KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY

(KMIT)

Narayanaguda, Hyderabad – 500029

(Affiliated to AICTE and JNTUH)

(Autonomous)

PROJECT TITLE:

BANKING PORTAL

GROUP MEMBERS (CSE-B) :

1.NIRUP REDDY-0533

2.RAMREDDY-053Z

3.MURALIDHAR-053D

INDEX

INTRODUCTION

ABOUT PYTHON:

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting

breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

PANDAS:

- A fast and efficient **DataFrame** object for data manipulation with integrated indexing;
- Tools for **reading and writing data** between in-memory data structures and different formats: CSV and text files, Microsoft Excel, SQL databases, and the fast HDF5 format;
- Intelligent **data alignment** and integrated handling of **missing data**: gain automatic label-based alignment in computations and easily manipulate messy data into an orderly form;
- Flexible **reshaping** and pivoting of data sets;
- Intelligent label-based **slicing**, **fancy indexing**, and **subsetting** of large data sets;
- Columns can be inserted and deleted from data structures for **size mutability**;
- Aggregating or transforming data with a powerful **group by** engine allowing split-apply-combine operations on data sets;
- High performance **merging and joining** of data sets;
- **Hierarchical axis indexing** provides an intuitive way of working with high-dimensional data in a lower-dimensional data structure;
- **Time series**-functionality: date range generation and frequency conversion, moving window statistics, date shifting and lagging. Even create domain-specific time offsets and join time series without losing data;
- Highly **optimized for performance**, with critical code paths written in Python or C.
- Python with *pandas* is in use in a wide variety of **academic and commercial** domains, including Finance, Neuroscience, Economics, Statistics, Advertising, Web Analytics, and more

CSV:

The so-called CSV (Comma Separated Values) format is the most common import and export format for spreadsheets and databases. The csv module implements classes to read and write tabular data in CSV format. It allows programmers to say, “write this data in the format preferred by Excel,” or “read data from this file which was generated by Excel,” without knowing the precise details of the CSV format used by Excel. Programmers can also describe the CSV formats understood by other applications or define their own special-purpose CSV formats.

The csv module’s reader and writer objects read and write sequences. Programmers can also read and write data in dictionary form using the DictReader and DictWriter classes

ABOUT BANKING PORTAL:

Banking portal project is written in Python. The project file contains a python script (main.py) using pandas module and a database (CSV) file. This is a simple console-based system which is very easy to understand and use. Talking about the system, it contains all the basic functions which include creating a new account, withdraws and deposit amount, balance inquiry ,transfer of amount from one account to another account and closing an account . In this mini project, there is login system. If he/she is an existing account holder he/she has to login with their login credentials to access all the banking functions.

Talking about the features of the Banking portal, a user can create an account by providing the name of the account holder, selecting amount type (Saving account or Current account),creating a username and password for login purpose and providing an initial amount more than or equal to 500. Then the user can also deposit and withdraw money just by providing his/her account and entering the amount. For certain purpose, he/she can also check for the balance inquiry which displays the amount.

This simple console-based Banking Portal provides the simplest management of bank account. There's an external database(CSV FILE) connection file used in this mini project to save user's data permanently.

CODE:

