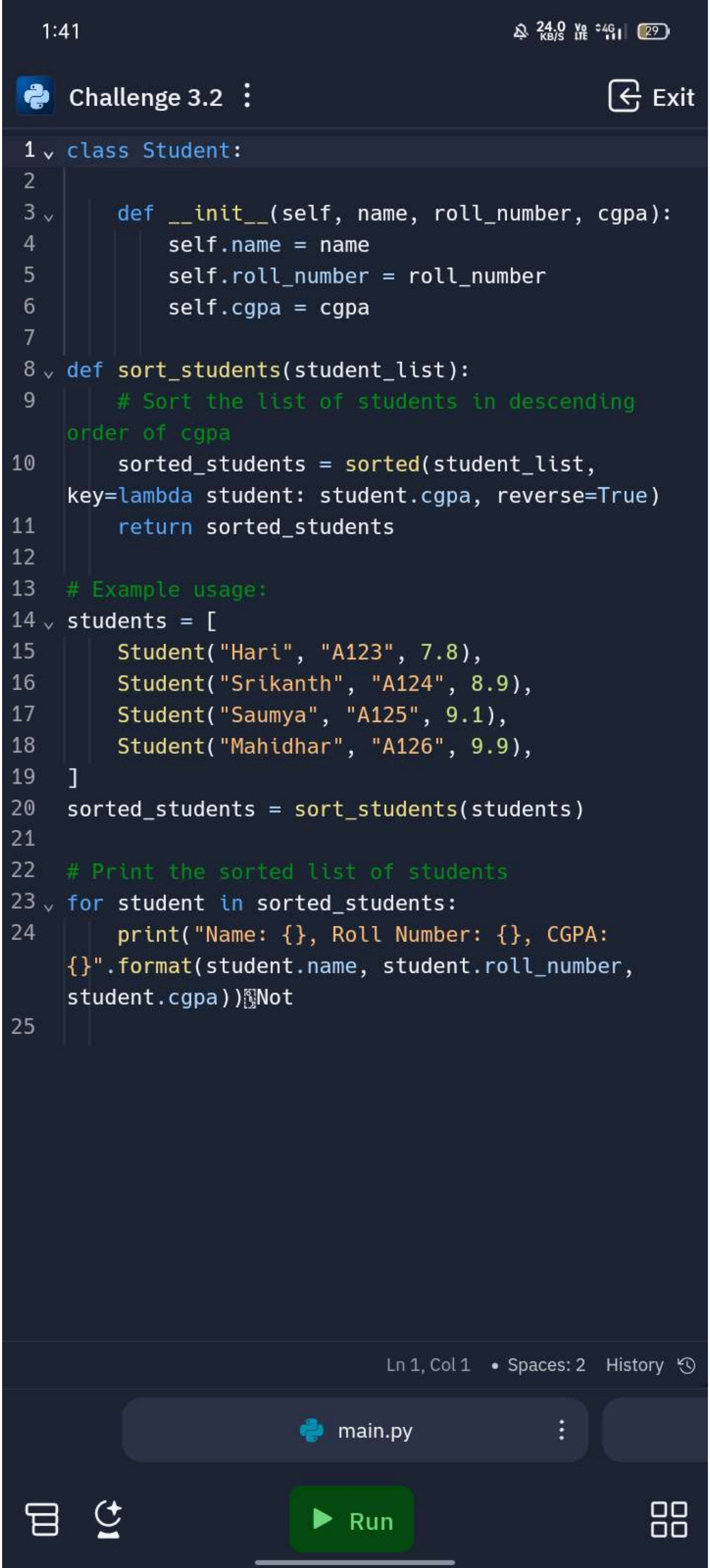
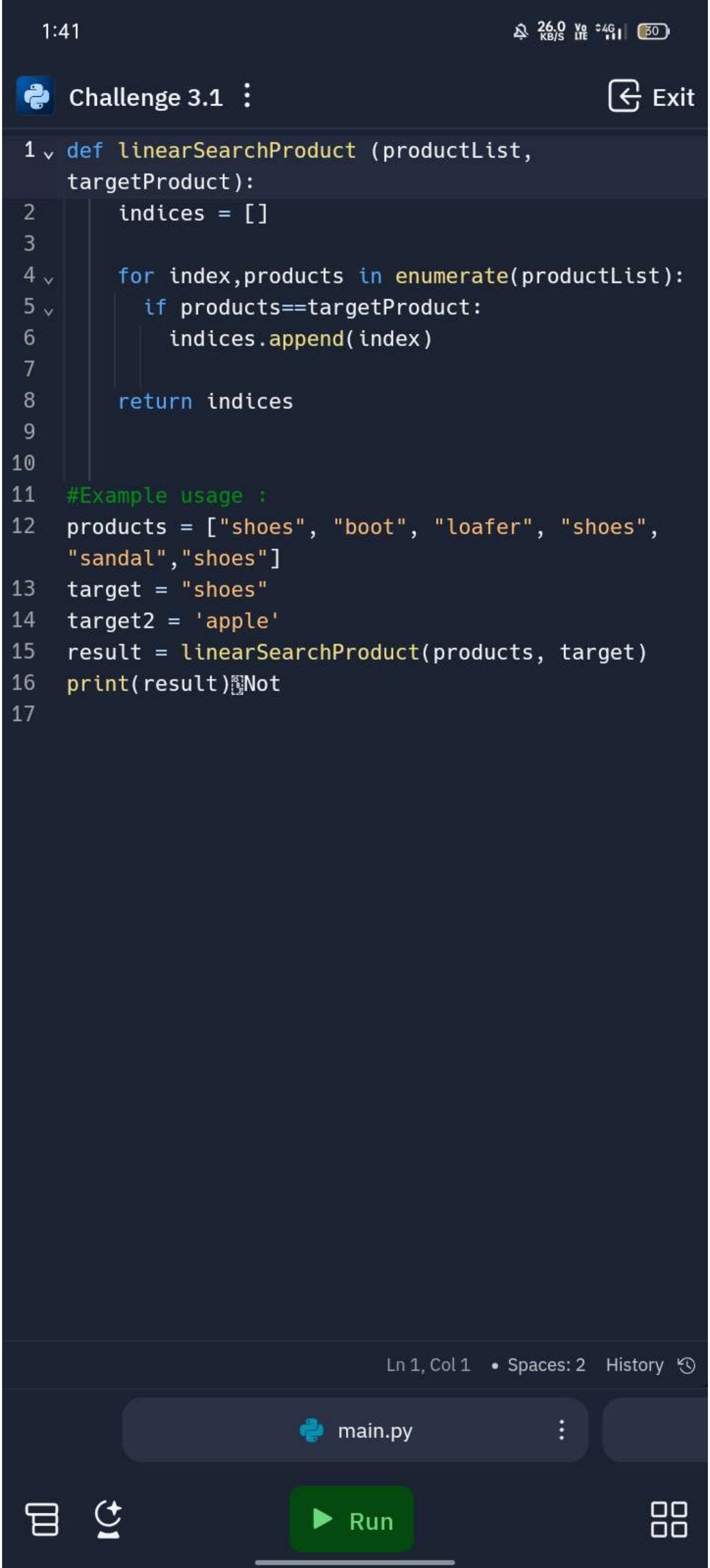
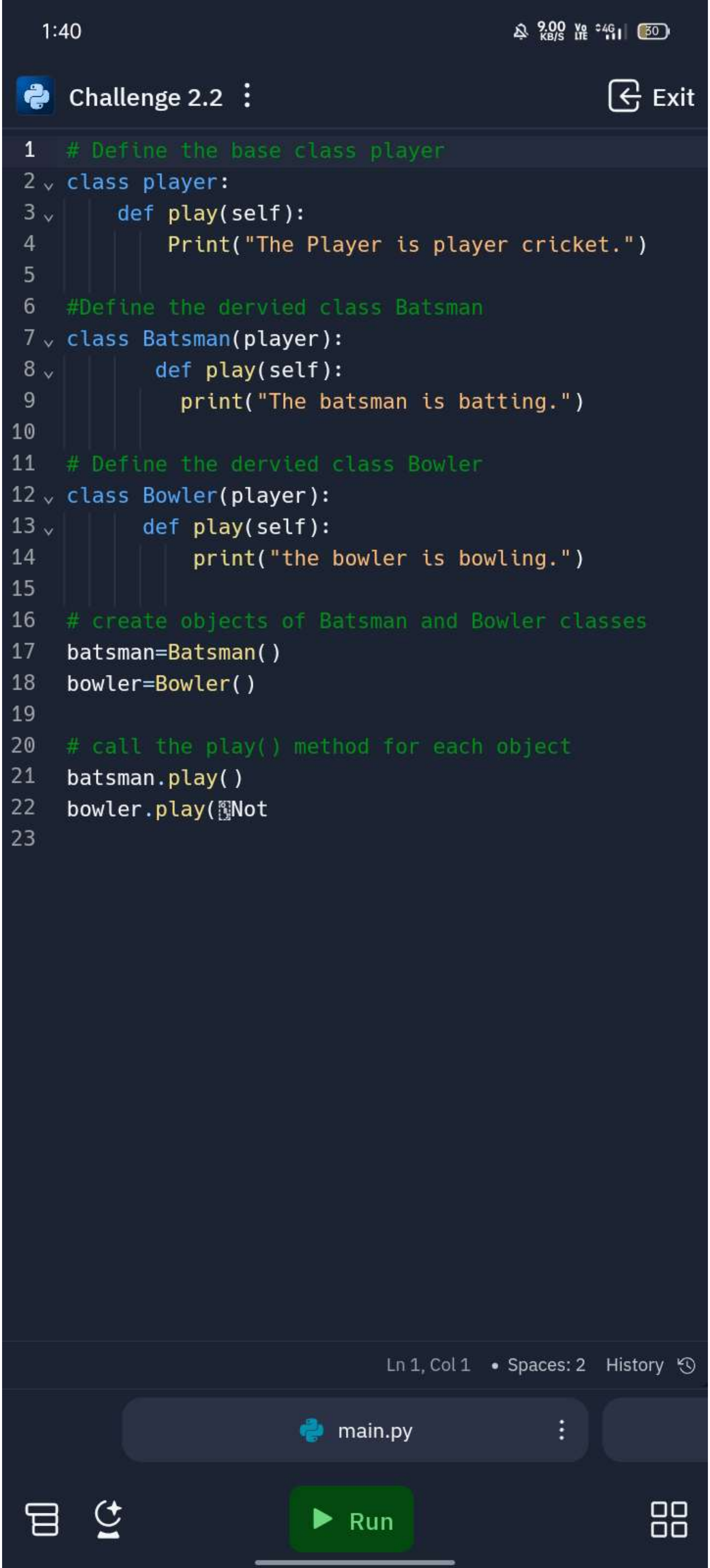


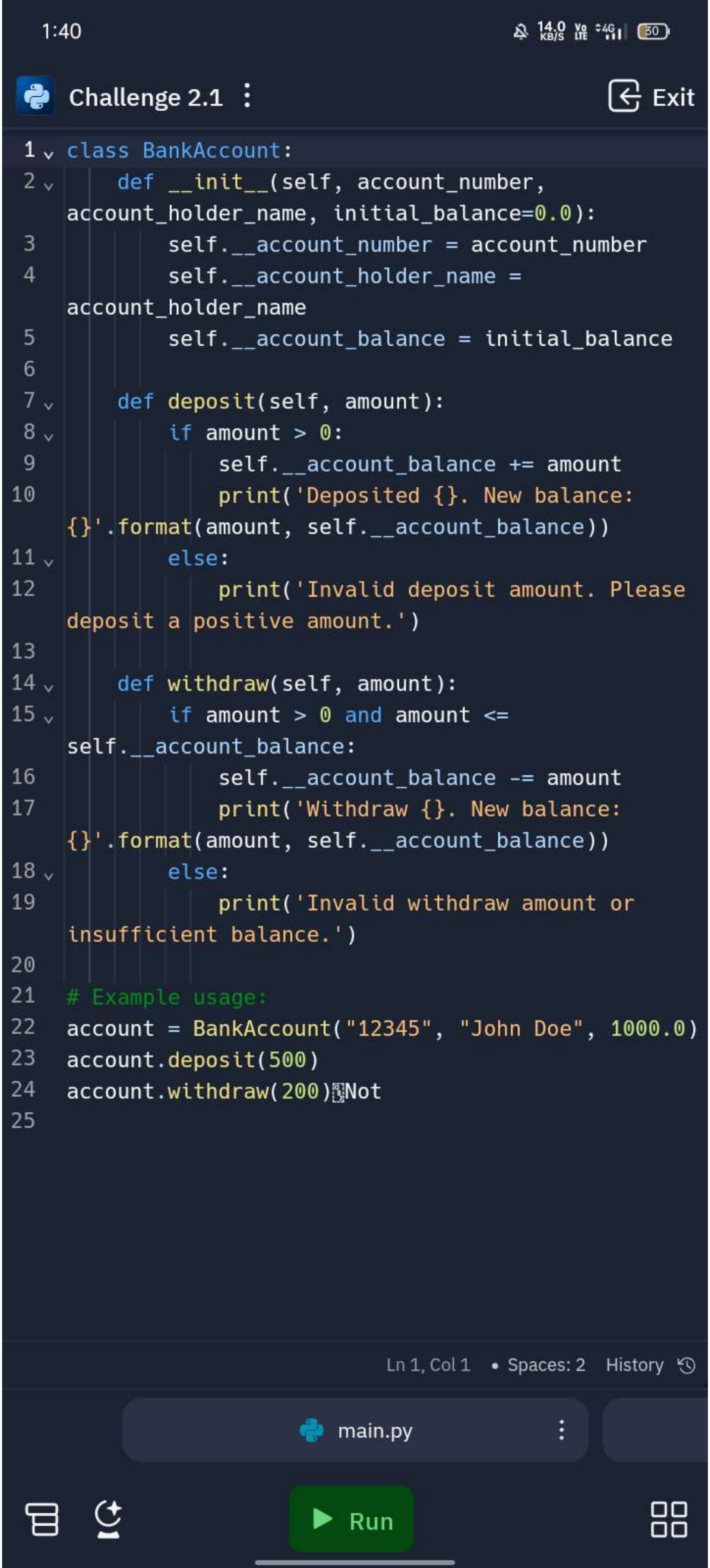
```python
class Student:

    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))Not

```



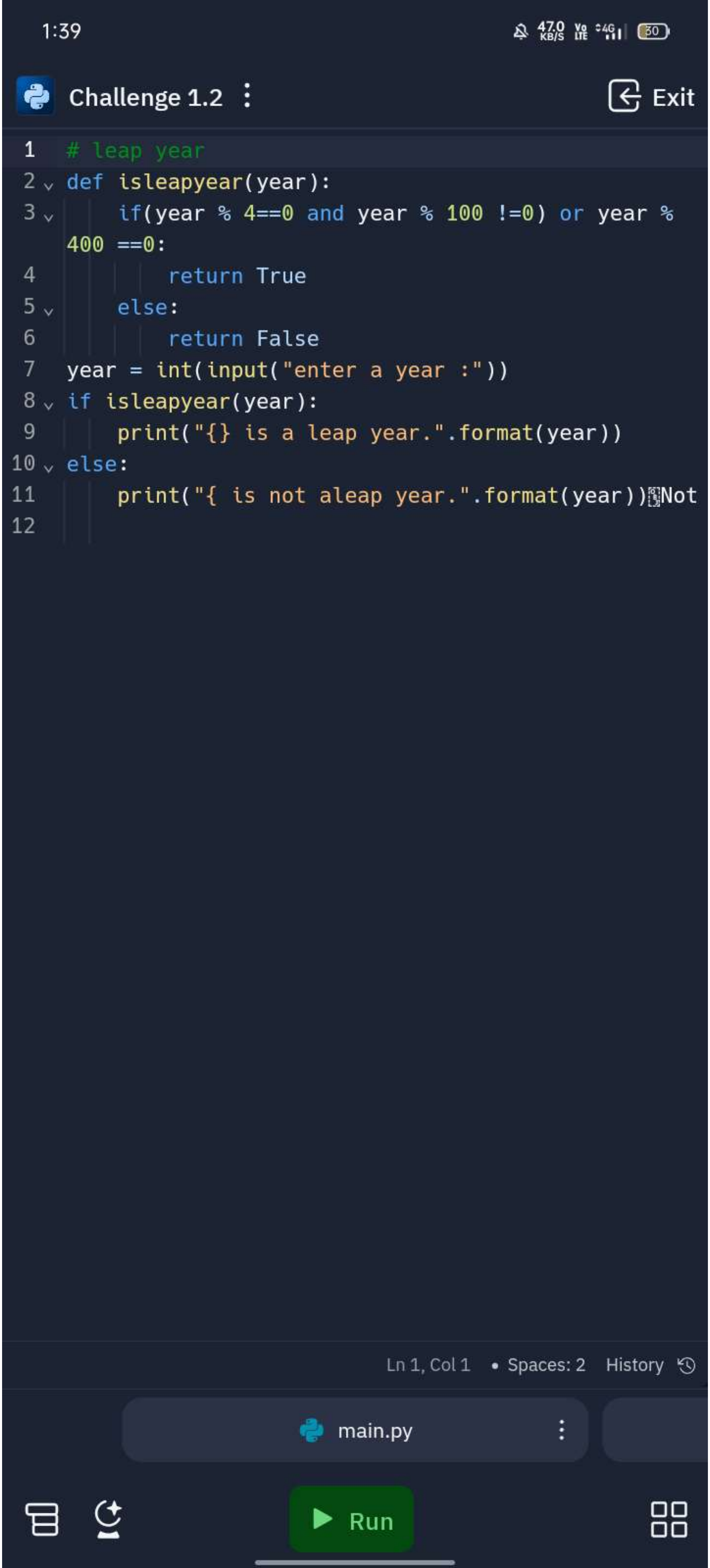
```python
def linearSearchProduct (productList, targetProduct):
