TypeScript Code

```
import React, { useState } from 'react';
import { FileDown } from 'lucide-react';

const PythonSolutions = () => {
  const [generating, setGenerating] = useState(false);

  const solutions = [
    {
      no: "632",
      question: "Implement the following hierarchy. The Book function has name, n (n
umber of authors), authors (list of authors), publisher, ISBN, and year as its data members
and the derived class has course as its data member.",
      code: `class Book:
    def __init__(self, name, n, authors, publisher, ISBN, year):
        self.name = name
        self.n = n
        self.authors = authors
        self.publisher = publisher
        self.ISBN = ISBN
        self.year = year

    def display(self):
        print(f"Name: {self.name}")
        print(f"Number of authors: {self.n}")
        print(f"Authors: {', '.join(self.authors)}")
        print(f"Publisher: {self.publisher}")
        print(f"ISBN: {self.ISBN}")
        print(f"Year: {self.year}")

class CourseBook(Book):
    def __init__(self, name, n, authors, publisher, ISBN, year, course):
        super().__init__(name, n, authors, publisher, ISBN, year)
        self.course = course

    def display(self):
        super().display()
        print(f"Course: {self.course}")

# Example Usage
book = Book("The Great Gatsby", 1, ["F. Scott Fitzgerald"], "Scribner", "97807432735
65", 1925)
book.display()
print()
course_book = CourseBook("Data Science from Scratch", 1, ["Joel Grus"], "O'Reilly",
"9781492041139", 2019, "Data Science")
course_book.display()`
    },
    {
      no: "633",
      question: "Implement the following hierarchy. The Staff function has name and
salary as its data members, the derived class Teaching has subject as its data member and th
e class NonTeaching has department as its data member.",
```

```
        code: `class Staff:
    def __init__(self, name, salary):
        self.name = name
        self.salary = salary

    def display(self):
        print(f"Name: {self.name}")
        print(f"Salary: {self.salary}")

class Teaching(Staff):
    def __init__(self, name, salary, subject):
        super().__init__(name, salary)
        self.subject = subject

    def display(self):
        super().display()
        print(f"Subject: {self.subject}")

class NonTeaching(Staff):
    def __init__(self, name, salary, department):
        super().__init__(name, salary)
        self.department = department

    def display(self):
        super().display()
        print(f"Department: {self.department}")

# Example Usage
staff_member = Staff("John", 50000)
teaching_member = Teaching("Jane", 60000, "Math")
non_teaching_member = NonTeaching("Joe", 40000, "Finance")

staff_member.display()
print()
teaching_member.display()
print()
non_teaching_member.display()`
    },
    {
        no: "634",
        question: "Create a class called Student, having name and email as its data members and __init__(self, name, email) and putdata(self) as bound methods. Create another class called PhDguide having name, email, and students as its data members.",
        code: `class Person:
    def __init__(self, name, email):
        self.name = name
        self.email = email

    def putdata(self):
        print("Name:", self.name)
        print("Email:", self.email)

class Student(Person):
    def __init__(self, name, email):
        super().__init__(name, email)
```

```
100
101 class PhDguide(Person):
102     def __init__(self, name, email):
103         super().__init__(name, email)
104         self.students = []
105
106     def putdata(self):
107         super().putdata()
108         print("Students:", [s.name for s in self.students])
109
110     def add(self, student):
111         self.students.append(student)
112
113     def remove(self, student):
114         if student in self.students:
115             self.students.remove(student)
116             print(f"{student.name} has been removed from the list of students.")
117         else:
118             print(f"{student.name} is not in the list of students.")
