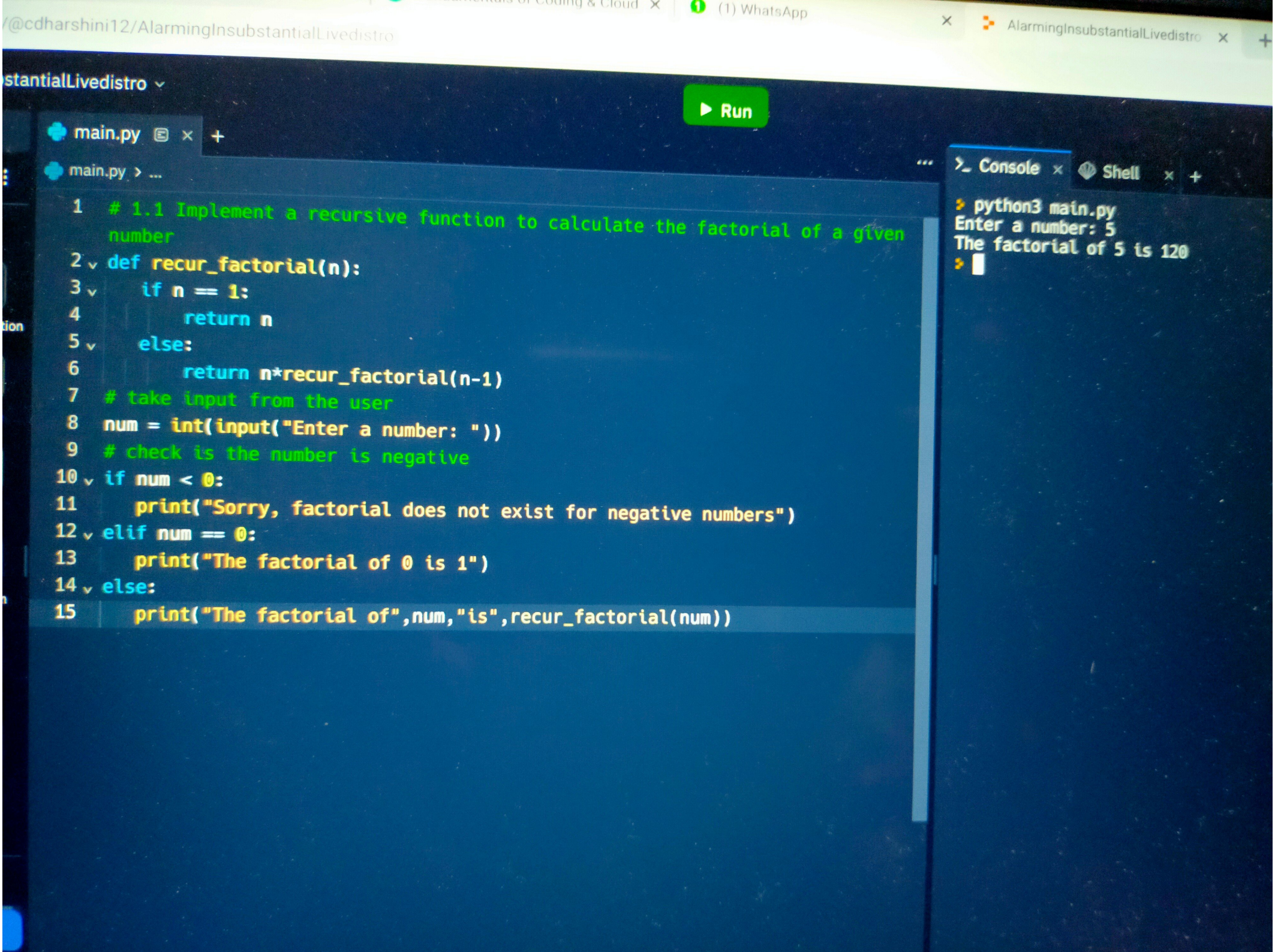
