EXPLORATION OF PYTHON

(To Create an ATM MACHINE)

An Internship Report Submitted in partial fulfilment

of the requirement for undergraduate degree of

Bachelor of Technology

in

COMPUTER SCIENCE AND ENGINEERING

By

Bathula Uday Kumar

HU21CSEN0100964

Under the Guidance of

Dr. Rajendra Prasad Babu

Assistant Professor

Department Of Computer Science and Engineering

GITAM School of Technology

GITAM (Deemed to be University)

Hyderabad-502329

December 2023

DECLARATION

I submit this internship work entitled **"To Create an ATM Machine"** to GITAM (Deemed to Be University), Hyderabad in partial fulfilment of the requirements for the award of the degree of "**Bachelor of Technology**" in "**Computer Science and Engineering**". I declare that it was carried out independently by me under the guidance of **Mr. Rajendra Prasad Babu Sir,** Asst. Professor, GITAM (Deemed to Be University), Hyderabad, India.

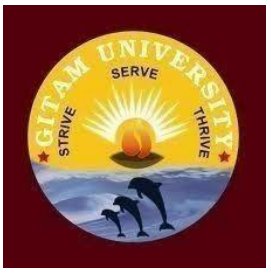
The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Place: HYDERABAD

Name: Bathula Uday Kumar

Date: 31-12-2023

Student Roll No: HU21CSEN0100964



GITAM (DEEMED TO BE UNIVERSITY)

Hyderabad-502329, India

Dated: 31 -12-2023

CERTIFICATE

This is to certify that the Report entitled **"To Create an ATM Machine"** is being submitted by Bathula Uday Kumar (HU21CSEN0100964) in partial fulfilment of the requirement for the award of **Bachelor of Technology in Computer Science And Engineering** at GITAM (Deemed to Be University), Hyderabad during the academic year 2023-24.

It is faithful record work carried out by him at the **Computer Science and Engineering Department**, GITAM University Hyderabad Campus under my guidance and supervision.

Dr. Rajendra Prasad Babu
Assistant Professor
Department of CSE

Dr.Mahaboob Basha Shaik
Professor and HOD
Department of CSE

ACKNOWLEDGEMENT

Apart from my effort, the success of this internship largely depends on the encouragement and guidance of many others. I take this opportunity to express my gratitude to the people who have helped me in the successful competition of this internship.

I would like to thank respected **Dr. D Sambasiva Rao**, Pro Vice Chancellor, GITAM Hyderabad, and **Dr. N. Seetharamaiah**, Principal, GITAM Hyderabad.

I would like to thank respected Mahaboob Basha Shaik sir**,** Head of the Computer Science and Engineering for giving me such a wonderful opportunity to expand my knowledge for my own branch and giving me guidelines to present an internship report. It helped me a lot to realize of what we study for.

I would like to thank the respected faculties **Mr. Rajendra Prasad Babu** who helped me to make this internship a successful accomplishment.

I would also like to thank my friends who helped me to make my work more organized and well-stacked till the end.

Bathula Uday Kumar

HU21CSEN0100964

ABSTRACT

Python programming has emerged as a versatile and powerful language, playing a pivotal role in various domains, from web development to scientific computing. This project focuses on leveraging Python for developing a robust and efficient solution in the realm of data analysis and manipulation. The primary objective is to explore and implement data processing techniques using Python, emphasizing its capabilities in handling diverse datasets, and performing complex computations.

The methodology involves preprocessing the dataset, feature extraction, and the implementation of a supervised learning model. Python's extensive ecosystem, including libraries like NumPy, Pandas, and Scikit-learn, facilitates seamless data manipulation, statistical analysis, and model training. The code is structured to be modular and scalable, allowing for easy adaptation to different datasets and sentiment analysis tasks.

The project concludes with an evaluation of the model's performance based on metrics such as accuracy, precision, and recall. Additionally, insights into potential enhancements and further applications of Python in sentiment analysis are discussed. This abstract underscore the significance of Python programming in empowering developers to create efficient and effective solutions for complex data analysis tasks.

