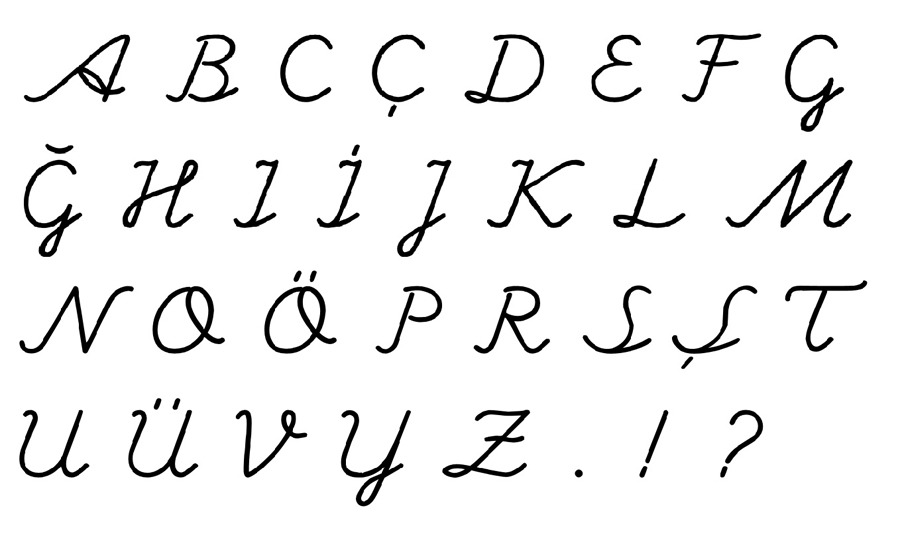
**Ertuğrul KANTAR**

**05190000086**

**Proje-2 Programcı ve Kullanıcı Kataloğu**



**Harcanan Süreler:**

**Kullanıcı Kataloğu:**

**Kaynak Kod Çıktısı:**

**İçindekiler:**

**1**

**2-5**

**6-15**

**Harcanan Süreler**

***Analiz için harcanan süre:*** *~1 saat*

***Tasarım için harcanan süre:*** *Yapılmadı-0 saat*

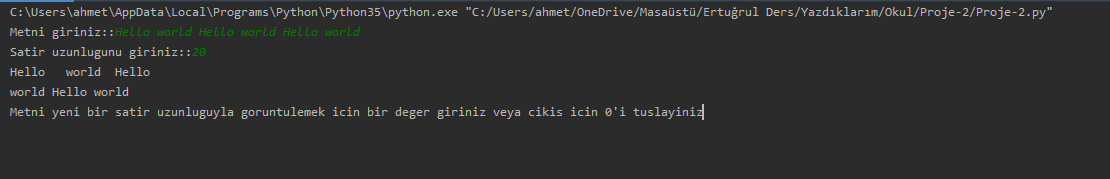
***Gerçekleştirim için harcanan süre:*** *>15 saat*

***Test için harcanan süre:*** *2-3 saat*

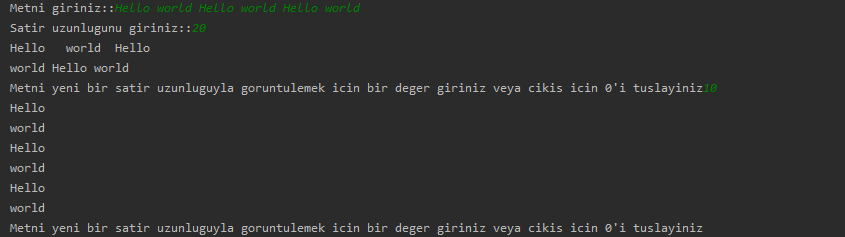
***Kataloglama için harcanan süre:*** *~1 saat*

