Challenge2.1 Exit

```python
class BankAccount:
    def __init__(self, account_number, account_holder_name, initial_balance=0.0):
        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
            print("Deposited {}. New balance: {}".format(amount, self.___account_balance))
        else:
            print("Invalid deposit amount. Please deposit a positive amount.")

    def withdraw(self, amount):
        if amount > 0 and amount <= self.___account_balance:
```

Ln 1, Col 1 • Spaces: 2 History

main.py

Run

Packager 8s on 08:58:52, 10/25 ✓

```
--> poetry lock --no-update
Resolving dependencies...

Writing lock file
--> poetry install
Installing dependencies from lock file
```

Run 5s on 08:59:02, 10/25 ✓

```
enter the year your wish :4
4 is not leap year
```

Challenge 1.2 Exit

```python
num = int(input("enter the year your wish :"))

if (num % 4 == 0):

  if (num % 100 == 0):

   if (num % 400 == 0):

     print("%d is a leap year " % num)

  else:

   print("%d is not leap year" % num)

else:

 print("%d is not " % num)
```

Ln 1, Col 1 • Spaces: 2 History

main.py

Run

Challenge 1.1 Exit

Packager 13s on 08:57:18, 10/25

```
--> poetry lock --no-update
Resolving dependencies...

Writing lock file
--> poetry install
Installing dependencies from lock file
```

Run 73ms on 08:57:32, 10/25

```
factorial of  2 is  2.
```

Console

Run

Challenge 1.1 Exit

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

Ln 1, Col 1 • Spaces: 2 History

main.py

Run

Challenge 1.1 Exit

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

Ln 1, Col 1 • Spaces: 2 History

main.py

Run

Challange 3.1 Exit

```python
def linearSearchProduct(productlist, targetProduct):
    indices = []

    for index, product in enumerate(productlist):
        if product == targetProduct:
            indices.append(index)


    return indices


# Example usage:
products = ["shoes", "boot", "loafes", "shoes", "sandal", "shoes"]
target = "shoes"
result = linearSearchProduct(products, target)
print(result)

```

Ln 1, Col 1 • Spaces: 2 History

main.py

Run

```
    sorted(student_list, key=lambda
    student: student.cgpa, reverse=True)
 9      return sorted_students
10
11  # Example usage
12  students = [
13      Student("Alice", "A001", 3.8),
14      Student("Bob", "A002", 3.5),
15      Student("Charlie", "A003", 3.9),
16      Student("David", "A004", 3.7)
17  ]
18
19  # Sort students based on CGPA in
    descending order
20  sorted_students =
    sort_students(students)
21
22  # Print the sorted list of students
23  for student in sorted_students:
24      print(f"Name: {student.name},
    Roll Number: {student.roll_number},
    CGPA: {student.cgpa}")
25
```

Challenge 3.2 Exit

```python
class Student:
    def __init__(self, name, roll_number, cgpa):
        self.name = name
        self.roll_number = roll_number
        self.cgpa = cgpa

def sort_students(student_list):
    sorted_students = sorted(student_list, key=lambda student: student.cgpa, reverse=True)
    return sorted_students

# Example usage
students = [
    Student("Alice", "A001", 3.8),
    Student("Bob", "A002", 3.5),
    Student("Charlie", "A003", 3.9),
    Student("David", "A004", 3.7)
]

# Sort students based on CGPA in descending order
sorted_students = sort_students(students)
```

Ln 1, Col 1 • Spaces: 2 History

main.py

Run

Challenge 2.2 Exit

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

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

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()
```

Ln 1, Col 1 • Spaces: 2 History

main.py

Run

Challenge2.1 Exit

```
{}".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="123456789",
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()
```

Ln 1, Col 1 • Spaces: 2 History

main.py

Run

Challenge2.1 Exit

```
        if amount > 0 and amount <= self.___account_balance:
            self.___account_balance -= amount  # Corrected this line
            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, self._account_number,


self.___account_balance))


# Create an instance of the BankAccount class
account = BankAccount(account_number="12345678q"
```

Ln 1, Col 1 • Spaces: 2 History

main.py

Run

Challenge2.1 Exit

```python
class BankAccount:
    def __init__(self, account_number, account_holder_name, initial_balance=0.0):
        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
            print("Deposited {}. New balance: {}".format(amount, self.___account_balance))
        else:
            print("Invalid deposit amount. Please deposit a positive amount.")

    def withdraw(self, amount):
        if amount > 0 and amount <= self.___account_balance:
```

Ln 1, Col 1 • Spaces: 2 History

main.py

Run

Challenge 1.2 Exit

```python
num = int(input("enter the year your wish :"))

if (num % 4 == 0):

  if (num % 100 == 0):

   if (num % 400 == 0):

     print("%d is a leap year " % num)

  else:

   print("%d is not leap year" % num)

else:

 print("%d is not " % num)
```

Ln 1, Col 1 • Spaces: 2 History

main.py

Run

Challenge 1.1 Exit

```python
def fact_rec(n):
  if n==0 or n==1:
    return 1
  else:
     return n *fact_rec(n-1)
number = 2
res = fact_rec(number)
print ("factorial of  {} is
{}.".format (number,res))
```

Ln 1, Col 1 • Spaces: 2 History

main.py

Run