

# Python OOP Mini-Project

## Banking System Case Study

Estimated Time: 3-5 hours



We regularly interact with banks in our day-to-day lives, but rarely think about the systems that power them. Banks power global finance, and the online systems that enable this are incredibly complex.

**In this project, you will create a simpler version of a banking system.**

You'll design the banking system using Python OOP techniques. In your bank model, you'll want to have entities for **customers**, **accounts**, **employees**, and **services** like loans and credit cards. Each of these entities will have distinct properties and **you'll need to decide what the relevant properties are for each entity**.

Customer records could have a customer's first name, last name, and address, while accounts could have account\_type (checking/savings), deposit, withdrawal, and balance. You'll also need to create getter and setter methods when necessary.

**Please make your own decisions around what type of data storage you'll want to use for the project**, whether that means a SQL database, CSV, JSON file, or another format (RECOMMENDED). You can also choose to keep the data in-memory - in which case, you might lose information if the program throws an error (NOT RECOMMENDED).

The final project should be structured in the manner outlined [here](#). **Your project should be executable via the command line**. It should accept user inputs and return results from the command line.

You should design the code in an OOP fashion. Here's an example:

```
#!/usr/bin/env python
class BankAccount:
    def __init__(self):
        self.balance = 0

    def withdraw(self, amount):
        self.balance -= amount
        return self.balance

    def deposit(self, amount):
        self.balance += amount
        return self.balance
```

A successful project will follow these requirements:

- Code follows [PEP-8 style](#)
- Code is readable and well documented with in-line and docstring comments for functions and classes.
- Logging is implemented to print messages on screen and to log errors and warnings to a file, which will be written to the logs folder in your project structure.
- [Exception handling](#) is implemented to capture errors gracefully.
- There is an [Markdown](#) (.md) file in your submission explaining how the program works, called README.md

Deliverables:

1. [UML Diagram](#) of the classes
2. Push code (in the project structure as outlined) to GitHub with the Readme Markdown file explaining the design and working of the program.