ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ

ELM 368 SAYISAL İŞARET İŞLEME LABORATUVARI

ÖN HAZIRLIK ÇALIŞMASI

VE

ÖDEV-1

1 AMAÇ

- İndis dizisi oluşturmak.
- İşaret dizisi oluşturmak.
- Grafik çizdirmek.
- Kompleks sayılarda dört işlem, kartezyen, polar gösterim.

2 AYRIK ZAMANLARIN İŞARETLERİN OLUŞTURULMASI VE ÇİZDİRİLMESİ

2.1 Vektör (Dizi) oluşturmak

Bu derste göreceğiniz işaretleri bilgisayar ortamında tek boyutlu bir dizi (1-D array) olarak düşünebilirsiniz. Python'da dizi oluşturmak için "Numpy" kütüphanesi kullanılacaktır. Bunun için önce Numpy kütüphanesinin import edilmesi gerekir.

```
import numpy as np
```

Yukarıda "*as np*" ifadesini Numpy kütüphanesinden bir modül çağırırken kısaltma amaçlı yazılır. Numpy kütüphanesinin *arange(A, B, x)* fonksiyonu ile *A* ve *B* sayıları arasında *x* adım mesafeli olacak şekilde vektör oluşturabilirsiniz. Dikkat e dilmesi gereken nokta oluşacak bu vektöre B sayısı dahil olmaz.

```
x=np.arange(0,10,2)
print(x)


Konsol çıktısı -> [0 2 4 6 8]
```

Dizi oluşturmada yine sıklıkla kullanacağımız diğer bir Numpy fonksiyonu ise *linspace(A,B,N)* komutu. Bu komutta ise *A*'dan başlayıp *B*'ye kadar (*B* dahil) *N* noktalı olacak şekilde vektör oluşturabilirsiniz.

```
x=np.linspace(0,10,5)
print(x)


Konsol çıktısı -> [  1. 3.25  5.5  7.75  10.]
```

2.2 Ayrık-zamanlı işaret oluşturmak

Ayrık zamanlı bir işareti oluşturmak için öncelikle işaretin tanımlı olduğu indis vektörünü oluşturmalısınız. İndis vektörü tam sayılardan oluşacağından dolayı Numpy kütüphanesinin *np.arange()* komutunu kullanmak uygun olacaktır. Örnek olarak 10 noktalı olacak şekilde $\cos(\frac{\pi}{3}n+\frac{\pi}{2})$ işareti oluşturalım.

```
n=np.arange(0,10)
x=np.cos(n*np.pi/3+np.pi/2)
print(x)
Ekran çıktısı ->
[  6.12323400e-17  -8.66025404e-01  -8.66025404e-01  -1.83697020e-16
   8.66025404e-01   8.66025404e-01   3.06161700e-16  -8.66025404e-01
  -8.66025404e-01  -4.28626380e-16]
```

Dikkat ederseniz hem kosinüs fonksiyonuna hem de π sayısına Numpy kütüphanesinden eriştik.

2.3 Sürekli zamanlı işaret oluşturmak

Sürekli zamanlı bir işaret sonsuz noktadan oluşacağı için bilgisayar ortamında oluşturmak mümkün değildir. Bunun yerine sürekli zamanlı bir işareti bilgisayar ortamında oluştururken işaretin tanımlı olduğu aralıkta çok fazla noktadan oluşacak şekilde oluşturacağız. Bu sayede aslında ayrık zamanlı olan işaretin grafiğini çizdirdiğimizde "sürekli zamanlı işaretmiş gibi" görünecek. Örnek olarak $x(t)=\cos(100\pi t-\pi/5)$ işaretini iki periyot olacak şekilde oluşturalım. $x(t)$ işaretinin periyodu $\frac{1}{50}$ saniye olduğu için zaman vektörünü 0'dan $\frac{2}{50}$'ye kadar 1000 noktalı olacak şekilde oluşturalım. Nokta sayısını ne kadar arttırırsanız işareti çizdirdiğinizde sürekli zaman işaretine daha fazla benzeyecektir ancak işareti oluşturma veya işaretle işlem yapma süresi artacaktır.

```
t=np.linspace(0,2/50,1000)
x=np.cos(100*np.pi*t-np.pi/5)
```