###--BANKING--###

```
import random
import pandas as pd
from csv import writer
import csv
from re import A


data=pd.read_csv("test.csv",index_col=0)
print("\t\t\t\tBANKING PORTAL\n")


#TO CREATE AN A BANK ACCOUNT
def createAccount(account_num,name_of_holder,typeof_account,amount,username,password):
 l=[account_num,name_of_holder,typeof_account,amount,username,password]
 with open("test.csv","a") as f:
  writer_object = writer(f,lineterminator='\n')
  writer_object.writerow(l)
  f.close()
 print("\nYOUR ACCOUNT HAS BEEN SUCCESSFULLY CREATED...\n")
 return
```

```
#TO CHECK ACCOUNT BALANCE

def checkBalance(account_num):

  data=pd.read_csv("test.csv",index_col=0)

  print("\nYOUR CURRENT BANK BALANCE:",data['amount'][account_num])



#TO WITHDRAW AMOUNT FROM AN ACCOUNT

def withdrawAmount(account_num,amount):

  data=pd.read_csv("test.csv",index_col=0)

  text = open("test.csv", "r")

  text = ''.join([i for i in text]) \

     .replace(str(data['amount'][account_num]),str(data['amount'][account_num]-amount))

  x = open("test.csv","w")

  x.writelines(text)

  x.close()


#TO DEPOSIT AMOUNT TO A BANK ACCOUNT

def depositAmount(account_num,amount):

  data=pd.read_csv("test.csv",index_col=0)

  text = open("test.csv", "r")

  text = ''.join([i for i in text]) \

     .replace(str(data['amount'][account_num]),str(data['amount'][account_num]+amount))

  x = open("test.csv","w")

  x.writelines(text)

  x.close()
```

```
#TRANSFER AMOUNT
def transferAmount(account_num1,account_num2,amount):
  withdrawAmount(account_num1,amount)
  depositAmount(account_num2,amount)


#TO CLOSE A BANK ACCOUNT
def deleteAccount(accnum):
  file=open("test.csv","r")
  data=csv.reader(file)
  list=[]
  for row in data:
    if str(accnum) not in row:
        list.append(row)
  file.close()
  csvfile=open("test.csv","w",newline="")
  write=csv.writer(csvfile)
  write.writerows(list)
  csvfile.close()


#LOGIN SECTION
def login():
    file=open("test.csv","r")
    data=csv.reader(file)
    dict={}
    for row in data:
```

```
        if len(row)!=0 and row[4]!="username" and row[5]!="password":
            dict[row[4]]=row[5]
    username=input("Enter USERNAME :")
    if username in dict.keys():
        password=input("ENTER PASSWORD: ")
        if(dict[username]==password):
            print("\nLOGIN SUCCESSFUL!\n")
            return 1
        else:
            print("INCORRECT PASSWORD")
            return 0
    else:
        print("INCORRECT USERNAME")
        return 0
    file.close()



#CODE
print("1.NEW ACCOUNT HOLDER")
print("2.EXISTING ACCOUNT HOLDER")
choice=int(input("ENTER YOUR CHOICE:"))
if choice==1:
    #ACCOUNT CREATION SECTION
    data=pd.read_csv("test.csv")
    while 1:
```

```
        account_num = random.randint(1111,9999)
        if account_num not in list(data["account_num"]):
            break
    name_of_holder=input("ENTER THE ACCOUNT HOLDER NAME:")
    typeof_account=input("ENTER THE TYPE OF ACCOUNT (S(SAVINGS)/C(CURRENT)) : ")
    username=input("ENTER THE USERNAME : ")
    password=input("ENTER PASSWORD : ")
    amount=int(input("ENTER THE INITIAL AMOUNT(>=500 FOR SAVING AND >=1000 FOR
CURRENT):"))
    createAccount(account_num,name_of_holder,typeof_account,amount,username,password)


elif choice==2:
    login_func=login()
    while (login_func):
        print("1.BALANCE ENQUIRY")
        print("2.WITHDRAW AMOUNT")
        print("3.DEPOSIT AMOUNT")
        print("4.TRANSFER THE AMOUNT")
        print("5.DELETE ACCOUNT")
        print("6.EXIT")
        n=int(input("ENTER YOUR CHOICE:"))
        if n==1:
            #BALANCE ENQUIRY SECTION
            account_num=int(input("ENTER THE ACCOUNT NUMBER:"))
            checkBalance(account_num)
```

```
elif n==2:
    #AMOUNT WITHDRAW SECTION
    account_num=int(input("ENTER THE ACCOUNT NUMBER:"))
    data=pd.read_csv("test.csv",index_col=0)
    total_amount=data['amount'][account_num]
    print("YOUR CURRENT ACCOUNT BALANCE IS = ",total_amount)
    amount=int(input("ENTER THE AMOUNT TO BE WITHDRAW:"))
    withdrawAmount(account_num,amount)
    print(amount,"AMOUNT HAS BEEN WITHDRAWN FROM XXXX ACCOUNT")

elif n==3:
    #AMOUNT DEPOSIT SECTION
    account_num=int(input("ENTER THE ACCOUNT NUMBER:"))
    amount=int(input("ENTER THE AMOUNT TO BE DEPOSITED:"))
    depositAmount(account_num,amount)
    print("SUCCESSFULLY DEPOSITED...")
elif n==4:
    #TRANSFER OF AMMOUNT
    data=pd.read_csv("test.csv",index_col=0)
    account_num1=int(input("ENTER YOUR ACCOUNT NUMBER:"))
    print("YOUR CURRENT ACCOUNT BALANCE IS = ",data['amount'][account_num1])
    account_num2=int(input("ENTER RECIPIENT ACCOUNT NUMBER:"))
    amount=int(input("ENTER THE AMOUNT TO TRANSFER:"))
    transferAmount(account_num1,account_num2,amount)
    print("TRANSFERED SUCCESSFULLY....")
```

```
elif n==5:
    #CLOSING AN ACCOUNT SECTION
    account_num=int(input("ENTER THE ACCOUNT NUMBER TO CLOSE:"))
    deleteAccount(account_num)
    print("YOUR ACCOUNT HAS BEEN SUCESSFULLY CLOSED..")
elif n==6:
    break
```

