**T.C**

**KOCAELİ SAĞLIK VE TEKNOLOJİ ÜNİVERSİTESİ**

**MÜHENDİSLİK FAKÜLTESİ**

**Python Dili ile Console Menü Oluşturma**

**Hazırlayan**

**EMRAH ŞAHİN MÜGE ÖZLER**

220502025 210501004

**DERS SORUMLUSU**

**PRF.DR. HÜSEYİN TARIK DURU**

**30 EKİM 2023**

**İÇİNDEKİLER**

[1.](#_1fob9te) ÖZET 3

[2.](#_3znysh7) GİRİŞ 3

[3.](#_2et92p0) YÖNTEM 3

[3.1](#_tyjcwt) Bir Liste İçinde İstenilen Sıradaki En Küçük Elemanı Bulma 3

[3.2](#_3dy6vkm) Bir Liste İçinde İstenilen Sayıya En Yakın Sonucu Veren İki Liste Elemanının Toplamını Bulma 4

[3.3](#_1t3h5sf) Bir Liste İçerisinde En Çok Tekrar Eden Elemanı Bulma 4

[3.4](#_4d34og8) Matris Çarpımı 4

[3.5](#_2s8eyo1) Bir Text Dosyası İçinde Kelimelerin Tekrar Etme Değerlerinin Bulma 5

[3.6](#_17dp8vu) Liste İçindeki En Küçük Değeri Bulma 5

[3.7](#_3rdcrjn) Karekök Bulma 6

[3.8](#_26in1rg) En Büyük Ortak Bölen Bulma 6

[3.9](#_lnxbz9) Bir Sayının Asal Olup Olmadığını Bulma 6

[3.10](#_35nkun2) Daha Hızlı Fibonacci Dizisi Bulma 7

[3.11](#_1ksv4uv) Çıkış Fonksiyonu 7

[3.12](#_44sinio) Arayüz Oluşturma 7

[4.](#_2jxsxqh) SONUÇ VE ÖĞRENİLEN DERSLER 8

[5.](#_z337ya) KAYNAKÇA 8

# ÖZET

Bu projedeki amacımız kodlama dillerinden bir olan Python kullanarak istenilen işlemler için giriş menüsü hazırlayıp daha sonra her işlem için ayrı fonksiyon tanımlayıp bu işlemlerin gerçekleşmesini sağlamaktır.

# GİRİŞ

Bu projede console menüye karar verirken yapılması istenen fonksiyonları seçenekler şeklinde sıralayarak proje taslağını oluşturma başlandı. Menü sıralaması yapıldıktan sonra yapılması istenen her işlem için çalışma butonu yerleştirmeyi planladık. Menüden seçilen her işlem için fonksiyon giriş verileri ve ara işlemleri tanımlanarak yaptığı işlemin sonucunu ekrana döndürmeyi amaçlandı. En son çıkış işlemi için seçenek belirleyip menünün istenildiği kadar çalışabilmesini sağlandı.

# YÖNTEM

Bu aşamada ilk olarak console menüsünü oluşturduk. Bu menüyü açılış ekranı olarak tasarladık. İlk sınıfımız ekran penceresi olarak belirleyip kodumuzu oluşturduk. Daha sonra istenilen her işlem için sınıflar oluşturarak burada yapılması gereken veri girişlerini ve işlemlerin kodlamasını oluşturduk