2.4 Grafik çizdirmek

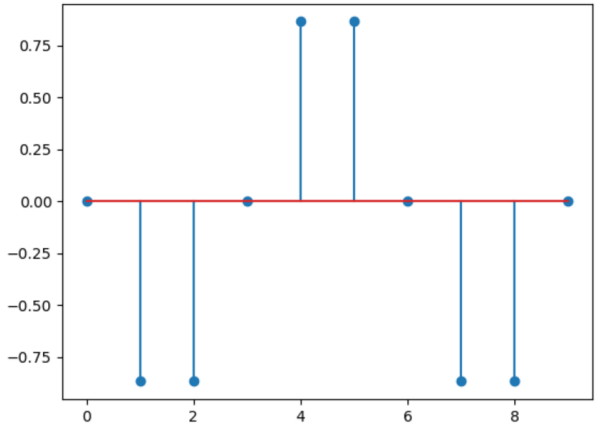
Grafik çizdirmek için "Matplotlib" kütüphanesinin pyplot modülü kullanılacaktır. Grafik çizilecekse aşağıda verildiği şekilde çizdirme komutlarından önce ilgili modülün import edilmesi gerekmektedir.

Kullanılacak tüm obje/fonksiyonların bulundukları kütüphaneler genellikle kodun en başında import edilir.

```
from matplotlib import pyplot as plt
```

2.2'de oluşturduğumuz ayrık-zamanlı işareti çizdirelim.

```
n=np.arange(0,10)
x=np.cos(n*np.pi/3+np.pi/2)
print(x)
plt.stem(n,x)
plt.show()
```



Şekil 1 $cos(\frac{\pi}{3}n + \frac{\pi}{2})$ *işareti, n=0'dan 9'a*

Pyplot modülündeki *stem(n,x)* komutu dikeyde *x* vektörünün değerlerini, yatayda *n* vektörünün aldığı değerlerle eşleştirerek çizdirir. plot() komutu kullandığınızda ise noktalar arasına düz çizgi ile birleştirilmiş olarak çizdirilir.

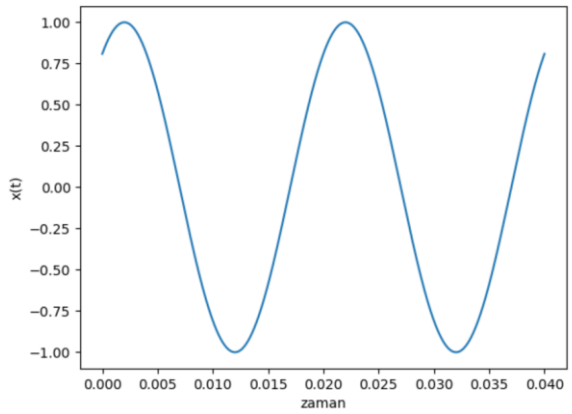
Yeni bir figür açmak için;

```
plt.figure()
```

Şimdi de Bölüm 2.3'te verilen sürekli zaman işaretini çizdirelim:

```
t=np.linspace(0,2/50,1000)
x=np.cos(100*np.pi*t-np.pi/5)
plt.plot(t,x)
plt.xlabel('zaman')
plt.ylabel('x(t)')
plt.show()
```

Pyplot modülünün *xlabel()* ve *ylabel()* komutları eksenlere isim vermek içindir.

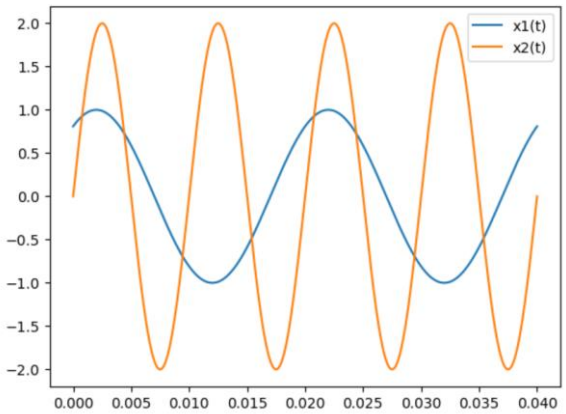


2.5 Tek figürde birden fazla işaret çizdirmek

Aynı figürde birden fazla işaretin grafiğini üst üste çizdirmek için şöyle bir örnek üzerinden gidebiliriz; $x_1(t) = \cos(100\pi t - \pi/5)$ ve $x_2(t) = 2\sin(200\pi t)$ işaretini üst üste çizdirelim. Bunun için Pyplot modülünün *plot()* fonksiyonunda kaç tane işaretin grafiğini çizdireceksek sırasıyla bağımsız ve bağımlı değişkenlerini virgülle ayırarak yazıyoruz.

```
t=np.linspace(0,2/50,1000)
x1=np.cos(100*np.pi*t-np.pi/5)
x2=2*np.sin(200*np.pi*t)
plt.plot(t,x1,t,x2)
plt.legend(('x1(t)','x2(t)'))
plt.show()
```

