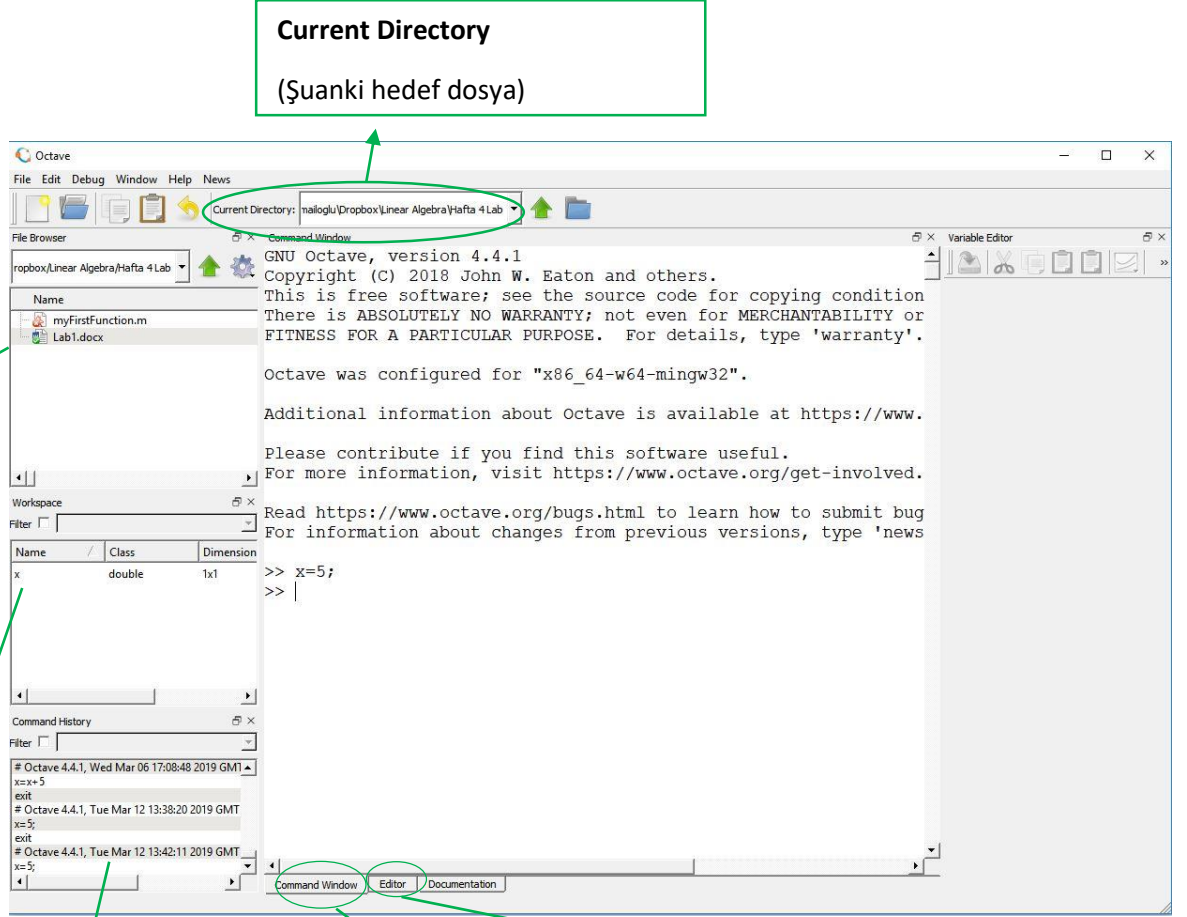


## LAB 1 Çalışma Kağıdı

Octave, MATLAB benzeri bir programlama dilidir. Fakat MATLAB'tan farklı olarak ücretsizdir.

<https://www.gnu.org/software/octave/download.html> adresinden indirilebilir.

### Default Octave Görüntüsü



### Bazı Kod Örnekleri

```
>> 3/10
```

ans = 0.30000 (ans burada yapılan en son işlemin sonucunu tutuyor)

```
>> ans*20
```

```
ans = 6
```

**Not:** Eğer Octave'in yapılan işlemin sonucunu göstermesini istemezsek işlem sonuna noktalı virgül koyarız.

Sonucu **ans** degiskeni yerine baska istedigimiz bir degiskende tutatbiliriz.

```
>> x = 3/10;
```

### Operatörlerin Öncelik Sırası

Parantez → Üs → Çarpma / Bölme → Toplama / Çıkarma

```
>> 4^2 *3-4/2
```

Yukarıdaki işlem öncelik sırasına göre şöyle yazilabilir:  $((4^2)*3) - (4/2) = (16*3) - 2 = 46$

```
>> 27^1/3+32^0.2
```

Yukarıdaki işlem öncelik sırasına göre şöyle yazilabilir:  $((27^1)/3)+(32^0.2) = (27/3)+(32^0.2)=11$

**Not:** Her ne kadar MATLAB’da operatorlerin yukarıdaki gibi islem sirasi olsada parantez kullanmak hata yapma riskinizi azaltir.