## Bir Liste İçinde İstenilen Sıradaki En Küçük Elemanı Bulma

Sınıfımızı console menü içerisinde tanımladığımız \_init\_ metodu ile ilgili olacak şekilde ana sınıfımızı çağırma işlemini başlattık. K-Küçük işlemi olarak başlığını kullandık. Daha düzenli görmek adına pencerenin konum ve boyutunu konumlandırıldı. Kullanıcıdan, kaçıncı en küçük elemanı bulunması isteniyor ise onun değerini girebilmesini sağlayan kod oluşturuldu. Daha sonra bir liste verisi için kullanıcıdan giriş işlemlerini yapabilmesi için komut grupları oluşturuldu. Oluşturulan butona basıldığında hesaplama işlemini gerçekleyen hesaplama işleminin fonksiyonunu oluşturduk. Bu kısımda istenilen k’ıncı en küçük elemanın girilen listenin eleman sayısının içinde mi kalıyor onun kontrolünü gerçekleştirdik. Eğer seçilen değer uygun ise fonksiyona gidilir uygun değilse yeni bir değer girilmesi istenir. Fonksiyon işleminde ise listeyi küçükten büyüğe doğru sıralamasını yaptıktan sonra girilen girdi elemanına göre yani istenilen girdi sıfırdan küçükse direk girdiyi indeks kabul eder ama girilen girdi sıfırdan büyükse k. Elemanın bir eksiğini indeks kabul eden liste elemanı sonuç olarak geri döndürülür.

## Bir Liste İçinde İstenilen Sayıya En Yakın Sonucu Veren İki Liste Elemanının Toplamını Bulma

İlk olarak pencere başlığımız ve boyutunu ayarlanır. Girdi etiketleri, girdi alanları, işlem butonu ve sonuç alanı oluşturulur. Kullanıcıdan girdiler alınır, ilk istenilen sayı daha sonra liste istenir. Oluşturulan en\_yakin\_cift fonksiyonu çağırılarak sonuç yazdırılır. Bu fonksiyonun içinde kullanıcının istediği sayı ve listeyi parametre olarak kabul eder. İlk sayı listesini sıralar. Ardından kullanıcıdan girileni tam sayıya dönüştürür. Burada en kücük fark oluşturduk çünkü toplanan sayı çiftleri ile kullanıcının istediği sayı arasındaki farkı bularak bunların arasındaki en küçüğünü bulma işlemini gerçekleştirmek istendi. En son en küçük fark hesaplandığında bize bu sayı çiftini döndürür.

## Bir Liste İçerisinde En Çok Tekrar Eden Elemanı Bulma

Daha önceki yöntemlerde uyguladığımız ekran pencere başlığı, boyutu konumu gibi komutlar girilerek standartlaştırma sağlanır. Kullanıcıdan liste elemanlarının girilmesini sağlayan kod grubu oluşturulur. Bunu salt okuyan bir metin alanına yönlendirildi. Kullanıcıdan virgül ile ayrılarak istenilen liste tam sayı listesine dönüştürüldü. List comprehension kullanılarak 1 den fazla tekrar eden elemanlar bir listeleye eklendi. Ardından ‘set’ fonksiyonu kullanarak aynı değerden birden fazla var ise onu silerek tek değere indirdi. İşlem sonucunda tekrar eden elemanların listesi geri döndürüldü.

## Matris Çarpımı

Matris çarpımı için yukarıda açıkladığımız yöntemlerdeki gibi bir sınıf oluşturup bu sınıfın içinde çarpım işlemlerini gerçekleştirildi. Bu sınıfın yer aldığı pencere başlığı, boyutu ve konumu oluşturuldu. Matrislerin satır ve sütun sayılarının girilmesi için gerekli etiketler oluşturuldu. Matris çarpımını yapabilmek için bir buton işlevi koyarak çarpma işlemini gerçekleyen fonksiyon çağırıldı. Bu fonksiyon içinde kullanıcıdan alınan girdiler, satır ve sütün şeklinde düzenlendi. Kullanıcı tarafından girilen listeler ‘map’ fonksiyonu kullanarak integer değere dönüştürülür ve ilk matrisin sütun sayısı ile ikinci matrisin satır sayısının eşit olmasına bakılır. Eşit ise işleme devam edilir, eşit değil ise işlem sonlandırılır. Eşit olması halında listeler uygun şekilde dönüştürülüp yani list comprehsion ile kullanıcının girdiği değere göre satır ve sütuna ayrılacak şekilde ayarlanıp çarpımını hesaplayan işlem fonksiyonuna girer. Matris çarpımı hesaplanarak text şeklinde ekrana yazdırılması için for döngüsü içinde tekrar ‘map’ fonksiyonunu kullanıp çarpılan matris sonuçlarını metine dönüştürüp metin alanına yazdırılır. ‘append’ metottunu kullanma nedeni ise for döngüsü her çalıştığında içindeki veriyi kaybetmeden metin alanına eklememizdir.

