Project 6: Buy an Umbrella In this project, you will build a text version of a shopping site where you can browse and filter a catalog of umbrellas before buying one. ## Overview The program starts by loading an inventory of umbrella models from a text file. Afterwards, it greets the user with this message: ```python Welcome to Umbrellas Unlimited, your online market for water protection. We have over 100 umbrellas for sale. Happy shopping! Use these commands to navigate our site: (n)ext - view the next page of items (p)revious - view the previous page of items (a)dd filter - narrow your search by adding criteria (r)emove filter - broaden your search by deleting criteria (m)odify filter - change your search criteria (b)uy - purchase an umbrella from the list shown (q)uit - exit our site ``` Then it allows the user to browse and filter the catalog of available umbrellas: ```python Showing items 1-5 of 162 items 1) Samsonite polyester compact umbrella. Automatic Open and close, Clear. 2 stars. $8.99 2) GustBuster plastic compact umbrella. Automatic Open only, Yellow. 0.5 stars. $9.04 3) totes plastic golf umbrella. Automatic Open only, Green. 4.5 stars. $9.84 4) GustBuster polyester standard umbrella. Automatic Open only, Red. 1.5 stars. $10.13 5) Rainlax canvas standard umbrella. Yellow. 3.5 stars. $10.16 (b)uy, (n)ext, (p)revious, (a)dd, (r)emove, (m)odify, (h)elp or (q)uit? ``` We are providing you with a text file that contains the [complete inventory][2] of all 100+ models carried by Umbrellas Unlimited. Your program should use a dictionary of strings to represent an umbrella and a list of such dictionaries to represent the full inventory. To manage the search criteria, your program should us a dictionary of lists that contain acceptable values for each attribute of an umbrella. Although you may build this entire program from scratch, we strongly suggest you start with the provided scaffolding program. Either way, we encourage you to follow the implementation plan listed below. ## User Experience Here's an example interaction session: ### Paging through items ```python (b)uy, (n)ext, (p)revious, (a)dd, (r)emove, (m)odify, (h)elp or (q)uit? **next** Showing items 6-10 of 162 items 6) Rainlax polyester golf umbrella. Red. 1 stars. $10.51 7) ShedRain plastic golf umbrella. Automatic Open only, Red. 2.5 stars. $10.51 8) Rainlax plastic standard umbrella. Automatic Open only, Yellow. 3.5 stars. $10.82 9) Rainlax plastic standard umbrella. Yellow. 0.5 stars. $11.08 10) Rainlax nylon standard umbrella. Clear. 2 stars. $11.48 (b)uy, (n)ext, (p)revious, (a)dd, (r)emove, (m)odify, (h)elp or (q)uit? **next** Showing items 11-15 of 162 items 11) Samsonite nylon golf umbrella. Yellow. 4 stars. $11.80 12) GustBuster plastic standard umbrella. Automatic Open only, Green. 4 stars. $12.29 13) Rainlax nylon standard umbrella. Red. 3 stars. $13.11 14) Rainlax canvas golf umbrella. Blue. 1 stars. $13.31 15) GustBuster polyester standard umbrella. Automatic Open and close, Green. 3 stars. $13.43 (b)uy, (n)ext, (p)revious, (a)dd, (r)emove, (m)odify, (h)elp or (q)uit? **next** Showing items 16-20 of 162 items 16) GustBuster nylon compact umbrella. Automatic Open only, Clear. 2.5 stars. $13.95 17) Rainlax nylon golf umbrella. Automatic Open and close, Clear. 3 stars. $14.25 18) ShedRain plastic compact umbrella. Automatic Open only, Blue. 1.5 stars. $14.25 19) Rainlax canvas standard umbrella. Automatic Open only, Yellow. 5 stars. $14.31 20) Rainlax plastic golf umbrella. Black. 2.5 stars. $15.28 (b)uy, (n)ext, (p)revious, (a)dd, (r)emove, (m)odify, (h)elp or (q)uit? **previous** Showing items 11-15 of 162 items 11) Samsonite nylon golf umbrella. Yellow. 4 stars. $11.80 12) GustBuster plastic standard umbrella. Automatic Open only, Green. 4 stars. $12.29 13) Rainlax nylon standard umbrella. Red. 3 stars. $13.11 14) Rainlax canvas golf umbrella. Blue. 1 stars. $13.31 15) GustBuster polyester standard umbrella. Automatic Open and close, Green. 3 stars. $13.43 ``` ### Adding filters ```python Showing items 1-5 of 162 items 1) Samsonite polyester compact umbrella. Automatic Open and close, Clear. 2 stars. $8.99 2) GustBuster plastic compact umbrella. Automatic Open only, Yellow. 0.5 stars. $9.04 3) totes plastic golf umbrella. Automatic Open only, Green. 4.5 stars. $9.84 4) GustBuster polyester standard umbrella. Automatic Open only, Red. 1.5 stars. $10.13 5) Rainlax canvas standard umbrella. Yellow. 