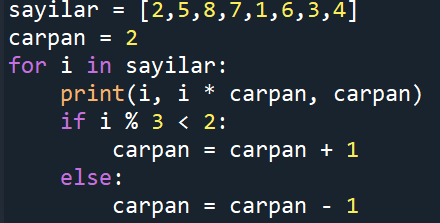
|  |  |  |
| --- | --- | --- |
| i | sayi | X |
| 2 | **4** | **2** |
| 5 | **5** | **1** |
| 7 | **-7** | **-1** |
| 1 | **0** | **0** |
| 6 | **6** | **1** |
| 3 | **6** | **2** |
| 4 | **12** | **3** |





1. **sayilar = [8,1,6,25,4,62,56,86,2,3,78,65]**

Yukarıda verilen dizinin geometrik ortalamasını bulan fonksiyonu yazınız.

**def geoOrt(dizi):**

**sonuc = 1**

**sayac = 0**

**for deger in dizi:**

**if isinstance(deger, int) or isinstance(deger, float):**

**sayac = sayac + 1**

**sonuc = sonuc \* deger**

**if sayac == 0:**

**print("Dizide hiç sayisal ifade yoktur!!")**

**sonuc = "Tanimsiz"**

**else:**

**sonuc = sonuc \*\* (1/sayac)**

**return sonuc**

1. **dizi1 = [1,5,9,12,16,18,22,26,75,89,92,95,98,100]**

**arananDeger = 98**

Yukarıdaki tanımlamalar yapılmış olsun. Bu tanımlamalara göre arananDeger değişkenini dizi1 dizisinin içerisinde arayan ve bulunması durumunda ekrana dizi içerisindeki index(kaçıncı sırada olduğunu)’ini veren Recursive(Kendi Kendini Çağıran) Binary Search(İkili Arama) algoritmasını fonksiyonunu yazınız.

**def recursiveBinarySearch(arr1, arananDeger):**

**ortanca = (int)(len(arr1)/2) #ortadaki indexi buldum**

**baslangic = 0**

**bitis = len(arr1)**

**if len(arr1) < 1:**

**return "Dizinin İcerisinde {} yoktur.".format(arananDeger)**

**if arr1[ortanca] == arananDeger:**

**return arananDeger**

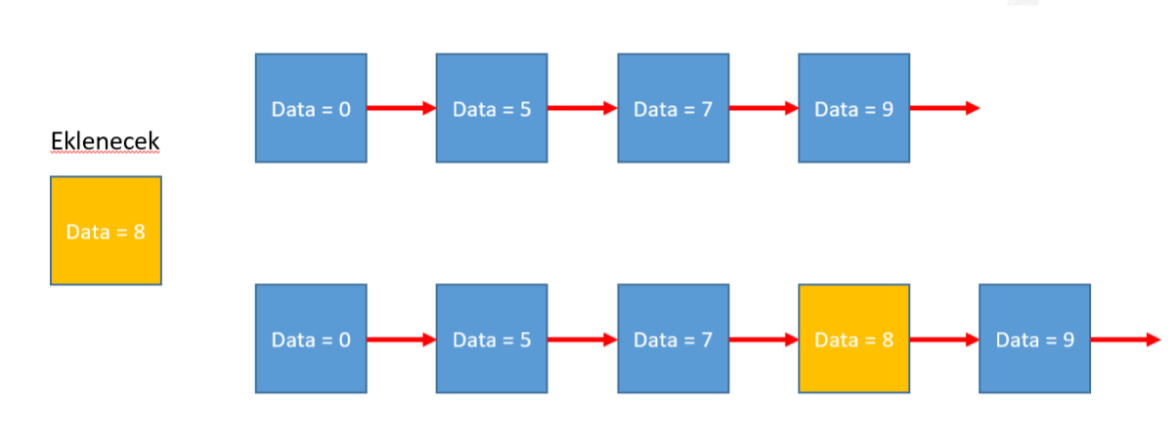
**elif arr1[ortanca] > arananDeger:**

**print("Yeni baslangic {} Yeni Bitis {}".format(baslangic, ortanca -1))**

**return recursiveBinarySearch(arr1[baslangic:ortanca -1], arananDeger)**

**elif arr1[ortanca] < arananDeger:**

**print("Yeni baslangic {} Yeni Bitis {}".format(ortanca + 1, bitis))**

 **return recursiveBinarySearch(arr1[ortanca + 1:bitis], arananDeger)**

1. Tek yönlü bağlı liste(Singulary Linked List) yapısı için sınıf(class) yapısını oluşturunuz. Bu oluşturduğunuz yapıya uygun bir şekilde aşağıdaki örnekteki gibi sıralı olacak şekilde araya eleman ekleme fonksiyonunu yazınız.

**class Node:**

**# Constructor (Yapıcı)**

**def \_\_init\_\_(self, data):**

**self.data = data**

**self.next = None**

**class LinkedList: #Bagli Liste**

**# Constructor (Yapıcı)**

**def \_\_init\_\_(self):**

**self.head = None**

**def sortedInsert(self, new\_node):**

**# Liste Bos mu?**

**if self.head is None:**

**new\_node.next = self.head**

**self.head = new\_node**

**# Yeni dugum en kucukse**

**elif self.head.data >= new\_node.data:**

**new\_node.next = self.head**

**self.head = new\_node**

**else :**

**# Eklenecek yerin belirlenmesi**

**current = self.head**

**while(current.next is not None and**

**current.next.data < new\_node.data):**

**current = current.next**

**new\_node.next = current.next**

**current.next = new\_node**

**# Listenin Yazdirilmasi**

**def printList(self):**

**temp = self.head**

**while(temp):**

**print temp.data,**

**temp = temp.next**