**ATM SIMULATER**

**Creating an ATM simulator involves creating a program that mimics the functionalities of a real ATM machine. Here's a simple Python program to simulate an ATM with basic operations like checking balance, depositing, and withdrawing money:**

print('Welcome to Northen Frock Bank ATM')

restart=('Y')

chances = 3

balance = 67.14

while chances >= 0:

pin = int(input('Please Enter You 4 Digit Pin: '))

if pin == (1234):

print('You entered you pin Correctly\n')

while restart not in ('n','NO','no','N'):

print('Please Press 1 For Your Balance\n')

print('Please Press 2 To Make a Withdrawl\n')

print('Please Press 3 To Pay in\n')

print('Please Press 4 To Return Card\n')

option = int(input('What Would you like to choose?'))

if option == 1:

print('Your Balance is Â£',balance,'\n')

restart = input('Would You you like to go back? ')

if restart in ('n','NO','no','N'):

print('Thank You')

break

elif option == 2:

option2 = ('y')

withdrawl = float(input('How Much Would you like to

withdraw? \nÂ£10/Â£20/Â£40/Â£60/Â£80/Â£100 for other enter 1: '))

if withdrawl in [10, 20, 40, 60, 80, 100]:

balance = balance - withdrawl

print ('\nYour Balance is now Â£',balance)

restart = input('Would You you like to go back? ')

if restart in ('n','NO','no','N'):

print('Thank You')

break

elif withdrawl != [10, 20, 40, 60, 80, 100]:

print('Invalid Amount, Please Re-try\n')

restart = ('y')

elif withdrawl == 1:

withdrawl = float(input('Please Enter Desired amount:'))

elif option == 3:

Pay\_in = float(input('How Much Would You Like To Pay In? '))

balance = balance + Pay\_in

print ('\nYour Balance is now Â£',balance)

restart = input('Would You you like to go back? ')

if restart in ('n','NO','no','N'):

print('Thank You')

break

elif option == 4:

print('Please wait whilst your card is Returned...\n')

print('Thank you for you service')

break

else:

print('Please Enter a correct number. \n')

restart = ('y')

elif pin != ('1234'):

print('Incorrect Password')

chances = chances - 1

if chances == 0:

print('\nNo more tries')

break

The best way to do this would be to take each of your sections, and give them each a function of their own. Also, when it's possible, separate user input functions from purely logic to improve readability!

def verify\_pin(pin):

if pin == '1234':

return True

else:

return False

def log\_in():

tries = 0

while tries < 4:

pin = input('Please Enter Your 4 Digit Pin: ')

if verify\_pin(pin):

print("Pin accepted!")

return True

else:

print("Invalid pin")

tries += 1

print("To many incorrect tries. Could not log in")

return False

def start\_menu():

print("Welcome to the atm!")

if log\_in():

# you will need to make this one yourself!

main\_menu()

print("Exiting Program")

start\_menu()

The intended output of the program is to simulate interacting with a very basic bank or ATM. I had no intention of actually storing any account information in a separate file, so each account pin is completely arbitrary and only serves as a medium to make the simulation a little more realistic. The program is fully functional as far as I can tell, though there is the possibility of some bugs slipping through the cracks.

import random

import time

class Accounts:

# Defining Account instance variables.

def \_\_init\_\_(self, pin, balance, annualInterestRate=3.4):

self.pin = pin

self.balance = balance

self.annualInterestRate = annualInterestRate

# Class function to return the monthly interest rate.

def getMonthlyInterestRate(self):

return self.annualInterestRate / 12

# class function to calculate difference between the balance and the amount withdrawn.

def withdraw(self, amount):

self.balance -= amount

# class function to calculate the sum between the balance and the amount deposited.

def deposit(self, amount):

self.balance += amount

# Class function to calculate the product of the balance and the annual interest rate.

def getAnnualInterest(self):

return self.balance \* self.annualInterestRate

# Class function to calculate the product of the balance and the monthly interest rate.

def getMonthlyInterest(self):

return self.balance \* self.getMonthlyInterestRate()

# Revieves pin from user input and validates input.

def getAccountPin():

while True:

pin = input("\nEnter four digit account pin: ")

try:

pin = int(pin)

if pin >= 1000 and pin <= 9999:

return pin

else:

print(f"\n{pin} is not a valid pin... Try again")

except ValueError:

print(f"\n{pin} is not a vaild pin... Try again")

# Recieves user input for option selection and validates selection.

def getSelection():

while True:

selection = input("\nEnter your selection: ")

try:

selection = int(selection)

if selection >= 1 and selection <= 4:

return selection

else:

print(f"{selection} is not a valid choice... Try again")

except ValueError:

print(f"{selection} is not a valid choice... Try again")

