Introduction

A new pizza place, Western Pizza, is opening at the end of October and they have hired you to create a program that allows customers to place an order. The business needs the program to be coded in Python and they have various requirements including the files and functions for the program to run, and the format of the receipt once an order has been placed. Read through all the instructions below to better understand the requirements and specifications of Western Pizza's vision for their ordering system. The actual implementation of each function is up to you based on the skills you have learned so far in this course.

In this assignment, you will get practice with:

- Creating functions
- Calling functions and using return values
- Using a function from a separate Python file (multi-file program)
- User input in a loop
- Conditionals
- Lists and tuples
- String formatting for clean output

Files

For this assignment, you must create two Python files: *pizzaReceipt.py* and *order.py*

You are provided with one file, Asst2Tests.py, which consists of several test cases to help you determine if your program is running correctly. These same tests cases will be run on Gradescope to determine a portion of your grade. Note that **additional**, **hidden** (not visible to you) tests will also be run on Gradescope to determine a portion of your grade. You are encouraged to create and run your own tests as well to ensure that your code works in many different cases.

When submitting your assignment on Gradescope, please submit pizzaReceipt.py and order.py only. Do not upload the Asst2Tests.py file.

pizzaReceipt.py

In the pizzaReceipt.py file, create a function called generateReceipt(pizzaOrder) which takes in the single parameter pizzaOrder which is described below. This function must print out the receipt of the entire order that is sent in via the pizzaOrder parameter. The receipt must include each pizza, its size, a hyphenated list of toppings on the pizza, the cost of each item in the order, the tax, and the total cost. It must be formatted in a very specific way, as explained and shown in the diagrams below.

pizzaOrder Parameter

The pizzaOrder parameter is a list that contains some number of pizzas. There could be zero pizzas meaning it is an empty list, or there could be one or more pizzas in the list. Each pizza is structured as a tuple (NOT a list) which first contains a string for the size as an abbreviation (S, M, L, or XL) followed by a list of toppings for that pizza. The pizza tuple will always have these 2 elements: one for size and one for toppings. The size will always be one of the 4 available sizes: S, M, L, or XL. The toppings list can contain any number of toppings, even zero meaning it is an empty list.

For example, pizzas would be structured like this:

```
pizza1 = ("M", ["PEPPERONI", "OLIVE"])
pizza2 = ("L", ["MUSHROOM", "BROCCOLI", "TOMATO"])
```

The function call to generateReceipt() would be like this:

```
generateReceipt([pizza1, pizza2])
```

In this example, pizzaOrder would be a list containing pizza1 and pizza2.

Pricing

Western Pizza has a simple menu for now. They only sell pizza and they have 4 sizes available: Small, Medium, Large, and Extra Large. All pizzas, regardless of the size, come with 3 free toppings. If someone wants a pizza with more than 3 toppings, there will be an additional cost for each topping after the first 3. The cost of the additional toppings will depend on the size of

the pizza. The following table shows the cost of each size of pizza and the cost of the additional toppings (after the free 3 toppings) for each size of pizza.

Pizza Size	Base Cost of Pizza	Cost of Additional Toppings (after first 3 which are free)
Small (S)	$7.99	$0.50
Medium (M)	~~$8.99~~ $9.99	$0.75
Large (L)	$11.99	$1.00
Extra Large (XL)	$13.99	$1.25

There is a 13% tax that is applied to any order which must be calculated after all the items from the order (including the possible additional toppings cost) have been added together.

Receipt Format

If a user decides they no longer want to place an order, i.e. the pizzaOrder parameter is empty, then simply print the message "You did not order anything" and exit the function immediately.

For any orders that are not empty, you must follow this specific format for displaying the receipt. The first line must say "Your order:", then each pizza in the order starting with the pizza's number in the order (i.e. Pizza 1) and its size as an abbreviation (S, M, L, or XL) and then the base cost of that pizza spaced to the right several tabs. Below the header line for each pizza, you must display a list of the toppings one per line with a hyphen before the topping name.

Any time there are more than 3 toppings selected for a pizza, you must include a line below that pizza's list of toppings that says "Extra Topping (*size*)" (where *size* is S, M, L, or XL) and the corresponding cost for that additional topping spaced to the right several tabs.

Once all the pizzas and any additional toppings are displayed in the correct order, you must display a line for the tax and then the final line for the total cost.

All costs must have 2 decimal places and be aligned cleanly along the rightmost digit. Do not include dollar signs for the costs.

```
Your order:
Pizza 1: M                    9.99
- CHICKEN
- BACON
- BROCCOLI
Tax:                          1.30
Total:                       11.29
```

Receipt Example 1

```
Your order:
Pizza 1: L                   11.99
- HAM
- PEPPERONI
- ONION
- GREEN PEPPER
Extra Topping (L)             1.00
Pizza 2: L                   11.99
- TOMATO
- ONION
- HOT PEPPER
Pizza 3: S                    7.99
- PEPPERONI
Tax:                          4.29
Total:                       37.26
```

Receipt Example 2

order.py

The order.py file is going to be the main heart of the program, mainly consisting of a user-input-driven system for taking orders. There must be a constant called TOPPINGS that is a tuple that contains the 15 toppings available for the pizzas (see list of toppings below). There also must be a variable that is a list (empty list to start) that will hold the whole order – this will be the list sent in to generateReceipt() as the pizzaOrder parameter so it must be formatted correctly.

Upon running order.py, the user must first me asked "Do you want to order a pizza?" If they type "Q" or "NO" in any case (i.e. NO or No or no), the program should end immediately and call generateReceipt() with the empty list.

If they typed anything else, it must then ask the user to choose a size out of S, M, L, or XL. If the user types any of those valid options, in any case, it stores that size in a variable for later use. If they type anything else, it must continue to ask them to choose a size until they type one of those 4 valid size options (note that the size must be an abbreviation, i.e. "medium" is not a valid choice).

After the size has been properly selected, the user must be prompted to enter in their toppings one at a time until they are finished, and then they must enter "X", in any case, to indicate that they are finished adding toppings. At any time during this process, the user should be allowed to type "LIST", in any case, to view a list of the 15 toppings. If they type anything other than "X" to proceed, "LIST" to see the list of toppings, or the name of any of the toppings (which can be any case but must match the exact spelling and spacing as specified below, i.e. BEEF is not a valid option but GROUND BEEF is valid), then the program must print "Invalid topping". The user must be prompted to continue in this toppings process until they type "X".

After all the toppings have been selected, the pizza must be completed by creating a tuple consisting of the size and then the list of selected toppings (just as explained in the "pizzaOrder Parameter" section). This tuple must be added to the end of the whole order list. The user must be asked if they want to continue ordering a pizza. This allows users to order as many pizzas as they wish in a single order. If they type "Q" or "NO" in any case (i.e. NO or No or no), the program should end immediately and call generateReceipt() with the whole order list sent in as the parameter. If they type anything else, then it must go back through all the steps for asking the size and then the toppings and then adding the pizza to the order. This process must continue until the user types "Q" or "NO", in any case, when asked about ordering a new pizza.

Toppings

ONION	SPINACH	HAM
TOMATO	BROCCOLI	BACON
GREEN PEPPER	PINEAPPLE	GROUND BEEF
MUSHROOM	HOT PEPPER	CHICKEN
OLIVE	PEPPERONI	SAUSAGE

Ordering System Examples

```
Do you want to order a pizza?  Yes

Choose a size:  S, M, L, or XL:  L
Type in one of our toppings to add it to your pizza. To see the list of toppings, enter
"LIST". When you are done adding toppings, enter "X"

X

Do you want to continue ordering?  No
```

