T.C
KOCAELİ SAĞLIK VE TEKNOLOJİ ÜNİVERSİTESİ MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ PROGRAMI

ÖDEV KONUSU

CONSOLE MENÜ

Hazırlayanlar
Nazlı Su KETÇİ 220501007

https://github.com/nzlktc02

Amine DERİN 220501004

https://github.com/aminederin

DERS SORUMLUSU
Prof. Dr. Hüseyin Tarık DURU

29.10.2023

Yeniköy Mahallesi Ilıca Caddesi No:29, Başiskele,Kocaeli
info@kocaelisaglik.edu.tr kocaelisaglik.edu.tr

1.SORU

```
from colorama import Fore, Style, init

def a_kucuk(a, sıra):
    if a > 0 and a <= len(sıra):

        sıra.sort()

        return sıra[a - 1]
    else:
        return None


sıra = [7, 10, 4, 3, 20, 15]
a_degeri = 3

veri = a_kucuk(a_degeri, sıra)
if veri is not None:
            print(f"Listenin {a_degeri}. en küçük elemanı: {veri}")
          else:
             print("Geçersiz a değeri.")
```

ÇIKTI :

```
Listenin 3. en küçük elemanı: 7
```

KOD ANLATIMI:

```
from colorama import Fore, Style, init
```

Burada colorama modülünü kullanırız.Colorama modülü terminaldeki renkli yazıyı yazmamızı sağlayan bir modüldür.

```
if a > 0 and a <= len(sıra
```

Fonksiyonun içinde, a değerinin 0'dan büyük ve sıranın uzunluğundan küçük veya eşit olduğu kontrol etmek için if fonksiyonunu kullanırız. Eğer bu şart sağlanmıyorsa none değerini döndürür.

`sıra.sort()`

Bu method sıradaki elemanın küçükten büyüğe sıralamamızı sağlar.

```
return sıra[a - 1]
```

Sıradaki eleman, sıranın (list'in) içindeki a-1 indisine karşılık gelen değerdir. Bu değer, belirli bir sıradaki küçük elemandır.

2.SORU

```
def yakın_sayılar(sayı, sıra):
    sıra.sort()
    yakın = None
    aradaki_fark = float('inf')

    for a in range(len(sıra) - 1):
        for b in range(a + 1, len(sıra)):
            toplam = sıra[a] + sıra[b]
            fark = abs(sayı - toplam)

            if fark < aradaki_fark:
                aradaki_fark = fark
                yakın = (sıra[a], sıra[b])

    return yakın sayılar

kullanımı = yakın_sayılar(90, [70, 22, 25, 178, 13, 230])
print( "En yakın sayılar" ,kullanımı)
```

ÇIKTI:

```
En yakın sayılar (38, 49)
```

KOD ANLATIMI:

```
yakın = None
```

```
    aradaki_fark = float('inf')
```

Yakın değişkeni, şu ana kadar bulunan en yakın sayı çiftini tutar. Aradaki_fark değişkeni ise şu ana kadar bulunan en küçük farkı tutar. İlk başta aradaki_fark'a sonsuz (inf) değeri atanır, çünkü henüz herhangi bir fark bulunmamıştır.

```
for a in range(len(sıra) - 1):
        for b in range(a + 1, len(sıra)):
```

Burada 2 kere döngü kullanıp bütün sayı çiftlerinin arasındaki farkı hesaplıyoruz.

```
toplam = sıra[a] + sıra[b]
            fark = abs(sayı - toplam)


            if fark < aradaki_fark:
                aradaki_fark = fark
                yakın = (sıra[a], sıra[b])
```

Her bir sayı çifti için, iki sayının toplamı ve sayı ile toplam arasındaki farkı hesaplarız. Eğer bu fark, daha önce bulunan en küçük farktan küçükse, aradaki_fark ve yakın değerleri güncellenir.

```
return yakın
```

Burada da en yakın olan sayıları döndürüyoruz.

3.SORU

```
def main(liste):
    return list({x for x in liste if liste.count(x) > 1})



liste = [1, 1, 2, 3, 3, 5, 6, 11, 8, 7, 7]
sırala = main(liste)
 print("Tekrar eden elemanlar:", sırala)
```

ÇIKTI

```
Tekrar eden elemanlar: [1, 3, 7]
```

KOD ANLATIMI:

```
return list({x for x in liste if liste.count(x) > 1})
```

Bu fonksiyon, bir set comprehension kullanarak listedeki tekrar eden elemanları bulur. liste.count(x) ifadesi, listede bulunan bir elemanın kaç defa tekrar ettiğini verir. Eğer bu değer 1'den büyükse (yani eleman tekrar ediyorsa), bu eleman sete eklenir. Sonuç olarak, bu set listeye dönüştürülerek tekrar eden elemanları içeren bir liste elde edilir.

```
liste = [1, 1, 2, 3, 3, 5, 6, 11, 8, 7, 7]
sırala = main(liste)
```

Bir liste tanımladık ve main fonksiyonuna geçirdik .