    indices = []

    for index,products in enumerate(productList):
      if products==targetProduct:
        indices.append( index)

    return indices


#Example usage :
products = ["shoes", "boot", "loafer", "shoes", "sandal","shoes"]
target = "shoes"
target2 = 'apple'
result = linearSearchProduct(products, target)
print(result)▯Not

```



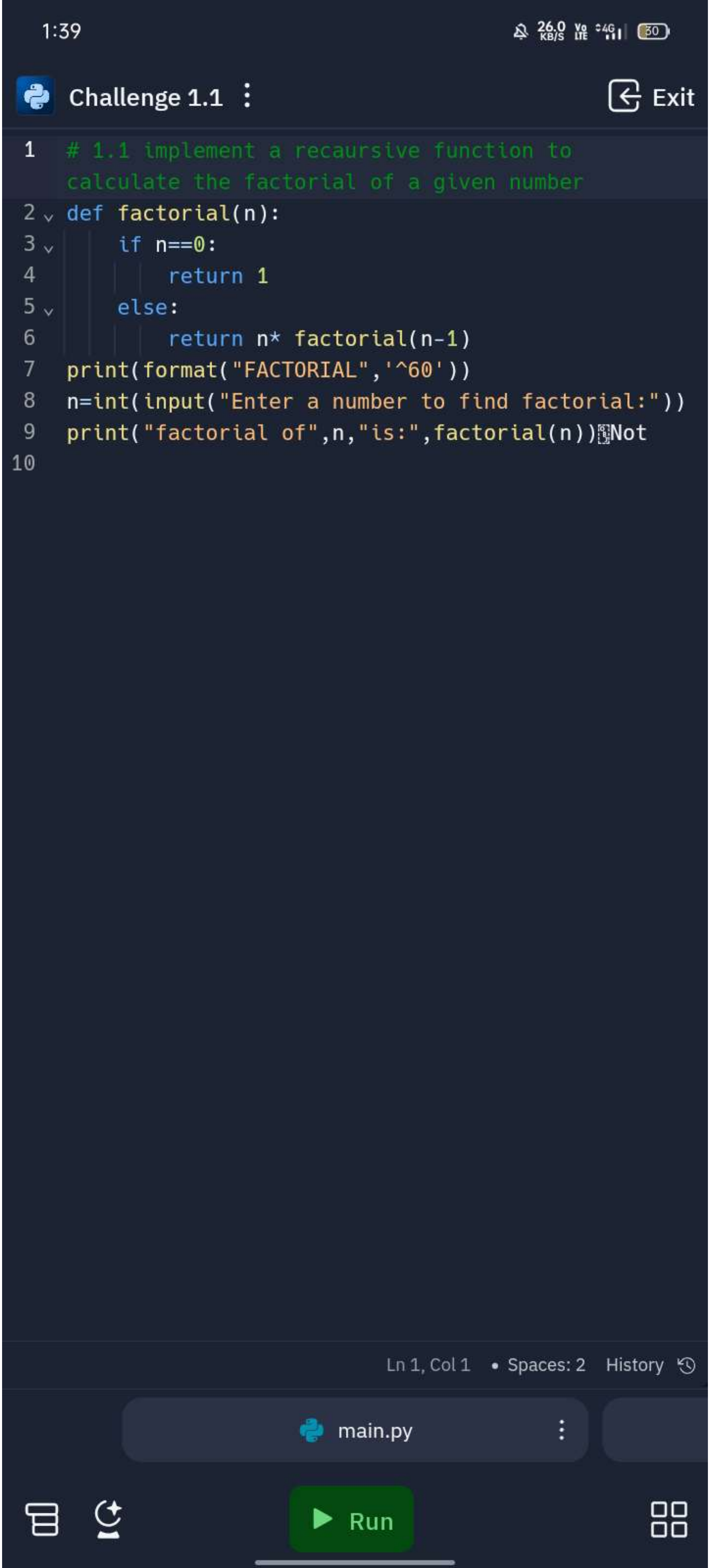
```python
# Define the base class player
class player:
    def play(self):
        Print("The Player is player cricket.")

#Define the dervied class Batsman
class Batsman(player):
       def play(self):
         print("The batsman is batting.")

# Define the dervied class Bowler
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(▯Not
```



```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:
            self.__account_balance -= amount
            print('Withdraw {}. New balance: {}'.format(amount, self.__account_balance))
        else:
            print('Invalid withdraw amount or insufficient balance.')

# Example usage:
account = BankAccount("12345", "John Doe", 1000.0)
account.deposit(500)
account.withdraw(200)▯Not

```



```python
# leap year
def isleapyear(year):
    if(year % 4==0 and year % 100 !=0) or year % 400 ==0:
        return True
    else:
        return False
year = int(input("enter a year :"))
if isleapyear(year):
    print("{} is a leap year.".format(year))
else:
    print("{ is not aleap year.".format(year))Not

```



```python
# 1.1 implement a recarsive function to calculate the factorial of a given number
def factorial(n):
    if n==0:
        return 1
    else:
        return n* factorial(n-1)
print(format("FACTORIAL",'^60'))
n=int(input("Enter a number to find factorial:"))
print("factorial of",n,"is:",factorial(n))Not
```