.

Function 2: sum_two_numbers

- *purpose*:
  - adds together 2 numbers and returns the result.
- *parameters*:
  - `a`:
    - a number (could be integer or float)
  - `b`
    - another number (could be integer or float)
- *logic and return values*
  - add `a` and `b` together and return the sum
- *calling the function*
  - Call `sum_two_numbers` with 5 and 7 as arguments, and then make another call with 1000.5 and -30 as arguments. Print out the result of each call.

Function 3: is_even

- *purpose*:
  - tells you whether an integer is even or odd.
- *parameters*:
  - `num`:
    - an integer.
- *logic and return values*:
  - Must return a boolean `True` if the number is even or `False` if the number is odd.
- *calling the function*:
  - Call the function with the number 7 and print out the result. Then call it again with the number 120 and print the result.

Function 4: get_number_parity

- *purpose*:
  - returns a message telling you whether an integer is even or odd. Uses the `is_even` function you already wrote
- *parameters*:
  - `num`:

        - an integer.
- *logic and return values*:
    - You need to call `is_even` from within `get_number_parity`. It should return `<number> is even.` if the number is even or `<number> is odd.` if the number is odd.
- *calling the function*:
    - Call the function with 5 as the argument and print out the result. Then call it again with the number 10 and print the result.

Function 5: fahrenheit_to_celsius

- *purpose*:
    - converts a temperature in Fahrenheit to Celsius.
- *parameters*:
    - `fahrenheit`:
        - an integer or float expressing a temperature in Fahrenheit.
- *logic and return values*:
    - The function should take the `fahrenheit` parameter and convert it to Celsius using this formula:
        - `°C = (°F - 32) * (5/9)`
    - Then return the Celsius value.
- *calling the function*:
    - Call the function with 32 as the argument and print out the returned value. Then call it with 75 as the argument and print out the returned value.

Function 6: min_max_mean

- *purpose*:
    - shows the smallest number, biggest number, and mean when given a list of numbers.
- *parameters*:
    - `numbers_list`:
        - a list that contains only integers or floats.
- *logic and return values*:
    - The function should return a list that has the lowest number in the first position, the highest number in the second position, and the mean of all numbers in the third position.
        - `[<lowest number>, <highest number>, <mean>]`
- *calling the function*:
    - Call the function using the list below as an argument, and print out the result:
        - `numbers_list_1 = [20, 45, 23, 2, 87, 3]`

Function 7: dog_message

- *purpose*:
    - returns a message about a dog and its age.
- *parameters*:
    - `name`:
        - a string representing a dog's name.
    - `age`:
        - an integer representing a dog's age. It should have a default value of 0.
- *logic and return values*:
    - The function should return a string that includes the parameters for the dog's `name` and `age`. Make sure the `age` displays as 0 if no argument for `age` is provided.
        - `I am a dog named <name> and I'm <age> years old!`
- *calling the function*:
    - Call the function twice, once using "Spot" for the `name` and 7 for the `age`, and another time using just "Peppy" for the `name`. Print out the returned values.

Function 8: classify_age

- *purpose*:
    - returns a string that classifies an age into one of 3 groups.
- *parameters*:
    - `age`:
        - an integer representing a person's age.
    - `senior_age`:
        - an integer representing the age at which someone is considered a "Senior". It should have a default value of 65.
- *logic and return values*:
    - The function should reference the `age` provided and return a string based on how old the person is.
        - If the `age` is below 18, it should return the string `Minor`.
        - If `age` is below the `senior_age`, it should return the string `Adult`.
        - If the `age` is equal to or above the `senior_age`, it should return the string `Senior`.
- *calling the function*:
    - Call the function twice, once using 60 for the `age` and 55 for the `senior_age`. Call the function another time using just 62 for the `age`. Print out the returned values.

Function 9: calculate_total

- *purpose*:
    - Calculates the total amount paid for a quantity of a product based on the quantity being bought, the discount percent, and whether a bonus discount is applied.
- *parameters*:
    - `price`:
        - a number representing the price of a single product.