## Bir Text Dosyası İçinde Kelimelerin Tekrar Etme Değerlerinin Bulma

Bu kısımda dosyadan alınıp okuma işlemi gerçekleştirmemiz gerektiği için bir dosya oluşturup bunun okunmasını sağlayan komut etiketleri oluşturulur, kullanılan ‘encoding=’utf-8’’ dosyayı Türkçe karakterler olduğunu bilerek okumamızı sağlıyor. Alınan dosyadaki metin bilgisinde kelime sayılarını hesaplaması için bir buton yardımıyla çağırılan fonksiyon oluşturduk. Bu fonksiyon içinde dosyayı açıp metin okunduktan sonra boşlukları saymayacak şekilde ‘.split()’ metotunu kullanarak kelime listesi oluşturuldu. Bu fonksiyonun içine işimizi kolaylaştırması için sayac(sonuc, eleman) adında ayrıca bir fonksiyon tanımladık. Bu fonksiyon iki parametre alır. ‘sonuc’ kelime frekanslarının saklandığı bir sözlüktür. ‘eleman’ işlenen kelimeyi temsil eder. Bu ‘sonuc’ adlı sözlüğümüze, ‘get()’ metotunu kullanarak her kelimenin kaç kez geçtiğini hesaplattık ve atadık. Tabi ‘sayac’ fonksiyonumuzu ‘reduce’ fonksiyonunu kullanırken çağırdık ve kullandık. ‘reduce’ fonksiyonunun kullanımı ise de ilk parametresi işlem yapılacak fonksiyon olur. İkinci parametresi, bu işlemin uygulanacağı öğelerin sırasını temsil eder. ‘reduce’ bu sıradaki öğeleri birer birer işlemek için bu sırayı kullanır. Son parametresi, işlemin ilk adımında kullanılacak başlangıç değerini temsil eder.

Her adımda ‘sayac’ işleminde, mevcut kelimenin frekansı zaten sonuc sözlüğünde saklanır. Yeni kelimeye geçildiğinde, bu işlem, yeni kelimenin frekansını hesaplamak ve sözlüğe eklemek için kullanılır. Bu nedenle, reduce işlemi sonunda kelime\_sayisi adındaki bir sözlük elde edilir ve bu sözlük, her kelimenin frekansını içerir. Bağıntı şu şekildedir: reduce, sayac işlevini kullanarak metin dosyasındaki her kelimenin frekansını hesaplar. İşlem her kelime için sırayla çalışır, akümülatör olan sonuc adındaki sözlüğü günceller ve sonunda bu sözlüğü döndürür. Yani, reduce, işlevi her kelime üzerinde uygulayarak sonuçları biriktirir ve nihai sonucu elde eder. Ve bu sonucu ‘return’ ile bize geri döndürür.

## Liste İçindeki En Küçük Değeri Bulma

Bu sınıf içerisinde standart ekran penceresi ve konum durumu oluşturulup, kullanıcıdan virgül ile ayrılarak alınan listeyi tam sayıya çevirip bunların en küçük elemanını bulmayı sağlayan fonksiyon yazılır. Bu yöntemde klasik buble sort algoritması kullanıldı. Yani sıralama yapmak yerine liste içinde en küçük olanını bulan bir yöntemdir. Listede tek eleman varsa bu listenin en küçük elemanı zaten kendisidir ve sonuç olarak liste elemanı döndürülür. Birden fazla elemana sahip ise liste içinde bir döngü oluşturulur elemanlar sıralanmaya başlanır sıralama işleminin sonucunda listenin sonunda liste içerisindeki en küçük eleman yer alır. Bu işlem liste boyutu azaltılarak rekürsif olarak devam ettirilir.