***1***

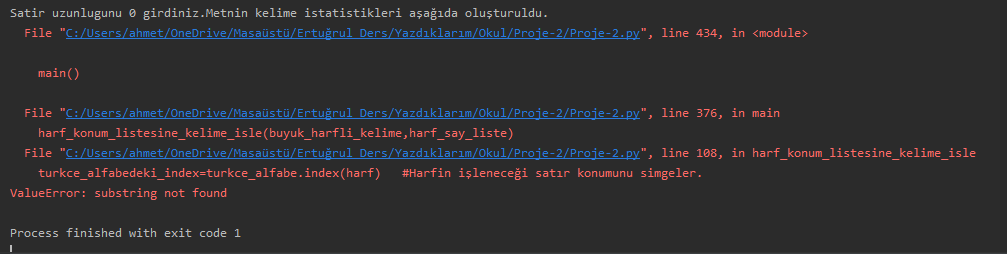
***Kullanıcı Kataloğu***



**1-**Hizalanmak istenen metin ve satır genişliği girilir. İstenen genişlikte çıktı alınır.

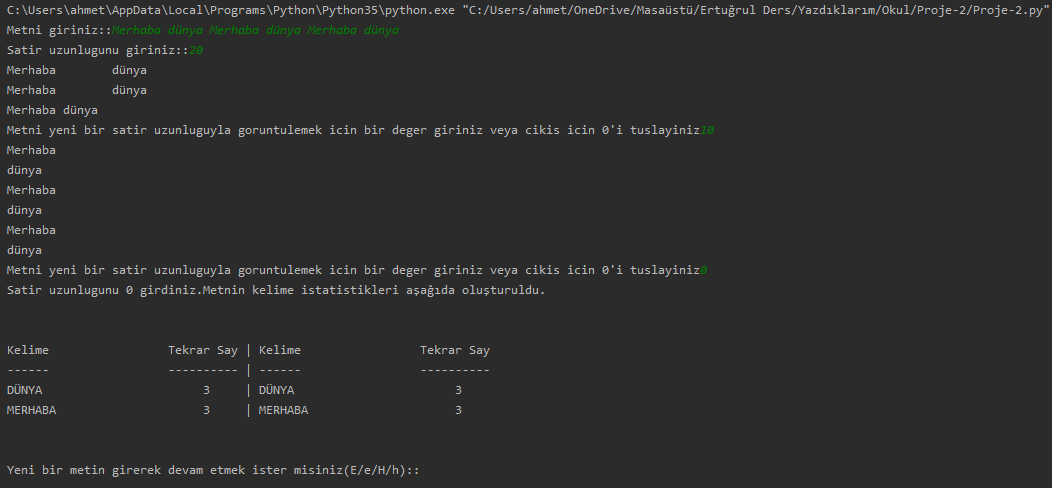


**2-**Metnin yeni bir satır genişliğinde görüntülenmesi için kullanıcıdan yeniden satır uzunluğu istenir, kullanıcı 0 değerini girene kadar işlem tekrarlanır.

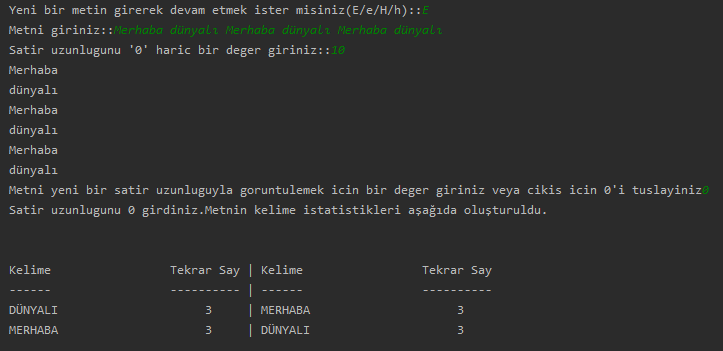


**3-**Yalnızca Türkçe karakterlerle çalışıldığından ve projede hata kontrolüne gerek olmadığından ‘w,q,x’ karakterleri programı sınırlıyor ve hataya neden oluyor.”

***2***

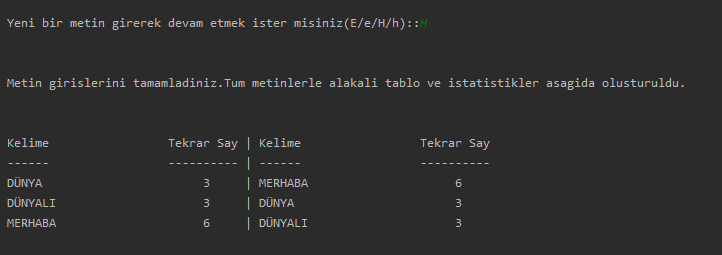


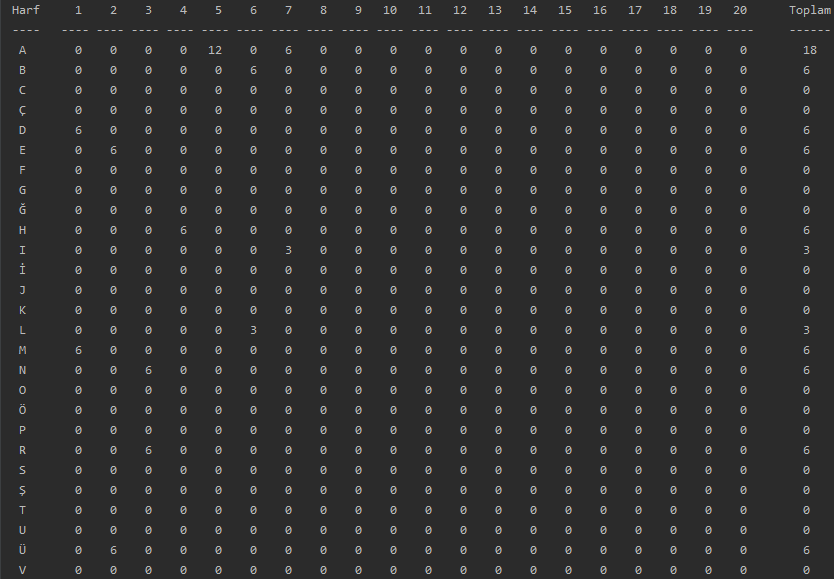
**4-**Kullanıcı satır genişliğine 0 değerini girdiğinde girdiği metindeki kelimelerin alfabetik ve tekrar sayısına göre sıralanmış listesini çıktı olarak alıyor. Kullanıcıya başka bir metin girmek isteyip istemediği soruluyor.