3.5 stars. $10.16 (b)uy, (n)ext, (p)revious, (a)dd, (r)emove, (m)odify, (h)elp or (q)uit? **add** Attributes: 1) brand 2) size 3) material 4) auto 5) color 6) stars 7) price Which attribute would you like to filter on? --> **1** Values: 1) GustBuster 2) NewSight 3) Rainlax 4) Samsonite 5) ShedRain 6) totes Which value would you like to allow ?--> **4** Current filters: brand: Samsonite Showing items 1-5 of 35 items 1) Samsonite polyester compact umbrella. Automatic Open and close, Clear. 2 stars. $8.99 2) Samsonite nylon golf umbrella. Yellow. 4 stars. $11.80 3) Samsonite polyester golf umbrella. Clear. 1.5 stars. $15.52 4) Samsonite plastic compact umbrella. Automatic Open and close, Black. 3.5 stars. $16.46 5) Samsonite canvas golf umbrella. Automatic Open only, Clear. 2 stars. $18.25 (b)uy, (n)ext, (p)revious, (a)dd, (r)emove, (m)odify, (h)elp or (q)uit? **add** Attributes: 2) size 3) material 4) auto 5) color 6) stars 7) price Which attribute would you like to filter on? --> **3** Values: 1) canvas 2) nylon 3) plastic 4) polyester Which value would you like to allow ?--> **2** Current filters: brand: Samsonite material: nylon Showing items 1-5 of 6 items 1) Samsonite nylon golf umbrella. Yellow. 4 stars. $11.80 2) Samsonite nylon standard umbrella. Blue. 3 stars. $23.98 3) Samsonite nylon standard umbrella. Red. 2 stars. $24.10 4) Samsonite nylon golf umbrella. Clear. 2.5 stars. $31.40 5) Samsonite nylon standard umbrella. Green. 4.5 stars. $32.95 (b)uy, (n)ext, (p)revious, (a)dd, (r)emove, (m)odify, (h)elp or

(q)uit? ``` Notice that the list now contains only 6 items, all Samsonite brand and all nylon materials. ### Removing filters Continuing the previous example, we can remove the brand filter: ```python (b)uy, (n)ext, (p)revious, (a)dd, (r)emove, (m)odify, (h)elp or (q)uit? **remove** Filters: 1) brand: Samsonite 2) material: nylon Which criterion would you like to remove? --> **1** Current filters: material: nylon Showing items 1-5 of 37 items 1) Rainlax nylon standard umbrella. Clear. 2 stars. $11.48 2) Samsonite nylon golf umbrella. Yellow. 4 stars. $11.80 3) Rainlax nylon standard umbrella. Red. 3 stars. $13.11 4) GustBuster nylon compact umbrella. Automatic Open only, Clear. 2.5 stars. $13.95 5) Rainlax nylon golf umbrella. Automatic Open and close, Clear. 3 stars. $14.25 (b)uy, (n)ext, (p)revious, (a)dd, (r)emove, (m)odify, (h)elp or (q)uit? ``` ### Modifying filters Sometimes you want to allow multiple values for a given attribute, say "Nylon or plastic". A user can modify their filter to allow this: ```python Current filters: brand: Samsonite material: nylon Showing items 1-5 of 6 items 1) Samsonite nylon golf umbrella. Yellow. 4 stars. $11.80 2) Samsonite nylon standard umbrella. Blue. 3 stars. $23.98 3) Samsonite nylon standard umbrella. Red. 2 stars. $24.10 4) Samsonite nylon golf umbrella. Clear. 2.5 stars. $31.40 5) Samsonite nylon standard umbrella. Green. 4.5 stars. $32.95 (b)uy, (n)ext, (p)revious, (a)dd, (r)emove, (m)odify, (h)elp or (q)uit? **modify** Filters: 1) brand: Samsonite 2) material: nylon Which filter would you like to modify? --> **2** Values: 1) canvas * 2) nylon 3) plastic 4) polyester Which value would you like to check/uncheck ?--> **3** Current filters: brand: Samsonite material: nylon, plastic Showing items 1-5 of 17 items 1) Samsonite nylon golf umbrella. Yellow. 4 stars. $11.80 2) Samsonite plastic compact umbrella. Automatic Open and close, Black. 3.5 stars. $16.46 3) Samsonite plastic compact umbrella. Automatic Open only, Yellow. 1 stars. $23.28 4) Samsonite nylon standard umbrella. Blue. 3 stars. $23.98 5) Samsonite nylon standard umbrella. Red. 2 stars. $24.10 (b)uy, (n)ext, (p)revious, (a)dd, (r)emove, (m)odify, (h)elp or (q)uit? ``` Note that both nylon and plastic umbrellas are now present. ### Completing a purchase Finally, the user can select an item they want and purchase it: ```python Current filters: brand: Samsonite material: nylon, plastic Showing items 6-10 of 17 items 6) Samsonite plastic standard umbrella. Blue. 1.5 stars. $25.02 7) Samsonite nylon golf umbrella. Clear. 2.5 stars. $31.40 8) Samsonite plastic golf umbrella. Yellow. 4.5 stars. $32.85 9) Samsonite nylon standard umbrella. Green. 4.5 stars. $32.95 10) Samsonite plastic compact umbrella. Automatic Open and close, Clear. 5 stars. $34.15 (b)uy, (n)ext, (p)revious, (a)dd, (r)emove, (m)odify, (h)elp or (q)uit? **buy** What item would you like to purchase?--> **8** You have purchased a: Samsonite plastic golf umbrella. Yellow. 