## Karekök Bulma

Karekök hesaplaması için kullanıcıdan bir değer girilmesini sağlayan ve hesaplama işlemi için fonksiyonun bağlandığı buton kısımları sınıf içerisinde yer alacak şekilde sınıf bloğu oluşturulur. Bu bloğun içinde karekök hesaplamamıza yardımcı olacak Newton Raphson yöntemini kullanıldı. Bu method için karekök fonksiyonu içinde karekökü bulması istenilen sayı, tahmini ilk değer, hesaplamanın durabilmesi için kabul edilebilir hata eşik değeri (10-10 gibi). İlk if bloğunda hata değeri ile eşik değerini karşılaştırıp eğer doğruysa sonucu geri döndürüyor ve fonksiyonun çalışması duruyor. İkinci if bloğunda ise bizim tanımladığımız bir maxiter değeri ile rekürsif için kullanılan tolerans değerini karşılaştırıp işlem sonuçlanıncaya kadar devam ediyor. Bu yöntemin tekrar etme sayısı, iterasyon sayısı gibi parametreleri kapsar. Böylece hesaplama işlemi eşik değeri veya maxiter iterasyon sayısı kadar devam eder.

## En Büyük Ortak Bölen Bulma

Bu hesaplamanın yapılacağı sınıf bloğu oluşturulur. Sınıf bloğu içerisinde kullanıcıdan sayı girilmesini sağlayan kod etiketleri oluşturulur. EBOB hesabı için en az iki sayı olması gerektiğinden kullanıcıdan bir sayı daha istenir. Fonksiyonun bağlandığı buton sayesinde hesaplama işlemi çağırılır. Bu fonksiyon içinde sayılardan biri sıfır mı diye kontrolü yapılır. Eğer sıfır ise ebob değeri geri döndürülür. İkinci kontrol ise bu iki sayı eşit mi diye kontrol edilir çünkü eğer eşit ise ebob sayıların kendisidir. Üçüncü kontrol de ise bu iki sayı eşit değil ve sıfırdan farklı ise mod alma işlemi kullanılarak yapılır. İkinci sayı ilk sayının modunu alır. Bu işlem, her adımda bir önceki adıma göre daha küçük sayılar ile devam eder.

## Bir Sayının Asal Olup Olmadığını Bulma

Bu sınıf bloğunun içinde girilen sayının asal sayı olup olmadığına bakmak için girilen sayı 2 mi diye kontrol edildi. Sayı 2 ise sayını asal sayı olduğu bilgisi döndürüldü. İkinci olarak sayı için bir döngü oluşturup kendisine kadar olan sayılara bölünüp bölünmediği kontrolü yapılır. Temel durumlar dışında ‘elif i \* i > int:’ satırının yaptığı işlem fonksiyonun belli bir aşamadan sonra çalışmasının gerek olmadan durdurmasıdır. Bu döngüyü de rekürsif şekilde oluşturuyoruz. Eğer sayı herhangi bir i değerine bölünürse bu sayının asal olmadığını söyleriz. Fakat sayı döngü içerisinde kendisinden başka bir bölen durumu olmaz ise sayının asal sayı oldu bilgisi döndürülür.

