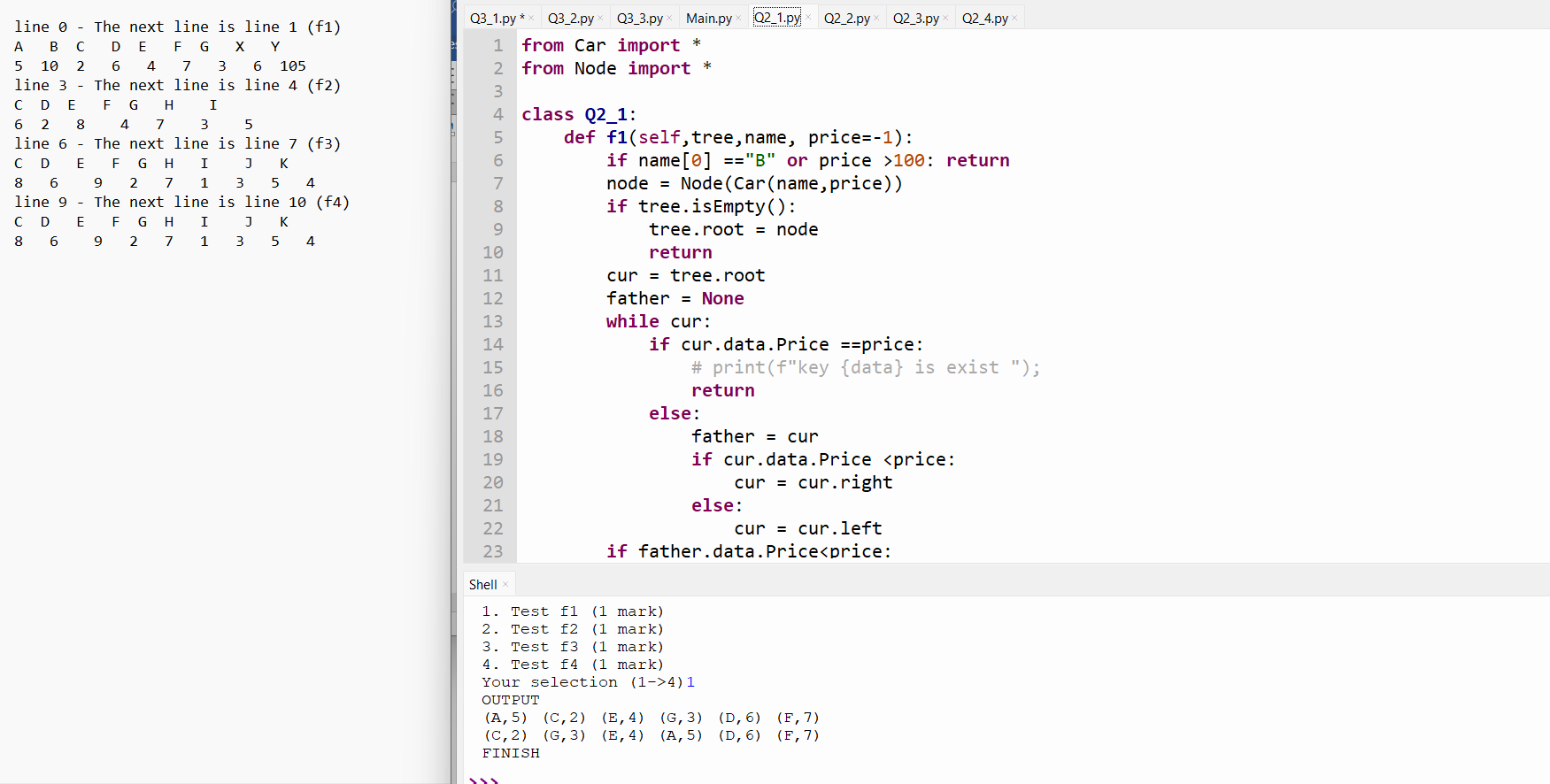
Q2\_1:



1. Kiểm tra điều kiện và tạo một node mới:

def f1(self,tree,name, price=-1):

if name[0] =="B" or price >100: return

Nếu ký tự đầu tiên của name là "B" hoặc price lớn hơn 100, hàm sẽ kết thúc ngay lập tức mà không thêm node mới vào cây tìm kiếm nhị phân.

