15-110

Quiz 5 Review Session

Privacy, Security, and Authentication

Privacy

Data privacy is the idea that we want to have control over our data; who has access to it and also what others are able to do with it.

As an example, what would data privacy be in the context of the students information that cmu has access to?

Authentication

Authentication is how we prove that we truly are who we are pretending to be on the web. Authentication are done from user standpoint and also from websites.

When is authentication present when students are trying to go into their sio?

Security

Data security is the idea that we want to keep our communications and data to be secured and reliable. No one other than the sender and the receiver should be able to modify the data being transmitted.

In the example of text messaging, how is data security enforced?

SECURITY ATTACKS AND ENCRYPTION

Security Attacks

With computers security attacks can happen when the vulnerability of the security of some data or service is compromised.

Security attacks on the internet are very common because most attacks can be automated through the computer, the attacks can be done from a very far distance and the techniques for the attacks can be distributed through the internet.

What are the 2 main examples of security attacks learned from class and how do they work?

Encryption

Encryption is how we are able to encode data so that only the sender and the receiver can read it.

What examples of encryption have we seen so far in class and how do they work?

Feature of Encryption

Asymmetric and Symmetric - A symmetric algorithm is that involves using a key to encrypt and decrypt, whereas an asymmetric algorithm is one where each member uses 2 keys.

Keyspace - The keyspace tells us how many possible keys an adversary needs to try in order to decrypt our message.

What are the differences between RSA and Ceasar Cipher? What is the Big-O of an algorithm with a keyspace of 10^10,000 keys? Is this algorithm good or can we make it better?

Read and writing to files, and helper functions

Read and write to files

When we open a file we need to specify whether we plan to **read from** or **write to** the file. This will change the **mode** we use to open the file.

```
filename = "sample.txt"
f = open(filename, "r") # read mode
lines = f.readlines() # reads the lines of a file as a list of strings
# or
text = f.read() # reads the whole file as a single string

f = open("sample2.txt", "w") # write mode
f.write(text) # writes a string to the file
```

Only one instance of a file can be kept open at a time, so you should always **close** a file once you're done with it.

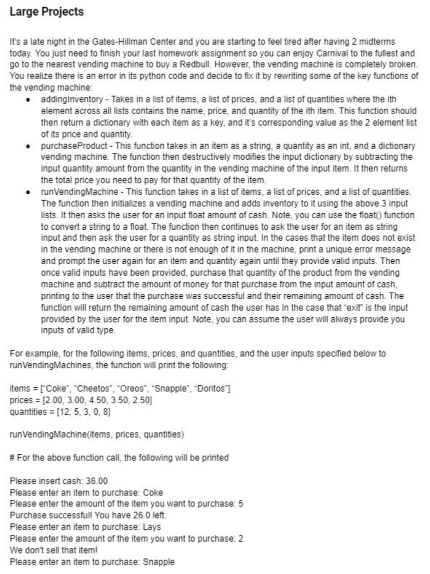
```
f.close()
```

Read and write to files

Write a function readEvenWords(filename) that given the path to a file it returns a string that holds every other word from the file.

Helper Functions

Helper functions are used to breakdown tasks from large projects into smaller components. By calling on the helper functions we are able to solve the problem into subsequent steps.



Large Projects

It's a late night in the Gates-Hillman Center and you are starting to feel tired after having 2 midterms today. You just need to finish your last homework assignment so you can enjoy Carnival to the fullest and go to the nearest vending machine to buy a Redbull. However, the vending machine is completely broken. You realize there is an error in its python code and decide to fix it by rewriting some of the key functions of the vending machine:

- addingInventory - Takes in a list of items, a list of prices, and a list of quantities where the ith element across all lists contains the name, price, and quantity of the ith item. This function should then return a dictionary with each item as a key, and it's corresponding value as the 2 element list of its price and quantity.
- purchaseProduct - This function takes in an item as a string, a quantity as an int, and a dictionary vending machine. The function then destructively modifies the input dictionary by subtracting the input quantity amount from the quantity in the vending machine of the input item. It then returns the total price you need to pay for that quantity of the item.
- runVendingMachine - This function takes in a list of items, a list of prices, and a list of quantities. The function then initializes a vending machine and adds inventory to it using the above 3 input lists. It then asks the user for an input float amount of cash. Note, you can use the float() function to convert a string to a float. The function then continues to ask the user for an item as string input and then ask the user for a quantity as string input. In the cases that the item does not exist in the vending machine or there is not enough of it in the machine, print a unique error message and prompt the user again for an item and quantity again until they provide valid inputs. Then once valid inputs have been provided, purchase that quantity of the product from the vending machine and subtract the amount of money for that purchase from the input amount of cash, printing to the user that the purchase was successful and their remaining amount of cash. The function will return the remaining amount of cash the user has in the case that "exit" is the input provided by the user for the item input. Note, you can assume the user will always provide you inputs of valid type.

