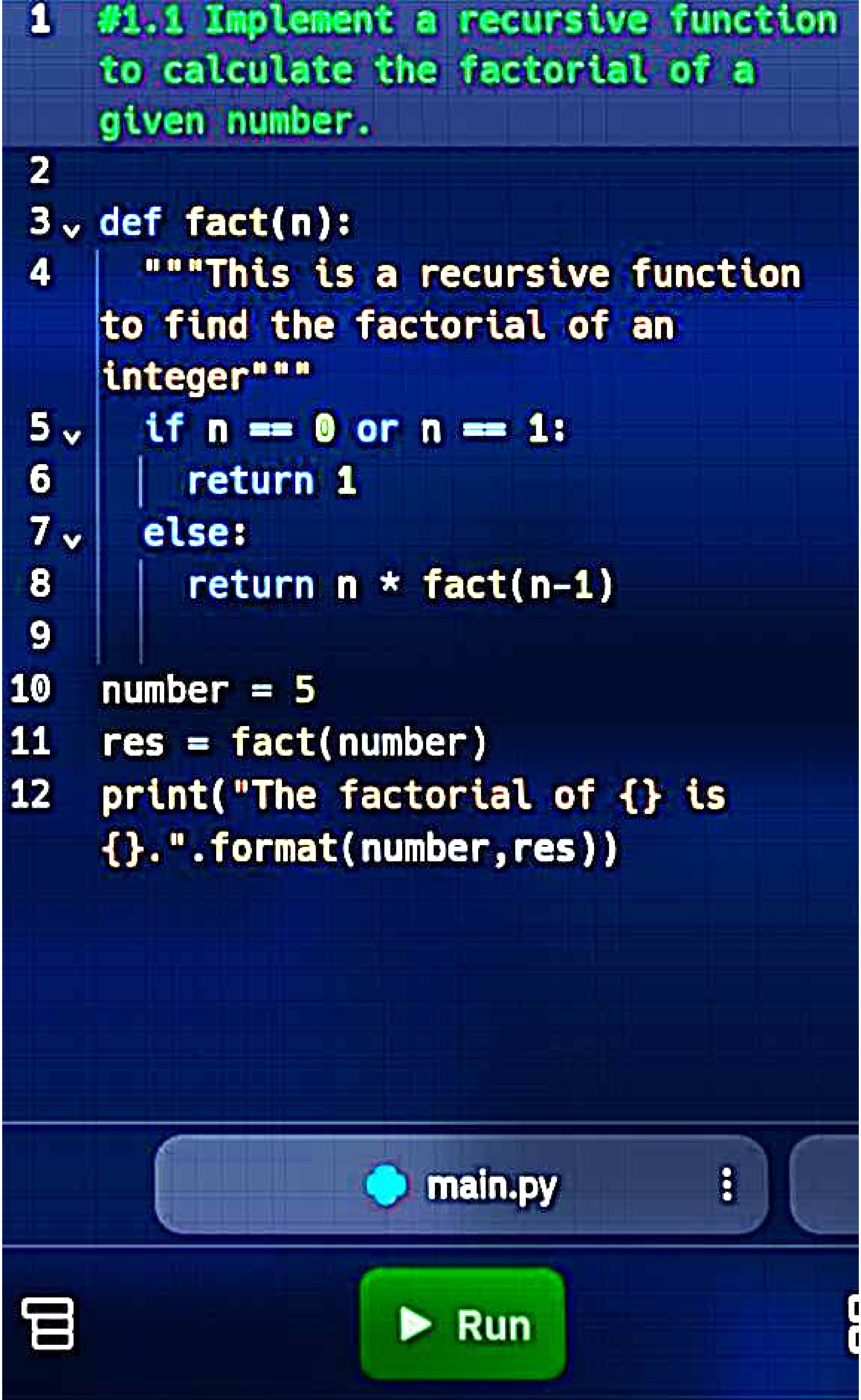
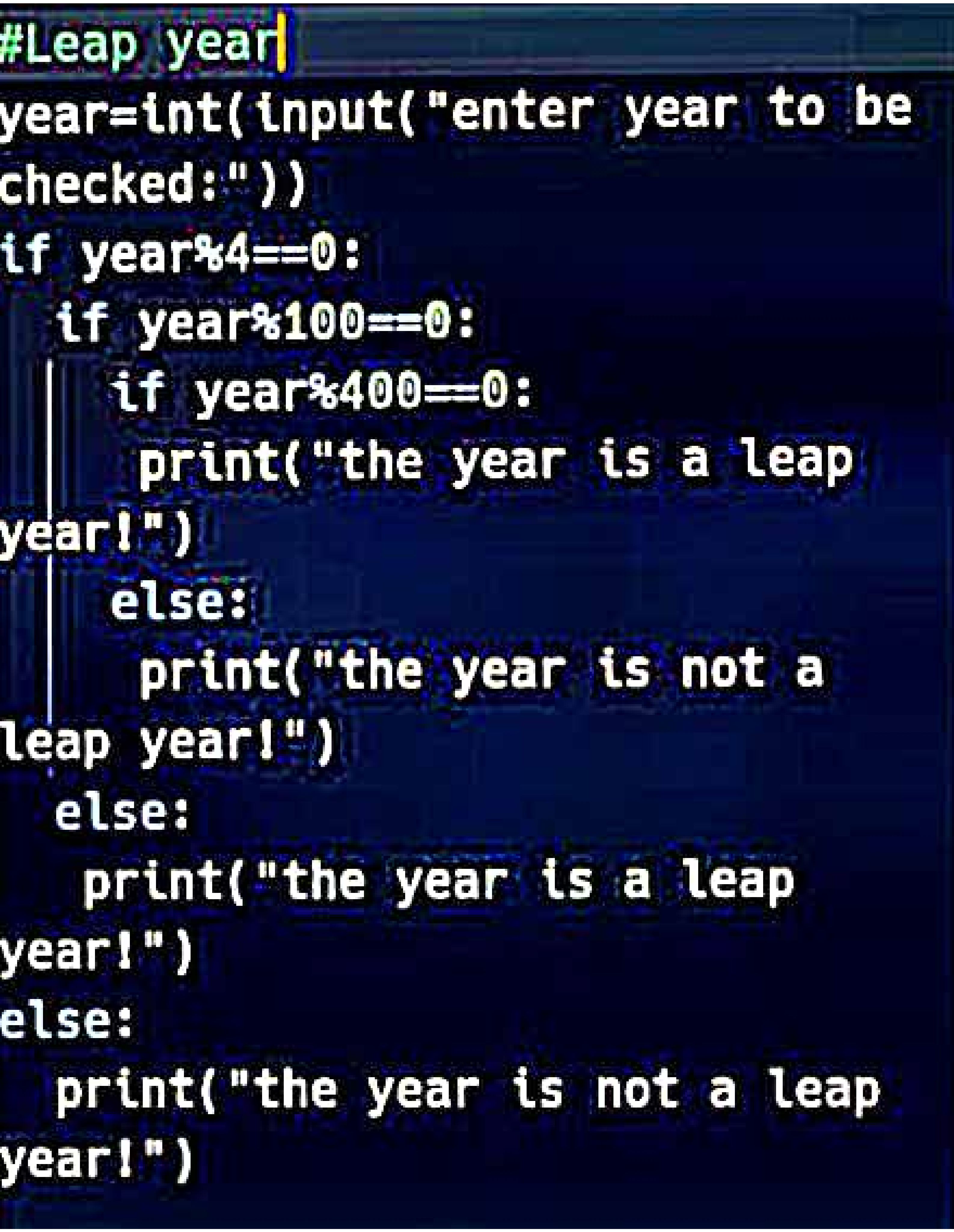