2. Tạo một node mới với đối tượng Car:

node = Node(Car(name,price))

Tạo một node mới với một đối tượng Car được tạo từ name và price đã cho.

3. Kiểm tra nếu cây là rỗng:

if tree.isEmpty():

tree.root = node

return

Nếu cây tìm kiếm nhị phân rỗng, node mới trở thành gốc của cây và hàm kết thúc.

4. Duyệt cây để tìm vị trí thích hợp cho node mới:

cur = tree.root

father = None

while cur:

if cur.data.Price ==price:

# print(f"key {data} is exist ");

return

else:

father = cur

if cur.data.Price <price:

cur = cur.right

else:

cur = cur.left

Duyệt cây để tìm vị trí thích hợp cho node mới dựa trên thuộc tính Price của các đối tượng Car.

Nếu đã tồn tại một node có giá trị Price tương tự, hàm sẽ kết thúc mà không thêm node mới.

5. Thêm node mới vào cây:

if father.data.Price<price:

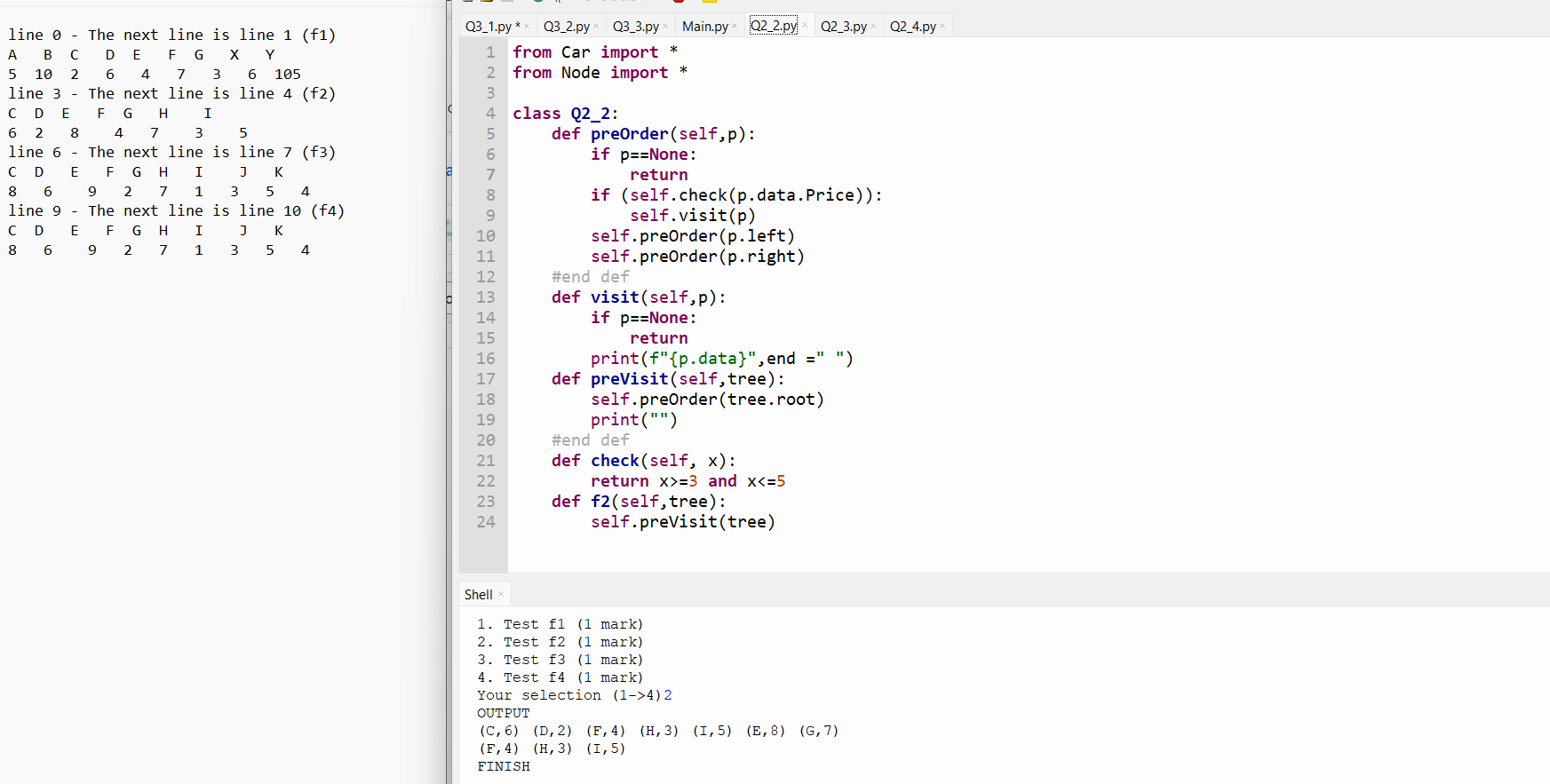
father.right = node

else:

father.left = node

Khi vị trí chính xác được tìm thấy (khi cur là None), thêm node mới vào cây làm con trái hoặc con phải của node father, tùy thuộc vào so sánh với thuộc tính Price.

Q2\_2:



1. Phương thức preOrder:

def preOrder(self,p):

if p==None:

return

if (self.check(p.data.Price)):

self.visit(p)

self.preOrder(p.left)

self.preOrder(p.right)

#end def

Duyệt cây theo thứ tự trước (pre-order).

Nếu giá trị Price của node p thỏa mãn điều kiện kiểm tra self.check, thì gọi phương thức self.visit(p).

Tiếp tục duyệt cây theo thứ tự trước trên cả hai nhánh con (left và right).

2. Phương thức visit:

def visit(self,p):

if p==None:

return

print(f"{p.data}",end =" ")

In thông tin của node p, nếu p không rỗng.

3. Phương thức preVisit:

def preVisit(self,tree):

self.preOrder(tree.root)

print("")

#end def

Gọi phương thức preOrder để duyệt cây theo thứ tự trước từ gốc của cây tree.

In một dòng trống để tạo khoảng trống giữa các lần duyệt.

4. Phương thức check:

def check(self, x):

return x>=3 and x<=5

Kiểm tra xem giá trị x có nằm trong khoảng từ 3 đến 5 không.

Trả về True nếu thỏa mãn điều kiện, ngược lại trả về False.

