TDQC

Bank of Nerds

Jack Spence

9 November 2018

1. Write-Up

1.1 Requirements

Requirements were to make a banking interface that supports the creation of an arbitrary amount of customers, who can have arbitrary amounts of checking, savings, and 401k accounts. The accounts can be access by deposits or withdraws at will, except for the 401k which has an age limit to withdraw set at 67.

1.2 Suggested Features

Suggested feature attempted in my submission is to persist a database of all the customers. Also attempted is the support of overdraft protection for all accounts, that allows users to go negative balances at a nominal fee of 35$.

1.3 Syntax

Usage: ./bank_of_nerds [-p]

Supports the -p option of persistence, but is ignored, due to implementing it as a base requirement.

2. Project Design Plans

2.1 Initial Design Plans

My initial plan was to use inheritance as much as I could when it came to the accounts. As well as make data persist from the beginning. I didn’t realize until later that feature was a suggested feature.

2.2 What didn’t work

This program did not have a lot of hiccups, but the problem that took the longest to resolve passwords in the customer objects after being pickled. Once the program was restarted the passwords were being loaded but I did not realize pythons builtin hash method seeds itself uniquely every time the program is ran. I changed my hashing algorithm for the passwords to md5 to correct it.

2.3 What went well

Implementing the classes went exceptionally well. Getting them to inherit things properly as well as making each class and subclass have all the relevant class and instance variables was quite easy.

2.4 Conclusion

This project had a bit more moving parts than the last project which added a layer of complexity, but that could just be with my implementation. Relying on nested dictionaries could get quite hectic but I believe that they were very appropriate for this project.

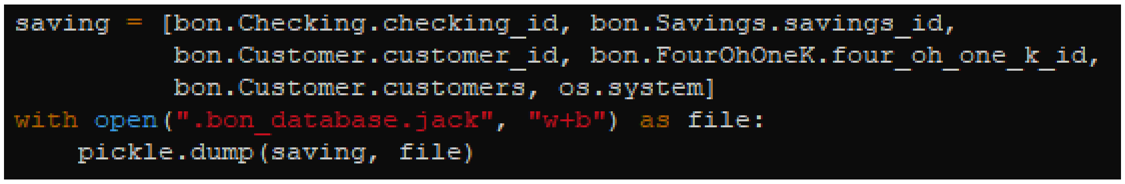
3. Test Procedures

3.1 Testing

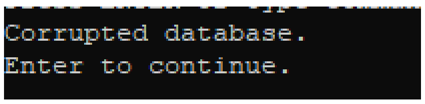
The biggest part of this testing was just making sure everything was being unpickled correctly. I was worried things wouldn’t work right because I had subclassed my own Unpickler to overwrite it’s class lookup.

3.2 Testing Procedure

Screen shots to show what a corrupted database looks like. In this scenario I raise the FileNotFoundError and handle the case as if it wasn’t found.

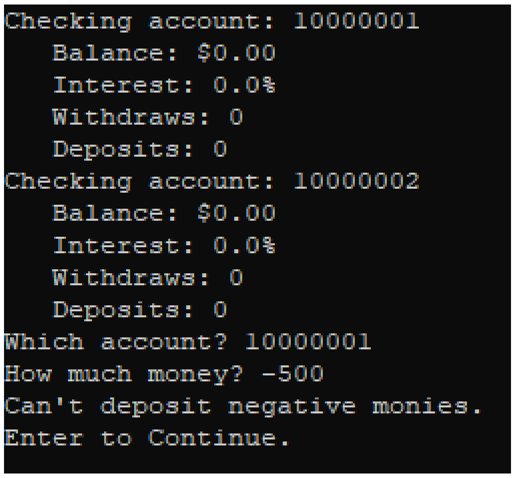


```
saving = [bon.Checking.checking_id, bon.Savings.savings_id,
          bon.Customer.customer_id, bon.FourOhOneK.four_oh_one_k_id,
          bon.Customer.customers, os.system]
with open(".bon_database.jack", "w+b") as file:
    pickle.dump(saving, file)
```

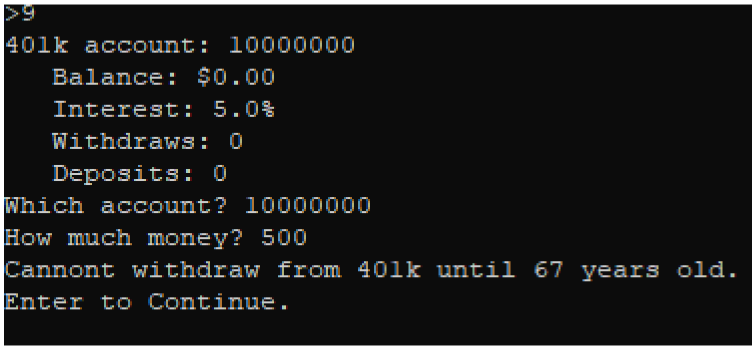


```
Corrupted database.
Enter to continue.
```

The rest of my testing consisted of creating users and managing their balances looking for expected output.

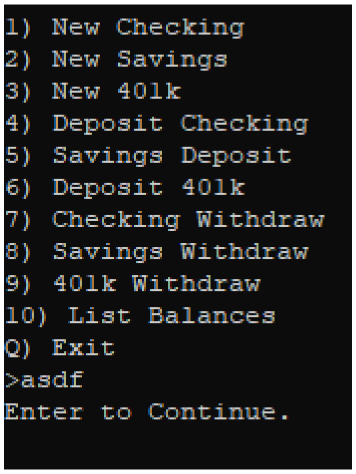


```
Checking account: 10000001
   Balance: $0.00
   Interest: 0.0%
   Withdraws: 0
   Deposits: 0
Checking account: 10000002
   Balance: $0.00
   Interest: 0.0%
   Withdraws: 0
   Deposits: 0
Which account? 10000001
How much money? -500
Can't deposit negative monies.
Enter to Continue.
```

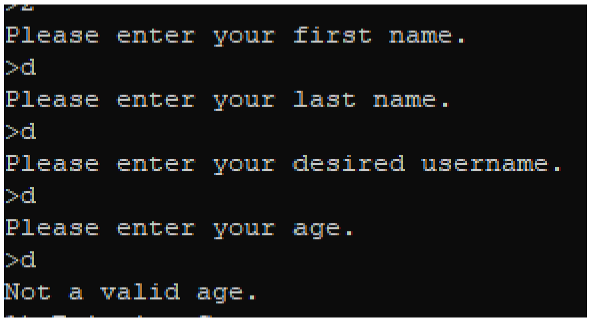


```
>9
401k account: 10000000
   Balance: $0.00
   Interest: 5.0%
   Withdraws: 0
   Deposits: 0
Which account? 10000000
How much money? 500
Cannont withdraw from 401k until 67 years old.
Enter to Continue.
```

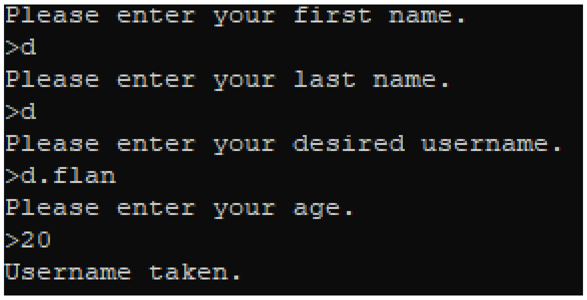
Invalid input options



```
1) New Checking
2) New Savings
3) New 401k
4) Deposit Checking
5) Savings Deposit
6) Deposit 401k
7) Checking Withdraw
8) Savings Withdraw
9) 401k Withdraw
10) List Balances
Q) Exit
>asdf
Enter to Continue.
```

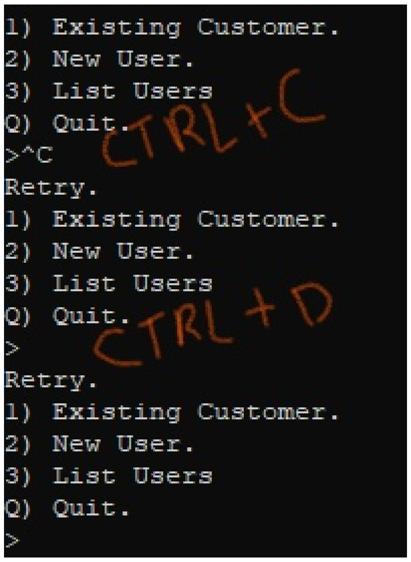


```
Please enter your first name.
>d
Please enter your last name.
>d
Please enter your desired username.
>d
Please enter your age.
>d
Not a valid age.
```



```
Please enter your first name.
>d
Please enter your last name.
>d
Please enter your desired username.
>d.flan
Please enter your age.
>20
Username taken.
```

I don't allow ctr+c or ctl+d on any input. I want to make sure the program will get to a spot in which data will be saved.



```
1) Existing Customer.
2) New User.
3) List Users
Q) Quit.
>^C
Retry.
1) Existing Customer.
2) New User.
3) List Users
Q) Quit.
>
Retry.
1) Existing Customer.
2) New User.
3) List Users
Q) Quit.
>
```