4.SORU

```
def islem(matrisa, matrisb):
    if len(matrisa[0]) != len(matrisb):
        return "Matrislerin boyutları uyumlu değil."

    matris_bulma = list(map(list, zip(*matrisb)))

    sonuc = [[sum(ele1 * ele2 for ele1, ele2 in zip(row1, row2)) for row2 in matris_bulma] for row1 in matrisa]

    return sonuc

matrisa = [[1, 2, 3], [4, 5, 6]]
matrisb = [[7, 8], [9, 10], [11, 12]]
sonuc = islem(matrisa, matrisb)
for row in sonuc:
        print(row)
```

ÇIKTI:

```
[58, 64]
[139, 154]
```

KOD ANLATIMI:

```
if len(matrisa[0]) != len(matrisb):
        return "Matrislerin boyutları uyumlu değil."
```

Fonksiyon içinde, iki matrisin çarpılabilir olup olmadığı kontrol edilir. Matris çarpımı için, birinci matrisin sütun sayısı ikinci matrisin satır sayısına eşit olmalıdır. Eğer uyumsuzluk varsa, kodumuz hata mesajı döndürür .

```
matris_bulma = list(map(list, zip(*matrisb)))
```

İkinci matrisin transpozunu alarak, sütunları satırlara ve satırları sütunlara dönüştürürüz. Bu, matris çarpımını daha etkili bir şekilde yapabilmek için kullanılır.

```
sonuc = [[sum(ele1 * ele2 for ele1, ele2 in zip(row1, row2)) for row2 in matris_bulma] for row1 in matrisa]
```

İki matrisin çarpımını hesaplamak için iç içe iki döngü kullanırız. Her bir eleman, karşılıklı elemanların çarpımının toplamıdır. Zip fonksiyonu, iki listenin karşılıklı elemanlarını çift olarak birleştirir.

5.SORU

```
from collections import Counter

def kelime_tekrarı(dosyanın_yeri):
    try:
        with open(dosyanın_yeri, 'r') as dosya:
            metin = dosya.read()
            kelimeler = metin.split()

            kelimelerin_sayisi= Counter(kelimeler)
            return kelimelerin_sayisi

    except FileNotFoundError:
        return "Dosya bulunamadı."


dosyanın_yeri = "/Users/maclock/Desktop/giris_metnii.txt"
tekrarlama = kelime_tekrarı(dosyanın_yeri)
if isinstance(tekrarlama, str):
                print(tekrarlama)
            else:
              for kelime, sayi in tekrarlama.items():
                print(f"{kelime}={sayi}")
```

ÇIKTI:

```
elma=3
armut=7
kiraz=8
```

KODU ANLATIMI:

```
from collections import Counter
```

Counter sınıfını içe aktarır. Bu sınıf, bir listedeki elemanların sayısını tutan bir sözlük yapısını sağlar.

```
try:
    with open(dosyanın_yeri, 'r') as dosya:
```

Dosyanın varlığını kontrol etmek için try-except bloğu kullanırız. Eğer dosya bulunursa, dosya okuma işlemi başlatılır.

```
metin = dosya.read()
kelimeler = metin.split()
```
Dosyanın içindeki metin okunur ve split fonksiyonu kullanılarak kelimelere ayrılır.

```
 kelimelerin_sayisi= Counter(kelimeler)
```
Counter sınıfı kullanılarak, her bir kelimenin kaç kez geçtiği hesaplanır.

```
except FileNotFoundError:
      return "Dosya bulunamadı."
```
Dosya bulunamazsa, FileNotFoundError hatası alınır ve "Dosya bulunamadı." mesajı döndürülür.

```
dosyanın_yeri = "/Users/maclock/Desktop/giris_metnii.txt"
tekrarlama = kelime_tekrarı(dosyanın_yeri)
if isinstance(tekrarlama, str):
            print(tekrarlama)
        else:
          for kelime, sayi in tekrarlama.items():
            print(f"{kelime}={sayi}")
```
Örnek bir dosya yolu belirlenir ve bu dosyadaki kelimelerin tekrar sayıları hesaplanarak ekrana yazdırılır.