### Eşleme Operatörü ( = )

Octave’da = işareti sağdaki değeri sola atar.

```
>> x = 3;
```

(sağdan sola okuyun: 3’ü al x değişkenine at. )

### Üs – Karakök

Bir **a** sayısının **b**. üssünü alırken **a^b** komutu girilir.

```
>> 3^4
```

```
ans = 81;
```

Octave’da karakök komutu **sqrt** dir.

```
>> sqrt(81)
```

```
ans = 9;
```

ör. Yüksekliği  $h$  ve yarıçapı  $r$  olan silindir şeklindeki bir tankin hacmi  $V = \pi \cdot r^2 \cdot h$  formulu ile hesaplanır. Belirli bir tankin yüksekliği 15 m ve yarıçapı 8 m dir. Bu tanktan hacimce %20 daha büyük bir tank yapmak istiyoruz. Yüksekliği aynı kalmak kosuluyla bu tankin yarıçapı kaç olur?

```
>> V= pi*8^2*15
```

```
V=3015.9
```

```
>>V=V*1.2 %yeni silinidirin hacmini de V’de tutuyoruz.
```

```
V= 3619.1
```

```
>>r=sqrt(V/(pi*15))
```

```
>>r=8.7636
```

### Format Secenekleri

MATLAB’da virgulden sonra kaç basamak geleceğini **format short** yada **format long** komutuyla ayarlarız.

```
>> format short %virgulden sonra en fazla 4 basamak
```

```
>>pi
```

```
ans = 3.1416
```

```
>> format long %virgulden sonra en fazla 16 basamak
```

```
>>pi
```

```
ans = 3.141592653589793
```

## DİZİLER – VEKTÖRLER (ARRAY)

Vektörler, Octave’da sıralanmış sayıların bir dizisi olarak algılanır ve `[]` içinde gösterilir.

```
>> a = [-7, 4, 11, -5, 4, 16];
```

### Vektör Operatörleri

Hatırlarsak iki temel vektör operasyonumuz vardı: bir vektörü skalerle çarpmak ve iki vektörü toplamak.

```
>> 3*a
```

```
ans = -21 12 33 -15 12 48
```

Görüldüğü gibi vektörün tüm elemanları 3 skaleriyle çarpıldı.

```
>> b = [ 4 , 4, 12, 11, -6, 3];
```

```
>> a+b
```

```
ans = -3 8 23 6 -2 19
```

Görüldüğü gibi iki vektörün karşılıklı elemanları birbiriyle toplandı.

### Eşit Aralıklı Vektör

Ardışık bileşenler arası farkın sabit olduğu bir vektörü kolayca elde edebiliriz:

```
>> v = i : j: k % i başlangıç değeri, j artış miktarı, k son değer
```

```
v = [i, i+j, i+2j, i+3j, ..., k]
```

```
>> v= 2: 4: 17
```

```
v=2 6 10 14
```

### Diğer Vektör Operatörleri

1. `length()` : bu komut ile vektörün boyutunu hesaplarız.

```
>> length(a)
```

```
ans = 6
```

2. `find()` : bu komut ile bir vektör içinde belirli bir özelliği sağlayan bileşenlerin indislerine (pozisyonlarına) erişebiliriz.

```
>> find(a>0)%a vektörünün 0'dan büyük bileşenlerinin indislerini aryoruz
```

```
ans = 2 3 5 6
```

```
>> a(find(a>0)) %a vektörünün 0'dan büyük bileşenleri
```

```
>> find(a==4)%a vektörünün değeri 4 olan bileşenlerinin indislerini arıyoruz
```

```
ans = 2 5
```

3. `min - max`

```
[enKucuk, indis]=min(a)
```

**enKucuk**: a vektöründeki en küçük değer, **indis**: a vektöründeki en küçük değerın pozisyonu

```
>> [enKucuk, indis]=min(a)
```

```
enKucuk = -7;
```

```
indis = 1
```

### Özel Vektörler

1. **zeros(1,n)** : n – boyutlu bütün bileşenleri 0 olan vektör.

```
>> zeros(1,5)
```

```
ans = 0 0 0 0 0
```

2. **ones(1,n)** : n – boyutlu bütün bileşenleri 1 olan vektör.

```
>> ones(1,5)*3
```

```
ans = 3 3 3 3 3
```