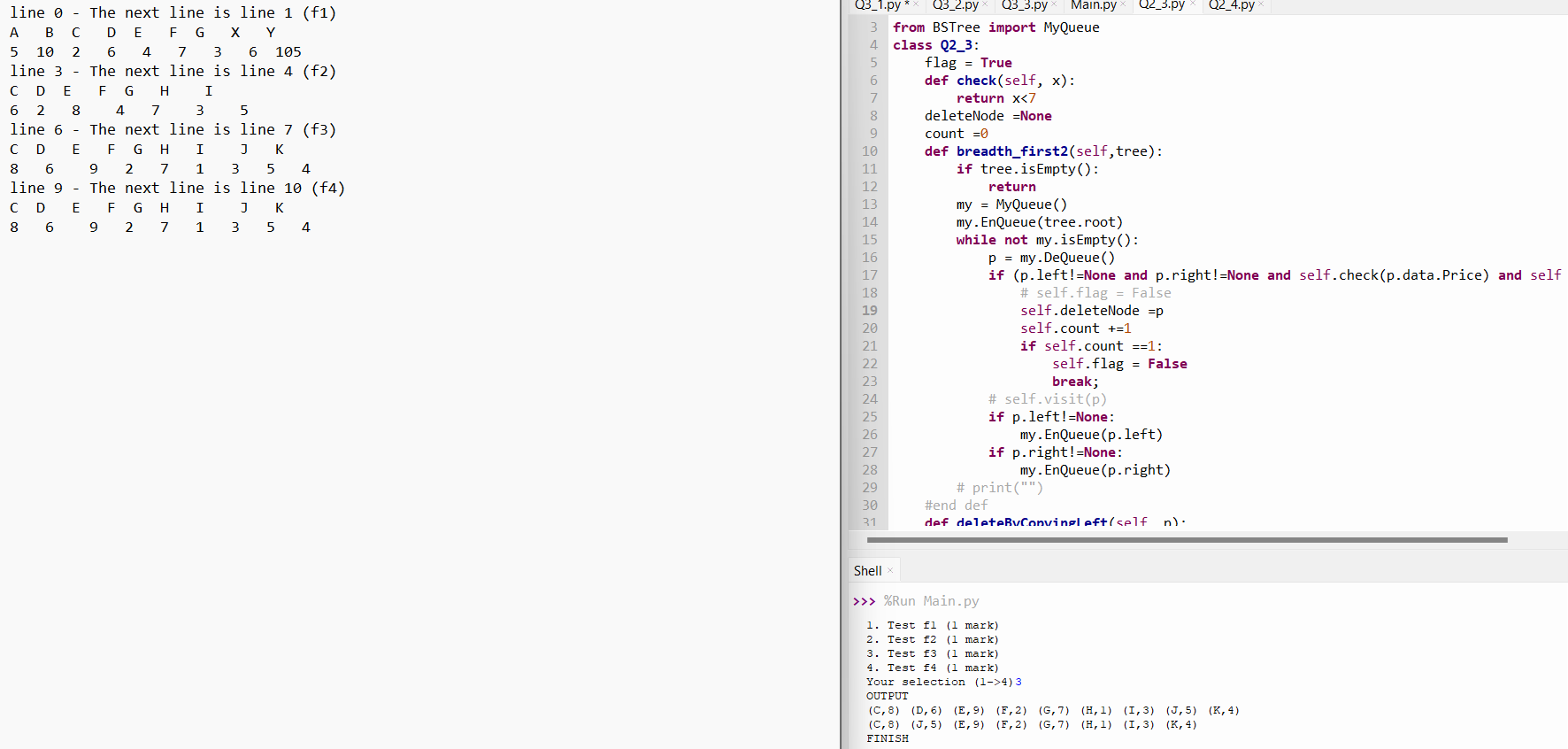
5. Phương thức f2:

def f2(self,tree):

self.preVisit(tree)

Gọi phương thức preVisit để thực hiện duyệt cây theo thứ tự trước và in các node thỏa mãn điều kiện.

Q2\_3:



1. Biến lớp flag, deleteNode, và count:

flag = True

RotateNode = None

count = 0

Biến flag được sử dụng để kiểm soát quá trình duyệt cây và dừng khi cần thiết.

Biến deleteNode sẽ giữ giá trị của node cần xóa.

Biến count đếm số lần duyệt và cũng được sử dụng để kiểm soát.

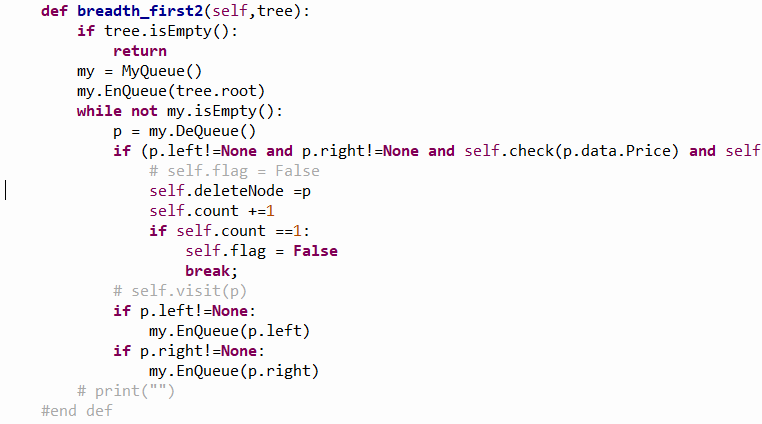
2. Phương thức check:



Kiểm tra xem giá trị x có nhỏ hơn 7 không.

Trả về True nếu thỏa mãn điều kiện, ngược lại trả về False.

3. Phương thức breadth\_first2:

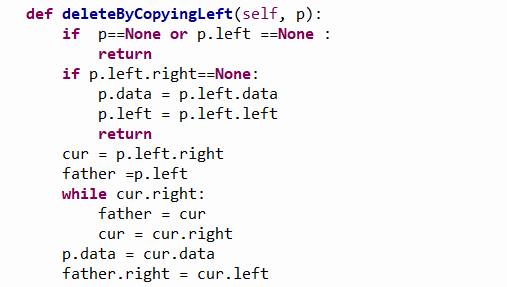


Duyệt cây theo chiều rộng (breadth-first).

Nếu node hiện tại có cả hai nhánh con (left và right), và giá trị Price thỏa mãn điều kiện (check) và biến flag đang được giữ, thì gán giá trị của node hiện tại cho deleteNode.

Dừng duyệt nếu đã tìm thấy một node thỏa mãn và count đạt đến 1.

4. Phương thức deleteByCopyingLeft:



Phương thức này được sử dụng để xóa một node từ cây bằng cách thay thế nó bằng node kế cận ở nhánh trái.

Nếu node cần xóa hoặc node trái của nó là None, thì không thực hiện gì cả.

Nếu node trái không có nhánh phải, gán giá trị của node trái cho node cần xóa và loại bỏ node trái.

Nếu node trái có nhánh phải, tìm node kế cận bên phải của node trái và thực hiện việc thay thế.