6.SORU

```
def en_küçük_değer_bulma(liste):
   if not liste:
      return None
   if len(liste) == 1:
      return liste[0]

      küçük_değer = en_küçük_değer_bulma(liste[1:])
   if not liste[0] >= küçük_değer:
      return liste[0]
   else:
      return küçük_değer
Liste=[-35, 98, 42, 120, 45, -23]
print(en_küçük_değer_bulma(Liste))
```

```

```

ÇIKTI:

```
-35
```

KODUN ANLATIMI:

```
if not liste:
    return None
```

Eğer liste boşsa, fonksiyon None döndürür.

```
if len(liste) == 1:
    return liste[0]
```

1 tane eleman varsa o döndürülüyor.

```
küçük_değer = en_küçük_değer_bulma(liste[1:])
```

Liste daha fazla eleman içeriyorsa, listenin ilk elemanını geride kalan kısmın en küçük değeri ile karşılaştırarak öz yineli olarak fonksiyonu çağırır.

```
if not liste[0] >= küçük_değer:
    return liste[0]
  else:
    return küçük_değer
```

Listenin ilk elemanı, geriye kalan kısmın en küçük değerinden küçükse, bu elemanı döndürür. Yoksa,geriye kalan kısmın en küçük değeri döndürülür.

7.SORU

```
def kare_kok(C, x, toplam=1e-10, karekök=10):
  for y in range(karekök):
    nz = 0.5 * (x + C / x)
    yanlıs = abs(nz**2 - C)
    if yanlıs < toplam:
```

```
            return nz
        x = nz
    print(f"{karekök} iterasyonda sonuca ulaşılamadı. 'yanlıs' veya 'karekök' değerlerini değiştirin")
    return nz


sonuc_1 = kare_kok(C=10, x=1)
print(sonuc_1)

sonuc_2 = kare_kok(C=10000, x=0.1)
print(sonuc_2)

sonuc_3 = kare_kok(C=10000, x=0.1, karekök=15)
print(sonuc_3)
```

ÇIKTI:

```
3.162277660168379
10 iterasyonda sonuca ulaşılamadı. 'yanlıs' veya 'maxiter' değerlerini değiştirin
129.61915927068787
100.0
```

KODUN ANLATIMI:

```
def kare_kok(C, x, toplam=1e-10, karekök=10):
```

C: Bulunmak istenen sayının karesine atadık

x: Başlangıç tahmini

toplam: Kabul edilebilir hata miktarı onu da 1e-10 arasında aldık.

karekök: Maksimum iterasyon sayısı onu da 10 aldık.

```
nz = 0.5 * (x + C / x)
```

Yeni tahmin, mevcut tahminin yarısı ve C ile mevcut tahminin bölümünün yarısıdır.

```
yanlıs = abs(nz**2 - C)
        if yanlıs < toplam:
            return nz
```

Yeni tahminin karesi ile C arasındaki farkın mutlak değeri, belirlenen hata miktarından küçükse, bulunan tahmini döndürür..

```
        x = nz
```

Eğer hata miktarı kabul edilebilir değilse, tahmin güncellenir ve döngü devam eder.

```
print(f"{karekök} iterasyonda sonuca ulaşılamadı. 'yanlıs' veya 'karekök' değerlerini değiştirin")
    return nz
```

Maksimum iterasyon sayısına ulaşıldığında bir hata mesajı yazdırır ve en son bulunan tahmin döndürülür.

8.SORU

```
def ebob(x, y):

    if y == 0:
        return abs(x)
    else:
        return ebob(y, x % y)

sayi1 = 36
sayi2 = 48
result = ebob(sayi1, sayi2)
print(f"SAYILARIN EBOB'U {result}")
```

ÇIKTI:

```
SAYILARIN EBOB'U 12
```

KODUN ANLATIMI:

```
def ebob(x, y):
```

Ebob adlı bir fonksiyon tanımlarız. Bu fonksiyon, iki sayının en büyük ortak bölenini (EBOB) hesaplamak için kullanılır. Parametreleri x ve y olarak adlandırdık.

```
if y == 0:
        return abs(x)
```

Eğer y 0 ise (temel durum), fonksiyon x'in mutlak değerini (abs(x)) döndürür. Bu, rekürsif çağrıların sonlandığı temel durumu temsil eder.

```
else:
        return ebob(y, x % y)
```

Eğer y 0 değilse, yani temel durum sağlanmıyorsa, fonksiyon kendisini y ve x % y parametreleri ile tekrar çağırır. Burada, x % y, x'in y'e bölümünden kalanı temsil eder.