**5-**Yeni metin girişi yapıldığında hizalama ve girilen metindeki sözcüklerin listesi tekrar çıktı olarak alınıyor.

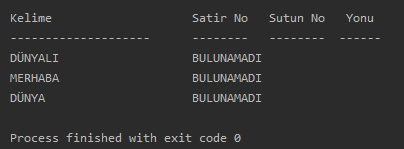
***3***

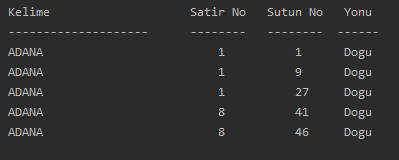




**6-**Metin girişleri de tamamlandıktan sonra kelime listeleri kullanıcının o ana kadar girdiği tüm kelimeler için oluşturuluyor. Daha sonra o ana kadar girilen her kelimedeki harflerin konumu ve kullanım toplamları çıktı olarak alınıyor.

***4***





**7-**Herhangi bir harf uzayında kelimeler 4 ana ve 4 ara yönde aranıyor. Eğer bulunursa başlangıç satırı, sütunu ve kelime harflerinin hangi yöne doğru sıralandığı çıktı olarak alınıyor.

***5***

**Program Çıktısı**

**def** kelime\_tekrari\_tablo\_sablonu():  
 *'''Kelime tekrarı tabloları için şablon.'''* print(format(**'Kelime'**,**'22'**),format(**'Tekrar Say'**,**'10'**),**'|'**,format(**'Kelime'**,**'22'**),format(**'Tekrar Say'**,**'10'**))  
 print(format(**'------'**,**'22'**),format(**'----------'**,**'10'**),**'|'**,format(**'------'**,**'22'**),format(**'----------'**,**'10'**))  
  
**def** harf\_konum\_tablo\_sablonu():  
 *'''Harf konum tablosu için şablon.'''* print(format(**'Harf'**,**'5'**),end=**''**)  
 **for** i **in** range(1,21):  
 print(format(i,**'5'**),end=**''**)  
 print(**' Toplam'**)  
 print(**'---- '**,end=**''**)  
 **for** i **in** range(20):  
 print(**'---- '**,end=**''**)  
 print(**' ------'**)  
  
  
**def** buyuk\_harfe\_donustur(kelime):  
 *'''Verilen kelimedeki harfleri büyük harfe dönüştürür.'''* kelime\_buyuk\_harfli=**''  
 for** harf **in** kelime:  
 **if** harf==**'i'**:  
 harf=**'İ'  
 elif** harf==**'ı'**:  
 harf=**'I'  
 else**:  
 harf=harf.upper()  
 kelime\_buyuk\_harfli+=harf  
 **return** kelime\_buyuk\_harfli  
  
**def** tekrar\_sayisina\_gore\_sirala(dict\_siralanacak):  
 *'''Sözlüğü tekrar sayısına göre sıralar.'''* key\_value\_listesi=[]  
 **for** key,value **in** dict\_siralanacak.items():  
 key\_value\_cifti=[key,value]  
 key\_value\_listesi.append(key\_value\_cifti)  
 key\_value\_listesi.sort(key=**lambda** pair:pair[1])  
 key\_value\_listesi.reverse()  
 **return** key\_value\_listesi  
  
**def** sozluk\_sirala(dict\_siralanacak):  
 *'''Buble sort yöntemiyle girilen sözlüğü sıralı hale getirir.'''* harfler=**"ABCÇDEFGĞHIİJKLMNOÖPRSŞTUÜVYZ,"** dict\_liste=list(dict\_siralanacak)  
 liste\_uzunlugu=len(dict\_liste)  
 yapilacak\_karsilastirma\_sayisi=liste\_uzunlugu-1  
 i=0  
  
 **while** yapilacak\_karsilastirma\_sayisi!=0:   
 incelenecek\_harf\_index=0  
 ilk\_kelime=dict\_liste[i]  
 **if** i==0:  
 swap\_yapildi\_mi=**False** yapilan\_karsilastirma\_sayisi=0  
 **try**:  
 ikinci\_kelime=dict\_liste[i+1]  
 **except**:  
 yapilacak\_karsilastirma\_sayisi-=1  
 i=0  
 **if** swap\_yapildi\_mi==**False**: *#Eğer listenin sonuna gelinmişse ve hala swap yapılmamışsa* **break** *#sıralamayı sonlandırabiliriz.* **continue** *#Öyle değilse devam edebiliriz.* flag=**True** ilk\_kelime\_harf\_sayisi=len(ilk\_kelime)  
 ikinci\_kelime\_harf\_sayisi=len(ikinci\_kelime)  
 **while** flag==**True and** yapilan\_karsilastirma\_sayisi<=yapilacak\_karsilastirma\_sayisi:  
 **try**:  
 ilk\_kelime\_incelenecek\_harf=ilk\_kelime[incelenecek\_harf\_index]  
 ikinci\_kelime\_incelenecek\_harf=ikinci\_kelime[incelenecek\_harf\_index]  
 ilk\_kelime\_incelenecek\_harf\_alfabe\_degeri=harfler.index(ilk\_kelime\_incelenecek\_harf)  
 ikinci\_kelime\_incelenecek\_harf\_alfabe\_degeri=harfler.index(ikinci\_kelime\_incelenecek\_harf)  
 **if** ilk\_kelime\_incelenecek\_harf\_alfabe\_degeri>ikinci\_kelime\_incelenecek\_harf\_alfabe\_degeri:  
 dict\_liste[i],dict\_liste[i+1]=dict\_liste[i+1],dict\_liste[i]  
 flag=**False** swap\_yapildi\_mi=**True  
 elif** ikinci\_kelime\_incelenecek\_harf\_alfabe\_degeri>ilk\_kelime\_incelenecek\_harf\_alfabe\_degeri:  
 **pass** *#Swap yapma.* flag=**False  
 else**:  
 incelenecek\_harf\_index+=1  
 flag=**True  
 except**:  
 **if** ilk\_kelime\_harf\_sayisi>ikinci\_kelime\_harf\_sayisi:  
 dict\_liste[i],dict\_liste[i+1]=dict\_liste[i+1],dict\_liste[i]  
 flag=**False** swap\_yapildi\_mi=**True  
 else**:  
 flag=**False** i+=1  
 yapilan\_karsilastirma\_sayisi+=1  
 **return** dict\_liste  
  