For example, for the following items, prices, and quantities, and the user inputs specified below to runVendingMachines, the function will print the following:

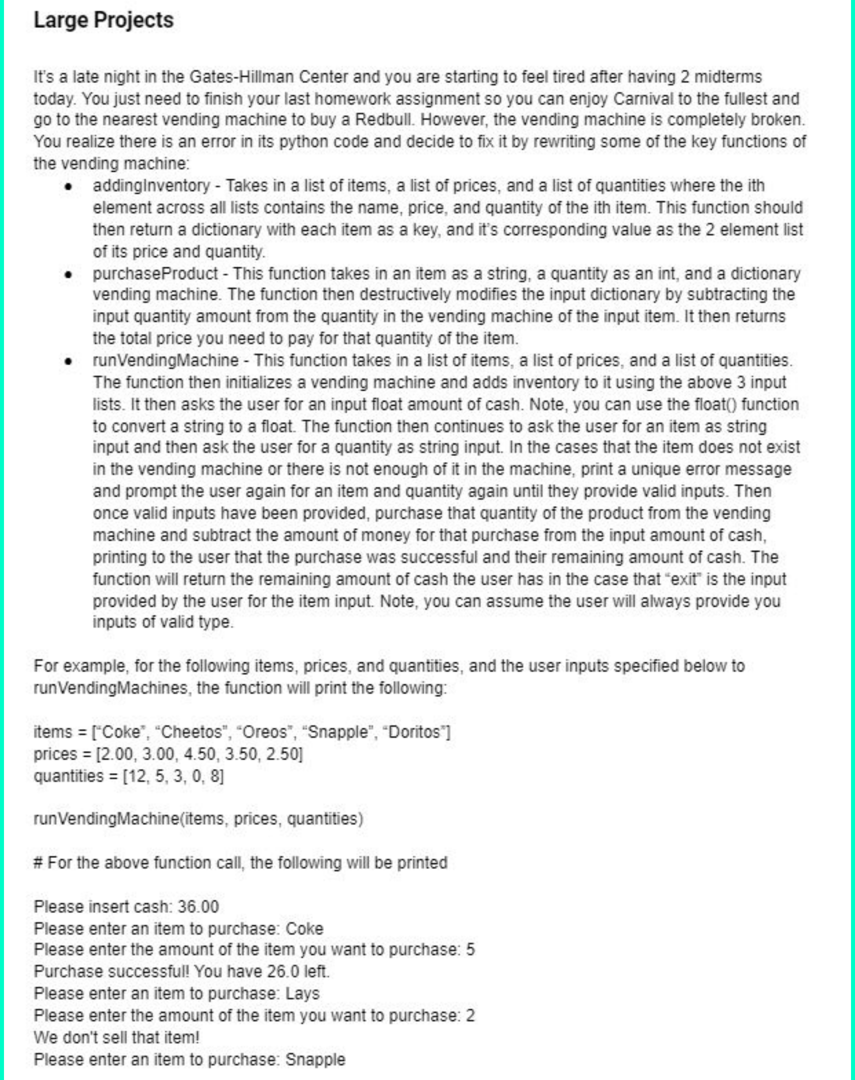
```
items = ["Coke", "Cheetos", "Oreos", "Snapple", "Doritos"]
prices = [2.00, 3.00, 4.50, 3.50, 2.50]
quantities = [12, 5, 3, 0, 8]

runVendingMachine(items, prices, quantities)

# For the above function call, the following will be printed

Please insert cash: 36.00
Please enter an item to purchase: Coke
Please enter the amount of the item you want to purchase: 5
Purchase successful! You have 26.0 left.
Please enter an item to purchase: Lays
Please enter the amount of the item you want to purchase: 2
We don't sell that item!
Please enter an item to purchase: Snapple
```

Helper Functions



Large Projects

It's a late night in the Gates-Hillman Center and you are starting to feel tired after having 2 midterms today. You just need to finish your last homework assignment so you can enjoy Carnival to the fullest and go to the nearest vending machine to buy a Redbull. However, the vending machine is completely broken. You realize there is an error in its python code and decide to fix it by rewriting some of the key functions of the vending machine:

- addingInventory - Takes in a list of items, a list of prices, and a list of quantities where the ith element across all lists contains the name, price, and quantity of the ith item. This function should then return a dictionary with each item as a key, and it's corresponding value as the 2 element list of its price and quantity.
- purchaseProduct - This function takes in an item as a string, a quantity as an int, and a dictionary vending machine. The function then destructively modifies the input dictionary by subtracting the input quantity amount from the quantity in the vending machine of the input item. It then returns the total price you need to pay for that quantity of the item.
- runVendingMachine - This function takes in a list of items, a list of prices, and a list of quantities. The function then initializes a vending machine and adds inventory to it using the above 3 input lists. It then asks the user for an input float amount of cash. Note, you can use the float() function to convert a string to a float. The function then continues to ask the user for an item as string input and then ask the user for a quantity as string input. In the cases that the item does not exist in the vending machine or there is not enough of it in the machine, print a unique error message and prompt the user again for an item and quantity again until they provide valid inputs. Then once valid inputs have been provided, purchase that quantity of the product from the vending machine and subtract the amount of money for that purchase from the input amount of cash, printing to the user that the purchase was successful and their remaining amount of cash. The function will return the remaining amount of cash the user has in the case that "exit" is the input provided by the user for the item input. Note, you can assume the user will always provide you inputs of valid type.

For example, for the following items, prices, and quantities, and the user inputs specified below to runVendingMachines, the function will print the following:

```
items = ["Coke", "Cheetos", "Oreos", "Snapple", "Doritos"]
prices = [2.00, 3.00, 4.50, 3.50, 2.50]
quantities = [12, 5, 3, 0, 8]

runVendingMachine(items, prices, quantities)

# For the above function call, the following will be printed

Please insert cash: 36.00
Please enter an item to purchase: Coke
Please enter the amount of the item you want to purchase: 5
Purchase successful! You have 26.0 left.
Please enter an item to purchase: Lays
Please enter the amount of the item you want to purchase: 2
We don't sell that item!
Please enter an item to purchase: Snapple
```

External modules and Documentations