Yukarıda Pyplot modülünün *legend()* komutu *plot()* komutunda çizdirdiğiniz işaretlerin renklerine isim ataması için kullanılır. Vereceğiniz isimlerin sıraları çizdirdiğiniz işaretlerin plot()'daki sıralamayla aynı olmalıdır.



2.6 İşaretleri ötelemek

Bir işareti ötelemek aslında o işaretin tanımlı olduğu indis vektörünü düzenlemeye karşılık gelir. Örnek vermek gerekirse,

$$x[n] = \delta[n] + 2\delta[n-2] - 1\delta[n-3]$$

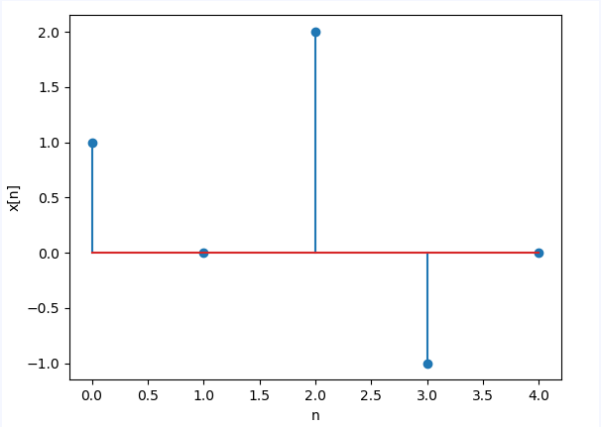
işaretini $n = 0,1,2,3,4$ noktalarından oluşan bir indis vektörü ile çizdirelim. Bunun için $x[n]$ vektörünün $[1, 0, 2, -1, 0]$ şeklinde olması gerektiğine dürtü işaretinin tanımından dolayı dikkat ediniz. Eğer işaretiniz sınırlı noktadan oluşuyor ise, indis vektörünü belirlerken işaretin sıfırdan farklı değer aldığı noktaları kapsayacak şekilde oluşturmanız gerekir. Örneğin, bu örnek için $n = -5$'den 5'e veya $n = 0$'dan 10'a olacak şekilde de belirleyebilirsiniz. Aşağıda sırasıyla farklı indis vektörleri için $x[n]$ işaretinin olması gereken diziyi verilmiştir.

$$n = [-5,-4,-3,-2,-1,0,1,2,3,4,5] \rightarrow x = [0,0,0,0,0,1,0,2-1,0]$$

$$n = [0,1,2,3,4,5,6,7,8,9,10] \rightarrow x = [1,0,2,-1,0,0,0,0,0,0,0]$$

Dikkat etmeniz gereken nokta, indis vektörü ve tanım kümesi o indis vektörü olan işaret vektörünüzün boyları **daima** eşit olmalıdır. Eğer eşit olmazsa grafik çizdirirken hata mesajı alırsınız.

```
n=np.array([0,1,2,3,4])
x3=np.array([1,0,2,-1,0])
plt.xlabel('n')
plt.ylabel('x[n]')
plt.stem(n,x3)
plt.show()
```



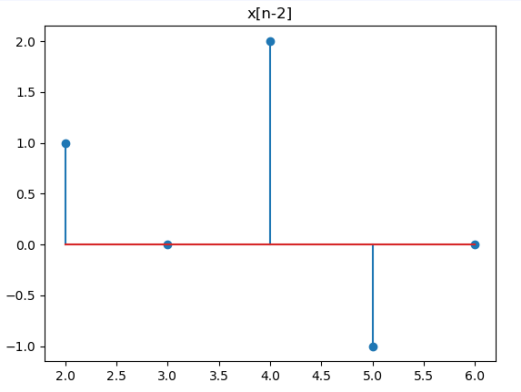
Şimdi yukarıdaki örnekte verilen $x[n]$ işaretini iki örnek sağa ötelenmiş işareti ($x[n-2]$) çizdirelim. Bunun i

çin iki farklı yol izleyebilirsiniz:

- Yol-1

x vektörüne dokunmadan doğrudan n vektörünü 2'den 6'ya olacak şekilde belirleyebilirsiniz.

