



ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ

ELM 368 SAYISAL İŞARET İŞLEME LABORATUVARI

ÖN HAZIRLIK ÇALIŞMASI

Laboratuvar 1

1 AMAÇ

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

2 KODLAR

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 edilmesi 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 Ayırık-zamanlı işaret oluşturmak

Ayrık zamanlı bir işareti oluşturmak için öncelikle işaretin tanımlı olduğu indis vektörünü oluşturmalısınız. Indis 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 değil. Bunun yerine biz sürekli zamanlı bir işareti bilgisayar ortamında oluştururken işaretin tanımlı olduğu aralıkta çok fazla noktadan oluşacak şekilde oluşturacağız. Bu sayede aslında ayırık 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 oluşturuyorum. Nokta sayısını ne kadar arttırırsanız işareti çizdirdiğinizde daha fazla sürekli zaman işaretine 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)
```

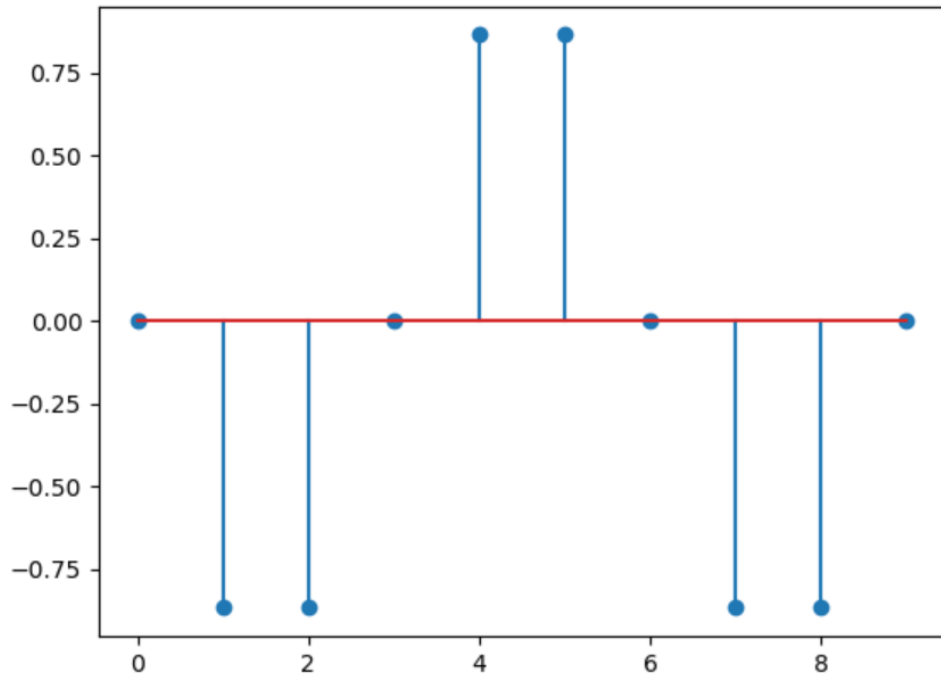
2.4 Grafik çizdirmek

Grafik çizdirmek için “Matplotlib” kütüphanesinin pyplot modülü kullanılacaktır. Grafik çizilecekse aşağıda verildiği şekilde çizdirme komutlarından önce ilgili modülün import edilmesi gerekmektedir. (Genellikle kullanılacak tüm obje/fonksiyonların bulundukları kütüphaneler 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.

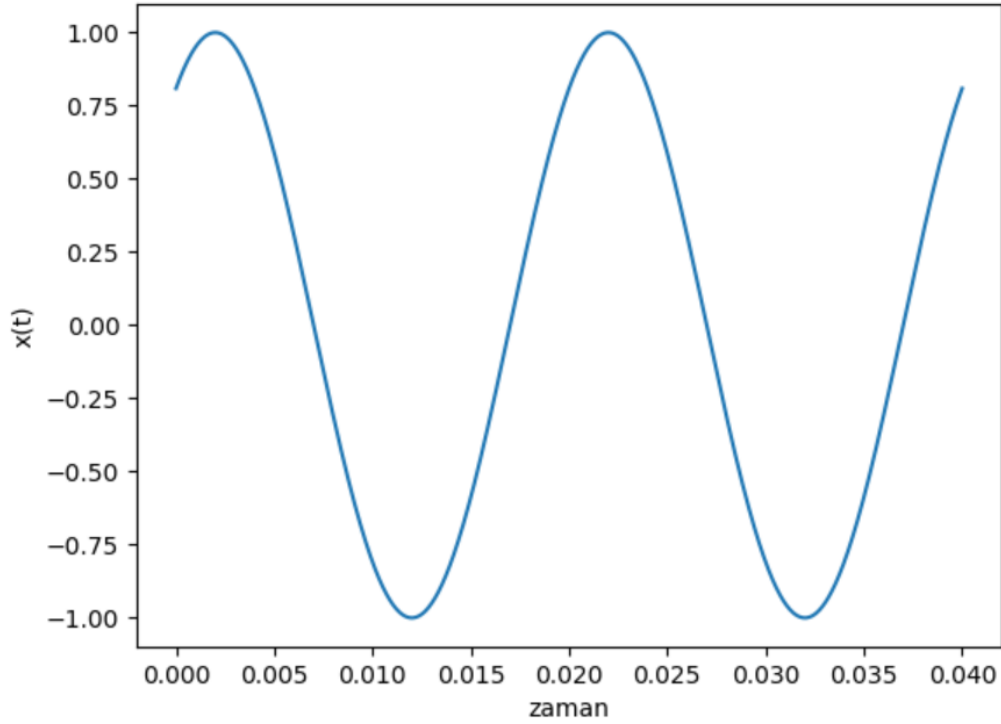
Yeni bir figür açmak için;

