To achieve this, we need to write a Python program that:

1. Defines a `BankAccount` class representing a savings account with balance and methods for deposits and withdrawals.

2. Creates 100 bank accounts with random initial balances.

3. Simulates transactions over a given number of months.

4. Outputs the final balance for each account and sorts them from lowest to highest balance.

Here's a step-by-step guide along with the code:

Step 1: Define the `BankAccount` class

```python

import random

class BankAccount:

    def \_\_init\_\_(self, account\_id, balance=0):

        self.account\_id = account\_id

        self.balance = balance

    def deposit(self, amount):

        if amount > 0:

            self.balance += amount

    def withdraw(self, amount):

        if amount > 0 and self.balance >= amount:

            self.balance -= amount

    def \_\_repr\_\_(self):

        return f"Account ID: {self.account\_id}, Balance: ${self.balance:.2f}"

```

### Step 2: Generate and simulate transactions for 100 accounts

```python

def create\_accounts(num\_accounts, seed):

    random.seed(seed)

    accounts = []

    for i in range(num\_accounts):

        initial\_balance = random.uniform(100.0, 10000.0)

        account = BankAccount(account\_id=i + 1, balance=initial\_balance)

        accounts.append(account)

    return accounts

def simulate\_transactions(accounts, months, transactions\_per\_month):

    for account in accounts:

        for \_ in range(months \* transactions\_per\_month):

            transaction\_type = random.choice(['deposit', 'withdraw'])

            amount = random.uniform(1.0, 500.0)

            if transaction\_type == 'deposit':

                account.deposit(amount)

            else:

                account.withdraw(amount)

def print\_sorted\_accounts(accounts):

    sorted\_accounts = sorted(accounts, key=lambda acc: acc.balance)

    for account in sorted\_accounts:

        print(account)

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

    num\_accounts = 100

    seed = 42

    months = 6

    transactions\_per\_month = 10

    accounts = create\_accounts(num\_accounts, seed)

    simulate\_transactions(accounts, months, transactions\_per\_month)

    print\_sorted\_accounts(accounts)

```

### Explanation

1. \*\*BankAccount Class\*\*: Defines methods for deposits and withdrawals and keeps track of the balance.

2. \*\*create\_accounts Function\*\*: Generates a list of `BankAccount` objects with random initial balances.

3. \*\*simulate\_transactions Function\*\*: Simulates a number of transactions (both deposits and withdrawals) for each account over a given number of months.

4. \*\*print\_sorted\_accounts Function\*\*: Sorts the accounts by their final balance and prints them.

### Output

When you run the above code, you will get a list of 100 bank accounts sorted by their final balance, from the lowest to the highest balance.

Here’s a sample output format:

```

Account ID: 3, Balance: $112.34

Account ID: 87, Balance: $223.45

...

Account ID: 16, Balance: $9876.54

```

### Report Creation

To create a report, you can save the output to a file:

```python

def save\_report(accounts, filename="bank\_accounts\_report.txt"):

    with open(filename, 'w') as file:

        sorted\_accounts = sorted(accounts, key=lambda acc: acc.balance)

        for account in sorted\_accounts:

            file.write(f"{account}\n")

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

    # Existing code

    save\_report(accounts)

This will generate a text file named `bank\_accounts\_report.txt` containing the sorted account balances.