# Returns the current working accounts balance.

def viewBalance(workingAccount):

return workingAccount.balance

# Recieves user input and validates if input is either yes, y, no, or n.

def correctAmount(amount):

while True:

answer = input(f"Is ${amount} the correct ammount, Yes or No? ")

try:

answer = answer.lower()

if answer == "y" or answer == "yes":

return True

elif answer == "n" or answer == "no":

return False

else:

print("Please enter a valid response")

except AttributeError:

print("Please enter a valid response")

# Recieves user input on amount to withdraw and validates inputed value.

def withdraw(workingAccount):

while True:

try:

amount = float(input("\nEnter amount you want to withdraw: "))

try:

amount = round(amount, 2)

if amount > 0 and ((workingAccount.balance) - amount) > 0:

answer = correctAmount(amount)

if answer == True:

print("Verifying withdraw")

time.sleep(random.randint(1, 2))

return amount

elif (((workingAccount.balance) - amount) < 0):

print("\nYour balance is less than the withdraw amount")

elif amount == 0:

answer = correctAmount(amount)

if answer == True:

print("Canceling withdraw")

time.sleep(random.randint(1, 2))

return amount

else:

print("\nPlease enter an amount greater than or equal to 0")

except TypeError:

print("\nAmount entered is invalid... Try again")

except ValueError:

print("\nAmount entered is invalid... Try again")

# Recieves user input on amount to deposit and validates inputed value.

def deposit(workingAccount):

while True:

try:

amount = float(input("\nEnter amount you want to deposit: "))

try:

amount = round(amount, 2)

if amount > 0:

answer = correctAmount(amount)

if answer == True:

print("Verifying deposit")

time.sleep(random.randint(1, 2))

return amount

elif amount == 0:

answer = correctAmount(amount)

if answer == True:

print("Canceling deposit")

time.sleep(random.randint(1, 2))

return amount

else:

print("\nPlease enter an amount greater than or equal to 0")

except TypeError:

print("\nAmount entered is invalid... Try again")

except ValueError:

print("\nAmount entered is invalid... Try again")

# End of program to print out account information and return false to end main loop

def exitATM(workingAccount):

print("\nTransaction is now complete.")

print("Transaction number: ", random.randint(10000, 1000000))

print("Current Interest Rate: ", workingAccount.annualInterestRate)

print("Monthly Interest Rate: ", workingAccount.annualInterestRate / 12)

print("Thanks for using this ATM")

return False

def main():

# Creating all accounts possible, could be stored or read from a file/database instead for better functionality overall.

accounts = []

for i in range(1000, 9999):

account = Accounts(i, 0)

accounts.append(account)

# ATM Processes loop

loop = True

while loop == True:

pin = getAccountPin()

print(pin)

# Account session loop

while loop == True:

# Menu Selection

print("\n1 - View Balance \t 2 - Withdraw \t 3 - Deposit \t 4 - Exit ")

selection = getSelection()

# Getting working account object by comparing pins

for acc in accounts:

# Comparing user inputted pin to pins created

if acc.pin == pin:

workingAccount = acc

break

# View Balance

if selection == 1:

print(f"\nYour balance is ${viewBalance(workingAccount)}")

# Withdraw

elif selection == 2:

workingAccount.withdraw(withdraw(workingAccount))

print(f"\nUpdated Balance: ${workingAccount.balance}")

# Deposit

elif selection == 3:

workingAccount.deposit(deposit(workingAccount))

print(f"\nUpdated Balance: ${workingAccount.balance}")

# Exit

elif selection == 4:

loop = exitATM(workingAccount)

# Invalid input

else:

print("Enter a valid choice")

if \_\_name\_\_ == "\_\_main\_\_":

main()

* **it's not realistic to use the pin as the account id. To make it more realistic you could ask for the account id first and then for the pin. Entering the account id would be the simulation of "inserting the card in the ATM".**
* **The function getAccountPin() requests input from the user, a better name would be requestAccountPin()**
* **The function viewBalance below could be a method of Accounts instead of a global function:**
* **def viewBalance(workingAccount):**
* **# Returns the current working accounts balance**
* **return workingAccount.balance**
* **To simplify the function withdraw(workingAccount) move the checks on the balance directly in Accounts.withdraw. For example:**
* **def withdraw(self, amount):**
* **if amount > 0 and self.balance - amount >= 0:**
* **self.balance -= amount**
* **return True**
* **return False**
* **Same for deposit(workingAccount), it can be simplified by moving some of the logic into Accounts.deposit:**
* **def deposit(self, amount):**
* **if amount > 0:**
* **self.balance += amount**
* **return True**
* **return False**
* **The class Accounts contains the information of a single account, so you can just call it Account**