```
plt.figure()
```

Şimdi de bölüm 2.3'de 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 ile eksenlere isim vermek için.

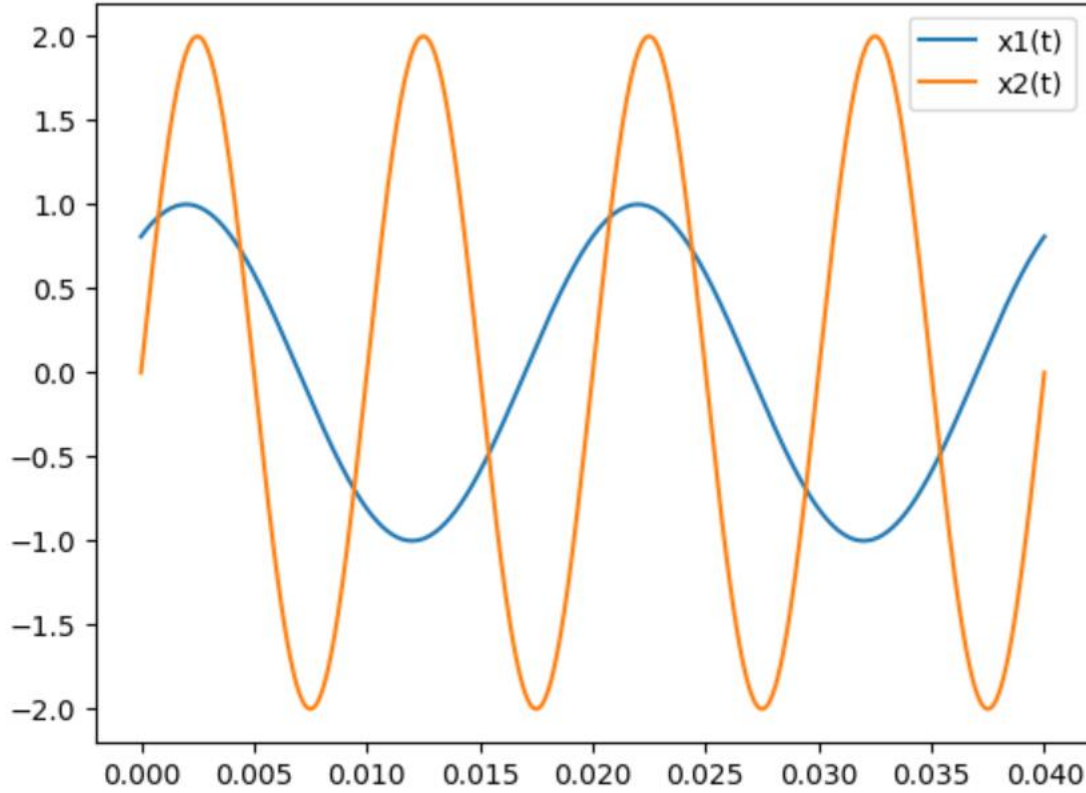


2.5 Tek figürde birden fazla işaret çizdirmek

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 atamanıza yarıyor (Not: vereceğiniz isimlerin sıraları çizdirdiğiniz işaretlerin plot()’daki sıralamayla aynı olmalı).



2.6 İşaretleri ötelemek

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

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

İşareti $n = 0, 1, 2, 3, 4$ noktalarından oluşan bir indis vektörü ile çizdirelim. (Not: Bunun için $x[n]$ vektörünün $[1, 0, 2, -1, 0]$ şeklinde olması gerektiğini dürtü işaretinin tanımından dolayı bilmenizi bekleriz. Eğer işaretiniz sınırlı noktadan oluşuyor ise, indis vektörünü belirlerken işaretin sıfırdan farklı değer aldığı noktaları kapsayacak şekilde oluşturmanız lazım. Mesela bu örnek için $n =$

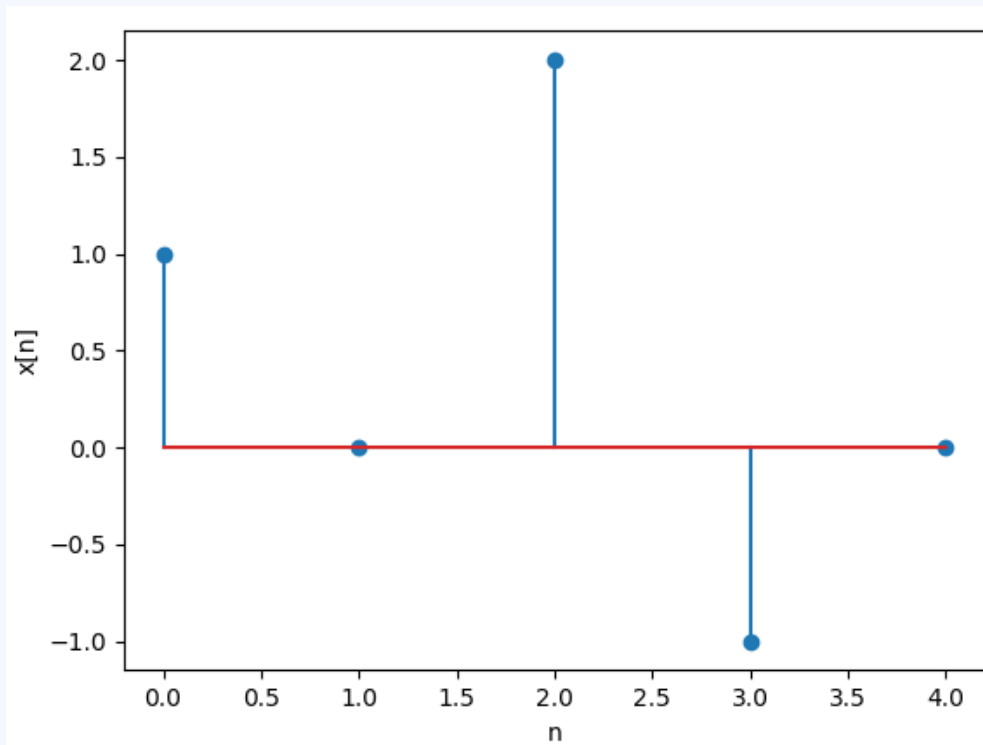
-5 'den 5 'e veya $n = 0$ 'dan 10 'a olacak şekilde de belirleyebilirsiniz. Aşağıda sırasıyla farklı indis vektörleri için $x[n]$ işaretinin olması gereken diziyi yazdım.

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

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

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 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()
```