OUTPUT:

```
                                BANKING PORTAL

1.NEW ACCOUNT HOLDER
2.EXISTING ACCOUNT HOLDER
ENTER YOUR CHOICE:
```

	A	B	C	D	E	F	G
1	account_n	name	type	amount	username	password	
2	7836	ram	s	20000	ram789	98765	
3	1425	murali	s	19000	murali123	12345	

#CREATING A NEW ACCOUNT:

```
                                BANKING PORTAL

1.NEW ACCOUNT HOLDER
2.EXISTING ACCOUNT HOLDER
ENTER YOUR CHOICE:1
ENTER THE ACCOUNT HOLDER NAME:person1
ENTER THE TYPE OF ACCOUNT (S(SAVINGS)/C(CURRENT)) : S
ENTER THE USERNAME : person1@123
ENTER PASSWORD : 87697
ENTER THE INITIAL AMOUNT(>=500 FOR SAVING AND >=1000 FOR CURRENT):10000

YOUR ACCOUNT HAS BEEN SUCCESSFULLY CREATED...
```

	A	B	C	D	E	F
1	account_n	name	type	amount	username	password
2	7836	ram	s	20000	ram789	98765
3	1425	murali	s	19000	murali123	12345
4	8331	person1	S	8000	person1@	87697

#BALANCE ENQUIRY

```
                                BANKING PORTAL

1.NEW ACCOUNT HOLDER
2.EXISTING ACCOUNT HOLDER
ENTER YOUR CHOICE:2
Enter USERNAME :person1@123
ENTER PASSWORD: 87697

LOGIN SUCCESSFUL!

1.BALANCE ENQUIRY
2.WITHDRAW AMOUNT
3.DEPOSIT AMOUNT
4.TRANSFER THE AMOUNT
5.DELETE ACCOUNT
6.EXIT
ENTER YOUR CHOICE:1
ENTER THE ACCOUNT NUMBER:8331

YOUR CURRENT BANK BALANCE: 12000
```

#WITHDRAW AMOUNT

```
                                BANKING PORTAL

1.NEW ACCOUNT HOLDER
2.EXISTING ACCOUNT HOLDER
ENTER YOUR CHOICE:2
Enter USERNAME :person1@123
ENTER PASSWORD: 87697

LOGIN SUCCESSFUL!

1.BALANCE ENQUIRY
2.WITHDRAW AMOUNT
3.DEPOSIT AMOUNT
4.TRANSFER THE AMOUNT
5.DELETE ACCOUNT
6.EXIT
ENTER YOUR CHOICE:2
ENTER THE ACCOUNT NUMBER:8331
YOUR CURRENT ACCOUNT BALANCE IS =  10000
ENTER THE AMOUNT TO BE WITHDRAW:2000
2000 AMOUNT HAS BEEN WITHDRAWN FROM XXXX ACCOUNT
```