119
120 # Example Usage
121 student1 = Student("John Doe", "johndoe@example.com")
122 student1.putdata()
123 print()
124
125 guide1 = PhDguide("Jane Smith", "janesmith@example.com")
126 guide1.putdata()
127 print()
128
129 guide1.add(student1)
130 guide1.putdata()
131 print()
132
133 guide1.remove(student1)
134 guide1.putdata()`
135     },
136     {
137       no: "635",
138       question: "Program to demonstrate the issue of invoking __init__() in case of
multiple inheritance",
139       code: `class A:
140     def __init__(self):
141         print("Class A __init__ called")
142         self.a = "A"
143
144 class B:
145     def __init__(self):
146         print("Class B __init__ called")
147         self.b = "B"
148
149 class C(A, B):
150     def __init__(self):
151         print("Class C __init__ called")
152         A.__init__(self)
153         B.__init__(self)
```

```
        self.c = "C"

# Example Usage
obj = C()
print("\\nObject attributes:")
print(f"a = {obj.a}")
print(f"b = {obj.b}")
print(f"c = {obj.c}")`
    },
    {
      no: "636",
      question: "Write program that has a class point. Define another class location which has two objects (Location and Destination) of class point. Also define function in Location that prints reflection of Destination on the x axis.",
      code: `class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

class Destination(Point):
    pass

class Location(Point):
    def __init__(self, x1, y1, x2, y2):
        super().__init__(x1, y1)
        self.destination = Destination(x2, y2)

    def reflect_destination_on_x_axis(self):
        self.destination.y = -self.destination.y
        print(f"Reflected Destination point on X axis: ({self.destination.x}, {self.destination.y})")

# Example Usage
l = Location(1, 2, 3, 4)
l.reflect_destination_on_x_axis()`
    },
    {
      no: "637",
      question: "Write a program that overload the + operator so that it can add two object of class fraction",
      code: `class Fraction:
    def __init__(self, numerator, denominator):
        self.numerator = numerator
        self.denominator = denominator

    def __add__(self, other):
        new_numerator = (self.numerator * other.denominator) + (other.numerator * self.denominator)
        new_denominator = self.denominator * other.denominator
        return Fraction(new_numerator, new_denominator)

    def display(self):
        print(f"{self.numerator}/{self.denominator}")

# Example Usage
```

```
f1 = Fraction(1, 2)
f2 = Fraction(1, 3)
f3 = f1 + f2

print("Fraction 1:", end=" ")
f1.display()
print("Fraction 2:", end=" ")
f2.display()
print("Sum:", end=" ")
f3.display()`
    },
    {
      no: "638",
      question: "Write a program that overload the * operator so that it can add two object of class fraction",
      code: `class Fraction:
    def __init__(self, numerator, denominator):
        self.numerator = numerator
        self.denominator = denominator

    def __mul__(self, other):
        new_numerator = self.numerator * other.numerator
        new_denominator = self.denominator * other.denominator
        return Fraction(new_numerator, new_denominator)

    def display(self):
        print(f"{self.numerator}/{self.denominator}")

# Example Usage
f1 = Fraction(2, 3)
f2 = Fraction(3, 4)
f3 = f1 * f2

print("Fraction 1:", end=" ")
f1.display()
print("Fraction 2:", end=" ")
f2.display()
print("Product:", end=" ")
f3.display()`
    },
    {
      no: "639",
      question: "Write a program to find the distance between two points in cartesian coordinate system",
      code: `import math

class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def distance(self, other):
        dx = self.x - other.x
        dy = self.y - other.y
        return math.sqrt(dx * dx + dy * dy)
```

```
257
258   # Example Usage
259   p1 = Point(3, 4)
260   p2 = Point(0, 0)
261   dist = p1.distance(p2)
262   print(f"Distance between points: {dist}")`
263       },
264       {
265         no: "640",
266         question: "Write a program to find the slope between two points in cartesian c
oordinate system",
267         code: `class Point:
268       def __init__(self, x, y):
269           self.x = x
270           self.y = y
271
272       def slope(self, other):
273           if self.x == other.x:
274               return "Undefined (vertical line)"
275           return (other.y - self.y) / (other.x - self.x)
276
277   # Example Usage
278   p1 = Point(1, 2)
279   p2 = Point(4, 8)
280   slope_value = p1.slope(p2)
281   print(f"Slope between points: {slope_value}")`
282       },
283       {
284         no: "641",
285         question: "Create a class student with following member attributes: roll no, n
ame, age and total marks. Write a python program to overload '==' operator to print the deta
ils of students having same marks.",
286         code: `class Student:
287       def __init__(self, roll_no, name, age, total_marks):
288           self.roll_no = roll_no
289           self.name = name
290           self.age = age
291           self.total_marks = total_marks
292
293       def display_student_info(self):
294           print(f"Roll No: {self.roll_no}")
295           print(f"Name: {self.name}")
296           print(f"Age: {self.age}")
297           print(f"Total Marks: {self.total_marks}")
298
299       def __eq__(self, other):
300           if isinstance(other, Student):
301               return self.total_marks == other.total_marks
302           return False
303
304   # Example Usage
305   s1 = Student(1, "John", 20, 90)
306   s2 = Student(2, "Mary", 21, 90)
307   s3 = Student(3, "Peter", 19, 85)
308
```

```
    if s1 == s2:
        print(f"{s1.name} and {s2.name} have the same marks!")
        s1.display_student_info()
        print()
        s2.display_student_info()
    else:
        print(f"{s1.name} and {s2.name} do not have the same marks.")`
        },
        {
          no: "642",
          question: "Write a program to create a class called Data having 'value' as its data member. Overload the (>) and the (<) operator for the class.",
          code: `class Data:
        def __init__(self, value):
            self.value = value

        def __lt__(self, other):
            return self.value < other.value

        def __gt__(self, other):
            return self.value > other.value

    # Example Usage
    d1 = Data(10)
    d2 = Data(20)
    d3 = Data(15)

    print(f"d1 > d2: {d1 > d2}")
    print(f"d2 > d3: {d2 > d3}")
    print(f"d3 < d1: {d3 < d1}")
    print(f"d3 < d2: {d3 < d2}")`
        },
        {
          no: "643",
          question: "The following illustration creates a class called data. If no argument is passed while instantiating the class a false is returned, otherwise a true is returned.",
          code: `class Data:
        def __init__(self, value=None):
            if value:
                self.value = value
                self.status = True
            else:
                self.status = False

        def __str__(self):
            return f"status: {self.status}"

    # Example Usage
    data_obj1 = Data()
    print(data_obj1)

    data_obj2 = Data(10)
    print(data_obj2)`
        },
```

```
      {
        no: "644",
        question: "Create a 5X2 integer array from a range between 100 to 200 such tha
t the difference between each element is 10",
        code: `import numpy as np

  # Create array with values from 100 to 200 with step 10
  arr = np.arange(100, 200, 10)

  # Reshape to 5x2
  result = arr.reshape(5, 2)

  print("5x2 Array:")
  print(result)`
      },
      {
        no: "645",
        question: "Following is the provided numpy array. Return array of items by tak
ing the third column from all rows",
        code: `import numpy as np

  # Sample array
  sampleArray = np.array([[11, 22, 33], [44, 55, 66], [77, 88, 99]])

  print("Original Array:")
  print(sampleArray)

  # Get third column from all rows
  third_column = sampleArray[:, 2]

  print("\\nThird column:")
  print(third_column)`
      },
      {
        no: "646",
        question: "Return array of odd rows and even columns from below numpy array",
        code: `import numpy as np

  # Sample array
  sampleArray = np.array([[3, 6, 9, 12], [15, 18, 21, 24], [27, 30, 33, 36], [39, 42,
45, 48], [51, 54, 57, 60]])

  print("Original Array:")
  print(sampleArray)

  # Get odd rows (index 0, 2, 4) and even columns (index 0, 2)
  result = sampleArray[::2, ::2]

  print("\\nOdd rows and even columns:")
  print(result)`
      },
      {
        no: "647",
        question: "Sort following NumPy array - Case 1: Sort array by the second row,
Case 2: Sort the array by the second column",
```

```
        code: `import numpy as np

  # Sample array
  sampleArray = np.array([[34, 43, 73], [82, 22, 12], [53, 94, 66]])

  print("Original Array:")
  print(sampleArray)

  # Case 1: Sort by second row
  sorted_by_row = sampleArray[:, sampleArray[1, :].argsort()]
  print("\\nCase 1 - Sorted by second row:")
  print(sorted_by_row)

  # Case 2: Sort by second column
  sorted_by_column = sampleArray[sampleArray[:, 1].argsort()]
  print("\\nCase 2 - Sorted by second column:")
  print(sorted_by_column)`
      },
      {
        no: "648",
        question: "Print max from axis 0 and min from axis 1 from the following 2-D ar
ray",
        code: `import numpy as np

  # Sample array
  sampleArray = np.array([[34, 43, 73], [82, 22, 12], [53, 94, 66]])

  print("Original Array:")
  print(sampleArray)

  # Max from axis 0 (column-wise max)
  max_axis_0 = np.max(sampleArray, axis=0)
  print("\\nMax from axis 0:")
  print(max_axis_0)

  # Min from axis 1 (row-wise min)
  min_axis_1 = np.min(sampleArray, axis=1)
  print("\\nMin from axis 1:")
  print(min_axis_1)`
      },
      {
        no: "649",
        question: "Write a NumPy array program to convert the values of Fahrenheit deg
rees into Celsius degrees",
        code: `import numpy as np

  # Fahrenheit values
  fahrenheit = np.array([0, 12, 45.21, 34, 99.91, 32])

  print("Values in Fahrenheit degrees:")
  print(fahrenheit)

  # Convert to Celsius using formula: C = (F - 32) * 5/9
  celsius = (fahrenheit - 32) * 5 / 9

```

```
print("\\nValues in Centigrade degrees:")
print(celsius)

# Sort the array
celsius_sorted = np.sort(celsius)
print("\\nSorted Celsius values:")
print(celsius_sorted)`
    },
    {
      no: "650",
      question: "Import numpy as np. arr = np.array([[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]]) Reshape arr into a 2D array and a 3D array.",
      code: `import numpy as np

# Original 1D array
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])

print("Original 1D array:")
print(arr)

# Reshape to 2D array (3x4)
arr_2d = arr.reshape(3, 4)
print("\\n2D Array (3x4):")
print(arr_2d)

# Reshape to 3D array (2x2x3)
arr_3d = arr.reshape(2, 2, 3)
print("\\n3D Array (2x2x3):")
print(arr_3d)`
    },
    {
      no: "651",
      question: "Imagine you own a call center. Use the following abstract class template to create three more classes: Respondent, Manager, and Director that inherit the Employee Abstract Class.",
      code: `from abc import ABC, abstractmethod

class EmployeeABC(ABC):
    @abstractmethod
    def receive_call(self):
        pass

    @abstractmethod
    def end_call(self):
        pass

    @abstractmethod
    def is_free(self):
        pass

    @abstractmethod
    def get_rank(self):
        pass

class Respondent(EmployeeABC):
```

```
    def __init__(self, name):
        self.name = name
        self.free = True

    def receive_call(self):
        print(f"{self.name} (Respondent) receiving call")
        self.free = False

    def end_call(self):
        print(f"{self.name} (Respondent) ending call")
        self.free = True

    def is_free(self):
        return self.free

    def get_rank(self):
        return "Respondent"

class Manager(EmployeeABC):
    def __init__(self, name):
        self.name = name
        self.free = True

    def receive_call(self):
        print(f"{self.name} (Manager) receiving call")
        self.free = False

    def end_call(self):
        print(f"{self.name} (Manager) ending call")
        self.free = True

    def is_free(self):
        return self.free

    def get_rank(self):
        return "Manager"

class Director(EmployeeABC):
    def __init__(self, name):
        self.name = name
        self.free = True

    def receive_call(self):
        print(f"{self.name} (Director) receiving call")
        self.free = False

    def end_call(self):
        print(f"{self.name} (Director) ending call")
        self.free = True

    def is_free(self):
        return self.free

    def get_rank(self):
        return "Director"
```

```
572
573   # Example Usage
574   r1 = Respondent("John")
575   m1 = Manager("Sarah")
576   d1 = Director("Michael")
577
578   r1.receive_call()
579   print(f"Is {r1.name} free? {r1.is_free()}")
580   r1.end_call()
581   print(f"Is {r1.name} free? {r1.is_free()}")`
582       },
583       {
584         no: "652",
585         question: "Write a python program to create a Bus child class that inherits fr
om the Vehicle class. The default fare charge of any vehicle is seating capacity * 100. If V
ehicle is Bus instance, we need to add an extra 10% on full fare as a maintenance charge.",
586         code: `class Vehicle:
587       def __init__(self, seating_capacity):
588           self.seating_capacity = seating_capacity
589
590       def fare(self):
591           return self.seating_capacity * 100
592
593   class Bus(Vehicle):
594       def fare(self):
595           base_fare = super().fare()
596           maintenance_charge = base_fare * 0.10
597           total_fare = base_fare + maintenance_charge
598           return total_fare
599
600   # Example Usage
601   v1 = Vehicle(50)
602   print(f"Vehicle fare: {v1.fare()}")
603
604   b1 = Bus(50)
605   print(f"Bus fare (with 10% maintenance): {b1.fare()}")
606
607   # Example with car
608   c1 = Vehicle(5)
609   print(f"\\nCar fare: {c1.fare()}")`
610       },
611       {
612         no: "653",
613         question: "Create an abstract class named Shape. Create an abstract method nam
ed calculate_area for the Shape class. Create two classes Rectangle and Circle that inherit
from Shape class. Implement calculate_area method in both classes.",
614         code: `from abc import ABC, abstractmethod
615   import math
616
617   class Shape(ABC):
618       @abstractmethod
619       def calculate_area(self):
620           pass
621
622   class Rectangle(Shape):
```

```
    def __init__(self, length, breadth):
        self.length = length
        self.breadth = breadth

    def calculate_area(self):
        return self.length * self.breadth

class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius

    def calculate_area(self):
        return math.pi * self.radius * self.radius

# Example Usage
rect = Rectangle(10, 5)
print(f"Area of Rectangle: {rect.calculate_area()}")

circle = Circle(7)
print(f"Area of Circle: {circle.calculate_area():.2f}")

# Check if Rectangle area is greater than Circle area
if rect.calculate_area() > circle.calculate_area():
    print("\\nRectangle has greater area")
else:
    print("\\nCircle has greater area")`
    },
    {
      no: "654",
      question: "Write a python program to demonstrate the use of super() method to call the method of base class.",
      code: `class Parent:
    def show(self):
        print("Inside Parent class method")

class Child(Parent):
    def show(self):
        print("Inside Child class method")
        super().show()
        print("Back to Child class method")

# Example Usage
obj = Child()
obj.show()`
    },
    {
      no: "655",
      question: "Create a class called Matrix containing constructor that initialized the number of rows and number of columns of a new Matrix object.",
      code: `class Matrix:
    def __init__(self, rows, cols):
        self.rows = rows
        self.cols = cols
        self.data = [[0 for j in range(cols)] for i in range(rows)]

```

```
    def set_value(self, row, col, value):
        if 0 <= row < self.rows and 0 <= col < self.cols:
            self.data[row][col] = value
        else:
            print("Invalid position")

    def get_value(self, row, col):
        if 0 <= row < self.rows and 0 <= col < self.cols:
            return self.data[row][col]
        return None

    def display(self):
        for row in self.data:
            print(row)

# Example Usage
m = Matrix(3, 3)
m.set_value(0, 0, 1)
m.set_value(0, 1, 2)
m.set_value(0, 2, 3)
m.set_value(1, 0, 4)
m.set_value(1, 1, 5)
m.set_value(1, 2, 6)
m.set_value(2, 0, 7)
m.set_value(2, 1, 8)
m.set_value(2, 2, 9)

print("Matrix:")
m.display()`
    },
    {
      no: "656",
      question: "Find the MRO of class Z of below program",
      code: `class A:
    pass

class B:
    pass

class C:
    pass

class D:
    pass

class E:
    pass

class K1(C, A, B):
    pass

class K3(A, D):
    pass

class K2(B, D, E):
```

```
    pass

class Z(K1, K3, K2):
    pass

# Find MRO (Method Resolution Order)
print("MRO of class Z:")
print(Z.__mro__)

print("\\nMRO in readable format:")
for i, cls in enumerate(Z.__mro__):
    print(f"{i+1}. {cls.__name__}")`
    },
    {
      no: "657",
      question: "Write a Python Program to find the Net Salary of Employee using Inheritance. Create three class Employee, Perks, NetSalary. Make an Employee class as an abstract class.",
      code: `from abc import ABC, abstractmethod

class Employee(ABC):
    def __init__(self, emp_id, name, salary):
        self.emp_id = emp_id
        self.name = name
        self.salary = salary

    @abstractmethod
    def print_details(self):
        pass

    @abstractmethod
    def get_salary(self):
        pass

class Perks:
    def __init__(self, da, hra, pf):
        self.da = da
        self.hra = hra
        self.pf = pf

    def calculate_total_perks(self, basic_salary):
        da_amount = basic_salary * (self.da / 100)
        hra_amount = basic_salary * (self.hra / 100)
        pf_amount = basic_salary * (self.pf / 100)
        return da_amount + hra_amount - pf_amount

class NetSalary(Employee):
    def __init__(self, emp_id, name, salary, da, hra, pf):
        Employee.__init__(self, emp_id, name, salary)
        self.perks = Perks(da, hra, pf)

    def print_details(self):
        print(f"Employee ID: {self.emp_id}")
        print(f"Employee Name: {self.name}")
        print(f"Basic Salary: {self.salary}")
```

```
        def get_salary(self):
            perks_amount = self.perks.calculate_total_perks(self.salary)
            net_salary = self.salary + perks_amount
            return net_salary

    # Example Usage
    emp = NetSalary(1, "John", 25000, 35, 17, 12)
    emp.print_details()
    print(f"DA: 35%")
    print(f"HRA: 17%")
    print(f"PF: 12%")
    print(f"Total Salary: {emp.get_salary():.2f}")`
        }
      ];

      const generatePDF = () => {
        setGenerating(true);

        setTimeout(() => {
          const content = document.getElementById('solutions-content');
