# FIZ220_EST_Odev_03_Donguler_ve_Kararlar_Cozumler

June 30, 2020

## 1 Ödev: 3 - Çözümler

### 1.1 FİZ220 - Bilgisayar Programlama II | 08/05/2020

**Döngüler ve Kararlar**

Dr. Emre S. Taşcı, emre.tasci@hacettepe.edu.tr
Fizik Mühendisliği Bölümü
Hacettepe Üniversitesi

### 1.2 NumPy Dizileri (ndarray) matrisler şeklinde olacak.

#### 1.2.1 1. Soru: Asal sayılar (hani hesaplanamıyordu?..)

Tanımlanan bir $n$ sayısından büyük ilk asal sayıyı bulan program yazınız.

[1]:
```
import numpy as np

n = 113
print("Tanımlanmış olan n değeri: ",n)

# Asal sayılar için sihirli bir formülümüz olmadığından,
# "en baştan", 3'den başlıyoruz

simdiye_kadarki_asallar = np.array([2])

# Once n'e kadar/dahil olan asal sayıları bulalım:
for i in np.arange(3,n+1,2):
    # Şu anda elimizdeki i sayısının asal olduğunu varsayalım
    i_asal_mi = True

    # Şimdi de birer birer "simdiye_kadarki_asallar" kumesinin
    # elemanlarına bölelim -- tâ ki bir tanesine kalansız bölününceye
    # kadar -- o zaman asal değil demektir!
    for j in simdiye_kadarki_asallar:
        if(i%j == 0):
            # Kalansız bölündü - demek ki asal değil!
```

```
            i_asal_mi = False

            # Diğer sayılara bölünüyor mu diye bakmaya gerek yok
            break

    # Buraya gelindiğinde iki şey olmuş olabilir:
    # * ya bir sayıya kalansız bölündü (i_asal_mi = False)
    # * ya da, hiçbirine kalansız bölünmedi de, j'nin döngüsü bitti
    #     (yani gerçekten asalmış -- i_asal_mi = True kaldı)
    if(i_asal_mi == True):
        simdiye_kadarki_asallar = np.append(simdiye_kadarki_asallar,i)
print("Şimdiye kadarki asallar: \n",simdiye_kadarki_asallar)

# Artık tek yapmamız gereken bunu takiben bir tane daha asal bulmak
while True:
    # Bu şekilde sonsuz bir döngüye girdik, sonumuz hayır olsun
    # (hemen korkmayın, istediğimiz an break ile kırar çıkarız 8)
    n = n + 1
    n_asal_mi = True
    for j in simdiye_kadarki_asallar:
        if(n%j == 0):
            n_asal_mi = False
            # print(j tarafindan tam bolundu")
            break
    if(n_asal_mi == True):
        print("Bulunan asal: ",n)
        break
```

```
Tanımlanmış olan n değeri:  113
Şimdiye kadarki asallar:
 [  2   3   5   7  11  13  17  19  23  29  31  37  41  43  47  53  59  61
  67  71  73  79  83  89  97 101 103 107 109 113]
Bulunan asal:  127
```

Kendinizi geliştirmek isterseniz, ikinci döngüde (n=n+1 olan) n'in değerini birer birer değil de, ikişer ikişer arttıralım, ne de olsa çift sayılar asal olamayacağından, boşu boşuna kontrol etmekle CPU'yu yormuş olmayız — ama bir saniye, ya bize başta verilen n çift sayı ise? o zaman bütün tekleri atlamış oluruz!!! O halde koda öyle bir ekleme yapın ki, eğer verilen n tek ise, öyle kabul edin ama eğer çift ise, ondan küçük en büyük tek sayıya eşitleyin (yani, halk tabiriyle: *1 çıkarın* 8)

### 1.2.2 2. Soru: Kuvvet hesabı

İki boyutta, konumları $(x_i\hat{i}+y_i\hat{j})$ m şeklinde, yükleri de Coulomb cinsinden verilen 5 adet parçacığın her birinin üzerine diğerlerinden binen kuvvetleri (vektörel olarak) hesaplayan program yazın.