## Daha Hızlı Fibonacci Dizisi Bulma

Bu diziyi hesaplayabilmek için özyinelemeli fonksiyon kullandık. Kullanıcıdan başlangıç, ilk ve ikinci terimlerini girmesini istedik ve girdileri aldık. Yine Fibonacci hesaplama fonksiyonunu bir buton yardımıyla birbirine bağladık. Fonksiyonun içinde hızlandırıcı fonksiyonunu çağırdık. Hızlandırıcı fonksiyonu içinde kullanıcının girdiği Fibonacci terimi, başlangıç, ilk ve ikinci terim vardır. İstenilen Fibonacci değeri 1 ise listeleme olarak sadece başlangıç değerini döndürür. Eğer istenilen değer 1 den daha fazla ise Fibonacci Dizisi kuralı gereği her terim kendinden önceki ilk iki terimin toplamıdır yöntemi ile istenilen Fibonacci dizisi hesaplanarak listelenir. Fibonacci dizisi hesaplamasını hızlandırmak için mevcut terim ‘fibk’ ve önceki terim ‘fibk1’ kullanılarak yeni terimler hesaplanır. ‘fibk’ güncellenmesi için mevcut terim ve bir önceki terim toplanarak (‘fibk = fibk + fibk1’) yeni terim hesaplanır. ‘fibk1’ bir önceki terim, mevcut terimden çıkartılarak (‘fibk1 = fibk – fibk1’) yeni bir önceki terim hesaplanır. Sonraki terim hesaplandıktan sonra, bu terimin ‘n’,’k’,’fibk’ ve ‘fibk1’ ile birlikte bilgileri bir liste içine eklenir. Bu, her adımın kaydedilmesini sağlar. Son olarak, eğer hedef terim ‘n’ ile ‘k’ eşitse, bu demek olur ki hesaplama tamamlanmıştır ve son hesaplanan Fibonacci terimi ‘fibk’ sonuç olarak döndürülür.

## Çıkış Fonksiyonu

İşlem yapılmak istenilmediği takdirde çıkış yapılan bir etiket verilerek bir sınıf bloğu oluşturuldu.