	A	B	C	D	E	F	G
1	account_n	name	type	amount	username	password	
2	7836	ram	s	20000	ram789	98765	
3	1425	murali	s	19000	murali123	12345	
4	8331	person1	S	8000	person1@	87697	

#DEPOSIT AMOUNT

```
                                BANKING PORTAL

 1.NEW ACCOUNT HOLDER
 2.EXISTING ACCOUNT HOLDER
 ENTER YOUR CHOICE:2
 Enter USERNAME :person1@123
 ENTER PASSWORD: 87697

 LOGIN SUCCESSFUL!

 1.BALANCE ENQUIRY
 2.WITHDRAW AMOUNT
 3.DEPOSIT AMOUNT
 4.TRANSFER THE AMOUNT
 5.DELETE ACCOUNT
 6.EXIT
 ENTER YOUR CHOICE:3
 ENTER THE ACCOUNT NUMBER:8331
 ENTER THE AMOUNT TO BE DEPOSITED:4500
 SUCCESSFULLY DEPOSITED...
```

	A	B	C	D	E	F	G
1	account_n	name	type	amount	username	password	
2	7836	ram	s	20000	ram789	98765	
3	1425	murali	s	19000	murali123	12345	
4	8331	person1	S	12500	person1@	87697	

#TRANSFER AMOUNT

```
                                BANKING PORTAL

 1.NEW ACCOUNT HOLDER
 2.EXISTING ACCOUNT HOLDER
 ENTER YOUR CHOICE:2
 Enter USERNAME :person1@123
 ENTER PASSWORD: 87697

 LOGIN SUCCESSFUL!

 1.BALANCE ENQUIRY
 2.WITHDRAW AMOUNT
 3.DEPOSIT AMOUNT
 4.TRANSFER THE AMOUNT
 5.DELETE ACCOUNT
 6.EXIT
 ENTER YOUR CHOICE:4
 ENTER YOUR ACCOUNT NUMBER:8331
 YOUR CURRENT ACCOUNT BALANCE IS =  12500
 ENTER RECIPIENT ACCOUNT NUMBER:7836
 ENTER THE AMOUNT TO TRANSFER:500
 TRANSFERED SUCCESSFULLY....
```

	A	B	C	D	E	F	G
1	account_n	name	type	amount	username	password	
2	7836	ram	s	20500	ram789	98765	
3	1425	murali	s	19000	murali123	12345	
4	8331	person1	S	12000	person1@	87697	

#CLOSING AN ACCOUNT

```
                                BANKING PORTAL

1.NEW ACCOUNT HOLDER
2.EXISTING ACCOUNT HOLDER
ENTER YOUR CHOICE:2
Enter USERNAME :person1@123
ENTER PASSWORD: 87697

LOGIN SUCCESSFUL!

1.BALANCE ENQUIRY
2.WITHDRAW AMOUNT
3.DEPOSIT AMOUNT
4.TRANSFER THE AMOUNT
5.DELETE ACCOUNT
6.EXIT
ENTER YOUR CHOICE:5
ENTER THE ACCOUNT NUMBER TO CLOSE:8331
YOUR ACCOUNT HAS BEEN SUCESSFULLY CLOSED..
```

	A	B	C	D	E	F	G
1	account_n	name	type	amount	username	password	
2	7836	ram	s	20500	ram789	98765	
3	1425	murali	s	19000	murali123	12345	
4							

CONCLUSION:

The console based banking portal project had been developed to an extinct. This project can handle smaller amount of data. This can be extended further by using different databases to optimise the project and can handle huge data. In this project more number of banking functions can be added. The login system can be made more secure. In this project the bugs or loop holes can be fixed such as, if any username exists previously it does not show any error (username already exists!). This project can be developed as an application which can be used in day-to-day banking.

THANK YOU!