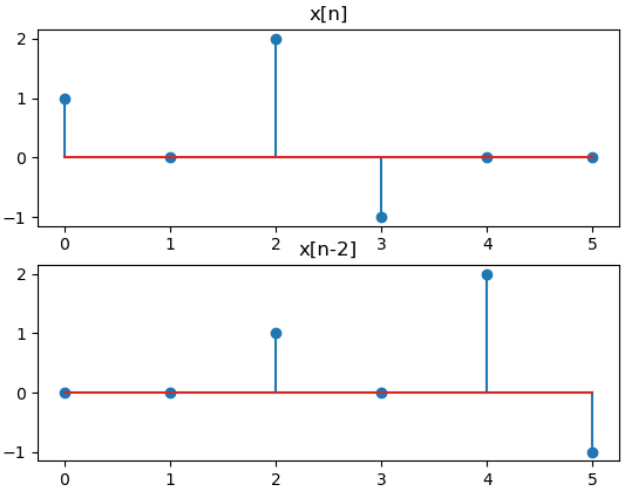
```
n=np.array([0,1,2,3,4])
n2=n+2 # n'de her elemana 2 ekler
x3=np.array([1,0,2,-1,0])
plt.title('x[n-2]') # Başlık koymak için
plt.stem(n2,x3)
plt.show()
```



- Yol-2

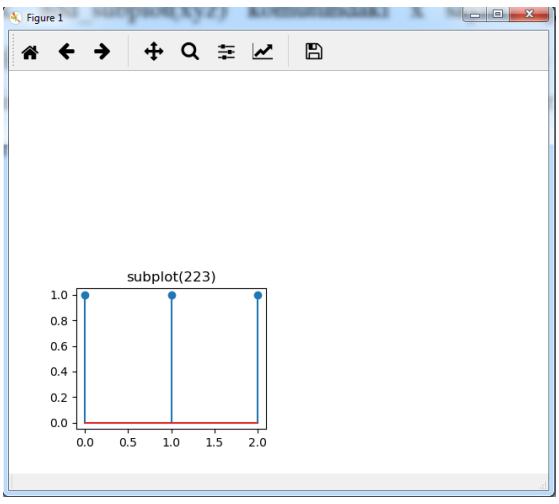
Diğer bir seçenek ise n vektörüne dokunmadan x vektörünün başına iki tane "0" ekleyip, sondaki iki elemanı vektörden silmek olabilir. Bu yolun avantajı işaretin tanımlı olduğu bölge (indis vektörü) sabit kaldığı için aynı indis vektörüyle tanımladığınız başka bir işaretle doğrudan işleme sokabilirsiniz. Dikkat etmeniz gereken nokta ise işaretin sonundan eleman silerken işaretin sıfırdan faklı bir değer aldığı noktayı kaybedebiliriz. Bu duruma engel olmak için n vektörünü en başta tanımlarken normalden biraz daha uzun (öteleme miktarı kadar mesela) olacak şekilde belirleyebilirsiniz.

```
n=np.array([0,1,2,3,4,5])
x=np.array([1,0,2,-1,0,0])
x_shifted=np.append(np.array([0,0]),x[:-2])
#yeni figür açmak için
fig = plt.figure()
ax1=fig.add_subplot(211)
ax2=fig.add_subplot(212)
ax1.stem(n,x)
ax1.title.set_text('x[n]')
ax2.stem(n,x_shifted)
ax2.title.set_text('x[n-2]')
```