REPORT CONTENT

Module 4: Application of Python for ATM Machine

MODULE 1

INTRODUCTION

History of Python:

Python was developed in 1980 by **Guido van Rossum** at the National Research Institute for Mathematics and Computer Science in the Netherlands as a successor of ABC language capableof exception handling and interfacing. Python features a dynamic type of system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional, and procedural, and has a large and comprehensive standard library.

Van Rossum picked the name Python for the new language from aTV show, Monty Python's Flying Circus.

In December 1989 the creator developed the 1st python interpreteras a hobby and then on 16 October 2000, Python 2.0 was released with many new features.

In December 1989, I was looking for a "hobby" programming project that would keep me occupied during the week around Christmas. My office ... would be closed, but I had a home computer, and not much else on my hands. I decided to write an interpreter for the new scripting language I had been thinking about lately: a descendant of ABC that would appeal to Unix/C hackers. I chose Python as a working title for the project, being in a slightly irreverent mood (and a big fan of Monty Python's Flying Circus)

Python Releases

- Python 1.0 - January 1994
- Python 1.5 - December 31, 1997
- Python 1.6 - September 5, 2000
- Python 2.0 - October 16, 2000
- Python 2.1 - April 17, 2001
- Python 2.2 - December 21, 2001
- Python 2.3 - July 29, 2003
- Python 2.4 - November 30, 2004
- Python 2.5 - September 19, 2006
- Python 2.6 - October 1, 2008
- Python 2.7 - July 3, 2010
- Python 3.0 - December 3, 2008
- Python 3.1 - June 27, 2009
- Python 3.2 - February 20, 2011
- Python 3.3 - September 29, 2012
- Python 3.4 - March 16, 2014
- Python 3.5 - September 13, 2015
- Python 3.6 - December 23, 2016

Why Python?

The language's core philosophy is summarized in the document The Zen of Python (PEP 20), which includes aphorisms such as…

- Beautiful is better than ugly.
- Simple is better than complex.
- Complex is better than complicated.
- Readability counts
- Explicit is better than implicit.

Characteristics of Python

- **Interpreted Language**: Python is processed at runtime by PythonInterpreter
- **Easy to read**: Python source-code is clearly defined and visible to theeyes.
- **Portable**: Python codes can be run on a wide variety of hardwareplatforms having the same interface.
- **Extendable**: Users can add low level-modules to Python interpreter.
- **Scalable**: Python provides an improved structure forsupporting large programs than shell-scripts.
- **Object-Oriented Language**: It supports object-orientedfeatures and techniques of programming.
- **Interactive Programming Language**: Users can interact with the python interpreter directly for writing programs.
- **Easy language**: Python is easy to learn language especially for beginners.
- **Straight forward Syntax**: The formation of python syntax is simple andstraight forward which also makes it popular.

Data Structures in Python

LISTS -

- Ordered collection of data.
- Supports similar slicing and indexing functionalities as in the case of strings
- They are mutable.
- Advantage of a list over a conventional array
    - Lists have no size or type constraints (no settingrestrictions beforehand).
    - They can contain different object types.
    - We can delete elements from a list by using Dellist_name[index_val]
- Example-
    - my_list = ['one', 'two','three',4,5]
    - len(my_list) would output 5.

DICTIONARY -

- Lists are sequences but the dictionaries are mappings.
- They are mappings between a unique key and a value pair.
- These mappings may not retain order.
- Constructing a dictionary.
- Accessing objects from a dictionary**.**
- Basic Syntax:

    - d= {} empty dictionary will be generated and assign keys andvalues to it, like d['animal'] = 'Dog'

TUPLES -

Immutable in nature, i.e. they cannot be changed.

- No type of restriction
- Indexing and slicing, everything's same as that in strings and lists.
- Constructing tuples.
- Basic tuple methods.
- Immutability.
- When to use tuples?
- We can use tuples to present things that shouldn't change, such asdays of the week, or dates on a calendar, etc.