## Arayüz Oluşturma

Öncelikle kodumuzun en başında bulunan ‘import ’ methotu ile sys modülünü kodumuza dahil ediyoruz. Sys modülü Python yorumlayıcısının etkileşimli işlemlerini kontrol etmeye ve işletim sistemi le etkileşimde bulunmaya olanak tanır.‘from PyQt5.QtWidgets import …. ‘ bu satır ile PyQt5 kütüphanesinin çeşitli bileşenlerini içe aktarır. Bu bileşenler arayüz oluşturmak için kullanılır. Böyle tanımlamamızın nedeni de her çağırışımızda tekrar tekrar’ QtWidgets’ yazmamaktır.‘from functools import reduce’ Bu satır, Python’ın ‘functools’ modülünden ‘reduce’ fonksiyonunu içe aktarır. ‘reduce’, veri yapıları üzerinde işlem yapmak için kullanılır. Fonksiyonları, arayüzde daha kolay kullanabilmek için her fonksiyonun kendi sınıfını oluşturduk.‘class fonksiyon\_adi(QDialog)’ Ve her sınıf tanımının içindeki parametre olarak QDialog metotunu almasının nedeni fonksiyonu çalıştırmak için butona tıkladığımızda açılan yeni pencerenin kapanmadan başka bir işlem yapamamamızı sağlıyor. Yani bir yeni açılan pencere işlemimizi bitirmeden bir önceki pencereye dönmemize izin vermiyor. Class içlerinde kullanılan komutlardan her bir işlem için ayrı yukarıda bahsedilmişti. ‘def fonkisyon\_adı\_calistir(self)’ şeklinde olan fonksiyonlar ise bizim ana fonksiyonumuzdan gelen geri dönüş değerini alarak yanlış doğruluğuna bakarak arayüzüne yansıtmayı sağlar. Bu fonksiyon tipleri ana fonksiyonları işlediğin için hata ayıklama satırlarını buraya koyuldu. Yani ‘try’ ‘except’ blokları, bu bloklar hata kontörlü için bir bloğun başlangıcını işaretler. Herhangi bir hata olursa ‘except’ bloğuna geçer. ‘except’ bloğunda yer alan QMessageBox() bir hata olduğunda olduğunda küçük yeni bir pencere açıp orda hata mesajını gösterir ve bu da QDialog sınıfından türetildiği için bu küçük pencereyi kapatmadan başka pencereye geçemiyorsunuz. Ve bu ‘except’ bloğunun en sonundaki ‘.exec\_()’ metotu iletişim kutusunu görüntülemeni sağlar. Bunun haricinde öncelikle arayüzümüzde iki pencere vardır birisi ana penceredir. Kodumuzdaki bütün fonksiyonlar butonlara atanmıştır. Çalıştırmak istediğiniz fonksiyon butonuna basarak ikinci pencereyi açarız. Bu ikinci pencerede fonksiyonu çalıştırıp işlemimizi yaparız. Ama bu ikinci pencereyi kapatmadan ilk penceredeki başka bir fonksiyon butonuna basıp çalıştırmak istesek onu yapamayız. İkinci pencereyi şöyle açarız : Class FirstWindow(QWidget) sınıfındaki oluşturulan def open\_fonksiyonadı\_window(self) şeklinde oluşturulan fonksiyonda diğer sınıflara bağlanıp .exec\_() metotu ile çalıştırırız.Kodumuzun sonundaki “if \_\_name\_\_ == '\_\_main\_\_':” ifadesi Python’da bir programın ana program olarak mı çalıştırıldığını kontrol eder. Bu blok, eğer fonksiyon doğrudan çalıştırılıyorsa yani başka bir program veya program tarafından içe aktarılmıyorsa çalışır.“app =QApplication(sys.argc)” ifadesi, Pyqt5 uygulamasını başlatır. ‘QApplication’ sınıfı PyQt5 uygulamalarının temelini oluşturur. ‘sys.argv’ komut satıru argümanlarını alır ver uygulamaya iletir.‘main\_window = FirstWindow()’ ifadesi, PyQt5’deki bir pencere sınıfını örneğini oluşturur. ‘FirstWindow’ burada ana pencerenin sınıfını temsil eder ve bu pencere uygulamanın ana penceresi olacaktır.‘main\_window.show()’ ifadesi, oluşturulan ana pencerenin görüntülenmesini sağlar.‘sys.exit(app.exec\_())’ ifadesi, uygulamanın çalıştığıı ve olay döngüsünü başlattığını belirtir. Bu kod, uygulamanın başlamasını ve pencere kapatılana kadar çalışmasını sağlar. ‘app.exec\_()’ olay döngüsünü başlatır ve pencere kapatıldığında uygulamayı sonlandırır.

# SONUÇ VE ÖĞRENİLEN DERSLER

Bu proje sayesinde Python dilinin kullanımı ve kullanırken dikkat edilmesi gereken hususları, istenilen yönte7mlerin hesaplanma işlemi sırasında sınıf bloklarının oluşturulması nesnelerin neler olması gerektiği, parametreleri ve işlemlerin doğru olup olmadığının kontrolünün yapılması gibi işlem basamaklarını tanımlamak gerekliliği gibi kavramlar üzerinde duruldu. Kod grubu için bir ara yüz tasarlandı ve kod bu ara yüze bağlandı. Grup çalışmasıyla birlikte ödevin ortak dil ile yazılması ve grubun tüm üyelerinin yöntemler hakkında bilgi sahibi olması gerektiği ve standart bir kod yazım şablonu oluşturulması gerektiğinin farkına varıldı.

# KAYNAKÇA

https://forum.yazbel.com/t/python-ile-matris-carpimi/15324/2

https://www.w3schools.com/python/python\_classes.asp

https://www.w3schools.com/python/python\_ref\_string.asp

https://www.w3schools.com/python/python\_lists\_comprehension.asp

https://www.w3schools.com/python/python\_ref\_list.asp

https://www.w3schools.com/python/python\_ref\_dictionary.asp

https://pypi.org/project/PyQt5/

https://www.sinanerdinc.com/python-map-zip-reduce-filter-kullanimi

<https://github.com/mugeozler/islemler/consele> menu