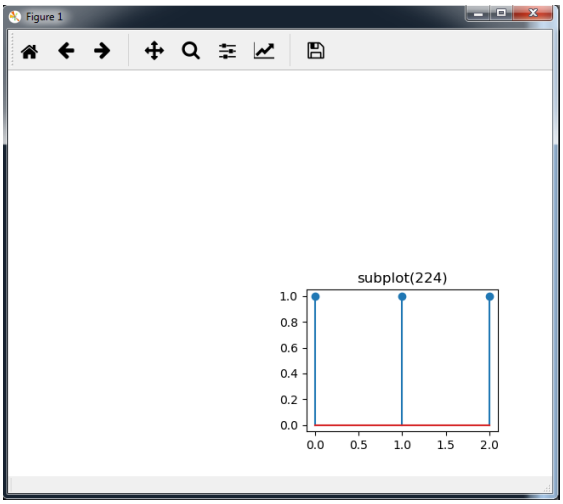


Yukarıdaki kod parçasında Numpy kütüphanesinin append(x1,x2) fonksiyonu x1 vektörünün sonuna x2 vektörünün eklenmiş versiyonunu döndürür. x[:-2] ise x vektörünün sondan iki elemanın vektörden çıkarılmış halini döndürür. Grafikleri aynı figürde alt alta göstermek için önce yeni bir figür objesi oluşturduk. Daha sonra oluşturduğumuz bu figür objesinin *add_subplot(xyz)* fonksiyonunu kullanarak ax1 ve ax2 alt figürlerini ekledik. add_subplot(xyz) komutundaki x sayısı figürü kaç satıra böleceğimize, y sayısı ise kaç sütüna bölüneceğini söyler. z sayısı ise figür matrisindeki kaçıncı bölüme grafiğin çizileceğini söyler. Örnek vermek gerekirse ;

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib notebook
plt.subplot(223)
plt.stem(np.array([0,1,2]),np.array([1,1,1]))
plt.title('subplot(223)')
```



Eğer subplot(224) yapsaydık aşağıdaki gibi bir figür elde ederdik.



2.7 Kompleks sayılar

Kompleks sayıları Numpy kütüphanesinin "complex" veri tipi ile saklarız. Bunun için isterseniz bir değişkene doğrudan a+bj sayısını atayabilirsiniz. Benzer şekilde Numpy'ın complex(a,b) fonksiyonu da size a+bj sayısını üretecektir.

```
x1=1+1j
x2=np.complex(1,1)
print(x1)
print(x2)
Konsol çıktısı →
1+j
1+j
```

Kompleks bir dizi oluşturmak için Numpy kütüphanesinin array fonksiyonunu kullanırız ancak data tipini 'complex' olarak belirleriz.

```
x=np.array([1+2j,2-6j],dtype=complex)
```

Kompleks bir sayının genliğini bulmak için Numpy'ın abs() fonksiyonunu kullanırız.

```
x=3+4j
print(np.abs(x))

konsol çıktısı → 5.0
```

Veya kompleks elemanlardan oluşan bir arrayin her bir elemanının genlik değerini bulmak için yine aynı fonksiyonu kullanabilirsiniz.

```
x=np.array([1+2j,3-4j],dtype=complex)
print(np.abs(x))
konsol çıktısı → [2.23606798 5.        ]
```

Kompleks bir sayının fazını bulmak için Numpy'ın angle() fonksiyonunu kullanırız. Bu fonksiyon size radyan cinsinden açı verecektir.

```
x1=1+1j
print(np.angle(x1))
konsol çıktısı → 0.7853981633974483
```

Polar koordinatlarda verilen bir kompleks sayıyı ($re^{j\theta}$) kartezyen koordinatlarda ifade etmek için aşağıdaki yol kullanılabilir;

