Challenge 1.1

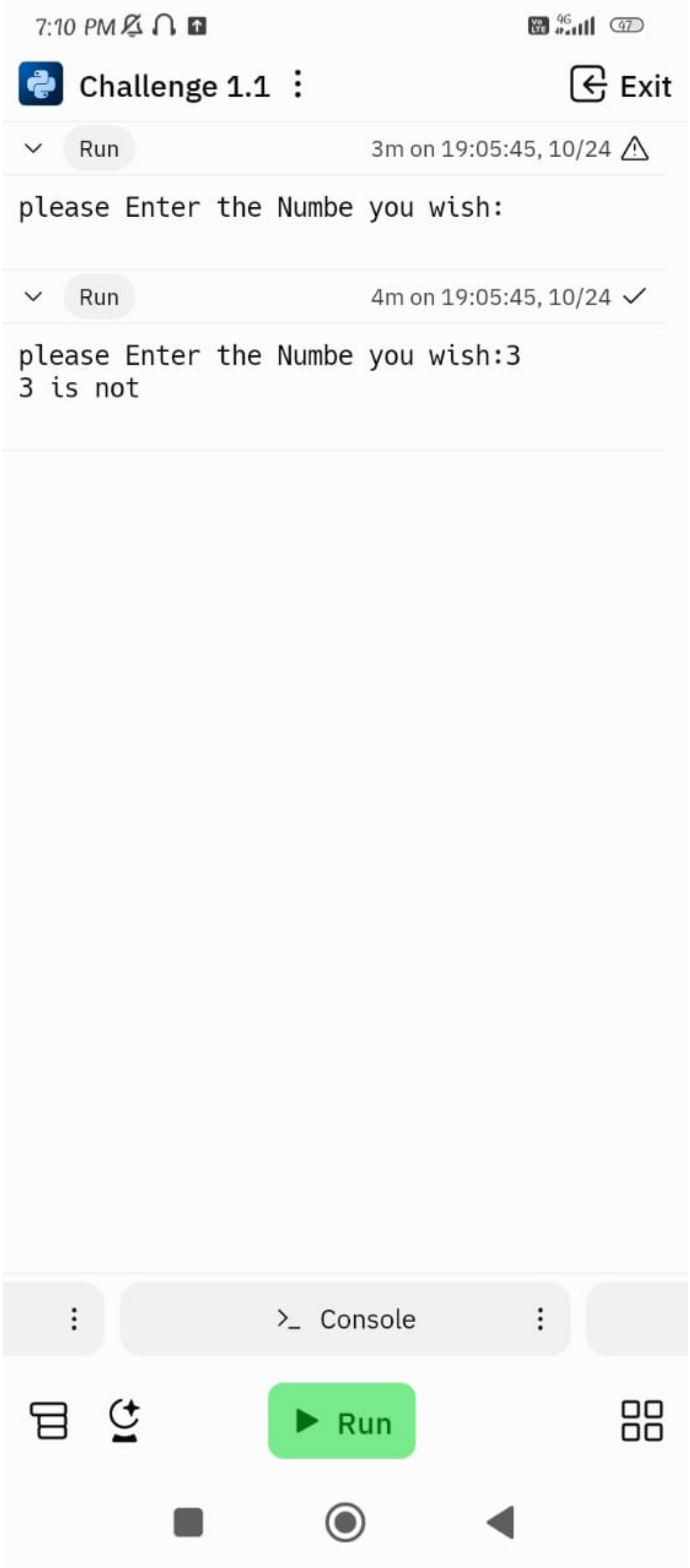
Exit

```
1  num=int(input("please Enter the
   Numbe you wish:"))
2  if (num%4==0):
3      if(num%100==0):
4          if (num%400==0):
5              print("%d is a leap year"%num)
6          else:
7              print("%d is not"%num)
8
9      else:
10         print("%d is a leap year"%num)
11 else:
12     print("%d is not"%num)
13
14
```

Ln 1, Col 1 • Spaces: 2 History

main.py

Run


7:10 PM
Challenge 1.1
Exit
Run
3m on 19:05:45, 10/24
please Enter the Numbe you wish:
Run
4m on 19:05:45, 10/24
please Enter the Numbe you wish:3
3 is not
Console
Run

7:11 PM

Challenge 1.2

Exit

```
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("the Factorial of {} is {} . " .
 format (number,res))

```

Ln 1, Col 1 • Spaces: 2 History

main.py

Run

Challenge 1.2

Exit

Run 293ms on 19:19:00, 10/24 ✓

```
the Factorial of 2 is 2 .
```

Console

Run

```
        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
17          print('Withdraw {}. New
        balance: {}'.format(amount,
        self.__account_balance))
18      else:
19          print('Invalid withdraw
        amount or insufficient balance.')
20
21  # Example usage:
22  account = BankAccount("12345",
    "John Doe", 1000.0)
23  account.deposit(500)
24  account.withdraw(200)
```


7:20 PM
Challenge 2.1
Exit
Run
160ms on 19:20:39, 10/24
Deposited 500. New balance: 1500.0
Withdraw 200. New balance: 1300.0
Console
Run

```
1  class player:
2    def play(self):
3      print("the player is playing
   cricket.")
4  class batsman(player):
5    def play(self):
6       print("The batsman  is
   batting.")
7  class bowler(player):
8    def play(self):
9      print("The bowler is bowler. ")
10 Batsman=batsman()
11 Bowler=bowler()
12 Batsman.play()
13 Bowler.play()
14
15
```

```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

students = [
    Student("Hari", "A123", 7.8),
    Student("Srikanth", "A124", 8.9),
    Student("Saunya", "A125", 9.1),
    Student("Mahidhar", "A126", 9.9),
]

sorted_students = sort_students(students)