[2]:
```
import numpy as np
```

```
np.random.seed(220)

n = 5 # Parcacik sayisi

k = 8.99E9 # Nm^2/C^2

# Rastgele konumlar ve yükler için:
# konumlar = 10*np.random.rand(n,2)-5 # orijine göre (m)
# yukler = 500*(np.random.rand(n,1)-0.5)*1E-6 # (C)

# Biz yüklerimizi merkezi orijinde olan,
# kenar uzunluğu 2m'lik bir karenin köşelerine
# ve merkezine koyalım

konumlar = np.zeros((n,2)) # (m)
konumlar[0,:] = [0,0]   # Merkez
konumlar[1,:] = [1,1]   # Sağ üst
konumlar[2,:] = [-1,1]  # Sol üst
konumlar[3,:] = [-1,-1] # Sol alt
konumlar[4,:] = [1,-1]  # Sağ alt


yukler = np.array([5,1,1,1,1])*1E-6 # (C)

print("konumlar (m):\n",konumlar)
print("-"*45)

print("yükler (C):\n",yukler)
print("-"*45)

kuvvetler = np.zeros((5,2))

print("\n"\
+"2 --- 1\n"\
+" \   /\n"\
+"|  0  |\n"\
+" /   \ \n"\
+"3 --- 4\n")



for i in np.arange(n-1):
    for j in np.arange(i+1,n):
        r_vec = konumlar[i,:] - konumlar[j,:]
        r_buyukluk = np.sqrt(r_vec[0]**2 + r_vec[1]**2)
        F = k*yukler[i]*yukler[j]*r_vec/r_buyukluk**3
        print("F_"+str(i)+str(j),": ",j," -> ",i," kuvveti: ",F,"N")
```

```
        kuvvetler[i] = kuvvetler[i] + F
        # j, i'ye F kuvveti uyguluyorsa,
        # i de j'ye -F kuvveti uygular:
        kuvvetler[j] = kuvvetler[j] - F
print("-"*45)
print("kuvvetler (N):\n",kuvvetler)
print("-"*45)
```

```
konumlar (m):
 [[ 0.  0.]
 [ 1.  1.]
 [-1.  1.]
 [-1. -1.]
 [ 1. -1.]]
---------------------------------------------
yükler (C):
 [5.e-06 1.e-06 1.e-06 1.e-06 1.e-06]
---------------------------------------------

2 --- 1
 \   /
|  0  |
 /   \
3 --- 4

F_01 :  1  ->  0  kuvveti:  [-0.01589222 -0.01589222] N
F_02 :  2  ->  0  kuvveti:  [ 0.01589222 -0.01589222] N
F_03 :  3  ->  0  kuvveti:  [0.01589222 0.01589222] N
F_04 :  4  ->  0  kuvveti:  [-0.01589222  0.01589222] N
F_12 :  2  ->  1  kuvveti:  [0.0022475 0.       ] N
F_13 :  3  ->  1  kuvveti:  [0.00079461 0.00079461] N
F_14 :  4  ->  1  kuvveti:  [0.        0.0022475] N
F_23 :  3  ->  2  kuvveti:  [0.        0.0022475] N
F_24 :  4  ->  2  kuvveti:  [-0.00079461  0.00079461] N
F_34 :  4  ->  3  kuvveti:  [-0.0022475  0.       ] N
---------------------------------------------
kuvvetler (N):
 [[ 0.          0.        ]
 [ 0.01893434  0.01893434]
 [-0.01893434  0.01893434]
 [-0.01893434 -0.01893434]
 [ 0.01893434 -0.01893434]]
---------------------------------------------
```

Parçacıklar üzerinden döngüyü alırken, nasıl saydığımıza dikkat edin: $i \to j$ etkileşimi ile $j \to i$ etkileşimi zıt yönlerde fakat aynı büyüklükte olduğundan, ayrı ayrı bir $i \to j$, bir de $j \to i$ etkileşimi hesaplamamak için $i$'leri 0,1,2,3'e kadar/dahil sayarken, $j$'leri de $i$'lerin başlangıcından 4'e kadar/dahil sayıyoruz. Böylelikle hesap sayısını en aza indirgemiş oluyoruz. İleride, bilimsel

makalelerde şöyle toplamlar göreceksiniz:

$$\frac{1}{2}\sum_{i\neq j} F_{ij}$$

işte o başındaki $\frac{1}{2}$'nin hikayesi bu çifte sayım mevzuu. Anlaşılsın diye o şekilde yazılır, koda uyarlanırken de:

$$\sum_{i=0}^{N-1}\sum_{j=i}^{N} F_{ij}$$

olarak uyarlanır. ;)

[ ]: