



# University Preparation Students and Adults

Module: Computer science

Assignment Type: Report

Term: 3

Date: 08/05/2025

Weighting:

Module Tutor: Claudia Papi

Class: Computer science 2

Student number: DZD3295913

## Marking Scheme

80-100 A\*

70-79 A

60-69 B

50-59 C

40-49 D

35-39 Narrow Fail (NF)

10-34 Clear Fail (CF)

0-9 Unclassified (U)

Student First Name:

Abdelkarim

Student Last Name:

Bouarroudj

**Student Declaration:** *I declare that the work submitted is my own.*

## Problem Definition and Analysis

The project I have selected for my final assessment is a Banking system (basically a bank) where a customer can do what most of us can imagine - Login, Check how much money I have, Deposit and Withdraw money securely. The reason I picked this idea is because it's a real life logic that is intuitively easy to understand but you'll still need to carefully plan and code for it. It also fits nicely with the Python that I've been learning in my course - loops, conditionals, input validation, and some basic error handling. As we've introduced in most of our lessons the use of Python, I was pretty comfortable I could come up with a system that is simply to use, easy to test, and that worked! A banking system sounds like something that might be valuable to people who just need something a little bit simple. The primary audience would be the beginners or those who just want to know how does banking work behind the scenes. The system flow is simple: the user should provide his username and password. Once they log on, they have four main options: to check their balance, to deposit money, to withdraw money or to exit. These are assumed after the user logs in successfully into the server first. While I was doing the planning, I reviewed the examples from our lessons and I also checked a couple of online tutorials, and it helped me to build up the logic and the structure of the system.

## Documented Design

### System Overview

The overall design is based on a simple menu-driven interface that interacts with the user through the terminal. The program starts by prompting the user for their login credentials. If the correct username and password are entered within three attempts, the user is granted access to the main menu. Otherwise, the system exits. Once logged in, the user can repeatedly access banking operations until they choose to exit.

### Main Components of the System

1. **Login System**
  - Inputs: Username and password
  - Function: Validates credentials and limits login attempts to three
  - Output: Grants or denies access
2. **Banking Menu**
  - Inputs: User's menu selection (1–4)
  - Function: Navigates the user to different banking operations
  - Output: Balance information, deposit confirmation, withdrawal confirmation, or exit
3. **Deposit/Withdraw Functions**
  - Inputs: Amount to deposit or withdraw
  - Function: Validates amount, updates balance
  - Output: Updated balance or error messages

### Data and Structures

- **Variables:**
  - `USERNAME` and `PASSWORD`: Strings storing login credentials
  - `balance`: Float representing the current amount in the account
  - `login_attempts`, `max_attempts`: Integers to control login limits
  - `logged_in`: Boolean to track login status
- **Data Types Used:**
  - Strings, floats, integers, booleans
- **Control Structures:**
  - While loops for login and menu repetition
  - If-elif-else statements for menu choices and validations
  - Try-except blocks for input error handling

## Algorithms and Logic

Here's a simplified version of the login and menu logic as pseudo-code:

```

Set USERNAME and PASSWORD
Set balance = 1000
Set max_attempts = 3

While attempts < max_attempts:
    Ask for username and password
    If correct:
        Login successful, proceed to menu
    Else:
        Increase attempts

If login failed:
    Exit program

While True:
    Show menu
    If choice is 1:
        Show balance
    If choice is 2:
        Ask for deposit amount, update balance
    If choice is 3:
        Ask for withdrawal amount, update balance if valid
    If choice is 4:
        Exit
    Else:
        Show invalid input message

```

## User Interface and Input/Output

Users enter the entire interface with a text interface. They simply type responses after words. Operations conducted through this interface might include login information and numeric figures needed for banking business. Like many systems in operation today, when we want to inform your screen of something, the command screen must be left.

## The Library Used

Not needing any external libraries, the script uses only those available in Python: input, print, and standard data types.

## Screenshots and Errors

When developing, a frequent problem I ran into was the need for error checking on input of numbers in string fields. If the user enters a word instead of a number for their deposit or withdrawal directly into the program, then it falls over. To get around this, I used 'try except blocks' to catch when ValueError happens and supply a friendly message.

## Testing and Evaluation



To ensure the program worked as expected, I tested it with different scenarios:

- Logging in with correct and incorrect credentials
- Entering more than three incorrect login attempts
- Depositing a valid amount, a negative amount, and a non-numeric input
- Withdrawing a valid amount, too much money, zero, or negative amounts
- Typing invalid menu choices (e.g., "5", letters, or blank input)

Each part of the system responded correctly based on the input. The balance updated accurately, and the program handled errors gracefully without crashing. I also asked a classmate to try it out to make sure it worked for someone else too.

## Meeting the Objectives

The original objectives were:

1. Build a secure login system –  Achieved with a 3-attempt limit.
2. Allow users to check, deposit, and withdraw money –  All functions work as planned.

Overall, I'm happy with the final result. The project helped me apply everything I've learned so far in Python.

1<sup>st</sup> Marker initials \_\_\_\_\_ 2<sup>nd</sup> Marker initials \_\_\_\_\_ Date \_\_\_\_\_