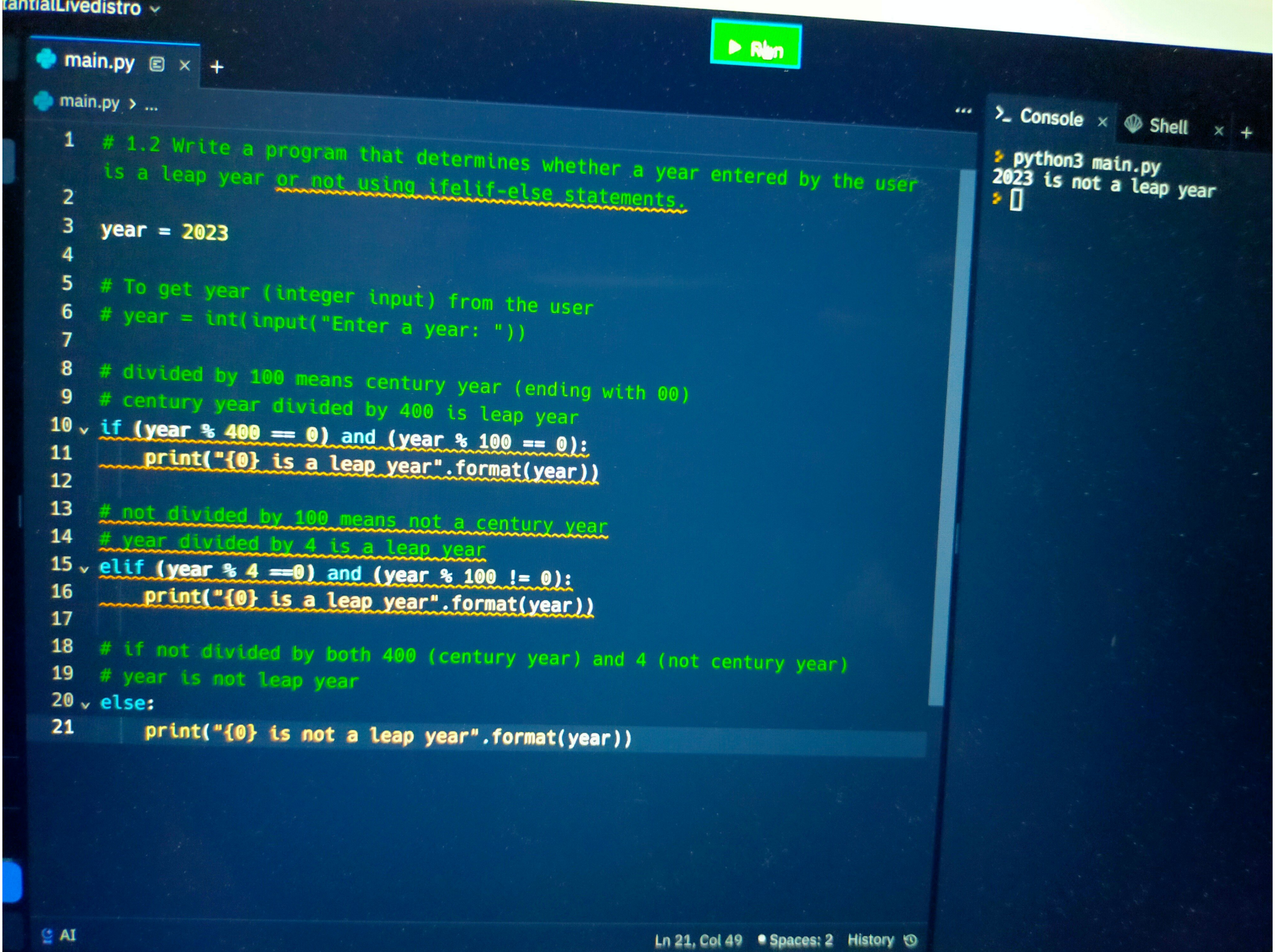

