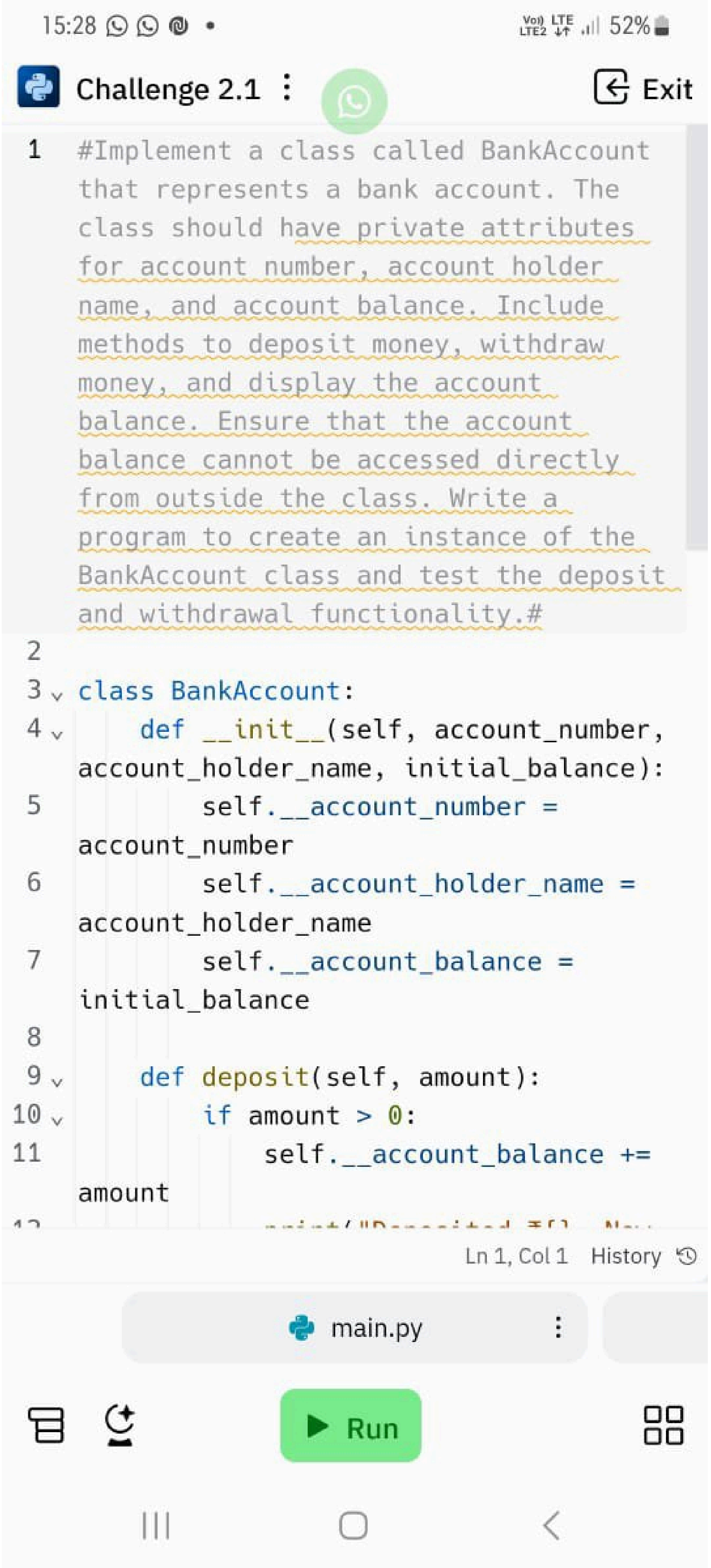
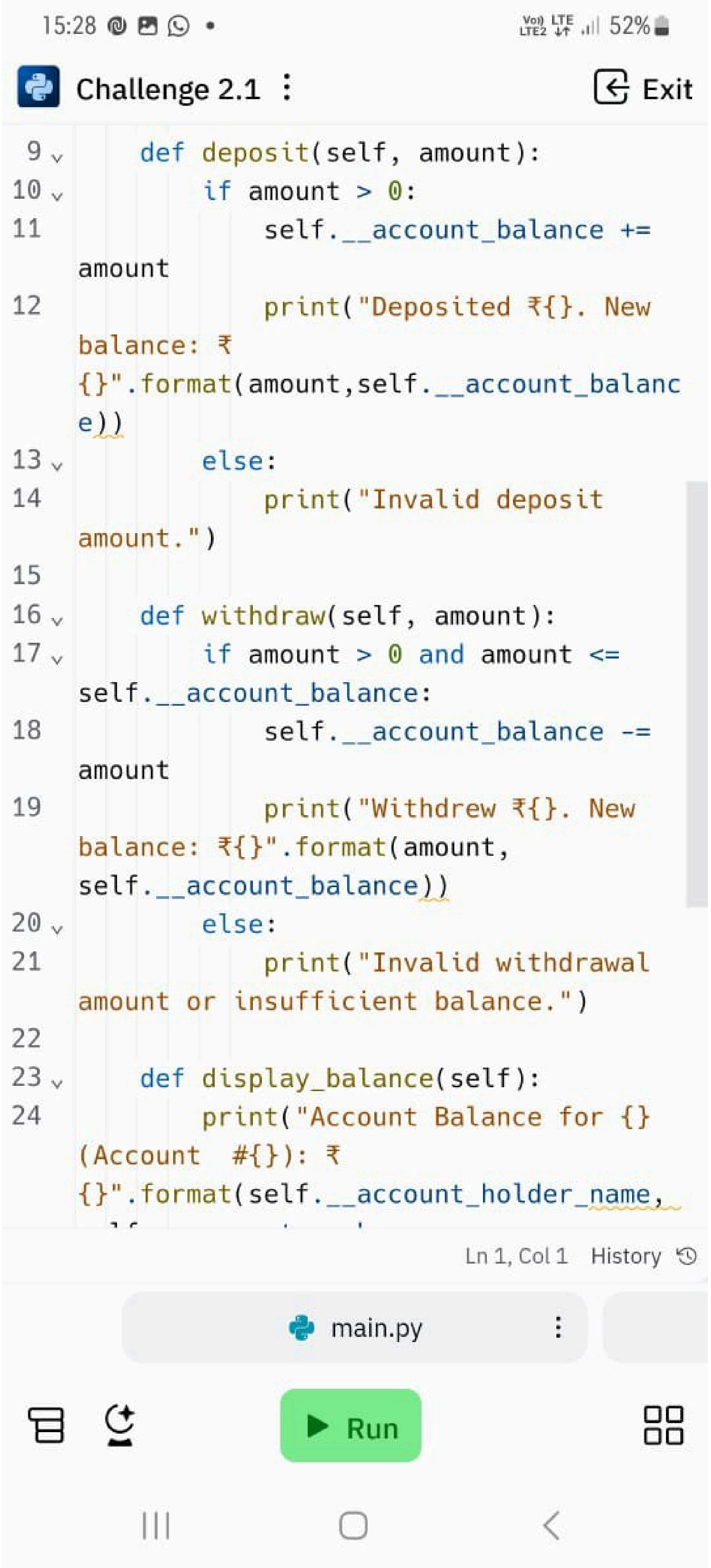
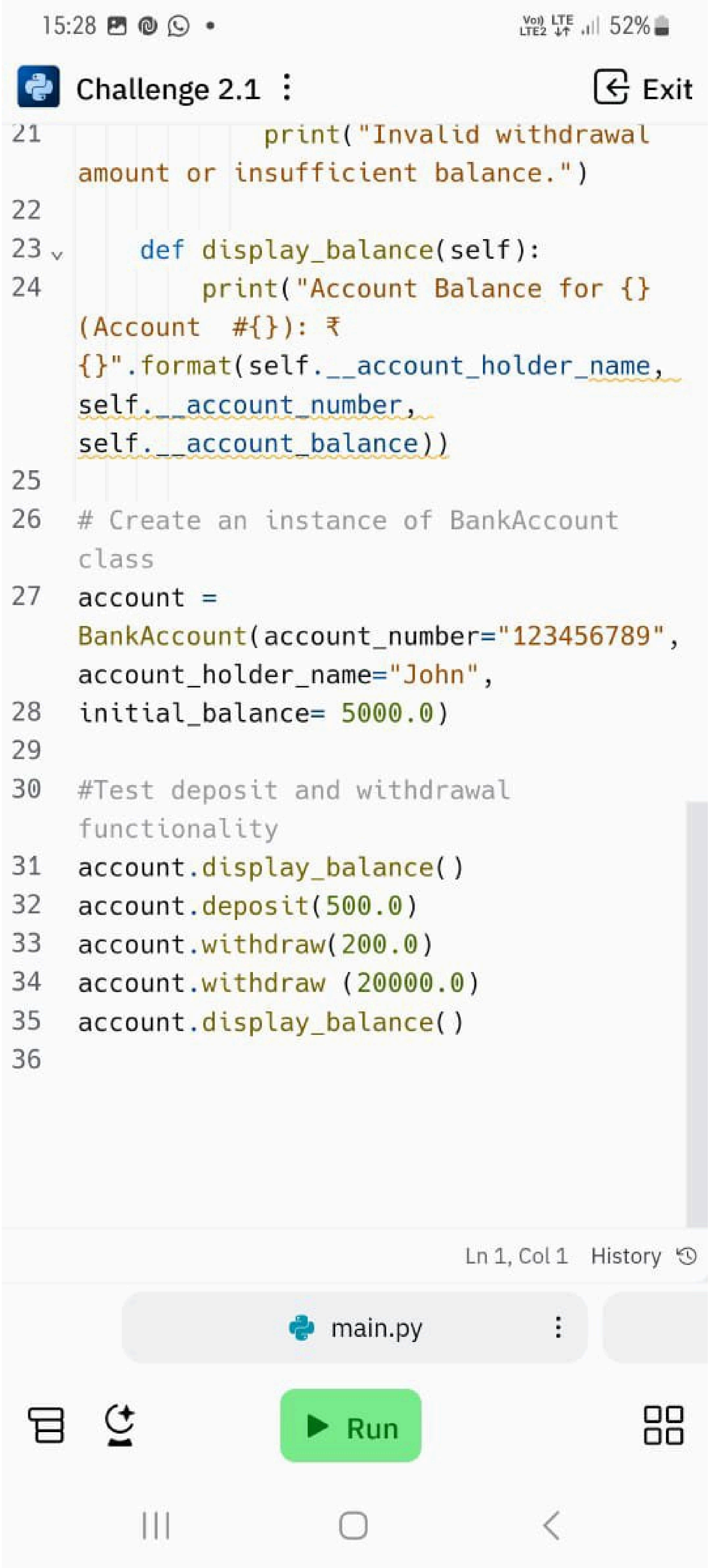


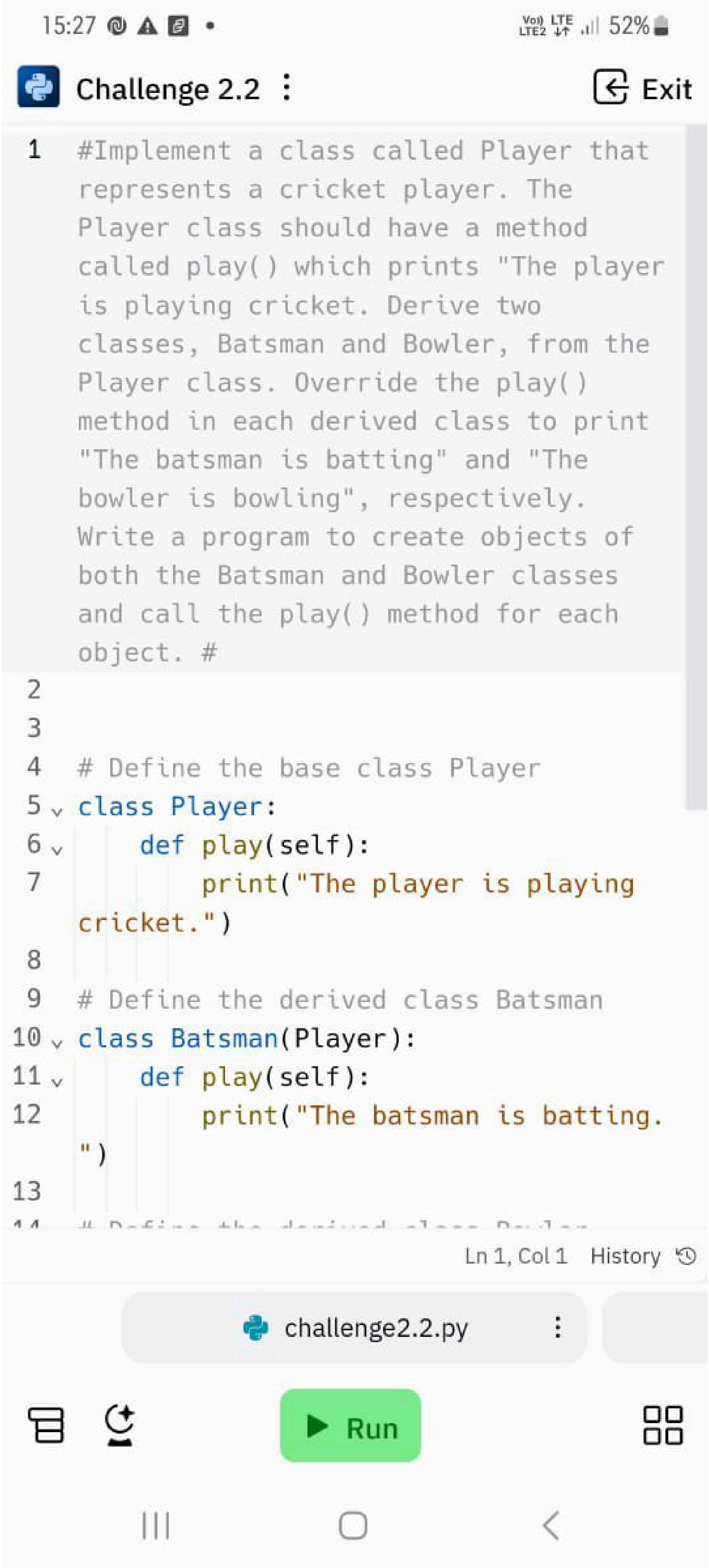
```python
#Implement a class called BankAccount that represents a bank account. The class should have private attributes for account number, account holder name, and account balance. Include methods to deposit money, withdraw money, and display the account balance. Ensure that the account balance cannot be accessed directly from outside the class. Write a program to create an instance of the BankAccount class and test the deposit and withdrawal functionality.#

class BankAccount:
    def __init__(self, account_number, account_holder_name, initial_balance):