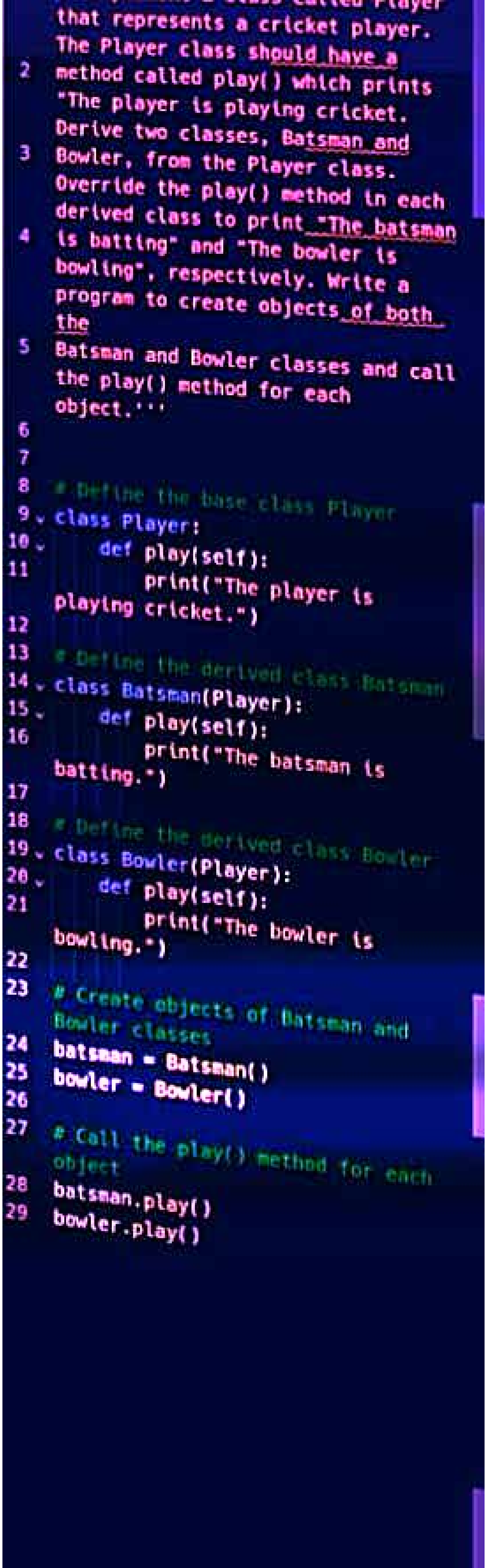