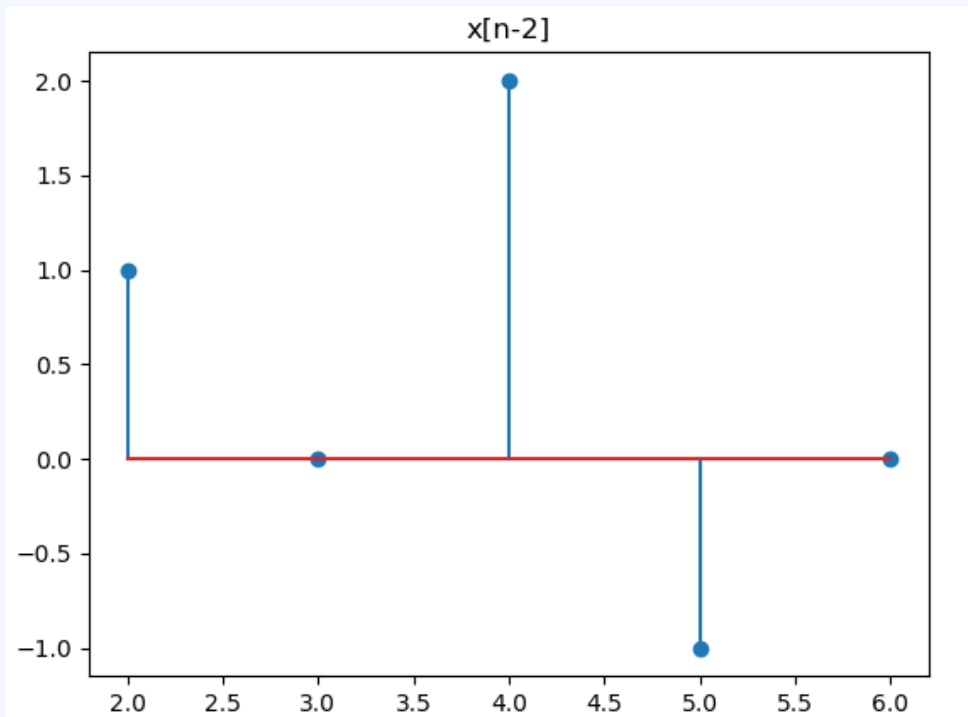


Şimdi yukarıdaki örnekte verilen $x[n]$ işaretini iki örnek sağa ötelenmiş işareti ($x[n - 2]$) çizdirelim. Bunun için iki farklı yol izleyebilirsiniz:

- 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()
```

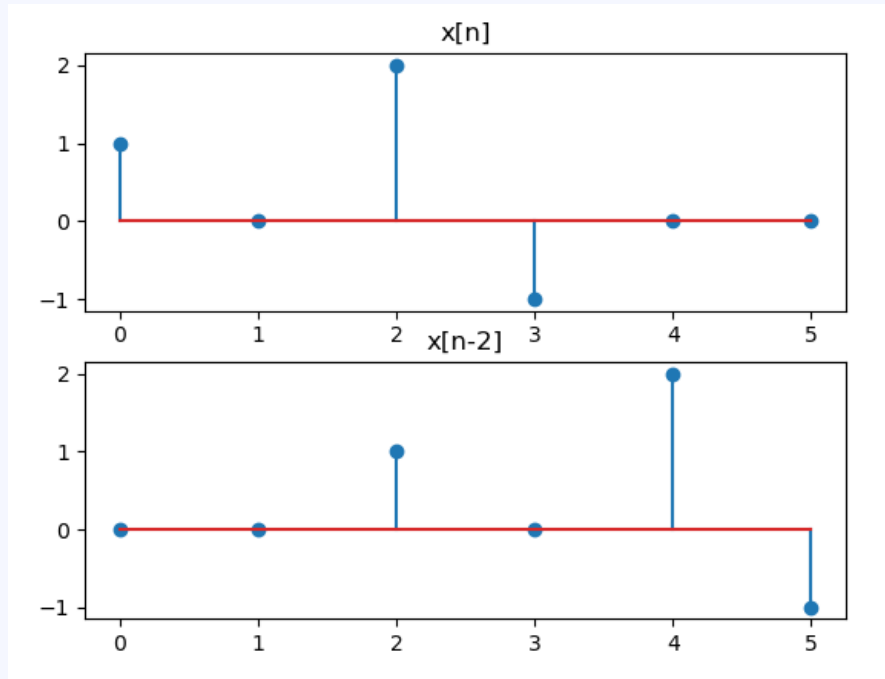


- Yol-2

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 farklı 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])
plt.subplot(2,1,1)
plt.stem(n,x)
plt.title('x[n]')
plt.subplot(2,1,2)
plt.stem(n,x_shifted)
plt.title('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.

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’nin `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])
```

Kompleks bir sayının genliğini bulmak için Numpy’nin `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])
```

```
print(np.abs(x))  
konsol çıktısı → [2.23606798 5.          ]
```

Kompleks bir sayının fazını bulmak için Numpy'nin `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 Çalışma Soruları

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

1) $a = 3 + 4j$ ve $b = 1 - 2j$ ise ;

$$a + b = C$$

$$\frac{a}{b} = D$$

C, D kompleks sayılarının gerçekte, sanal, genlik ve faz bileşenlerini print() komutu ile bastırın.

2) $x[n] = 2\cos(\frac{2\pi}{7}n + \frac{\pi}{5})$ işaretini 2 tam periyot olacak şekilde oluşturun ve çizdirin.

3) $x_1[n] = e^{\frac{j\pi}{4}n}$, $x_2[n] = e^{\frac{j\pi}{4}}$, $x_3[n] = e^{\frac{\pi}{4}}$ işaretlerini 10 noktalı olacak şekilde çizdirin.

4 TESLİM ŞEKLİ ve ZAMANI

Kodlar bölümünde yazılan kodları kendiniz bir Jupyter Notebook'ta yazarak sonuçları gözlemleyin. Jupyter Notebook'ta yaptığınız çalışmayı **OgrenciNo_Ad_Soyad_LABno.ipynb** formatına bir isimle kaydedip Ders Kutusu'na yükleyiniz. Laboratuvar ön çalışmaları (ev ödevi), laboratuvarın yapılacağı gün sabah 05:00'e kadar sisteme yüklenmelidir. Sisteme geç yüklenen dosyalar kabul edilmeyecektir. Ön hazırlık çalışmasını yapmamış (sisteme ön çalışmasını yüklememiş) öğrenciler aynı yapılacak laboratuvar çalışmasına giremez. Ekte, örnek bir ödev çözümü şablonu verilmektedir (bkz: **101024099_AYSE_SEN_LAB1.ipynb**). Jupyter Notebook'ta yapacağınız çözümler **bu şablona göre hazırlanmalıdır**.