        self.__account_number = account_number
        self.__account_holder_name = account_holder_name
        self.__account_balance = initial_balance

    def deposit(self, amount):
        if amount > 0:
            self.__account_balance += amount
```



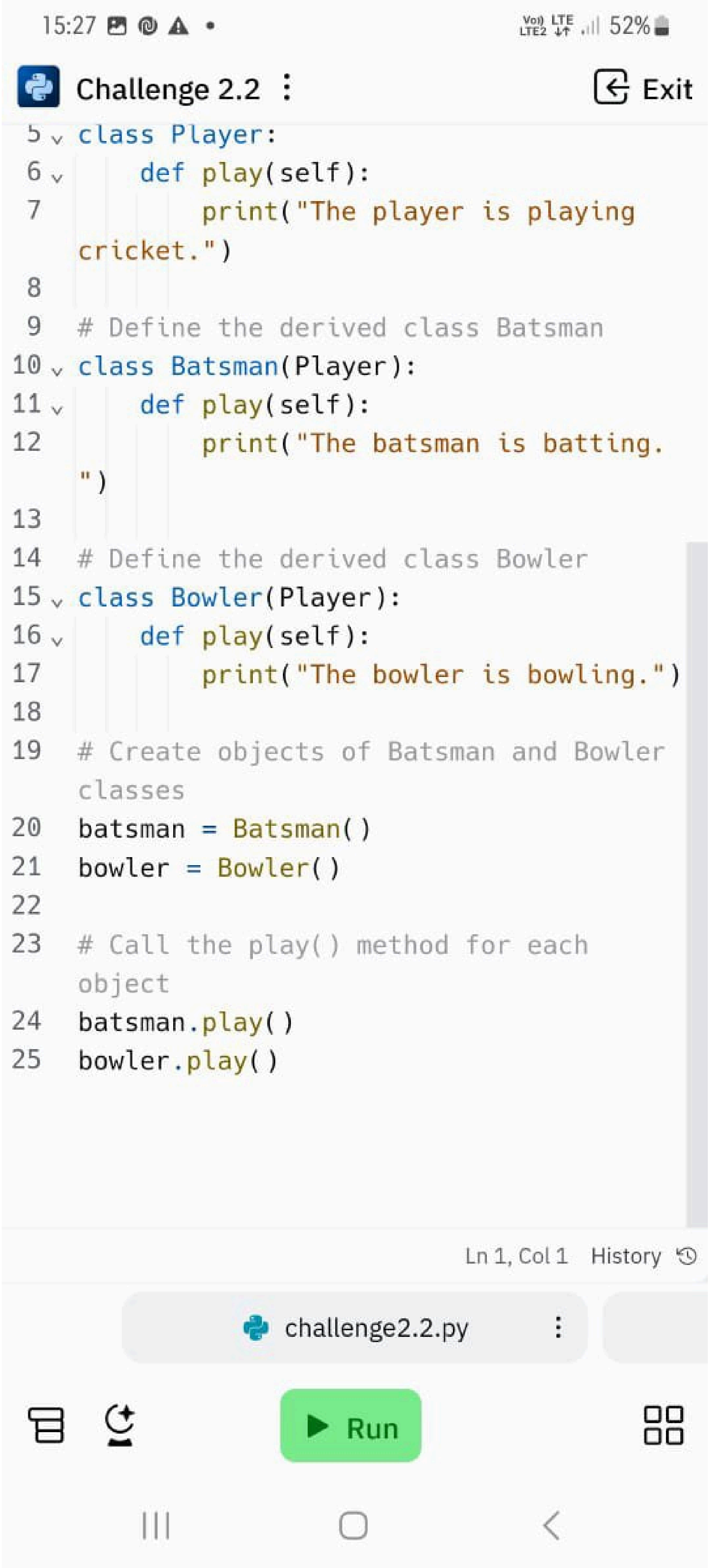
```python
    def deposit(self, amount):
        if amount > 0:
            self.__account_balance += amount
            print("Deposited ₹{}. New balance: ₹{}".format(amount,self.__account_balance))
        else:
            print("Invalid deposit amount.")

    def withdraw(self, amount):
        if amount > 0 and amount <= self.__account_balance:
            self.__account_balance -= amount