**def** harf\_konum\_listesi\_olustur():  
 *'''Kelimelerin işleneceği harf-konum listesini oluşturur ve döndürür.'''* harf\_say\_liste=[]  
 **for** i **in** range(29): *#29 harf için 29 satır.* bir\_harf=[0]\*20  
 harf\_say\_liste.append(bir\_harf)  
 **return** harf\_say\_liste  
  
**def** harf\_konum\_listesine\_kelime\_isle(kelime,harf\_say\_liste):  
 *'''Kelimeleri harf-konum listesindeki uygun konumlara işler.'''* turkce\_alfabe=**'ABCÇDEFGĞHIİJKLMNOÖPRSŞTUÜVYZ'** harf\_say=-1  
 **for** harf **in** kelime:  
 harf\_say+=1 *#Harfin işleneceği sütun konumunu simgeler.* turkce\_alfabedeki\_index=turkce\_alfabe.index(harf) *#Harfin işleneceği satır konumunu simgeler.* harf\_say\_liste[turkce\_alfabedeki\_index][harf\_say]+=1  
 **return** harf\_say\_liste  
  
**def** son\_kelime\_haric\_her\_kelime\_sonuna\_bosluk\_ekle(satirdaki\_kelimelerin\_listesi):  
 *'''Kelimeler split ile bölündüğünden boşluk içermiyor,içermeli.Sadece son kelimenin sonunda boşluk olmaması gerekiyor.Fonksiyon bunları hallediyor'''* satirdaki\_kelimeler\_tek\_bosluk\_ayarli\_liste=[]  
 **for** i **in** range(len(satirdaki\_kelimelerin\_listesi)):  
 **if** i!=len(satirdaki\_kelimelerin\_listesi):  
 bosluk\_ayarli\_kelime=satirdaki\_kelimelerin\_listesi[i]+**' '** satirdaki\_kelimeler\_tek\_bosluk\_ayarli\_liste.append(bosluk\_ayarli\_kelime)  
 **else**:  
 son\_kelime=satirdaki\_kelimelerin\_listesi[-1]  
 satirdaki\_kelimeler\_tek\_bosluk\_ayarli\_liste.append(son\_kelime)  
 **return** satirdaki\_kelimeler\_tek\_bosluk\_ayarli\_liste  
  
**def** bosluklari\_dagit(kelimeler\_arasi\_tek\_bosluk\_olarak\_ayarlanmis\_liste,satir\_genisligi,satir\_no\_say,olmasi\_gereken\_satir\_genisligi):  
 *'''Boşlukları satır numarasına ve genişliğine göre uygun şekilde dağıtıyor.Son satırsa veya kelime satırda tek kalmalıysa bunu sondaki 'else' hallediyor.'''* bosluk\_karakteri=**' '  
 if** satir\_no\_say%2==1 **and** satir\_no\_say!=-1 **and** len(kelimeler\_arasi\_tek\_bosluk\_olarak\_ayarlanmis\_liste)!=1:  
 j=0  
 **while** satir\_genisligi<olmasi\_gereken\_satir\_genisligi:  
 kelimeler\_arasi\_tek\_bosluk\_olarak\_ayarlanmis\_liste[j]=kelimeler\_arasi\_tek\_bosluk\_olarak\_ayarlanmis\_liste[j]+bosluk\_karakteri  
 satir\_genisligi+=1  
 j=j+1  
 **if** len(kelimeler\_arasi\_tek\_bosluk\_olarak\_ayarlanmis\_liste)-1==j:  
 j=0  
 **elif** satir\_no\_say%2==0 **and** satir\_no\_say!=-1 **and** len(kelimeler\_arasi\_tek\_bosluk\_olarak\_ayarlanmis\_liste)!=1:  
 k=-1  
 **while** satir\_genisligi<olmasi\_gereken\_satir\_genisligi:  
 kelimeler\_arasi\_tek\_bosluk\_olarak\_ayarlanmis\_liste[k]=bosluk\_karakteri+kelimeler\_arasi\_tek\_bosluk\_olarak\_ayarlanmis\_liste[k]  
 satir\_genisligi+=1  
 k=k-1  
 **if** kelimeler\_arasi\_tek\_bosluk\_olarak\_ayarlanmis\_liste[k]==kelimeler\_arasi\_tek\_bosluk\_olarak\_ayarlanmis\_liste[0]:  
 k=-1  
 **else**: *#Satır sonuna gelinmişse yani satir\_no\_say==-1 ise veya satırda tek kelime var ise* **pass** *#Hiçbir şey yapma.* bosluklari\_ayarlanmis\_liste=kelimeler\_arasi\_tek\_bosluk\_olarak\_ayarlanmis\_liste.copy() *#Direkt olarak kopyası alınan liste de gönderilebilir.* **return** bosluklari\_ayarlanmis\_liste *#Ancak yine anlaşılırlığı arttırmak için liste adını değiştirildi.***def** harf\_uzayi\_matrisi\_olustur():  
 harf\_uzayi\_dosyasi=open(**'harf\_uzayi.txt'**, **'r'**, encoding=**'utf-8'**)  
 harf\_uzayi\_matris=[]  
 harf\_uzayi\_islenecek\_satir=(harf\_uzayi\_dosyasi.readline()).rstrip(**'\n'**)  
 **while** harf\_uzayi\_islenecek\_satir!=**""**:  
 harf\_uzayi\_matris.append(harf\_uzayi\_islenecek\_satir)  
 harf\_uzayi\_islenecek\_satir=(harf\_uzayi\_dosyasi.readline()).rstrip(**'\n'**)  
 harf\_uzayi\_dosyasi.close()  
 **return** harf\_uzayi\_matris  
  
