**Chapter 12, Problem 12.3**

**Here is my account class that will be used with the ATM machine program:**

'''

Account class is a class to emulate an interest earning

bank account.

'''

**class** **Account:**

'''constructor initializes instance variables

'''

**def** \_\_init\_\_**(**self**,** i **=** 0**,** b **=** 100**,** rate **=** 0**):**

self**.**\_\_id **=** i

self**.**\_\_balance **=** b

self**.**\_\_annualInterestRate **=** rate

'''

The following methods are the getter and setter methods for

the instance variables

'''

**def** setId**(**self**,** i**):**

self**.**\_\_id **=** i

**def** getId**(**self**):**

**return** self**.**\_\_id

**def** setBalance**(**self**,** b**):**

self**.**\_\_balance **=** b

**def** getBalance**(**self**):**

**return** self**.**\_\_balance

**def** setAnnualInterestRate**(**self**,** r**):**

self**.**\_\_annualInterestRate **=** r

**def** getAnnualInterestRate**(**self**):**

**return** self**.**\_\_annualInterestRate

'''

Returns the monthly interest rate (rate/12)

'''

**def** getMonthlyInterestRate**(**self**):**

**return** self**.**\_\_annualInterestRate **/** 12

'''

Returns the monthly interest earned based on the current balance

and the monthly interest rate

'''

**def** getMonthlyInterest**(**self**):**

**return** **((**self**.**getBalance**()\*** **(**1 **+** self**.**getMonthlyInterestRate**()))** **-** self**.**getBalance**())**

'''

The following two methods allow the balance to be adjusted by either

subtracting from or adding to it.

'''

**def** withdrawal**(**self**,** amount**):**

self**.**\_\_balance **-=** amount

**def** deposit**(**self**,** amount**):**

self**.**\_\_balance **+=** amount

'''

Override the toString method so that a string view of an Account object

can be displayed

'''

**def** \_\_str\_\_**(**self**):**

string **=** "Account ID:" **+** str**(**self**.**getId**())** **+** "\n"

string **+=** "Balance: " **+** str**(**self**.**\_\_balance**)** **+** "\n"

string **+=** "Annual Interest Rate: " **+** str**(**self**.**getAnnualInterestRate**())** **+** "\n"

string **+=** "Monthly Interest: " **+** str**(**self**.**getMonthlyInterest**())**

**return** string

**Here is the ATM machine program:**

**from** Account **import** Account

'''

Frank Mock, CS122

Chapter 12, Problem 12.3

'''

''' Global Variables'''

account **=** **[]**

id **=** **-**1

'''

Displays a menu of ATM choices and returns the users intger choice

'''

**def** menu**():**

**print(**"Main Menu\n1: Check Balance\n2: Withdraw\n3: Deposit\n4: Exit"**)**

c **=** input**(**"Enter a choice: "**)**

**return** c

'''

Takes an input integer and returns the users choice

'''

**def** userSelection**(**c**):**

c**.**strip**()**

b **=** account**[**id**]** # get the account object to work with

**if(**c **==** "1"**):**

#b = account[id]

bal **=** b**.**getBalance**()**

**print(**"The Balance is " **+** str**(**bal**)** **+** "\n"**)**

**elif(**c **==** "2"**):**

amount **=** eval**(**input**(**"Enter the amount to withdraw: "**))**

b**.**withdrawal**(**amount**)**

**print(**"\n"**)**

**elif(**c **==** "3"**):**

amount **=** eval**(**input**(**"Enter the amount to deposit: "**))**

b**.**deposit**(**amount**)**

**print(**"\n"**)**

**else:**

**print(**"Exit"**)**

'''

Prompts the user for an id and returns a valid id

or prints message informing the users to enter a

valid id

'''

**def** getUserId**():**

a **=** **-**1 # an invalid ID

**while** isValidId**(**a**)** **==** **False:**

a **=** eval**(**input**(**"Enter an Account id: "**))**

**if(**isValidId**(**a**)):**

**return** a

**else:**

**print(**"Id enter is not valid"**)**

'''

Predicate function that returns true if id is valid

and false otherwise

'''

**def** isValidId**(**id**):**

**if(**id **>=** 0 **and** id **<** 10**):**

**return** **True**

**else:**

**return** **False**

'''

The start point of the ATM program

'''

**def** main**():**

c **=** ""

'''Create 10 Account objects'''

**for** i **in** range**(**10**):**

a **=** Account**(**i**,**100.0**,**0**)**

a**.**setAnnualInterestRate**(**.05**)**

account**.**append**(**a**)**

'''

for n in account:

print(n)

'''

''' The main program loop '''

**while** **True:**

id **=** getUserId**()**

**while(**c **!=** "4"**):**

c **=** menu**()**

userSelection**(**c**)**

**if** \_\_name\_\_ **==** "\_\_main\_\_"**:** main**()**

**Here is the program run and resulting output:**

Enter an Account id: 22

Id enter is not valid

Enter an Account id: 1

Main Menu

1: Check Balance

2: Withdraw

3: Deposit

4: Exit

Enter a choice: 1

The Balance is 100.0

Main Menu

1: Check Balance

2: Withdraw

3: Deposit

4: Exit

Enter a choice: 2

Enter the amount to withdraw: 25

Main Menu

1: Check Balance

2: Withdraw

3: Deposit

4: Exit

Enter a choice: 1

The Balance is 75.0

Main Menu

1: Check Balance

2: Withdraw

3: Deposit

4: Exit

Enter a choice: 3

Enter the amount to deposit: 300

Main Menu

1: Check Balance

2: Withdraw

3: Deposit

4: Exit

Enter a choice: 1

The Balance is 375.0

Main Menu

1: Check Balance

2: Withdraw

3: Deposit

4: Exit

Enter a choice: 4

Exit

Enter an Account id:

**Chapter 13, Problem 13.8**

# Frank Mock CS 122

# Chapter 13, Problem 13.8

# Encrypt a file by adding 5 to every byte

**def** main**():**

inputFile **=** input**(**"Enter input filename: "**).**strip**()**

outputFile **=** input**(**"Enter output filename: "**).**strip**()**

# Open file for reading

fh **=** open**(**inputFile**,** 'r'**)**

s **=** fh**.**read**()** # Read complete file

word **=** ""

**for** i **in** range**(**len**(**s**)):**

word **+=** chr**(**ord**(**s**[**i**])** **+** 5**)**

# Close input file handle

fh**.**close**()**

# Open output file for writing

fh2 **=** open**(**outputFile**,** 'w'**)**

# Write to the file

**print(**word**,** file **=** fh2**,** end **=** ""**)**

**print(**outputFile **+** " contains the contents of " **+** inputFile **+** " encrypted"**)**

fh2**.**close**()** # Close the output file

main**()**

**Program run with output:**

Enter input filename: words.txt

Enter output filename: words\_encrypted.txt

words\_encrypted.txt contains the contents of words.txt encrypted

**Here is the contents of both files:**

**words.txt**

horse cow pig sheep goat chicken rabbit duck dog cat fish

**words\_encrypted.txt**

mtwxj%ht|%unl%xmjju%ltfy%hmnhpjs%wfggny%izhp%itl%hfy%knxm

**Chapter 13, Problem 13.9**

# Frank Mock CS 122

# Chapter 13, Problem 13.9

# Decrypt a file that had 5 added to every byte

**def** main**():**

inputFile **=** input**(**"Enter filename to be decrypted: "**).**strip**()**

outputFile **=** input**(**"Enter output filename: "**).**strip**()**

# Open file for reading

fh **=** open**(**inputFile**,** 'r'**)**

s **=** fh**.**read**()** # Read all from the file

w **=** ""

**for** i **in** range**(**len**(**s**)):**

w **+=** chr**((**ord**(**s**[**i**])** **-** 5**)%**128**)** # Wrap back to beginning using %128

# Close input file handle

fh**.**close**()**

# Open output file for writing

fh2 **=** open**(**outputFile**,** 'w'**)**

# Write to the file

**print(**w**,** file **=** fh2**,** end **=** ""**)**

**print(**outputFile **+** " contains the contents of " **+** inputFile **+** " decrypted"**)**

fh2**.**close**()** # Close the output file

main**()**

**Program run with output:**

Enter filename to be decrypted: words\_encrypted.txt

Enter output filename: words\_output.txt

words\_output.txt contains the contents of words\_encrypted.txt decrypted

**Contents of words\_encrypted.txt:**

mtwxj%ht|%unl%xmjju%ltfy%hmnhpjs%wfggny%izhp%itl%hfy%knxm

**Contents of words\_output.txt:**

horse cow pig sheep goat chicken rabbit duck dog cat fish

**Chapter 14, Problem 14.2**

# This program determines which numbers occur the

# from a given list of numbers

**def** main**():**

**print(**"This program determines which numbers occur the most "**)**

**print(**"frequently in a list of numbers"**)**

**print(**""**)**

# Main loop of program

**while** **True:**

s **=** input**(**"Enter numbers, each separated by a space: "**).**strip**()**

numbers **=** **[**eval**(**x**)** **for** x **in** s**.**split**()]**

# Create an empty dictionary

dictionary **=** **{}**

# Count the number occurences and store values in

# dictionary

**for** number **in** numbers**:**

**if** number **in** dictionary**:**

dictionary**[**number**]** **+=** 1

**else:**

dictionary**[**number**]** **=** 1

# Get the maximum values

maxCount **=** max**(**dictionary**.**values**())**

pairs **=** list**(**dictionary**.**items**())**

# Reverse pairs in the list

items **=** **[[**x**,** y**]** **for** **(**x**,** y**)** **in** pairs**]**

# Display numbers to the user

**print(**"The numbers that occur the most are "**,** end **=** ""**)**

**for** **(**x**,** y**)** **in** items**:**

**if** y **==** maxCount**:**

**print(**x**,** end **=** " "**)**

# Ask the user if they would like to play again

again **=** input**(**"Again? (y/n)"**)**

**if** again **==** 'n'**:**

**print(**"Goodbye"**)**

**break**

main**()**

**Program run and output on the next page**

This program determines which numbers occur the most

frequently in a list of numbers

Enter numbers, each separated by a space: 16 22 5 4 8 60 9 5 22 40 75

The numbers with the most occurrence are 5 22 Again? (y/n)y

Enter numbers, each separated by a space: 12 8 7 99 32 16 2 80 2 35 19

The numbers with the most occurrence are 2 Again? (y/n)n

Goodbye