SETS -

- A set contains unique and unordered elements, and we can constructthem by using a set () function.
- Convert a list into Set-
- l= [1,2,3,4,1,1,2,3,6,7]
- k = set(l)
- k becomes {1,2,3,4,6,7}
- Basic Syntax-
- x=set ()
- x.add (1)
- x = {1}
- x.add (1)
- This would make no change in x now.

MODULE 2

FOUNDATION OF DATA MANIPULATION AND VISUALISATION

File Handling in Python:

Python too supports file handling and allows users to handle files i.e., to read and write files, along with many other file handling options, to operate on files. The concept of file handling has stretched over to various other languages, but the implementation is either complicated or lengthy, but alike other concepts of Python, this concept here is also easy and short. Python treats files differently as text or binary and this is important. Each line of code includes a sequence of characters, and they form a text file. Each line of a file is terminated with a special character, called the EOL or End of Line characters like comma.
{,} or newline character. It ends the current line and tells the interpreter a new one has begun. Let's start with Reading and Writing files.

We use the open () function in Python to open a file in read or write mode. As explained above, open () will return a file object. To return a file object we use **open () function** along with two arguments, that accepts file name and the mode, whether to read or write. So, the syntax being: **open (filename, mode).** There are three kinds of mode, that Python provides and how files can be opened:

- "r ", for reading.
- "w ", for writing.
- "a ", for appending.
- "r+ ", for both reading and writing.

Use of NumPy:

NumPy is a Python package. It stands for 'Numerical Python'. It is alibrary consisting of multidimensional array objects and a collection of routines for processing of array.

Numeric, the ancestor of NumPy, was developed by Jim Hugunin. Another package Num array was also developed, having some additional functionalities. In 2005, Travis Oliphant created NumPy package by incorporating the features of Num array into Numeric package. There are many contributors to this open-source project.

Operations using NumPy.

Using NumPy, a developer can perform the following operations −

- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.
- Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

Use of Matplotlib-

Matplotlib is a library for making 2D plots of arrays in Python. Although it has its origins in emulating the MATLAB graphics commands, it is independent ofMATLAB, and can be used in a Pythonic, object-oriented way. Although Matplotlib is written primarily in pure Python, it makes heavy use of NumPy and other extension code to provide good performance even for large arrays.

Matplotlib is designed with the philosophy that you should be able to create simple plots with just a few commands, or just one! If you want to see a histogram of your data, you shouldn't need to instantiate objects, call methods, set properties, and so on; it should just work.

Use of Pandas-

Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide rangeof fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tools using its powerful data structures.The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data.

Key Features of Pandas-

- Fast and efficient DataFrame object with default and customized indexing**.**
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.

Pandas deals with the following three data structures –

- Series
- DataFrame
- Panel

These data structures are built on top of Numpy array, which means they are fast.

Use of OpenCV-

OpenCV was started at Intel in 1999 by Gary Bradsky and the first release came out in 2000. Vadim Picaresque joined Gary Bradsky to manage Intel's Russian software OpenCV team. In 2005, OpenCV was used on Stanley, the vehicle who won 2005 DARPA Grand Challenge.

Below is the list of contributors who submitted tutorials to OpenCV-Python.

-Alexander Mordvintsev (GSoC-2013 mentor)

-Abid Rahman K. (GSoC-2013 intern)

Use the function: **cv2.imread()** to read an image. The image should be in the working directory, or a full path of the image should be given.

The second argument is a flag which specifies the way an image should be read.

- cv2.IMREAD_COLOR: Loads a color image. Any Transparency of image will be neglected. It is the default flag.
- cv2.IMREAD_GRAYSCALE: Loads image in grayscale mode
- cv2.IMREAD_UNCHANGED: Loads image as suchincluding alpha channel.

