



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         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
16            self.__account_balance +=
17                amount
18
19            print("Deposited
20                {}. New balance:
21                {}".format(amount,
```

Ln 1, Col 1 • Spaces: 2 History 🕒



main.py



Run





Challenge 1.1 :

Exit

```
1
2 ✓ def fact_rec(n):
3 ✓     if n==0 or n==1:
4         return 1
5 ✓     else:
6         return n*fact_rec(n-1)
7
8 number=2
9 res=fact_rec(number)
10 print("The factorial of {} is
    {}".format(number,res))
```

Ln 10, Col 37 • 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 2.2 :

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 3.2 :

Exit

```
1 v class Student:
2 v     def __init__(self, name,
    roll_number, cgpa):
3         self.name = name
4         self.roll_number =
    roll_number
5         self.cgpa = cgpa
6
7 v def
    sort_students(student_list):
8     sorted_students =
    sorted(student_list,
    key=lambda student:
    student.cgpa, reverse=True)
9     return sorted_students
10
11 # Example usage
12 v students = [
13     Student("Alice", "A001",
    3.8),
14     Student("Bob", "A002",
    3.5).
```

Ln 1, Col 1 • Spaces: 2 History 🕒



main.py



Run





Challenge 3.1 :

Exit

```
10 # Example usage
11 products = ["Apple", "Banana",
12             "Orange", "Apple", "Orange"]
13
14 # Perform a linear search for
   the target product
15 result_indices =
   linear_search_product(products,
   target_product)
16
17 # Print the result
18 v if result_indices:
19     print(f"The product
   '{target_product}' was found
   at indices: {result_indices}")
20 v else:
21     print(f"The product
   '{target_product}' was not
   found.")
22
```

Ln 10, Col 16 • Spaces: 2 History 🕒



main.py



Run





Challenge 3.2 :

Exit

```
13     Student("Alice", "A001",  
14           3.8),  
15     Student("Bob", "A002",  
16           3.5),  
17     Student("Charlie", "A003",  
18           3.9),  
19     Student("David", "A004",  
20           3.7)  
21 ]  
22  
23 # Sort students based on CGPA  
24 # in descending order  
25 sorted_students =  
26     sort_students(students)  
27  
28 # Print the sorted list of  
29 students  
30 for student in sorted_students:  
31     print(f"Name:  
32           {student.name}, Roll Number:  
33           {student.roll_number}, CGPA:  
34           {student.cgpa}")
```

Ln 1, Col 1 • Spaces: 2 History 🕒



main.py



Run





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 3.1 :

Exit

```
1  ~
   linear_search_product(product_l
ist, target_product):
2  ~ indices = []
3
4  ~     for index, product in
enumerate(product_list):
5  ~         if product ==
target_product:
6
       indices.append(index)
7
8     return indices
9
10 # Example usage
11 products = ["Apple", "Banana",
"Orange", "Apple", "Orange"]
12 target_product = "Apple"
13
14 # Perform a linear search for
the target product
15 result indices =
```

Ln 10, Col 16 • Spaces: 2 History 🕒



main.py



Run





Challenge 3.2 :

Exit

```
1 class Student:
2     def __init__(self, name,
3         roll_number, cgpa):
4         self.name = name
5         self.roll_number =
6         roll_number
7         self.cgpa = cgpa
8
9     def
10         sort_students(student_list):
11         sorted_students =
12         sorted(student_list,
13             key=lambda student:
14                 student.cgpa, reverse=True)
15         return sorted_students
16
17 # Example usage
18 students = [
19     Student("Alice", "A001",
```



```
udent("Bob", "A002",
```

Ln 1, Col 1 • Spaces: 2 History 🕒



main.py





Challenge 1.2 :

Exit

```
1 ✓ def isLeapYear (year):  
2   if (year % 4 == 0 and year %  
   100 != 0) or year % 400 == 0:  
3       return True  
4   else:  
5       return False  
6   year = int(input("Enter a year  
   : "))  
7   if isLeapYear(year):  
8       print('{} is a leap  
   year.'.format(year))  
9   else:  
10      print('{} is not a leap  
   year.'.format(year))
```

Ln 1, Col 1 • Spaces: 2 History 🕒



main.py



Run