  
  
**def** uzayda\_kelime\_ara(kelime,harf\_uzayi\_matrisi):  
 buyuk\_harfli\_kelime=buyuk\_harfe\_donustur(kelime)  
 matris\_satir\_sayisi=len(harf\_uzayi\_matrisi)  
 i=0 *#Matrisin satır indexini simgeliyor.Aynı zamanda döngüde kelimenin ilk harfinin matristeki satır indexi.* dongulerde\_kullanilacak\_satir\_indexi=i  
 harf\_konum\_tupleleri\_listesi=[]  
 **while** i<matris\_satir\_sayisi:  
 aranacak\_satir=harf\_uzayi\_matrisi[i]  
 j=aranacak\_satir.find(buyuk\_harfli\_kelime[0]) *#Kelimenin ilk harfinin matristeki sütun indexi.* dongulerde\_kullanilacak\_sutun\_indexi=j  
 **if** j==-1: *#Satırda mevcut değilse* i+=1 *#Diğer satıra geç ve döngüyü başa sar.* **continue  
 else**:  
 harf\_bulundu=**True  
  
 while** harf\_bulundu==**True**:  
 dongulerde\_kullanilacak\_satir\_indexi=i  
 dongulerde\_kullanilacak\_sutun\_indexi=j  
 **try**:  
 **for** harf **in** buyuk\_harfli\_kelime:  
 **if** harf==harf\_uzayi\_matrisi[dongulerde\_kullanilacak\_satir\_indexi][dongulerde\_kullanilacak\_sutun\_indexi] **and** dongulerde\_kullanilacak\_satir\_indexi>=0 **and** dongulerde\_kullanilacak\_sutun\_indexi>=0:  
 dongulerde\_kullanilacak\_sutun\_indexi+=1  
 **else**:  
 hata\_yap *#Burada ve diğer döngülerde akışı bilerek except'e yöneltmek için* **except**: *#exception oluşturdum.* **pass  
 else**:  
 harf\_konum\_tuplesi=(i+1,j+1,3)  
 harf\_konum\_tupleleri\_listesi.append(harf\_konum\_tuplesi)  
  
 dongulerde\_kullanilacak\_satir\_indexi=i  
 dongulerde\_kullanilacak\_sutun\_indexi=j  
 **try**:  
 **for** harf **in** buyuk\_harfli\_kelime:  
 **if** harf==harf\_uzayi\_matrisi[dongulerde\_kullanilacak\_satir\_indexi][dongulerde\_kullanilacak\_sutun\_indexi] **and** dongulerde\_kullanilacak\_satir\_indexi>=0 **and** dongulerde\_kullanilacak\_sutun\_indexi>=0:  
 dongulerde\_kullanilacak\_sutun\_indexi-=1  
 **else**:  
 hata\_yap  
 **except**:  
 **pass  
 else**:  
 harf\_konum\_tuplesi=(i+1,j+1,7)  
 harf\_konum\_tupleleri\_listesi.append(harf\_konum\_tuplesi)  
  
 dongulerde\_kullanilacak\_satir\_indexi=i  
 dongulerde\_kullanilacak\_sutun\_indexi=j  
 **try**:  
 **for** harf **in** buyuk\_harfli\_kelime **and** dongulerde\_kullanilacak\_satir\_indexi>=0 **and** dongulerde\_kullanilacak\_sutun\_indexi>=0:  
 **if** harf==harf\_uzayi\_matrisi[dongulerde\_kullanilacak\_satir\_indexi][dongulerde\_kullanilacak\_sutun\_indexi]:  
 dongulerde\_kullanilacak\_satir\_indexi+=1  
 **else**:  
 hata\_yap  
 **except**:  
 **pass  
 else**:  
 harf\_konum\_tuplesi=(i+1,j+1,5)  
 harf\_konum\_tupleleri\_listesi.append(harf\_konum\_tuplesi)  
  