MODULE 3

PYTHON FUNDAMENTALS AND BEYOND

Data Types in Python:

- Data Types In previous installments of these notes, we've alluded "data types".
- All values in python have a type.
- Intuitively, the number 7 and the piece of text "saber-toothed tiger" are not the same type of value.
- There are 4 basic (or primitive) types in python:

type	Short name	example values
integer	int	-10, -7, -4, 0, 2, 13, 1117, 98372
float	float	-5.3, -4.0, -0.756, 0.0, 1.25, 12386.2
string	str	"hello", "sentences are strings", "3.0", "remember to start. your homework early!"
boolean	bool	True, False

Integers

Integers are all whole numbers. As soon as there is a decimal point, the number is no longer an int, it's a float instead. Note: Numbers that you might be used to writing with a comma (such as 1,000,000) are written as 1000000 in python. This is because python would try to interpret 1,000,000 as three separate numbers!

Floats

Floats are all numbers with a decimal point. Even 1.0 and 6.0 and -7.0!You may also hear floats referred to as "doubles". This is a second name that comes from the history of computers and programming languages. This is because floats and doubles represent two different ways to store high-precision (decimal) numbers behind the scenes. We don't have to worry about this because of advancements in computer architectures!

Strings

In programming, we call all sequences of characters (letters, but including the numbers and symbols like "1" and "@") strings. Notice that there are always quotation marks around strings. It's important to know the difference because the int 3 acts very differently than the string "3"! In python, we can enclose strings in either single quotes 'like this' or double quotes "like this". See the section on strings for lots more information about how special they are!

Boolean

The name of these values comes from George Boolean who developed Boolean algebra in The Mathematical Analysis of Logic and The Laws of Thought in the mid-1800s. It was named after Boole in the early 1900s. Boolean values are only the values True or False. We can use them to represent all sorts of useful things like whether or not to execute a particular block of code. (We'll talk a lot more about boolean values later when we talk about conditionals.)

Operations

Operations are used to combine two different values to calculate a single new value. In python, we have several operators, some of which are used in algebra and other math courses, and some of which are not!

Let's Understand Operators in Detail

operator	name
+	plus
-	minus
*	times
/	divide
//	double divide or integer divide
**	power
%	modulus

Statements and Syntax in Python:

Statements:

In Python, statements are instructions that the interpreter can execute. They are the basic building blocks of a Python program.

Here are some common types of statements in Python:

the Python if statement, which is the main statement used for selecting from alternative actions based on test results. Because this is our first in-depth look at compound statements—statements that embed other statements—we will also explore the general concepts behind the Python statement syntax model here in more detail. Because the if statement introduces the notion of tests, this chapter will also deal with Boolean expressions and fill in some details on truth tests in general.

if Statements

In simple terms, the Python if statement selects actions to perform. It's the primary selection tool in Python and represents much of the logic a Python program possesses.

General Format

The Python if statement is typical of if statements in most procedural languages. It takes the form of an if test, followed by one or more optional elif ("else if") tests and a final optional else block. The tests and the else part each have an associated block of nested statements, indented under a header line. When the if statement runs, Python executes the block.

Python File Operations

Python is the extensively used web-based or website development programming language that facilitates any type of input to be incorporated in the codes. In python, files can be created, read, updated, deleted, and managed while considering. The types of files are taken differently, especially the binary files and text files.

Opening a Python File

The very first operation to work on a file is to open it. In Python, the open () function (built-in function) is used to open a file in both read and write mode. This function returns a file object. In the open () function, we define two arguments in which the first is the file name and the second is the mode in which we want to open that file.

Syntax in Python:

Syntax: file = open (“abc.txt”, “r”)

In the above example, the user wants to open a file named “abc.txt” in the read mode. Similarly, users can open a file in different modes like “w” for write mode and “a” for append mode. In Python, the user can also specify the binary or textual mode in which 18 he wants to open a file. It is not mandatory for a user to specify the mode of the file; if no mode is specified, then by default Python will open a file in reading “r” mode.