            print("Withdrew ₹{}. New balance: ₹{}".format(amount, self.__account_balance))
        else:
            print("Invalid withdrawal amount or insufficient balance.")

    def display_balance(self):
        print("Account Balance for {} (Account  #{}): ₹{}".format(self.__account_holder_name,
```



```python
            print("Invalid withdrawal amount or insufficient balance.")

    def display_balance(self):
        print("Account Balance for {} (Account  #{}): ₹ {}".format(self.__account_holder_name, self.__account_number, self.__account_balance))

# Create an instance of BankAccount class
account = BankAccount(account_number="123456789", account_holder_name="John",
initial_balance= 5000.0)

#Test deposit and withdrawal functionality
account.display_balance()
account.deposit(500.0)
account.withdraw(200.0)
account.withdraw (20000.0)
account.display_balance()

```



```python
#Implement a class called Player that represents a cricket player. The Player class should have a method called play() which prints "The player is playing cricket. Derive two classes, Batsman and Bowler, from the Player class. Override the play() method in each derived class to print "The batsman is batting" and "The bowler is bowling", respectively. Write a program to create objects of both the Batsman and Bowler classes and call the play() method for each object. #


# Define the base class Player
class Player:
    def play(self):
        print("The player is playing cricket.")

# Define the derived class Batsman
class Batsman(Player):
    def play(self):
        print("The batsman is batting.")

```



```python
class Player:
    def play(self):
        print("The player is playing cricket.")

# Define the derived class Batsman
class Batsman(Player):
    def play(self):
        print("The batsman is batting.")

# Define the derived class Bowler
class Bowler(Player):
    def play(self):
        print("The bowler is bowling.")

# Create objects of Batsman and Bowler classes
batsman = Batsman()
bowler = Bowler()

# Call the play() method for each object
batsman.play()
bowler.play()
```