(1) WhatsApp
AlarmingInsubstantialLivedistro
Run
main.py
main.py > ...
# 1.1 Implement a recursive function to calculate the factorial of a given number
def recur_factorial(n):
if n == 1:
return n
else:
return n*recur_factorial(n-1)
# take input from the user
num = int(input("Enter a number: "))
# check is the number is negative
if num < 0:
print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
print("The factorial of 0 is 1")
else:
print("The factorial of",num,"is",recur_factorial(num))
Console
Shell
python3 main.py
Enter a number: 5
The factorial of 5 is 120


tantialLivedistro
main.py
Run
main.py > ...
# 1.2 Write a program that determines whether a year entered by the user is a leap year or not using ifelif-else statements.
year = 2023
# To get year (integer input) from the user
# year = int(input("Enter a year: "))
# divided by 100 means century year (ending with 00)
# century year divided by 400 is leap year
if (year % 400 == 0) and (year % 100 == 0):
print("{0} is a leap year".format(year))
# not divided by 100 means not a century year
# year divided by 4 is a leap year
elif (year % 4 ==0) and (year % 100 != 0):
print("{0} is a leap year".format(year))
# if not divided by both 400 (century year) and 4 (not century year)
# year is not leap year
else:
print("{0} is not a leap year".format(year))
Console
Shell
python3 main.py
2023 is not a leap year
AI
Ln 21, Col 49
Spaces: 2
History

main.py

Run

main.py > BankAccount > __init__ > ...

```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.")

    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, self.__account_number, self.__account_balance))
```

Ln 2, Col 7 • Spaces: 2 History

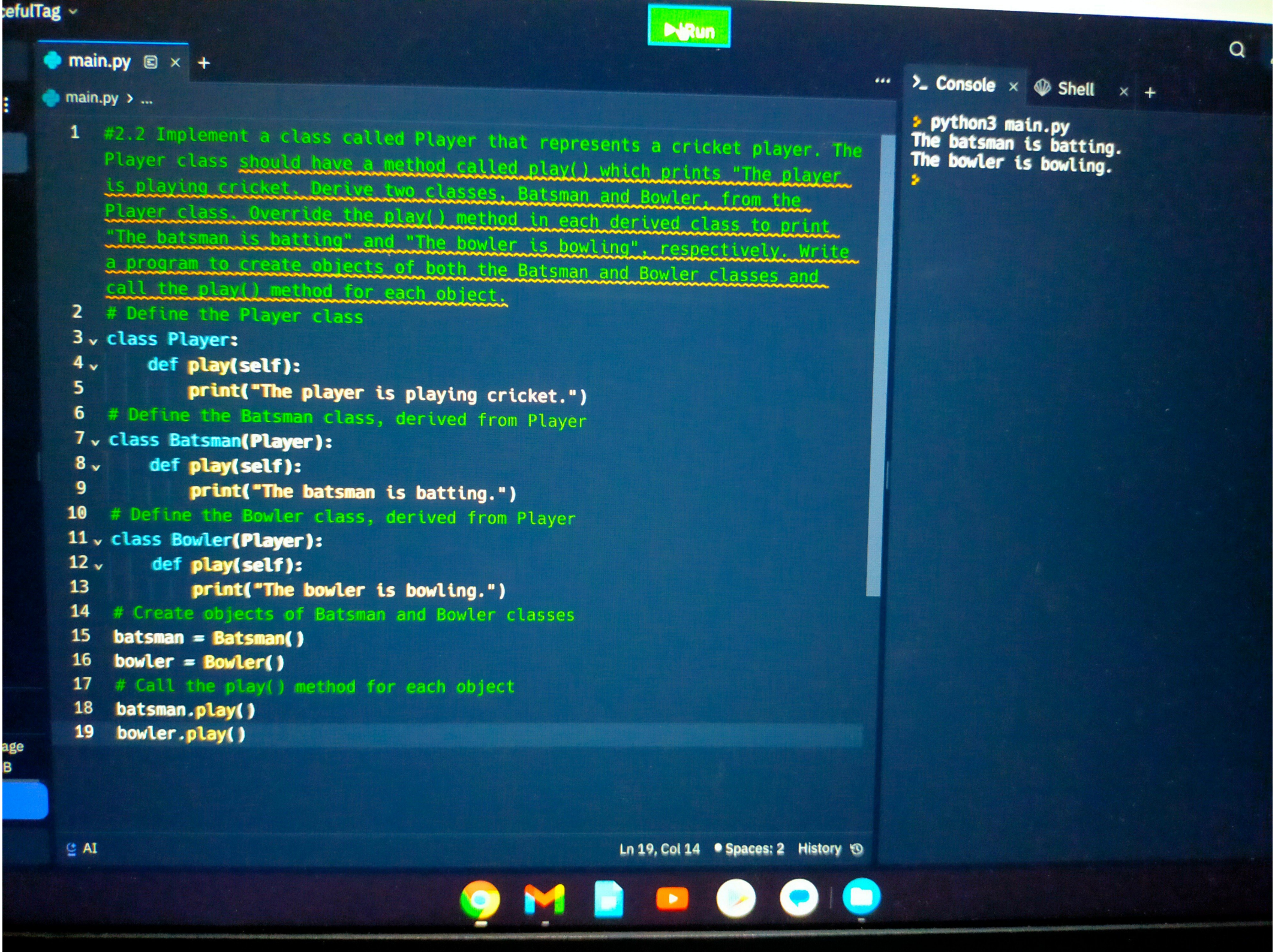
AI

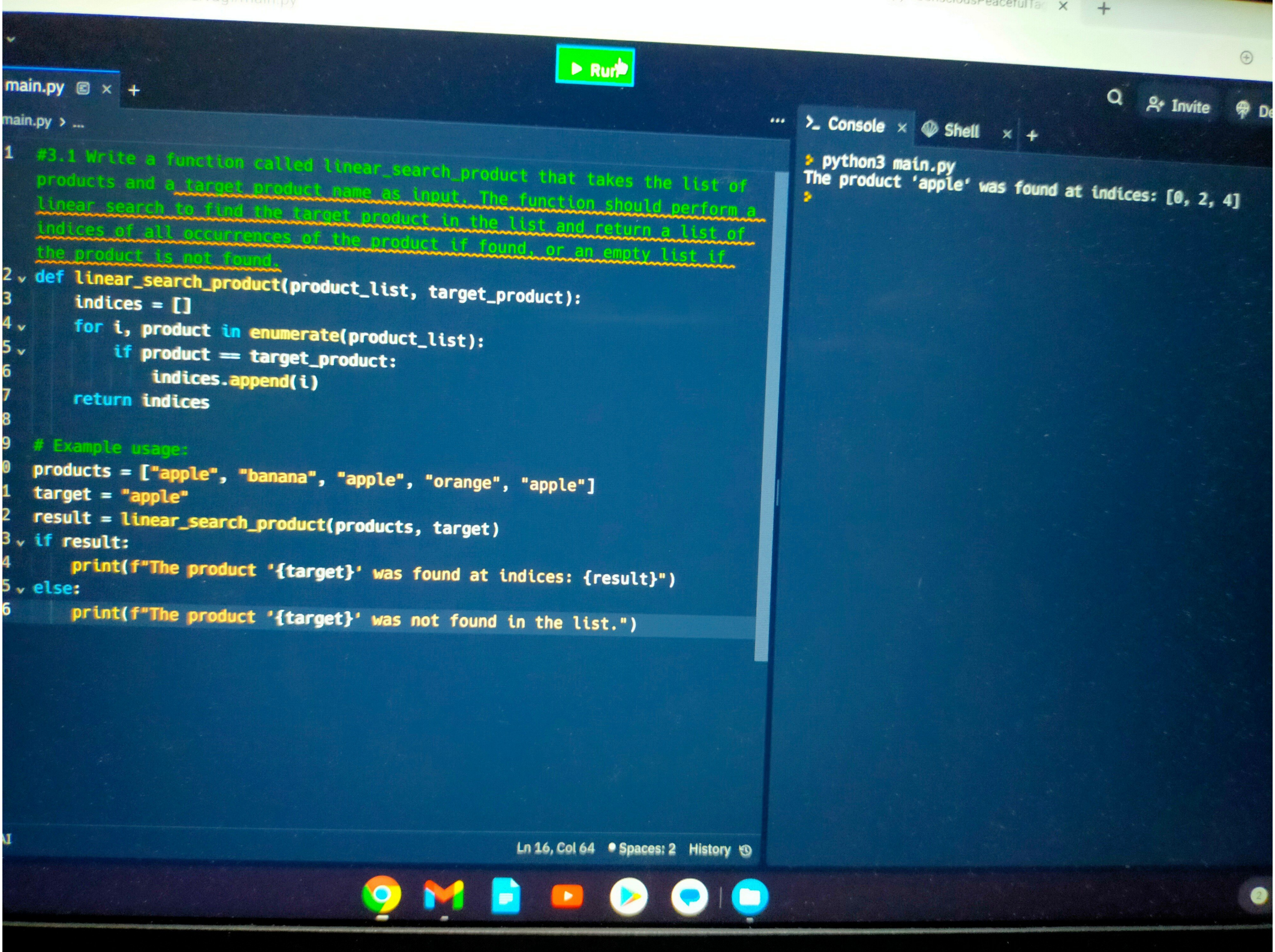
Console

Invite Deploy

```
> python3 main.py
Account balance for Hari Prabu (Account #123456789): ₹5000.0
Deposited ₹500.0. New balance: ₹5500.0
Withdrew ₹200.0. New balance: ₹5300.0
Invalid withdrawal amount or insufficient balance.
Account balance for Hari Prabu (Account #123456789): ₹5300.0
>
```

8:52


Run
main.py
main.py > ...
#2.2 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 Player class
class Player:
def play(self):
print("The player is playing cricket.")
# Define the Batsman class, derived from Player
class Batsman(Player):
def play(self):
print("The batsman is batting.")
# Define the Bowler class, derived from Player
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()
Console
Shell
python3 main.py
The batsman is batting.
The bowler is bowling.
AI
Ln 19, Col 14
Spaces: 2
History


Run
main.py
main.py > ...
#3.1 Write a function called linear_search_product that takes the list of products and a target product name as input. The function should perform a linear search to find the target product in the list and return a list of indices of all occurrences of the product if found, or an empty list if the product is not found.
def linear_search_product(product_list, target_product):
indices = []
for i, product in enumerate(product_list):
if product == target_product:
indices.append(i)
return indices
# Example usage:
products = ["apple", "banana", "apple", "orange", "apple"]
target = "apple"
result = linear_search_product(products, target)
if result:
print(f"The product '{target}' was found at indices: {result}")
else:
print(f"The product '{target}' was not found in the list.")
Ln 16, Col 64
Spaces: 2
History
Console
Shell
Invite
python3 main.py
The product 'apple' was found at indices: [0, 2, 4]

main.py

```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):
    # Sort the list of students in descending order of CGPA
    sorted_students = sorted(student_list,
                             key=lambda student: student.cgpa,
                             reverse=True)
    return sorted_students

# Example usage:
students = [
    Student("Hari", "A123", 7.8),
    Student("Srikanth", "A124", 8.9),
    Student("Saumya", "A125", 9.1),
    Student("Mahidhar", "A126", 9.9),
]

sorted_students = sort_students(students)

# Print the sorted list of students
for student in sorted_students:
    print("Name: {}, Roll Number: {}, CGPA: {}".format(student.name,
                                                       student.roll_number,
                                                       student.cgpa))
```

Ln 28, Col 66 • Spaces: 2 History

Console Shell

```
$ python3 main.py
Name: Mahidhar, Roll Number: A126, CGPA: 9.9
Name: Saumya, Roll Number: A125, CGPA: 9.1
Name: Srikanth, Roll Number: A124, CGPA: 8.9
Name: Hari, Roll Number: A123, CGPA: 7.8
$
```