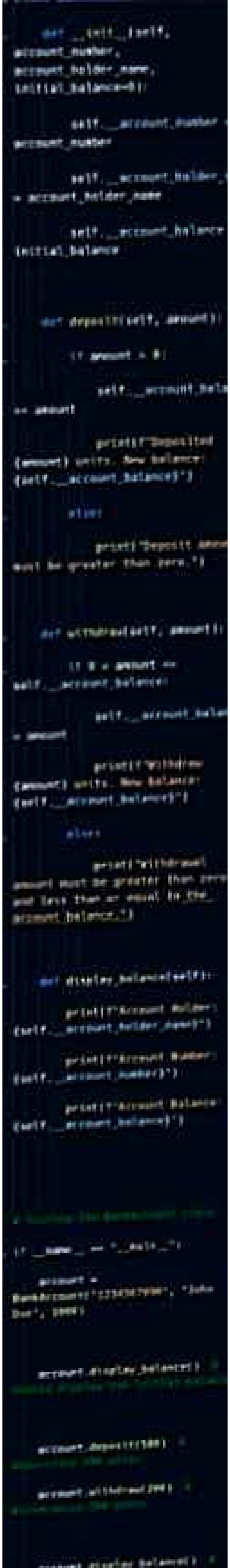
```python
#1.1 Implement a recursive function to calculate the factorial of a given number.

def fact(n):
    """This is a recursive function to find the factorial of an integer"""
    if n == 0 or n == 1:
        return 1
    else:
        return n * fact(n-1)

number = 5
res = fact(number)
print("The factorial of {} is {}.".format(number,res))
```