Syntax: file = open (“abc.txt”)

The above two ways of opening a file will perform the same action, i.e. open a Python file in read mode. Let’s understand different file modes in Python:

Mode	Function Description
“r”	It opens a file in reading mode
“w”	It opens a file in write mode
“a”	Opens a file in append mode (adding text at the end of the file)
“x”	Creates a specified file, returns an error if the file already exists
“r+”	It opens a file in both reading and writing mode
“b”	Opens a file in binary mode (in case of images, .exe files)
“t”	It opens a file in text mode

Closing a file

It is a good practice to close a file after the desired operations are done on it as this will free up all the resources used in that file and be allocated somewhere else by the Operating System. For closing a file in Python, the close() method is used. Although it is not mandatory to close a file as the Python uses the garbage collector to clean the unreferenced objects, but it is a good practice, and we must do it.

Functions in Python:

What is a function in Python?

In Python, a function is a group of related statements that performs a specific task. Functions help break our program into smaller and modular chunks. As our program grows larger and larger, functions make it more organized and manageable. Furthermore, it avoids repetition and makes the code reusable.

Syntax of Function

```
def function_name(parameters):
        """docstring"""
        statement(s)
```

Above shown is a function definition that consists of the following components.

Keyword def that marks the start of the function header.

In Python, a function is a reusable block of code that performs a specific task or set of tasks. Functions allow you to organize your code into modular and manageable pieces, making it easier to read, understand, and maintain. Functions also promote code reuse, as you can call a function multiple times from different parts of your program.

Here's a breakdown of key concepts related to functions in Python:

Defining a Function:

To define a function in Python, you use the def keyword followed by the function name and a set of parentheses. Any parameters the function accepts are listed within the parentheses. The function code block is indented beneath the function definition.

Example:

```
def greet(name):
    print(f"Hello, {name}!")
```

Calling a Function:

To execute the code within a function, you "call" the function by using its name followed by parentheses. If the function accepts parameters, you provide values for those parameters within the parentheses.

Example:

```
greet("John")
```

Parameters and Arguments:

Parameters are variables that are used in the function definition, while arguments are the values passed to the function when it is called.

Example:

```
def add_numbers(x, y):
    result = x + y
    return result
sum_result = add_numbers(3, 5)
```

Return Statement:

Functions can return a value using the return statement. The returned value can be assigned to a variable or used in expressions.

Example:

```
def square(x):
    return x * x
result = square(4)
```

Default Arguments:

You can provide default values for function parameters, allowing callers to omit them if they are not needed.

Example:

```
def power(base, exponent=2):
    return base ** exponent
result1 = power(3) # Uses default exponent (2)
result2 = power(3, 4)  # Uses specified exponent (4)
```

Variable-Length Argument Lists:

Functions can accept a variable number of arguments using *args (for variable positional arguments) and **kwargs (for variable keyword arguments).

Example:

```
def print_args(*args, **kwargs):
    print("Positional arguments:", args)
    print("Keyword arguments:", kwargs)
print_args(1, 2, 3, name='John', age=25)
```

Lambda Functions:

Lambda functions are anonymous functions defined using the lambda keyword. They are often used for short, simple operations.

Example:

```
square = lambda x: x * x
result = square(3)
```

Functions are a fundamental building block in Python programming, enabling you to write modular and reusable code. They play a crucial role in structuring your programs and improving code readability and maintainability.

MODULE 4

APPLICATION OF PYTHON FOR ATM MACHINE

Section 1: Introduction

The code introduces a simple Automated Teller Machine (ATM) system implemented in Python using object-oriented programming principles. The system is designed to manage customer accounts, handle transactions such as withdrawals and deposits, and provide a mini statement of account activity.

Section 2: ATM Class Definition

The chapter begins by defining the ATM class, which serves as the blueprint for creating ATM objects. The class has an __init___method to initialize customer details, ATM number, PIN, balance, and a list to track transactions. Key methods include check_balance, withdraw, deposit, mini_statement, and update_file.

2.1 __init___Method:

- Initializes customer information, ATM number, PIN, balance, and an empty transaction list.