 dongulerde\_kullanilacak\_satir\_indexi=i  
 dongulerde\_kullanilacak\_sutun\_indexi=j  
 **try**:  
 **for** harf **in** buyuk\_harfli\_kelime:  
 **if** harf==harf\_uzayi\_matrisi[dongulerde\_kullanilacak\_satir\_indexi][dongulerde\_kullanilacak\_sutun\_indexi] **and** dongulerde\_kullanilacak\_satir\_indexi>=0 **and** dongulerde\_kullanilacak\_sutun\_indexi>=0:  
 dongulerde\_kullanilacak\_satir\_indexi-=1  
 **else**:  
 hata\_yap  
 **except**:  
 **pass  
 else**:  
 harf\_konum\_tuplesi=(i+1,j+1,1)  
 harf\_konum\_tupleleri\_listesi.append(harf\_konum\_tuplesi)  
  
 dongulerde\_kullanilacak\_satir\_indexi=i  
 dongulerde\_kullanilacak\_sutun\_indexi=j  
 **try**:  
 **for** harf **in** buyuk\_harfli\_kelime:  
 **if** harf==harf\_uzayi\_matrisi[dongulerde\_kullanilacak\_satir\_indexi][dongulerde\_kullanilacak\_sutun\_indexi] **and** dongulerde\_kullanilacak\_satir\_indexi>=0 **and** dongulerde\_kullanilacak\_sutun\_indexi>=0:   
 dongulerde\_kullanilacak\_satir\_indexi+=1  
 dongulerde\_kullanilacak\_sutun\_indexi+=1  
 **else**:  
 hata\_yap  
 **except**:  
 **pass  
 else**:  
 harf\_konum\_tuplesi=(i+1,j+1,4)  
 harf\_konum\_tupleleri\_listesi.append(harf\_konum\_tuplesi)  
  
 dongulerde\_kullanilacak\_satir\_indexi=i  
 dongulerde\_kullanilacak\_sutun\_indexi=j  
 **try**:  
 **for** harf **in** buyuk\_harfli\_kelime:  
 **if** harf==harf\_uzayi\_matrisi[dongulerde\_kullanilacak\_satir\_indexi][dongulerde\_kullanilacak\_sutun\_indexi] **and** dongulerde\_kullanilacak\_satir\_indexi>=0 **and** dongulerde\_kullanilacak\_sutun\_indexi>=0:  
 dongulerde\_kullanilacak\_satir\_indexi-=1  
 dongulerde\_kullanilacak\_sutun\_indexi+=1  
 **else**:  
 hata\_yap  
 **except**:  
 **pass  
 else**:  
 harf\_konum\_tuplesi=(i+1,j+1,2)  
 harf\_konum\_tupleleri\_listesi.append(harf\_konum\_tuplesi)  
  
 dongulerde\_kullanilacak\_satir\_indexi=i  
 dongulerde\_kullanilacak\_sutun\_indexi=j  
 **try**:  
 **for** harf **in** buyuk\_harfli\_kelime:  
 **if** harf==harf\_uzayi\_matrisi[dongulerde\_kullanilacak\_satir\_indexi][dongulerde\_kullanilacak\_sutun\_indexi] **and** dongulerde\_kullanilacak\_satir\_indexi>=0 **and** dongulerde\_kullanilacak\_sutun\_indexi>=0:  
 dongulerde\_kullanilacak\_satir\_indexi+=1  
 dongulerde\_kullanilacak\_sutun\_indexi-=1  
 **else**:  
 hata\_yap  
 **except**:  
 **pass  
 else**:  
 harf\_konum\_tuplesi=(i+1,j+1,6)  
 harf\_konum\_tupleleri\_listesi.append(harf\_konum\_tuplesi)  
  
 dongulerde\_kullanilacak\_satir\_indexi=i  
 dongulerde\_kullanilacak\_sutun\_indexi=j  
 **try**:  
 **for** harf **in** buyuk\_harfli\_kelime:  
 **if** harf==harf\_uzayi\_matrisi[dongulerde\_kullanilacak\_satir\_indexi][dongulerde\_kullanilacak\_sutun\_indexi] **and** dongulerde\_kullanilacak\_satir\_indexi>=0 **and** dongulerde\_kullanilacak\_sutun\_indexi>=0:  
 dongulerde\_kullanilacak\_satir\_indexi-=1  
 dongulerde\_kullanilacak\_sutun\_indexi-=1  
 **else**:  
 hata\_yap  
 **except**:   
 **pass  
 else**:  
 harf\_konum\_tuplesi=(i+1,j+1,8)  
 harf\_konum\_tupleleri\_listesi.append(harf\_konum\_tuplesi)  
  
 j=aranacak\_satir.find(buyuk\_harfli\_kelime[0],j+1) *#Bu satırda kelimenin ilk harfinden daha da var mı diye kontrol.* **if** j==-1:  
 harf\_bulundu=**False** *#Yok ise döngüden çıkıp* i+=1 *#diğer satır ile üstte uğraşıyor.* **else**:  
 harf\_bulundu=**True** *#Bulursa dönmeye devam ediyor.* **return** harf\_konum\_tupleleri\_listesi  
  
