



## Challenge 2.1 :



Exit

```
1 #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.#

2
3 class BankAccount:
4     def __init__(self, account_number,
5         account_holder_name, initial_balance):
6         self.__account_number =
7         account_number
8         self.__account_holder_name =
9         account_holder_name
10        self.__account_balance =
11        initial_balance
12
13    def deposit(self, amount):
14        if amount > 0:
15            self.__account_balance +=
16            amount
17
18    def withdraw(self, amount):
19        if amount > 0:
20            self.__account_balance -=
21            amount
22
23    def display(self):
24        print("Account Number:", self.__account_number,
25              "Account Holder Name:", self.__account_holder_name,
26              "Account Balance:", self.__account_balance)
```

Ln 1, Col 1 History



main.py



Run





## Challenge 2.1 :



Exit

```
9  def deposit(self, amount):
10     if amount > 0:
11         self.__account_balance +=
amount
12         print("Deposited ₹{}. New
balance: ₹
{}".format(amount, self.__account_balanc
e))
13     else:
14         print("Invalid deposit
amount.")
15
16     def withdraw(self, amount):
17         if amount > 0 and amount <=
self.__account_balance:
18             self.__account_balance -=
amount
19             print("Withdrew ₹{}. New
balance: ₹{}".format(amount,
self.__account_balance))
20         else:
21             print("Invalid withdrawal
amount or insufficient balance.")
22
23     def display_balance(self):
24         print("Account Balance for {}
(Account #{}): ₹
{}".format(self.__account_holder_name,
```

Ln 1, Col 1 History



main.py



Run





## Challenge 2.1 :



Exit

```
21         print("Invalid withdrawal
amount or insufficient balance.")
22
23     def display_balance(self):
24         print("Account Balance for {}
(Account #{}): ₹
{}".format(self.__account_holder_name,
self.__account_number,
self.__account_balance))
25
26 # Create an instance of BankAccount
class
27 account =
BankAccount(account_number="123456789",
account_holder_name="John",
28 initial_balance= 5000.0)
29
30 #Test deposit and withdrawal
functionality
31 account.display_balance()
32 account.deposit(500.0)
33 account.withdraw(200.0)
34 account.withdraw (20000.0)
35 account.display_balance()
36
```

Ln 1, Col 1 History



main.py



Run







## Challenge 2.2 :



Exit

1 #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. #

```
2
3
4 # Define the base class Player
5 class Player:
6     def play(self):
7         print("The player is playing
8             cricket.")
9
10 # Define the derived class Batsman
11 class Batsman(Player):
12     def play(self):
13         print("The batsman is batting.
14             ")
15
16 # Define the derived class Bowler
```

Ln 1, Col 1 History



challenge2.2.py



Run





## Challenge 2.2 :



Exit

```
5 ✓ class Player:
6 ✓     def play(self):
7         print("The player is playing
           cricket.")
8
9     # Define the derived class Batsman
10 ✓ class Batsman(Player):
11 ✓     def play(self):
12         print("The batsman is batting.
           ")
13
14     # Define the derived class Bowler
15 ✓ class Bowler(Player):
16 ✓     def play(self):
17         print("The bowler is bowling.")
18
19     # Create objects of Batsman and Bowler
           classes
20     batsman = Batsman()
21     bowler = Bowler()
22
23     # Call the play() method for each
           object
24     batsman.play()
25     bowler.play()
```

Ln 1, Col 1 History



challenge2.2.py



Run