9.SORU

```
def asal_bulma(a, b=2):
    if a < 2:
        return "False"
    if a == 2:
        return "True"

if a % b == 0:
        return "False"
    if b * b > a:
        return "True"
    return asal_bulma(a, b + 1)


        print(asal_bulma(87))
        print(asal_bulma(43))
```

ÇIKTI:

```
False
True
```

KODUN ANLATIMI:

```
def asal_bulma(a, b=2):
```

Burada 2 den başlatıyoruz çünkü en küçük asal sayı 2 den başlıyor.

```
if a < 2:
        return "False"
    if a == 2:
        return "True"
```

Kodda ilk olarak, a 2'den küçükse, bu sayı asal değildir, bu nedenle "False" döndürülür. Eğer a 2 ise, bu sayı asaldır, bu nedenle "True" döndürülür.

```
if a % b == 0:
        return "False"
```

```
    if b * b > a:
        return "True"
    return asal_bulma(a, b + 1)
```

Eğer a b'ye bölünüyorsa, yani a % b == 0 ise, a asal değildir, bu nedenle kod "False"değerini döndürür. Ayrıca, eğer b * b (b'nin karesi) a'dan büyükse, a asaldır, bu nedenle "True" değerini döndürür. Eğer bu durumlar sağlanmazsa, fonksiyon kendisini (asal_bulma) a ve b + 1 parametreleri ile tekrar çağırarak devam eder.

10.SORU

```
def fibonacci_dizisi(x, y, z, t):
    if x == y:
        return z
    else:
        return fibonacci_dizisi(x, y + 1, z + t, z)

def hızlı_fibonacci(x):
    return fibonacci_dizisi(x, 1, 1, 0)

terim_sayısı = 13
sonuçç = hızlı_fibonacci(terim_sayısı)
print(f"Fibonaccinin {terim_sayısı}.teriminin değeri  = {sonuçç}")
```

ÇIKTI:

```
Fibonaccinin 13.teriminin değeri  = 233
```

KODUN ANLATIMI:

```
def fibonacci_dizisi(x, y, z, t):
```

x: Hesaplanmak istenen Fibonacci terimidir
y: Bir sayıcı, mevcut terimi temsil eder.
z: x. Fibonacci teriminin değerini temsil eder.
t: x-1. Fibonacci teriminin değerini temsil eder.

Fonksiyon, x == y durumunda mevcut Fibonacci teriminin değerini (z) döndürür. Aksi takdirde, fonksiyon kendisini y + 1, z + t, ve z parametreleri ile öz yinelemeli olarak çağırır.

```
if x == y:
        return z
    else:
        return fibonacci_dizisi(x, y + 1, z + t, z)
```

Bu fonksiyon, Fibonacci dizisinin x. terimini hesaplamak için fibonacci_dizisi fonksiyonunu çağırır. Başlangıç değerleri olarak 1, 1 ve 0 kullandık.

CONSOL KISMI:

```
def main():
    while True:
        print(Fore.GREEN +"1- a'nıncı En Küçük Elemanı Bulma")
        print(Fore.BLUE+"2- En Yakın Çifti Bulma")
        print(Fore.RESET+"3- Bir Listenin Tekrar Eden Elemanlarını Bulma")
        print(Fore.YELLOW+"4- Matris Çarpımı")
        print(Fore.BLACK+"5- Bir Text Dosyasındaki Kelimelerin Frekansını Bulma")
        print(Fore. WHITE+"6- Liste İçinde En Küçük Değeri Bulma")
        print(Fore.CYAN+"7- Karekök Fonksiyonu")
        print(Fore.MAGENTA +"8- En Büyük Ortak Bölen")
        print(Fore.BLUE+"9- Asallık Testi")
        print(Fore.RED +"10- Daha Hızlı Fibonacci Hesabı")
        print(Fore.YELLOW +"11- EXIT")


        seçenek_seç = input("Yapmak istediğiniz işlemi tuşlayabilir misiniz? ")
```

Buranın renkli olması için colorama kütüphanesini kullandık.While döngüsü ile döngünün devamlılığını sağladık. Tuşladığı sayı ile işlem yapması istenilen menüye gidilir.Ve sonucu yazar.

KAYNAK:

https://stackoverflow.com/
https://www.udemy.com
https://forum.yazbel.com