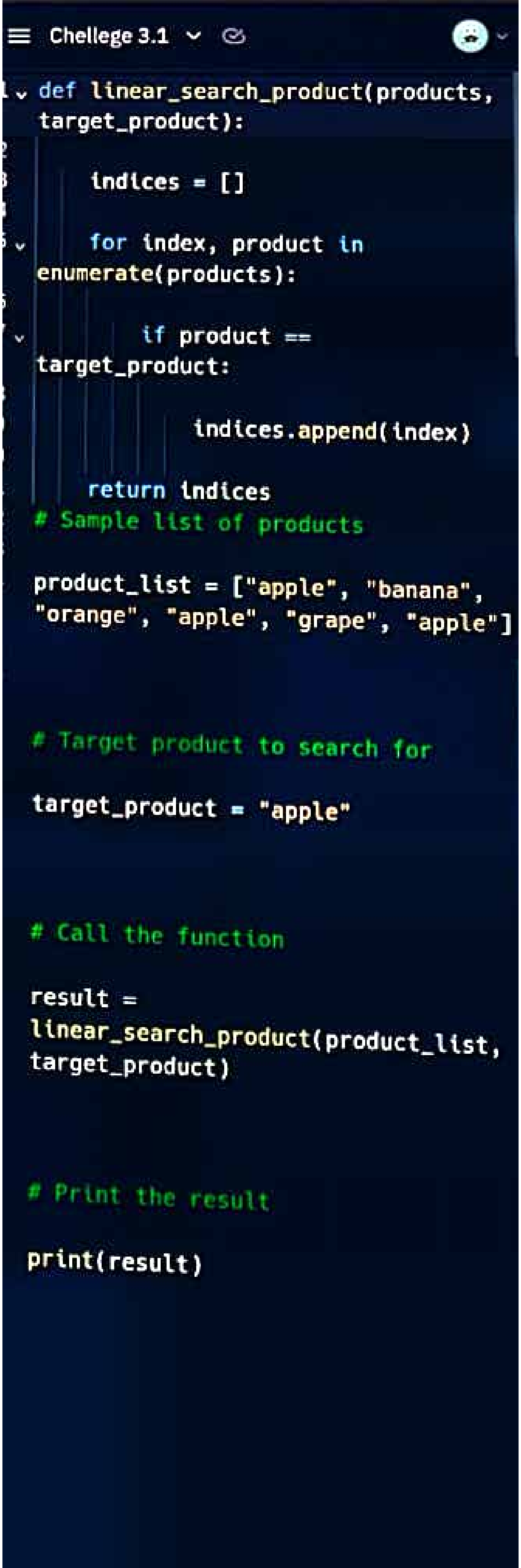
```
#Leap year
year=int(input("enter year to be
checked:"))
if year%4==0:
  if year%100==0:
    if year%400==0:
     print("the year is a leap
year!")
    else:
     print("the year is not a
leap year!")
  else:
   print("the year is a leap
year!")
else:
  print("the year is not a leap
year!")
```



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

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

# Define the derived 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()
```

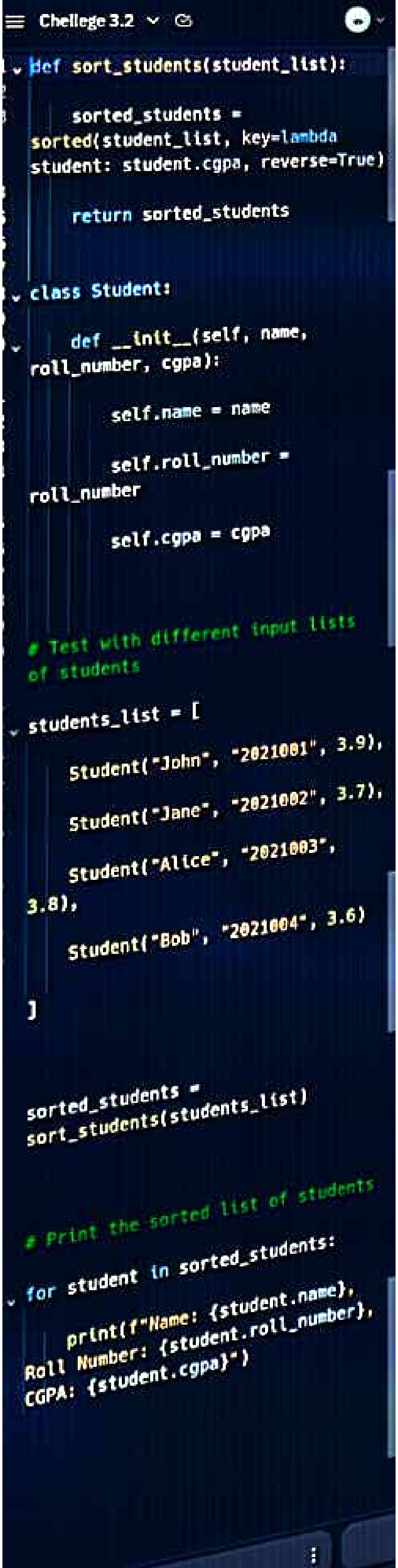


```python
    def __init__(self,
account_number,
account_holder_name,
initial_balance=0):

        self.__account_number =
account_number

        self.__account_holder_
= account_holder_name

        self.__account_balance
initial_balance



    def deposit(self, amount):

        if amount > 0:

            self.__account_bala
+= amount

            print(f"Deposited
{amount} units. New balance:
{self.__account_balance}")

        else:

            print("Deposit amou
must be greater than zero.")


    def withdraw(self, amount):

        if 0 < amount <=
self.__account_balance:

            self.__account_balan
-= amount

            print(f"Withdrew
{amount} units. New balance:
{self.__account_balance}")

        else:

            print("Withdrawal
amount must be greater than zero
and less than or equal to the
account balance.")


    def display_balance(self):

        print(f"Account Holder:
{self.__account_holder_name}")

        print(f"Account Number:
{self.__account_number}")

        print(f"Account Balance:
{self.__account_balance}")




# Testing the BankAccount class

if __name__ == "__main__":

    account =
BankAccount("1234567890", "John
Doe", 1000)


    account.display_balance()  #
[illegible]


    account.deposit(500)  #
[illegible]

    account.withdraw(200)  #
[illegible]