    - `quantity`:
        - an integer representing how many units of the product are being purchased.
    - `discount_percent`:
        - a percent expressed as a decimal (float). Represents the base discount that will be applied. For example, a 10% discount would be `.1`
    - `threshold_total`:
        - The threshold for the total price (`price * quantity`) that needs to be passed for the `bonus_discount` to be applied. The default value should be 100.
    - `bonus_discount`:
        - a percent expressed as a decimal (float). Represents the discount that will be added to the `discount_percent` if the `threshold_total` is met. The default value should be 0.02.
- *logic and return values*:
    - The function should calculate the total price (`price * quantity`), and depending on the total price, the discount applied will be different.
    - If the total price is equal to or below the `threshold_total`, then the discount applied would just be the `discount_percent`.
    - If the total price is above the `threshold_total`, then the applied discount should be the `discount_percent + bonus_discount`.
    - Return the total price, rounded to the second decimal, after the discount is applied.
    - You can reference the table below for example inputs and outputs. In the first row, the bonus_discount is not applied since it doesn't reach the threshold_total, but in the second row, the bonus_discount is applied since the price*quantity is over 100.
    -

Price	Quantity	Discount_percent	Threshold_total	Bonus_discount	Returned value
5	15	0.1	100	0.02	67.5
5	25	0.1	100	0.02	110.0

- *calling the function*:
    - Write a loop that runs 2 times. In each iteration, ask the user to input:
        - a `price`:
            - `Enter the price for the product purchased:`
        - a `quantity`
            - `Enter the quantity of the product purchased:`
        - a `discount_percent`
            - `Enter the discount percent (formatted as a decimal):`
    - Then call the function using the `price`, `quantity`, and `discount_percent` gathered and print out a message using the returned value that shows the total:
        - `The total price after discounts is: $<total price returned from calculate_total>`

Upload your .py file to Learning Suite to get credit. Remember to consider leaving feedback on the AI Assistant by clicking here.

Example Output

```
 Hello Diego, welcome to IS 303!
Hello Mai, welcome to IS 303!
12
970.5
False
True
5 is odd.
10 is even.
0.0
23.88888888888889
[2, 87, 30.0]
I am a dog named Spot and I'm 7 years old!
I am a dog named Peppy and I'm 0 years old!
Senior
Adult
Enter the price for the product purchased: 5
Enter the quantity of the product purchased: 15
Enter the discount percent (formatted as a decimal): .1
The total price after discounts is: $67.5
Enter the price for the product purchased: 5
Enter the quantity of the product purchased: 25
Enter the discount percent (formatted as a decimal): .1
The total price after discounts is: $110.0
```

Grading Rubric

Test	Description	Points
	Arguments / Return Values	

Test	Arguments	Description	Return Values	Points
1. welcome message	`Mai`		`None`	5
2. sum two numbers	`1, 1`		`2`	5
	`-3, 7`		`4`	
	`2000, 42`		`2042`	
3. is even	`1`		`False`	5
	`-3`		`False`	
	`64`		`True`	
4. get number parity	`1`		`1 is odd.`	10
	`-3`		`-3 is odd.`	
	`64`		`64 is even.`	
5. fahrenheit to celsius	`32`		`0`	10
	`122`		`50`	
	`-13`		`-25`	
6. min max mean	`[1, 2, 3]`		`[1, 3, 2.0]`	10
	`[354, 87, -7, 92, 34]`		`[-7, 354, 112.0]`	
7. dog message	`Jet`		`I am a dog named Jet and I'm 0 years old!`	10
	`Poof, 92`		`I am a dog named Poof and I'm 92 years old!`	
8. classify age	`18`		`Adult`	15
	`1, 78`		`Minor`	
	`70`		`Senior`	
	`70, 75`		`Adult`	

Test	Arguments	Description	Return Values	Points
9. calculate total	`10, 9, 0.3`		`63`	20
	`10, 11, 0.3`		`74.8`	
	`10, 11, 0.3, 120`		`77`	
	`10, 11, 0.3, 90, 0.5`		`22`	
10. Runs properly	Your code should all run properly, printing out the required statements and asking for the proper inputs.			9
11. Sufficient Comments	Your code must include at least `10` comments. You can use any form of commenting: • `#` • `''' '''` • `""" """`			1
Total Points				100