```
r=5
theta=np.pi/2
x=r*np.exp(1j*theta)
print(x)
konsol çıktısı → (3.061616997868383e-16+5j)
```

3 ÖDEV-1:

Aşağıdaki sorularda istenenleri Python'da kodlayınız. Kodlarınızı .py formatında değil "jupyter notebook" formatı olan .ipynb formatında tek bir dosya halinde teslim ediniz.

1) (**20 PUAN**) $a = 3 - 4j$ ve $b = 1 + 2j$ olmak üzere D=$\frac{a}{b}$'dir.

D kompleks sayısının

I. gerçek,
II. sanal,
III. genlik ve
IV. faz

bileşenlerini print() komutu ile bastırın.

2) (**30 PUAN**) Aşağıdaki işaretlerin periyodunu bulun ve 2 tam periyot olacak şekilde çizdirin. Çizdiğiniz grafiğe bakarak periyodu teyit edin.

I. $x[n] = 2\sin(\frac{2\pi}{5}n - \frac{\pi}{5}) + \sin(\frac{3\pi}{5}n - \frac{2\pi}{5})$

3) (**50 PUAN**) Aşağıdaki işaretleri belirtilen aralıkta çizdiriniz, karşılaştırınız ve yorumlayınız. İndis vektörü ile çiziminizde yatay eksen değerleri uyuşmalıdır. Grafiğin başlık kısmında çizdirilen fonksiyonun adı ($x_1[n]$, $x_2[n]$, vb.) belirtilmelidir.

I. $x_2[n] = e^{\frac{j\pi}{5}}$, $n = 0,1, \dots ,15$

4 ÖDEV-1 EK ÇALIŞMA:

Aşağıdaki sorularda istenenleri Python'da kodlayınız. Kodlarınızı .py formatında değil "jupyter notebook" formatı olan .ipynb formatında hazırlayınız. Bu bölümde verilen soruların cevapları puanlanmayacaktır. Ödev-1 Ek Çalışma, Ödev-1'in bir parçasıdır ve teslim edilmesi zorunludur; rastgele seçilen öğrencilere bu bölümden soru sorulabilir; sorulacak sorulara cevap vermeye hazırlıklı olunuz.

1) $a = 3 - 4j$ ve $b = 1 + 2j$ olmak üzere $C = a + b$'dir.

C kompleks sayısının

I. gerçek,
II. sanal,
III. genlik ve
IV. faz

bileşenlerini print() komutu ile bastırın.

2) Aşağıdaki işaretlerin periyodunu bulun ve 2 tam periyot olacak şekilde çizdirin. Çizdiğiniz grafiğe bakarak periyodu teyit edin.

I. $x[n] = 2\sin(\frac{2\pi}{5}n - \frac{\pi}{5})$

II. $x[n] = \sin(\frac{3\pi}{5}n - \frac{2\pi}{5})$

3) Aşağıdaki işaretleri belirtilen aralıkta çizdiriniz, karşılaştırınız ve yorumlayınız. İndis vektörü ile çiziminizde yatay eksen değerleri uyuşmalıdır. Grafiğin başlık kısmında çizdirilen fonksiyonun adı ($x_1[n]$, $x_2[n]$, vb.) belirtilmelidir.

I. $x_1[n] = e^{\frac{\pi}{5}}$, $n = 0,1,\dots,15$

II. $x_3[n] = e^{\frac{j\pi}{5}n}$, $n = 0,1,\dots,15$

III. $x_4[n] = \cos\left(\frac{\pi}{5}\right) + j\sin\left(\frac{\pi}{5}\right)$, $n = 0,1,\dots,15$

IV. $x_5[n] = \cos\left(\frac{\pi}{5}n\right) + j\sin\left(\frac{\pi}{5}n\right)$, $n = 0,1,\dots,15$

V. $x_6[n] = \cos\left(\frac{\pi}{5}n\right) + j\sin\left(\frac{\pi}{5}n\right)$, $n = -10,-9,\dots,3,4$

5 TESLİM ŞEKLİ ve ZAMANI

Bu dokümanda verilen örnek kodları kendiniz bir Jupyter Notebook'ta yazarak verilen sonuçlarla karşılaştırınız. Aynı dokümanın devamında olacak şekilde, ÖDEV-1 başlığı altında verilen sorularda istenenleri Python'da (Jupyter Notebook kullanarak) kodlayınız. Yaptığınız çalışmayı aşağıdaki formata uygun isimle kaydediniz:

ÖDEV-1: **ÖDEV1_OgrenciKOD.ipynb**

Ayrı bir doküman olarak ÖDEV-1 EK ÇALIŞMA başlığı altında verilen sorularda istenenleri Python'da kodlayınız. Yaptığınız çalışmayı aşağıdaki formata uygun isimle kaydediniz:

ÖDEV-1 EK ÇALIŞMA: **ÖDEV1_EK_OgrenciKOD.ipynb**

Bu iki .ipynb uzantılı dosyayı GTUZEM'e yükleyiniz. Sisteme geç yüklenen dosyalar kabul edilmeyecektir. Ekte, örnek bir ödev çözümü şablonu verilmektedir (bknz: **ÖDEV1_OgrenciKOD.ipynb**). Jupyter Notebook'ta yapacağınız çözümler **bu şablona göre hazırlanmalıdır.**