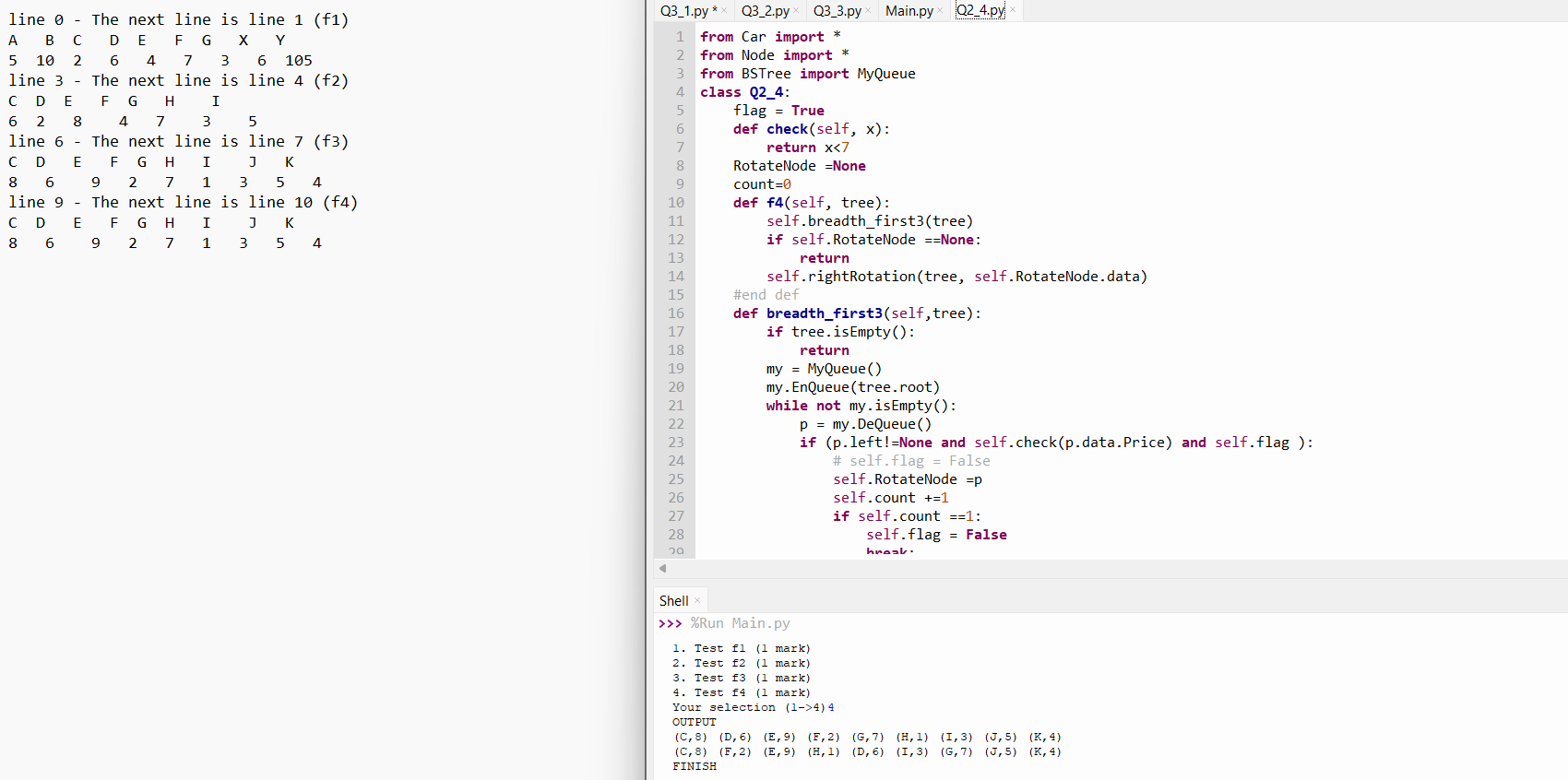
5. Phương thức f3:



Gọi phương thức breadth\_first2 để tìm và gán giá trị node cần xóa.

Gọi phương thức deleteByCopyingLeft để thực hiện việc xóa node được gán giá trị từ bước trước.

Q2\_4:



1. Biến lớp flag, RotateNode, và count:

flag = True

RotateNode = None

count = 0

Biến flag được sử dụng để kiểm soát quá trình duyệt cây và dừng khi cần thiết.

Biến RotateNode sẽ giữ giá trị của node cần thực hiện quay phải.

Biến count đếm số lần duyệt và cũng được sử dụng để kiểm soát.

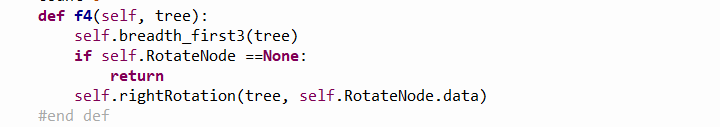
2. Phương thức check:



Kiểm tra xem giá trị x có nhỏ hơn 7 không.

Trả về True nếu thỏa mãn điều kiện, ngược lại trả về False.