**def** harf\_uzayi\_aramasi\_tuple\_inin\_ikinci\_indexini\_adlandir\_ve\_yazdir(sayi):  
 **if** sayi==1:  
 print(**'Kuzey'**)  
 **elif** sayi==2:  
 print(**'Kuzeydogu'**)  
 **elif** sayi==3:  
 print(**'Dogu'**)  
 **elif** sayi==4:  
 print(**'Guneydogu'**)  
 **elif** sayi==5:  
 print(**'Guney'**)  
 **elif** sayi==6:  
 print(**'Guneybati'**)  
 **elif** sayi==7:  
 print(**'Bati'**)  
 **else**:  
 print(**'Kuzeybati'**)  
  
**def** main():  
 i=0  
 satir\_no\_say=0  
 dict\_lokal={}  
 dict\_global={}  
 harf\_say\_liste=harf\_konum\_listesi\_olustur() *#Bu liste 'harf\_konum\_listesine\_eleman\_isle' fonksiyonuna argüman olarak verilecek.* metin=input(**'Metni giriniz::'**)  
 liste=metin.split()  
 olmasi\_gereken\_satir\_uzunlugu=int(input(**'Satir uzunlugunu giriniz::'**))  
 flag=**True** *#2.while'a ilk giriş için atama.* **while** olmasi\_gereken\_satir\_uzunlugu!=0:  
 satir=**''  
 while** flag==**True**:  
 **try**:  
 kelime\_eklenecek=**' '**+liste[i]  
 **except**:  
 **break** satir=satir+kelime\_eklenecek  
 **if** satir[0]==**' '**: *#Kelimeleri satıra eklerken boşlukları kelimelerin sağına değil soluna koydum.Bu satırın ilk karakteri için problem oluşturuyor,* satir=satir.lstrip() *#Bu sebeple ilk karakter boşluk ise onu silmeliyim.* **try**:  
 kelime\_sonraki=**' '**+liste[i+1]  
 **except**:   
 satir\_no\_say=-1 *#Eğer bir sonraki kelime yok ise satır sonu gelmiş demektir.Bunu saçma bir değer yaparak 'bosluklari\_dagit' fonksiyonunda* **break** *#son satırı ayırt edebilmek için kullandım.* sonraki\_satir\_uzunlugu=len(satir)+len(kelime\_sonraki)  
 **if** sonraki\_satir\_uzunlugu<=olmasi\_gereken\_satir\_uzunlugu:  
 flag=**True** *#Satır dolmamış ve sıradaki kelime boşluğu ile birlikte satıra sığıyor.Devam edilmeli.* i+=1  
 **else**:  
 flag=**False** *#Satır tamamlandı.* satir\_no\_say+=1 *#O yüzden artık kaçıncı satır olduğunu saydırabilirim.* satir\_genisligi=len(satir)  
 satirdaki\_kelimelerin\_listesi=satir.split()  
 kelimeler\_arasi\_tek\_bosluk\_olarak\_ayarlanmis\_liste=son\_kelime\_haric\_her\_kelime\_sonuna\_bosluk\_ekle(satirdaki\_kelimelerin\_listesi)  
 bosluklari\_ayarlanmis\_kelimeler\_liste=bosluklari\_dagit(kelimeler\_arasi\_tek\_bosluk\_olarak\_ayarlanmis\_liste,satir\_genisligi,satir\_no\_say,olmasi\_gereken\_satir\_uzunlugu)  
 **for** el **in** bosluklari\_ayarlanmis\_kelimeler\_liste:  
 print(el,end=**''**)  
 print(**'\n'**,end=**''**)  
 **if** satir\_no\_say!=-1:  
 flag=**True** i+=1  
 **else**:  
 olmasi\_gereken\_satir\_uzunlugu=int(input(**'''Metni yeni bir satir uzunluguyla goruntulemek icin bir deger giriniz veya cikis icin 0'i tuslayiniz'''**))  
 **if** olmasi\_gereken\_satir\_uzunlugu!=0:  
 i=0 *#Metni satır genişliğine göre tekrar düzenleyebilmek için bu* satir\_no\_say=0 *#ve bu değişkenler sıfırlandı.* flag=**True** *#2.while'a tekrar giriş için atama.* **if** olmasi\_gereken\_satir\_uzunlugu==0:  
 print(**'Satir uzunlugunu 0 girdiniz.Metnin kelime istatistikleri aşağıda oluşturuldu.'**)  
 print(**'\n'**)  
 **for** eleman **in** liste:  
 buyuk\_harfli\_kelime=buyuk\_harfe\_donustur(eleman)  
 dict\_lokal[buyuk\_harfli\_kelime]=dict\_lokal.get(buyuk\_harfli\_kelime,0)+1  
 dict\_global[buyuk\_harfli\_kelime]=dict\_global.get(buyuk\_harfli\_kelime,0)+1  
 harf\_konum\_listesine\_kelime\_isle(buyuk\_harfli\_kelime,harf\_say\_liste)  
 local\_dict\_in\_alfabetik\_siralanmis\_key\_listesi=sozluk\_sirala(dict\_lokal)  
 local\_dict\_in\_tekrar\_sayisina\_gore\_siralanmis\_listesi = tekrar\_sayisina\_gore\_sirala(dict\_lokal)  
 kelime\_tekrari\_tablo\_sablonu()  
 **for** i **in** range(len(local\_dict\_in\_alfabetik\_siralanmis\_key\_listesi)): *#Sözcük sayısı kadar eleman yazdıracağız her ikisi için de.* alfabetik\_siralanmis\_kelime=local\_dict\_in\_alfabetik\_siralanmis\_key\_listesi[i]   
 tekrar\_sayisina\_gore\_siralanmis\_kelime=local\_dict\_in\_tekrar\_sayisina\_gore\_siralanmis\_listesi[i][0]  
 print(format(alfabetik\_siralanmis\_kelime,**'22'**),format(dict\_lokal[alfabetik\_siralanmis\_kelime],**'6'**),**' | '**,end=**''**)  
 print(format(tekrar\_sayisina\_gore\_siralanmis\_kelime,**'23'**),format(dict\_lokal[tekrar\_sayisina\_gore\_siralanmis\_kelime],**'5'**))  
 print(**'\n'**)  
 yeni\_metinle\_devam=input(**'Yeni bir metin girerek devam etmek ister misiniz(E/e/H/h)::'**)  
 **if** yeni\_metinle\_devam==**'E' or** yeni\_metinle\_devam==**'e'**:  
 metin=input(**'Metni giriniz::'**)  
 olmasi\_gereken\_satir\_uzunlugu=int(input(**'''Satir uzunlugunu '0' haric bir deger giriniz::'''**))  
 liste=metin.split()  
 dict\_lokal={}  
 i=0  
 satir\_no\_say=0  
 print(**'\n'**)  
 print(**'Metin girislerini tamamladiniz.Tum metinlerle alakali tablo ve istatistikler asagida olusturuldu.'**)  
 print(**'\n'**)  
  