          const printWindow = window.open('', '', 'height=600,width=800');

          printWindow.document.write('<html><head><title>Python Programming Solutions - Unit 9</title>');
          printWindow.document.write('<style>');
          printWindow.document.write(`
            body { font-family: Arial, sans-serif; padding: 20px; line-height: 1.6; }
            h1 { color: #2563eb; text-align: center; border-bottom: 3px solid #2563eb; padding-bottom: 10px; }
            h2 { color: #1e40af; margin-top: 30px; background: #eff6ff; padding: 10px; border-left: 4px solid #2563eb; }
            .question { background: #f3f4f6; padding: 10px; margin: 10px 0; border-radius: 5px; }
            pre { background: #1e293b; color: #e2e8f0; padding: 15px; border-radius: 5px; overflow-x: auto; font-size: 12px; }
            code { font-family: 'Courier New', monospace; }
            .page-break { page-break-after: always; }
          `);
          printWindow.document.write('</style></head><body>');
          printWindow.document.write('<h1>Python Programming Solutions - Unit 9</h1>');
          printWindow.document.write('<p style="text-align: center; color: #666;">Complete Solutions with Question Numbers</p>');

          solutions.forEach((sol, idx) => {
            printWindow.document.write(`<h2>Question ${sol.no}</h2>`);
            printWindow.document.write(`<div class="question"><strong>Problem:</strong> ${sol.question}</div>`);
            printWindow.document.write(`<pre><code>${sol.code.replace(/</g, '&lt;').replace(/>/g, '&gt;')}</code></pre>`);
            if ((idx + 1) % 3 === 0 && idx < solutions.length - 1) {
              printWindow.document.write('<div class="page-break"></div>');
            }
          });
```

```
        printWindow.document.write('</body></html>');
        printWindow.document.close();

        printWindow.onload = function() {
          printWindow.focus();
          printWindow.print();
          setGenerating(false);
        };
      }, 500);
    };

    return (
      <div className="min-h-screen bg-gradient-to-br from-blue-50 to-indigo-100 p-8">
        <div className="max-w-6xl mx-auto">
          <div className="bg-white rounded-lg shadow-xl p-8 mb-6">
            <h1 className="text-4xl font-bold text-blue-600 mb-2 text-center">
              Python Programming Solutions
            </h1>
            <p className="text-gray-600 text-center mb-6">Unit 9 - Complete Solutions
</p>

            <div className="flex justify-center mb-8">
              <button
                onClick={generatePDF}
                disabled={generating}
                className="bg-blue-600 hover:bg-blue-700 text-white font-bold py-3 px-
6 rounded-lg flex items-center gap-2 transition-all disabled:opacity-50 disabled:cursor-not-
allowed"
              >
                <FileDown size={20} />
                {generating ? 'Generating PDF...' : 'Download as PDF'}
              </button>
            </div>
          </div>

          <div id="solutions-content" className="bg-white rounded-lg shadow-xl p-8">
            {solutions.map((sol, idx) => (
              <div key={idx} className="mb-12 pb-8 border-b border-gray-200 last:borde
r-b-0">
                <h2 className="text-2xl font-bold text-blue-700 mb-3">
                  Question {sol.no}
                </h2>
                <div className="bg-gray-50 p-4 rounded-lg mb-4">
                  <p className="text-gray-700">{sol.question}</p>
                </div>
                <div className="bg-gray-900 rounded-lg overflow-hidden">
                  <div className="bg-gray-800 px-4 py-2 text-sm text-gray-300">
                    Python Code
                  </div>
                  <pre className="p-4 overflow-x-auto">
                    <code className="text-sm text-gray-100">{sol.code}</code>
                  </pre>
                </div>
              </div>
            ))}
```

```
          </div>

          <div className="bg-white rounded-lg shadow-xl p-6 mt-6 text-center">
            <p className="text-gray-600">
              Total Solutions: <span className="font-bold text-blue-600">{solution
s.length}</span>
            </p>
            <p className="text-sm text-gray-500 mt-2">
              All solutions use basic Python syntax for easy understanding
            </p>
          </div>
        </div>
      </div>
    );
  };

  export default PythonSolutions;
```