4.5 stars. $32.85. Enjoy! ``` ## Data Structures The key idea of this project is that you can use Python dictionaries and Python lists together in interesting combinations. We will be using simple dictionaries of strings, but also dictionaries of lists and lists of dictionaries! ### Representing an Umbrella We'll use a simple dictionary of strings to represent each umbrella. Here's an example umbrella, represented as a dictionary: ```python one_umbrella = { 'brand': 'Samsonite', 'size': 'compact', 'material': 'polyester', 'auto': 'Open and close', 'color': 'Clear', 'stars': '2', 'price': '8.99', } ``` The umbrella has various attributes like brand, size, material, etc. and each has the value shown. ### Representing the Umbrella Inventory One umbrella isn't very interesting. Our online marketplace sells over 100 models of umbrellas. Your program should keep track of them as a list of dictionaries: ```python umbrellas = [umbrella0, umbrella1, umbrella2, ...] ``` Each umbrella is a dictionary. Of course, you won't want to create this list of dictionaries by hand -- there's over 100 of them. Rather, you will use the computer to construct these dictionaries for you. We have provided you with a simple text file named `inventory.txt` that lists all the models that Umbrellas Unlimited sells. Here's the first few lines of that file. ```python
brand,size,material,auto,color,stars,price Samsonite,compact,polyester,Open and close,Clear,2,8.99 GustBuster,compact,plastic,Open only,Yellow,0.5,9.04 totes,golf,plastic,Open only,Green,4.5,9.84 GustBuster,standard,polyester,Open only,Red,1.5,10.13 ``` This file is in a popular format named "comma-separated values". Each item is described by a number of attributes, like its brand, size, price, etc. The first line (the "header line") of the file names what each of these attributes is. The rest of the file is one line per item, each line containing the values of those attributes for the particular item it represents. The order of the values is consistent with the order of the attributes in the header line. For example the first non-header line here lists an umbrella whose price is $8.99, color is Clear, and brand is Samsonite. In the inventory file, each item is represented by a string. In the program, we will be representing the the items as dictionaries. So one of the important functions you will be writing in this lab is `load_items(filename)`. In this example, the first non-header line would become the dictionary `one_umbrella` shown above. ### Representing Search Filter The structure for representing a search filter is more complex. Here's a filter that represents "A Samsonite umbrella made of nylon or plastic": ```python my_criteria = { 'brand': ['Samsonite'], 'material': ['nylon', 'plastic'], } ``` Notice that the keys are still the names of the attributes, but the values are now *lists* of acceptable descriptions of the umbrella. One curious thing about this representation is that if a key is absent, then implicitly any value is acceptable. So for

example, any color umbrella could satisfy `my_criteria`. In the example above, does `one_umbrella` satisfy `my_criteria` ? No, it does not. You can see this by checking each of the attributes in the criteria. First is `brand`. Does `one_umbrella` have an acceptable brand? Yes. `one_umbrella['brand']` is `'Samsonite'`, which is in the allowed list `my_criteria[`brand`]`. How about material? Here it fails. `one_umbrella['material']` is `'polyester'`, but `'polyester'` is not in the list `['nylon', 'plastic']` (which is `my_criteria['material']`. One of the key functions you will write in this project is `filter_items(criteria, items)`, which takes a list of items (e.g. umbrellas) and returns a smaller list of just the ones that satisfy the conditions specified in the dictionary `criteria`. ## Development Strategy ### Milestones We strongly encourage you to write your program in stages; at the end of each stage your program must run without error, and deliver an additional increment of functionality. Here's the suggested list of development milestones. 1. Program can print the welcome message and action prompt, but can't actually take any actions. It just states "X not implemented" for any action the user tries to take. (The starter code described below gets students to this milestone "for free".) 1. Program can load the inventory of umbrella models from the text file, and print out a nice representation of the top 10 of them. 1. Program supports the actions `(n)ext` and `(p)revious`, allowing users to browse through the full inventory of umbrellas. 