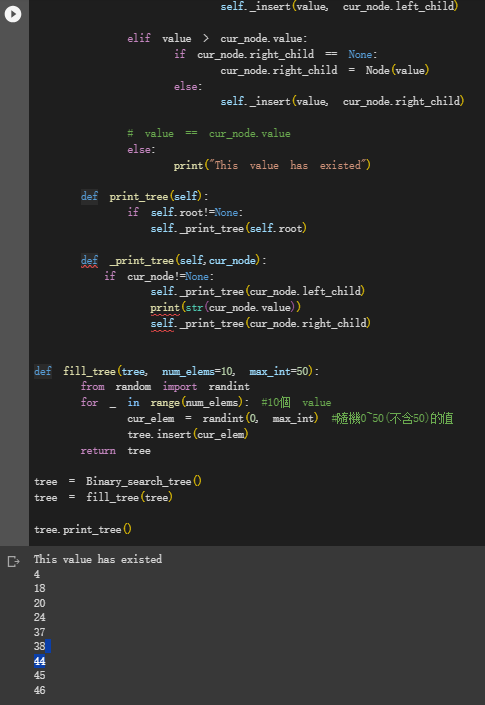
10824238 徐崇恩 10824301 董瑜祥



1.建立一個 Class Node,指向 left\_child 和 right\_child 初始化都指向 None

2.建立Class binary\_search\_tree,將 instructor 初始化為 self.root = None

3.創建一個 insert() 的function,插入資料前要先檢查樹是否為空 if self.root == None,如果 root == None 代表樹是空的,因此 value 就等於 self.root;反之如果 root != None 的話,就呼叫 private function \_insert(),接著在這個 \_insert() 內要傳入 value 和 root

4. 接下來就開始比較「傳入的value 是否 大於 cur\_node」了,  
當 value < cur\_node （value小於目前的node）  
如果 cur\_node.left\_child == None 也就是目前的node「沒有」指向 left\_child 的 pointer  
理所當然 cur\_node.left\_child = Node(value)  
如果 cur\_node.left\_child != None 也就是目前的node「有」指向 left\_child 的 pointer  
我們會把 left\_child 當成 cur\_node 並重新呼叫 \_insert()right\_child 同理,最後 else 要處裡 value 與 cur\_node 的 value相同的問題

5. 將樹印出來要先檢查 self.root 是不是 Node  
如果 self.root != None,則呼叫 \_print\_tree(self.root)  
同理一樣是用「遞迴」的方式處理，所以將它寫成private function

6. 印出一棵樹是從root開始查找  
故將cur\_node設為self.root  
則 \_print\_tree(self, cur\_node)

7. 最後在class外建立一個 function來自動建立一個二搜樹