



Challenge 2.2 :

Exit

```
6  def play(self):
7      print("The batsman is
    batting.")
8
9  class Bowler(Player):
10     def play(self):
11         print("The bowler is
            bowling.")
12
13     # Create objects of Batsman
    and Bowler classes
14     batsman = Batsman()
15     bowler = Bowler()
16
17     # Call the play() method for
    each object
18     batsman.play()
19     bowler.play()
20
```

Ln 1, Col 1 • Spaces: 2 History 🕒



main.py



Run





Challenge 2.2 :

Exit

```
1 v class Player:
2 v     def play(self):
3         print("The player is
   playing cricket.")
4
5 v class Batsman(Player):
6 v     def play(self):
7         print("The batsman is
   batting.")
8
9 v class Bowler(Player):
10 v     def play(self):
11         print("The bowler is
   bowling.")
12
13 # Create objects of Batsman
   and Bowler classes
14 batsman = Batsman()
15 bowler = Bowler()
16
17 # Call the play() method for
   each object
```

Ln 1, Col 1 • Spaces: 2 History ↻



main.py



Run





Challenge 2.1 :

Exit

```
25
26 # Create an instance of the
    BankAccount class
27 account =
    BankAccount(account_number="123
    456789",
28
    account_holder_name="Hari
    Prabu",
29
    initial_balance=5000.0)
30
31 # Test deposit and withdrawal
    functionality
32 account.display_balance()
33 account.deposit(500.0)
34 account.withdraw(200.0)
35 account.display_balance()
36
```

Ln 1, Col 1 • Spaces: 2 History 🕒



main.py



Run





Challenge 2.1 :

Exit

```
insufficient balance.")
20
21 def display_balance(self):
22     print("Account balance
for {} (Account #{}):
{}".format(self.__account_holder_name, self._account_number,
23
        self.__account_balance))
24
25
26 # Create an instance of the
BankAccount class
27 account =
BankAccount(account_number="123
456789",
28
account_holder_name="Hari
Prabu",
29
initial_balance=50000)
```

Ln 1, Col 1 • Spaces: 2 History 🕒



main.py



Run





Challenge 2.1 :

Exit

```
    {}. New balance:
    {}.format(amount,
self.__account_balance))
11  else:
12      print("Invalid
deposit amount. Please deposit
a positive amount.")
13
14  def withdraw(self, amount):
15      if amount > 0 and
amount <=
self.__account_balance:
16
self.__account_balance -=
amount # Corrected this line
17      print("Withdrew
{}. New balance:
{}.format(amount,
self.__account_balance))
18  else:
19      print("Invalid
withdrawal amount or
```

Ln 1, Col 1 • Spaces: 2 History 🕒



main.py



Run





Challenge 2.1 :

Exit

```
1 class BankAccount:
2     def __init__(self,
3         account_number,
4         account_holder_name,
5         initial_balance=0.0):
6         self.__account_number =
7         account_number
8
9         self.__account_holder_name =
10        account_holder_name
11
12        self.__account_balance =
13        initial_balance
14
15    def deposit(self, amount):
16        if amount > 0:
17
18            self.__account_balance +=
19            amount
20
21            print("Deposited
22            {}. New balance:
23            {}".format(amount,
```

Ln 1, Col 1 • Spaces: 2 History



main.py



Run