1. Program supports the `(b)uy` action, allowing users to select one of the shown models and "purchase" it (simply prints a description of the selected item and exits). 1. Temporarily hard code the filter `my_criteria` into your program as above. Program applies that filter to the full inventory and displays just the subset that match it. 1. Program supports the `(a)dd` filter action, prompting user for an attribute and a value, creates a corresponding search criteria dictionary, and applies it to the inventory (instead of `my_criteria`, which you can now delete). 1. Program supports the `(r)emove` filter action, displaying the current attributes that are being filtered on and asking the user which one to remove. 1. Program supports the `(m)odify` filter action. First it prompts the user to choose one of the attributes currently being filtered on, then it prompts the user to toggle (switch from allowed to excluded or vice-versa) one of the values for that attribute. ### Starter Project This is a lengthy project, and writing it entirely from scratch in the time available is fairly challenging. We encourage students to use [this starter project][1] and extend it as instructed to make a finished project. Here is a tour of the starter project and a list of parts that students must implement. The starter project contains 7 files: | File | Description | | ---- | ----------- | | `inventory.txt` | A CSV file containing descriptions of 100+ models of umbrellas | | `main.py` | Contains the top-level logic of the program, including the action loop and handlers for each action | | `items.py` | Functions related to representing shopping items as python dictionaries | | `criteria.py` | Functions related to search criteria | | `dictionaries.py` | Utilities for manipulating dictionaries | | `pages.py` | Functions to help display a paginated list | | `TEALS_utils.py` | General utilities for many TEALS labs | Code from 5 of these files is incorporated into `main.py` via `import` statements at the top of that program. The file `inventory.txt` is loaded at that start of the main program. You can run the starter project as-is. It will run without error, but it won't do much. In order to make it useful, you will have to implement a number of functions. They are already defined in the various `.py` files, but have stub implementation that do little besides announce that they are not yet implemented. The starter code files also contain some fully implemented functions that you do not need to change. Here's a table listing all of the functions present in the starter code, together with an indication of whether you need to modify them or not. | Student must implement? | Function name | File | | ---------- | ---------- | ---- | | **Yes** | `handle_purchase()` | `main.py` | | **Yes** | `handle_add_criterion()` | `main.py` | | **Yes** | `handle_remove_criterion()` | `main.py` | | **Yes** | `handle_modify_criterion()` | `main.py` | | **Yes** | `load_items()` | `items.py` | | **Yes** | `item_to_string()` | `items.py` | | No | `print_page()` | `items.py` | | **Yes** | `filter_items()` | `criteria.py` | | **Yes** | `criteria_to_string()` | `criteria.py` | | **Yes** | `union_of_dictionaries()` | `dictionaries.py` | | No | `first_index()` | `pages.py` | | No | `last_index()` | `pages.py` | | No | `next_page_number()` | `pages.py` | | No | `previous_page_number()` | `pages.py` | | No | `safe_to_integer()` | `TEALS_utils.py` | | No | `get_valid_integer()` | `TEALS_utils.py` | Of course you are free to implement additional functions that you find useful. Students will have to decide what is the best order to implement these functions in; we hope they will be guided by the Milestones listed above. ### Bonus You may earn bonus points on the project by trying some of these ideas: * The current user interface and data representation are pretty poor for continuous quantities like price and number. Users typically want to add criteria like "price < $X" or "stars > Y". Build this feature into your program. * Introduce a "sort by" feature that can list the items by increasing / decreasing prices, alphabetically, etc. * Write a program to generate a random assortment of umbrella models and output them as an inventory file in CSV format. * Update your inventory generator to produce something other than umbrellas: smart phones, bicycles, cars, clothing, etc. Update your main program in a corresponding way to create a shopping site for a different product line. [1]: starter_code [2]: starter_code/inventory.txt