```

Challenge 3.1 ⋮ Exit

```
6            self.cgpa = cgpa
7
8   def sort_students(student_list):
9       sorted_students =
    sorted(student_list, key=lambda
    student: student.cgpa, reverse=True)
10      return sorted_students
11
12  students = [
13      Student("Hari", "A123", 7.8),
14      Student("Srikanth", "A124",
    8.9),
15      Student("Saunya", "A125", 9.1),
16      Student("Mahidhar", "A126",
    9.9),
17  ]
18
19  sorted_students =
    sort_students(students)
20
21  for student in sorted_students:
22      print("Name: {}, Roll Number:
    {}, CGPA: {}".format(student.name,
    student.roll_number, student.cgpa))
```

Ln 1, Col 1 • Spaces: 2 History

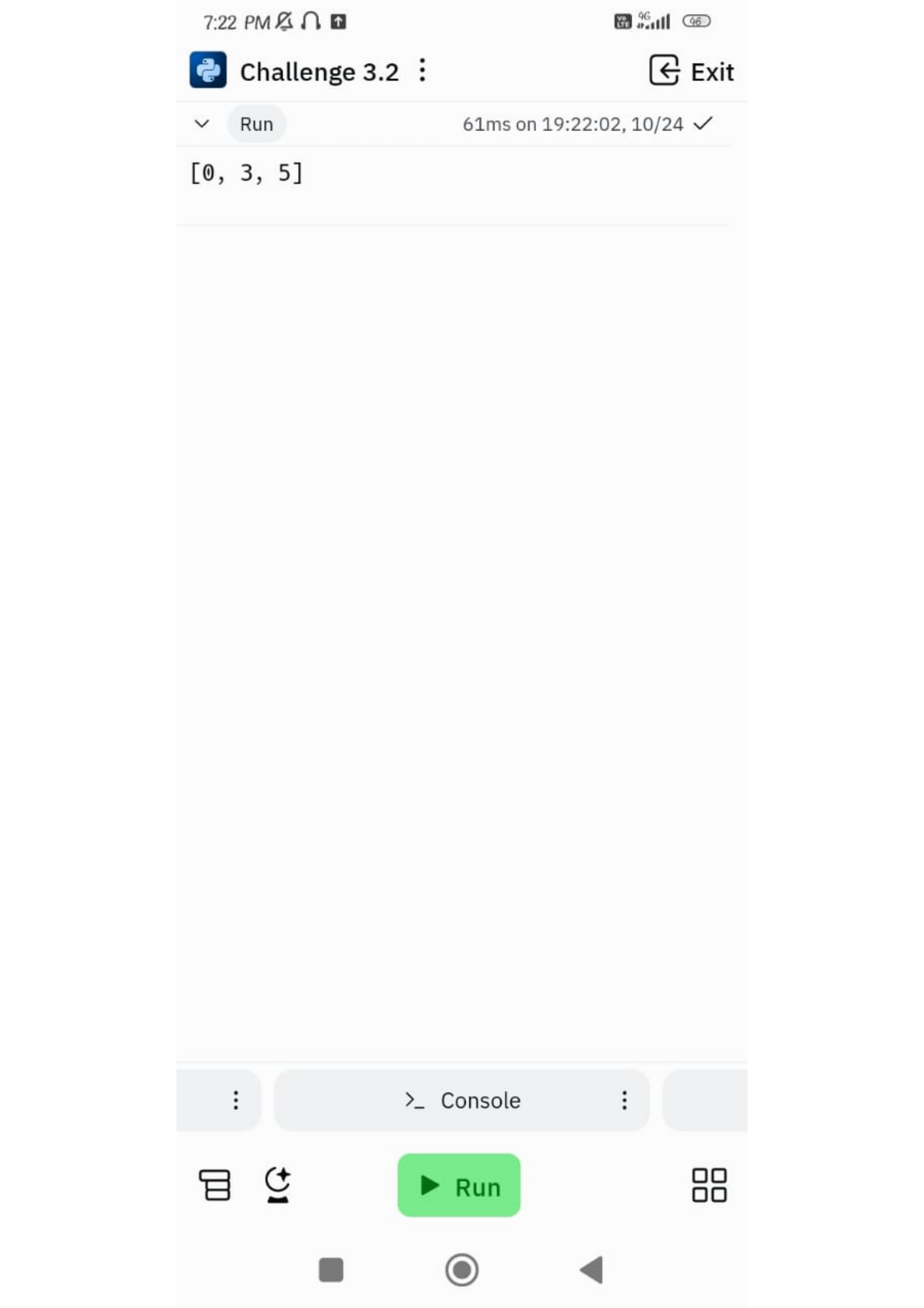
main.py ⋮

Run

```
Name: Mahidhar, Roll Number: A126, CGPA
: 9.9
Name: Saunya, Roll Number: A125, CGPA:
9.1
Name: Srikanth, Roll Number: A124, CGPA
: 8.9
Name: Hari, Roll Number: A123, CGPA: 7.
8
```

```python
def linear_search_product_list(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"
target2 = "apple"
result = linear_search_product_list(products, target)
print(result)
```


7:22 PM
Challenge 3.2
Exit
Run
61ms on 19:22:02, 10/24
[0, 3, 5]
Console
Run