2.2 check_balance Method:

- Returns the current account balance.

2.3 withdraw Method:

- Allows the customer to withdraw a specified amount, updating the balance and recording the transaction if the withdrawal is valid.

2.4 deposit Method:

- Enables customers to deposit money into their accounts, updating the balance and recording the transaction if the deposit is valid.

2.5 mini_statement Method:

- Displays a mini statement showing the available balance and recent transactions.

2.6 update_file Method:

- Updates the external file ('atm_info.txt') with the latest account information.

Class Definition Code:

```
class ATM:
    def___init_(self, customer_name, atm_number, pin, balance):
        self.customer_name = customer_name
        self.atm_number = atm_number
        self.pin = pin
        self.balance = balance
        self.transactions = []
    def check_balance(self):
        return self.balance
    def withdraw(self, amount):
        if amount > 0 and amount <= self.balance:
            self.balance -= amount
            self.transactions.append(f"Withdrawal of {amount} from {self.customer_name}'s
account")
            self.update_file()
            return True
        else:
            return False
    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            self.transactions.append(f"Deposit of {amount} into {self.customer_name}'s
account")
            self.update_file()
            return True
        else:
            return False
    def mini_statement(self):
```

```
    print("Mini Statement for", self.customer_name)
    print("- Available balance:", self.balance)
    if not self.transactions:
        print("- No transactions to display.\n")
    else:
        for transaction in self.transactions:
            print("-", transaction)
        print()
def update_file(self):
    with open('atm_info.txt', 'r') as file:
        lines = file.readlines()
    with open('atm_info.txt', 'w') as file:
        for line in lines:
            data = line.strip().split()
            if len(data) >= 4:
                atm_number, pin, balance = map(int, data[1:])
                if self.atm_number == atm_number and self.pin == pin:
                    line = f"{self.customer_name} {self.atm_number} {self.pin} {self.balance}\n"
            file.write(line)
```

Section 3: Main Function

The chapter then introduces the main function that serves as the entry point of the program. It reads account information from an external file ('atm_info.txt'), prompts the user for their ATM number and PIN, and provides a menu for different operations.

3.1 Reading ATM Information:

- Reads account information from 'atm_info.txt' and creates ATM objects for each account.

3.2 User Authentication:

- Prompts the user for their ATM number and PIN, ensuring authentication.

3.3 Main ATM Menu:

- Presents a menu with options such as checking balance, withdrawing, depositing, viewing a mini statement, and exiting.

3.4 Handling User Choices:

- Implements logic to execute user-selected operations and interact with the ATM object accordingly.

3.5 Exiting the Program:

- Allows users to exit the program with a friendly message.

Main Function Code:

```
def main():
    atm_info = []
    with open('atm_info.txt', 'r') as file:
        for line in file:
            data = line.strip().split()
            if len(data) >= 4:
                customer_name = data[0]
                atm_number, pin, balance = map(int, data[1:])
                atm_info.append(ATM(customer_name, atm_number, pin, balance))
    user_ATM_number = int(input("Enter ATM number: "))
    user_PIN = int(input("Enter PIN: "))
    for atm in atm_info:
        if user_ATM_number == atm.atm_number and user_PIN == atm.pin:
            print("\nWelcome to our ATM,", atm.customer_name + "!\n")
            while True:
                print("Please select an option:")
                print("1. Check Available Balance")
                print("2. Withdraw Amount")
                print("3. Deposit Amount")
                print("4. View Mini Statement")
                print("5. Exit")
                choice = int(input())
                if choice == 1:
                    print("Your balance is", atm.check_balance(), "\n")
                elif choice == 2:
                    withdraw_amount = int(input("Enter amount to withdraw: "))
                    if atm.withdraw(withdraw_amount):
```

```
                print(f"{withdraw_amount} has been withdrawn from your account. Your new balance is {atm.check_balance()}\n")
            else:
                print("Invalid amount or insufficient funds.\n")
        elif choice == 3:
            deposit_amount = int(input("Enter amount to deposit: "))
            if atm.deposit(deposit_amount):
                print(f"{deposit_amount} has been deposited into your account. Your new balance is {atm.check_balance()}\n")
            else:
                print("Invalid deposit amount.\n")
        elif choice == 4:
            atm.mini_statement()
        elif choice == 5:
            print("Thank you for using our ATM!")
            break
        else:
            print("Invalid choice. Please try again.\n")
    break
else:
    print("Invalid ATM number or PIN. Please try again.")
if __name__== "_main_":
    main()
```