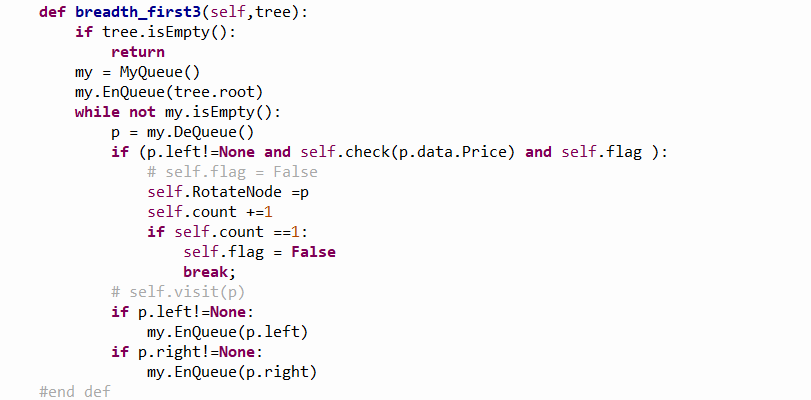
3. Phương thức f4:



Gọi phương thức breadth\_first3 để tìm và gán giá trị node cần thực hiện quay phải.

Nếu RotateNode không rỗng, gọi phương thức rightRotation để thực hiện quay phải.

4. Phương thức breadth\_first3:

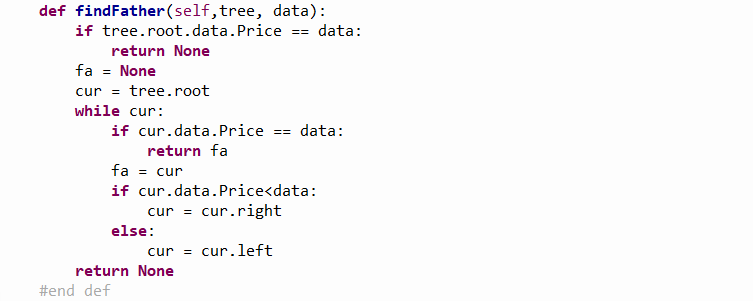


Duyệt cây theo chiều rộng (breadth-first).

Nếu node hiện tại có nhánh con trái (left) không rỗng, giá trị Price thỏa mãn điều kiện (check) và biến flag đang được giữ, thì gán giá trị của node hiện tại cho RotateNode.

Dừng duyệt nếu đã tìm thấy một node thỏa mãn và count đạt đến 1.

5. Phương thức findFather:



Tìm node cha của node có giá trị Price là data trong cây tìm kiếm nhị phân.

Nếu node cần tìm là gốc của cây, trả về None.

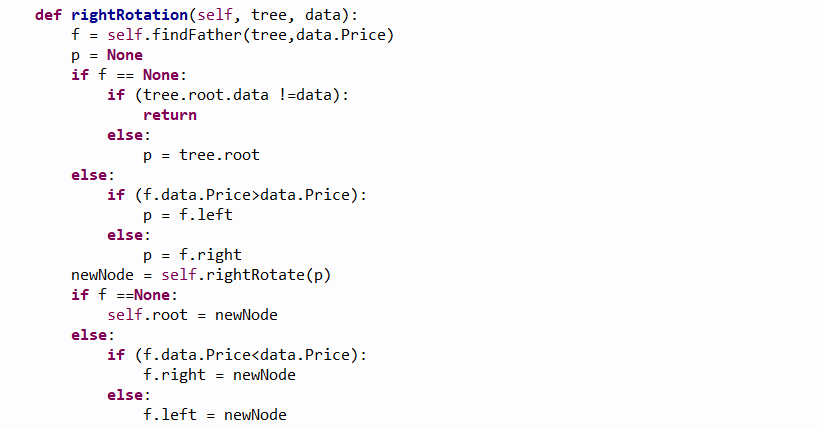
6. Phương thức rightRotate:



Thực hiện phép quay phải trên cây tìm kiếm nhị phân.

Trả về node mới là node thay thế sau khi quay phải.

7. Phương thức rightRotation:



Gọi phương thức findFather để tìm node cha của node cần thực hiện quay phải.

Xác định vị trí của node cần quay phải (p).

Gọi phương thức rightRotate để thực hiện quay phải và nhận node mới