Ordering System Example 1 – an order of one Large pizza with no toppings

```
Do you want to order a pizza?  Yes

Choose a size:  S, M, L, or XL:  M
Type in one of our toppings to add it to your pizza. To see the list of toppings, enter
"LIST". When you are done adding toppings, enter "X"

LIST
('ONION', 'TOMATO', 'GREEN PEPPER', 'MUSHROOM', 'OLIVE', 'SPINACH', 'BROCCOLI',
'PINEAPPLE', 'HOT PEPPER', 'PEPPERONI', 'HAM', 'BACON', 'GROUND BEEF', 'CHICKEN',
'SAUSAGE')
Type in one of our toppings to add it to your pizza. To see the list of toppings, enter
"LIST". When you are done adding toppings, enter "X"

CHICKEN
Added CHICKEN to your pizza
Type in one of our toppings to add it to your pizza. To see the list of toppings, enter
"LIST". When you are done adding toppings, enter "X"

BACON
Added BACON to your pizza
Type in one of our toppings to add it to your pizza. To see the list of toppings, enter
"LIST". When you are done adding toppings, enter "X"

BROCCOLI
Added BROCCOLI to your pizza
Type in one of our toppings to add it to your pizza. To see the list of toppings, enter
"LIST". When you are done adding toppings, enter "X"

X

Do you want to continue ordering?  No
```

Ordering System Example 2 – using the LIST option to see the toppings while ordering a pizza

```
Do you want to order a pizza?  Y

Choose a size:  S, M, L, or XL:  large

Choose a size:  S, M, L, or XL:  L
Type in one of our toppings to add it to your pizza. To see the list of toppings, enter
"LIST". When you are done adding toppings, enter "X"

beef
Invalid topping
Type in one of our toppings to add it to your pizza. To see the list of toppings, enter
"LIST". When you are done adding toppings, enter "X"

ground beef
Added GROUND BEEF to your pizza
Type in one of our toppings to add it to your pizza. To see the list of toppings, enter
"LIST". When you are done adding toppings, enter "X"

x

Do you want to continue ordering?  q
```

Ordering System Example 3 – entering an invalid size and an invalid topping while order a pizza

Tips and Guidelines

- Constants must be named with all uppercase letters, i.e. TOPPINGS
- Variables should be named in lowercase for single words and camel case for multiple words, i.e. extraToppingCost
- All user input in this program should be case-insensitive, meaning that they can type in lowercase or uppercase or a mixture, and it should work the same regardless
- Add comments throughout your code to explain what each section of the code is doing and/or how it works

Rules

- Read and follow the instructions carefully.
- Only submit the Python files described in the Files section of this document.
- Submit the assignment on time. Late submissions will receive a late penalty of 10% per day (except where late coupons are used).
- Forgetting to submit a finished assignment is **not** a valid excuse for submitting late.
- Submissions must be done on Gradescope. They will not be accepted by email.
- You may re-submit your code as many times as you would like. Gradescope uses your last submission as the one for grading by default. There are no penalties for re-submitting. However, re-submissions that come in after the due date **will** be considered late and subject to penalties (or to the use of late coupons).
- Assignments will be run through a similarity checking software to check for code that looks very similar to that of other students. Sharing or copying code in any way is considered plagiarism and may result in a mark of zero (0) on the assignment and/or reported to the Dean's Office. Plagiarism is a serious offence. Work is to be done **individually**.

Submission

Due: Wednesday, October 26, 2022 at 11:55pm

You must submit the two files (pizzaReceipt.py and order.py) to the Assignment 2 submission page on Gradescope. There are several tests that will automatically run when you upload your

files. Some of the tests are visible to you so it will give you an idea of how well your program is working. However, we will also run some hidden tests so you won't see those tests nor the grade you get from those tests. It is recommended that you create your own test cases to check that the code is working properly for a multitude of different scenarios.

Assignments will not be accepted by email or any other form. They **must** submitted on Gradescope.

Marking Guidelines

The assignment will be marked as a combination of your auto-graded tests (both visible and hidden tests) and manual grading of your code logic, comments, formatting, style, etc. Below is a breakdown of the marks for this assignment:

[50 marks] Auto-graded Tests

[20 marks] Code logic and completeness

[10 marks] Comments

[10 marks] Code formatting

[10 marks] Meaningful and properly formatted variables

Total: 100 marks

The weight of this assignment is 8% of the course mark.