 kelime\_tekrari\_tablo\_sablonu()  
 global\_dict\_in\_alfabetik\_siralanmis\_key\_listesi=sozluk\_sirala(dict\_global)  
 global\_dict\_in\_tekrar\_sayisina\_gore\_siralanmis\_listesi=tekrar\_sayisina\_gore\_sirala(dict\_global)  
 **for** i **in** range(len(global\_dict\_in\_alfabetik\_siralanmis\_key\_listesi)): *#Sözcük sayısı kadar eleman yazdıracağız her ikisi için de.* alfabetik\_siralanmis\_kelime = global\_dict\_in\_alfabetik\_siralanmis\_key\_listesi[i]  
 tekrar\_sayisina\_gore\_siralanmis\_kelime=global\_dict\_in\_tekrar\_sayisina\_gore\_siralanmis\_listesi[i][0]  
 print(format(alfabetik\_siralanmis\_kelime, **'22'**), format(dict\_global[alfabetik\_siralanmis\_kelime],**'6'**),**' | '**,end=**''**)  
 print(format(tekrar\_sayisina\_gore\_siralanmis\_kelime, **'23'**),format(dict\_global[tekrar\_sayisina\_gore\_siralanmis\_kelime], **'5'**))  
 print(**'\n'**)  
  
  
 harf\_konum\_tablo\_sablonu()  
 turkce\_alfabe=**'ABCÇDEFGĞHIİJKLMNOÖPRSŞTUÜVYZ'  
 for** i **in** range(len(turkce\_alfabe)):  
 print(**''**,turkce\_alfabe[i],**' '**,end=**''**)  
 goruntulenecek\_satir=harf\_say\_liste[i]  
 **for** i **in** goruntulenecek\_satir:  
 print(format(i,**'5'**),end=**''**)  
 print(**' '**,format(sum(goruntulenecek\_satir),**'<5'**))  
  
  
  
 harf\_uzayi\_matrisi=harf\_uzayi\_matrisi\_olustur()  
 print(**'\n'**)  
 print(format(**'Kelime'**,**'25'**),format(**'Satir No'**,**'10'**),format(**'Sutun No'**,**'10'**),format(**'Yonu'**,**'8'**))  
 print(**'-------------------- -------- -------- ------'**)   
 **for** el **in** dict\_global.keys():  
 harf\_uzay\_konum\_listesi=list(uzayda\_kelime\_ara(el,harf\_uzayi\_matrisi))  
 **if** len(harf\_uzay\_konum\_listesi)!=0:  
 **for** tuple1 **in** harf\_uzay\_konum\_listesi:  
 print(format(el,**'29'**),format(tuple1[0],**'<10'**),format(tuple1[1],**'<7'**),end=**''**)  
 tuple1\_index\_2\_degeri=int(tuple1[2])  
 harf\_uzayi\_aramasi\_tuple\_inin\_ikinci\_indexini\_adlandir\_ve\_yazdir(tuple1\_index\_2\_degeri)  
 **else**:  
 print(format(el,**'25'**),**'BULUNAMADI'**)  
  
main()

***6***

***7***

***8***

***9***

***10***

***11***

***12***

***13***

***14***

***15***