    account.display_balance()  #
```

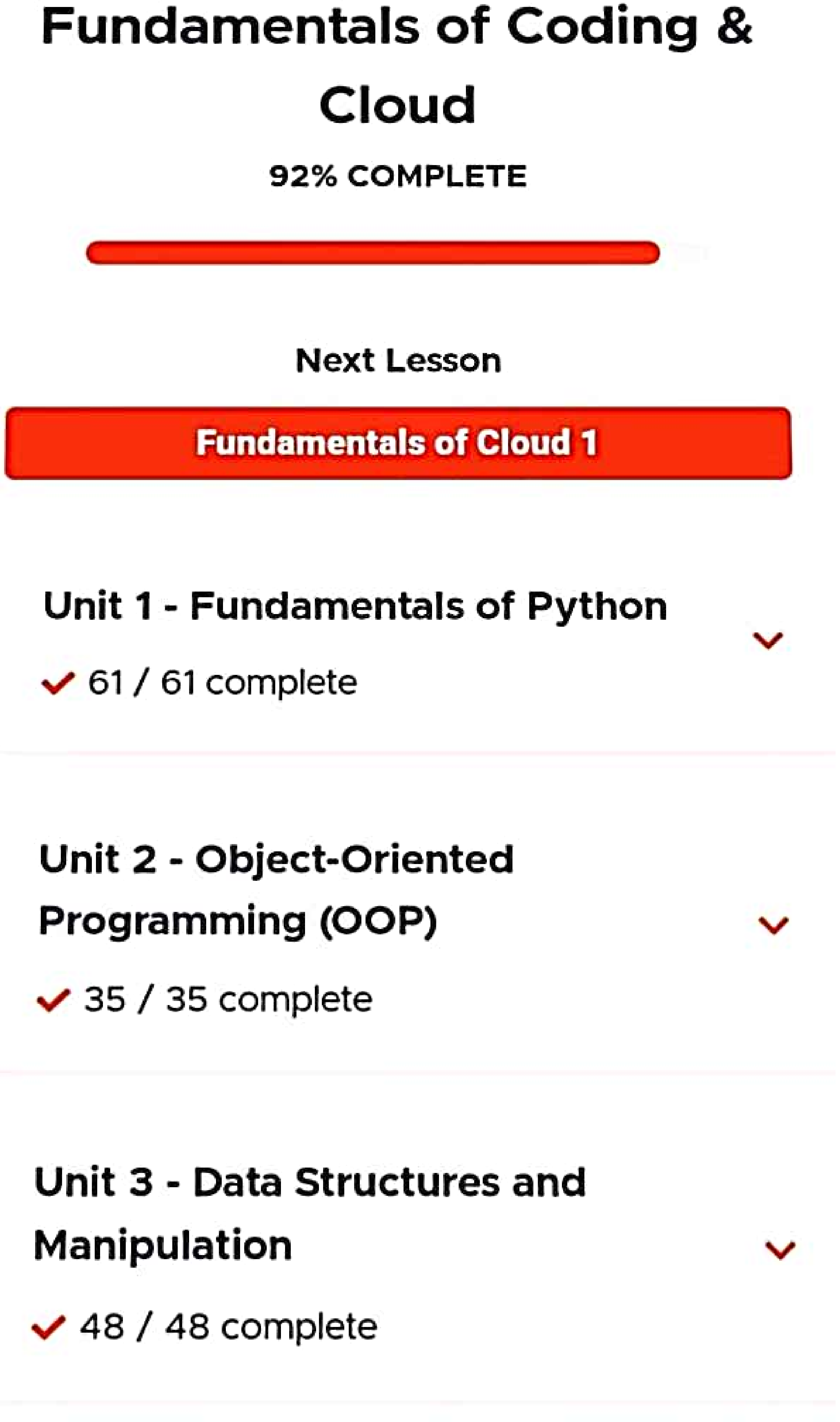


Chellege 3.1

```python
def linear_search_product(products, target_product):

    indices = []

    for index, product in enumerate(products):

        if product == target_product:

            indices.append(index)

    return indices
# Sample list of products

product_list = ["apple", "banana", "orange", "apple", "grape", "apple"]



# Target product to search for

target_product = "apple"



# Call the function

result = linear_search_product(product_list, target_product)



# Print the result

print(result)
```



```
def sort_students(student_list):

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

    return sorted_students


class Student:

    def __init__(self, name,
roll_number, cgpa):

        self.name = name

        self.roll_number =
roll_number

        self.cgpa = cgpa


# Test with different input lists
of students

students_list = [

    Student("John", "2021001", 3.9),

    Student("Jane", "2021002", 3.7),

    Student("Alice", "2021003",
3.8),

    Student("Bob", "2021004", 3.6)

]


sorted_students =
sort_students(students_list)


# Print the sorted list of students

for student in sorted_students:

    print(f"Name: {student.name},
Roll Number: {student.roll_number},
CGPA: {student.cgpa}")
```



Fundamentals of Coding & Cloud

92% COMPLETE

Next Lesson

Fundamentals of Cloud 1

Unit 1 - Fundamentals of Python

✓ 61 / 61 complete

Unit 2 - Object-Oriented Programming (OOP)

✓ 35 / 35 complete

Unit 3 - Data Structures and Manipulation

✓ 48 / 48 complete