Section 4: Execution

The chapter concludes by explaining the correct execution of the program and emphasizes the importance of secure user authentication in ATM systems. It highlights the functionality of each method within the ATM class and provides insights into designing basic financial systems using object-oriented programming.

OUTPUT:

```
Enter ATM number: 123456
Enter PIN: 1234

Welcome to our ATM, Uday!

Please select an option:
1. Check Available Balance
2. Withdraw Amount
3. Deposit Amount
4. View Mini Statement
5. Exit
1
Your balance is 18752

Please select an option:
1. Check Available Balance
2. Withdraw Amount
3. Deposit Amount
4. View Mini Statement
5. Exit
2
Enter amount to withdraw: 33
33 has been withdrawn from your account. Your new balance is 18719

Please select an option:
1. Check Available Balance
2. Withdraw Amount
3. Deposit Amount
4. View Mini Statement
5. Exit
4
Mini Statement for Uday
- Available balance: 18719
- Withdrawal of 33 from Uday's account

Please select an option:
1. Check Available Balance
2. Withdraw Amount
3. Deposit Amount
4. View Mini Statement
5. Exit
5
Thank you for using our ATM!
```

CONCLUSION:

In conclusion, understanding and effectively using functions in Python is essential for writing modular, reusable, and well-structured code. Functions provide a means of organizing code into manageable units, promoting readability, maintainability, and code reuse. Key concepts such as defining functions, handling parameters and arguments, utilizing return statements, setting default arguments, working with variable-length argument lists, and employing lambda functions offer a versatile toolkit for Python developers.

By leveraging these concepts, developers can create efficient and modular programs, making it easier to collaborate, maintain, and scale their projects. Whether it's defining custom functions for specific tasks, utilizing default arguments for flexibility, or employing lambda functions for concise operations, a solid understanding of Python functions is crucial for building robust and maintainable software solutions.

The code also incorporates file handling to persist account information in a text file (atm_info.txt). The user interface, implemented in the main function, guides users through a menu-driven interaction, allowing them to perform various transactions securely. Furthermore, the revised version streamlines the original code, making it more concise and enhancing its overall clarity.

This project serves as a practical example of object-oriented programming principles and demonstrates effective file handling for persistent data storage. While it is a simplified representation, the structure provides a solid foundation that can be extended and enhanced for more complex banking systems or educational purposes. Overall, the project strikes a balance between functionality, readability, and simplicity, making it a valuable resource for learning and understanding Python programming concepts.

REFERENCES:

1. Python Documentation:

   - Python Classes and Objects: https://docs.python.org/3/tutorial/classes.html

   - File Input/Output: https://docs.python.org/3/tutorial/inputoutput.html#reading-and-writing-files

   - Control Flow Statements: https://docs.python.org/3/tutorial/controlflow.html

   - Built-in Functions: https://docs.python.org/3/library/functions.html

2. Object-Oriented Programming (OOP):

   - Introduction to OOP Concepts: https://realpython.com/python3-object-oriented-programming/

   - Python OOP: https://realpython.com/python-object-oriented-programming/

3. Security Considerations:

   - Python Security Practices: https://docs.python-guide.org/writing/style/#security-best-practices

4. General Python Programming:

   - Python Official Website: https://www.python.org/

   - W3Schools Python Tutorial: https://www.w3schools.com/python/

   - GeeksforGeeks Python: https://www.geeksforgeeks.org/python-programming-language/