### Bir Vektörün Tüm Elemanlarını Toplamak

```
>> sum(a) % a vektörünün tüm elemanlarını toplar
```

```
ans = 23
```

### Vektörlerin Karşılıklı Bileşenlerini Çarpmak

Aynı boyutlu iki vektörün karşılıklı elemanlarını çarparak, bu çarpımlardan oluşan yeni bir vektör elde edebiliriz.

```
>> a = [-2, 3, 4, 1];
```

```
>> b = [1, 3, -2, 0];
```

```
>>a.*b
```

```
ans = -2 9 -8 0
```

**ör.** Bir pazarda bir kilo elma 3 TL, bir kilo armut 4.5 TL ve bir kilo domates 7 lira olsun. 3.5 kg elma, 2 kg armut ve 2.5 kg domates alan biri bu alışveriş için ne kadar öder?

```
>> fiyat = [3, 4.5, 7];
```

```
>> sepet = [3.5, 2, 2.5];
```

```
>> toplamFiyat= sum(fiyat.*sepet)
```

```
>> toplamFiyat = 37
```

### İç Çarpım (Dot Product)

Aynı boyutlu iki vektörün karşılıklı elemanlarını çarpıp toplayarak iç çarpımı elde ediyorduk. Octave’da bunun için **dot** komutunu kullanacağız.

```
>> dot(fiyat, sepet)
```

```
ans = 37
```

İç çarpımı ayrıca transpoze ( kullanarak da yapabiliriz:

```
>> fiyat*sepet'
```

```
ans = 37
```

' komutu ile vektörün transpozunu alıyoruz. Böylece satır vektörü sütun, sütun vektörü satır olur .

## IF Statment

Genel Sintaks

```
if <test edilen boolean ifade>
    .....
    .....
end
```

If bloğu (Boolean ifade doğru iken olacaklar)

ör.

```
x=input('Bir sayı giriniz ');
if rem(x,2)==0 %eger x'in 2 ile bolumunden elde edilen 0 ise
    disp('Girdiginiz sayı çifttir);
else
    disp('the number you have entered is an odd number');
end
```

ör. Girilen sayı 3 ile 8 arasında ise yada 15'ten büyükse OK yazan kod:

```
if (x>3 && x< 8) || x > 15
    disp('OK');
end
```

**Not:** Octave'da ve && ile veya || gösterilir.

## Döngüler (Looplar)

### 1. For Döngüsü

Önceden belirlenen sayıda çalıştırılacak ifadeler için for loop kullanılır.

Genel Sintaks

```
for i = başlangicDegeri: artışMiktarı:bitişDeğeri
    ....
    ....
end
```

for bloğu (herbir for döngüsünde olacaklar)

ör. Ekrana 5 defa ben zekiyim yazdıralım.

```
for i = 1:5
    disp('Ben zekiyim');
end
```

ör. 1 ile 1000 arasında 13'ün katları olan sayıları toplayan program:

```
t =0;
for i = 1: 1000
    if rem(i, 13) == 0
        t=t+i;
    end
end
```

ör. Bir vektörde lineer arama yapalım.

```
v = [3, -2, 7, 11, 5, 99]
x = 11; % aradığımız sayı
ustSınır = length(v);
for i=1:ustSınır
    if v(i)==x
        disp('Aradığımız sayı bulundu! ');
        break; % sayıyı bulduğumuzdan for looptan çıkabiliriz.
    end
end
```

ör. Bir vektörü diğerine eşleyelim.

```
v = [3, -2, 7, 11, 5, 99];
vecUzun=length(v);
u = zeros(1, vecUzun); %bu vektörü v'nin elemanları ile dolduracağız
for i = 1: vecUzun
    u(i) = v(i);
end
```

ör. İki vektörün toplamı:

```
v = [3, -2, 7, 11, 5, 99];
u = [3, 6, 1, 0, 0, -4];
vecUzun=length(v);
z = zeros(1, vecUzun); %bu toplam vektörü olacak
for i = 1:vecUzun
    z(i) = v(i)+u(i);
end
```

## 2. While Döngüsü

While da for gibi bir döğüdür; ama bu döngünün kaç defa döneceği başlangıçta belli değildir. Belirli bir şart sağlanana kadar while dönmeye devam eder.

Genel Sintaks

```
while <test edilen boolean ifade>
```

```
.....  
.....  
.....
```



while bloğu (herbir while döğüsünde olacaklar)

```
end
```

ör. 1'den 100'e kadar olan sayıların toplamı

```
t=0;
```

```
summ=0;
```

```
while t<=100
```

```
    summ=summ+t;
```

```
    t=t+1; %test edilen degeri guncelliyoruz
```

```
end
```

ör. Kisi '0' girene kadar giridigi sayıları toplayan program

```
flag = true;
```

```
t = 0;
```

```
while flag
```

```
    x = input(' Bir sayı giriniz yada cikmak icin 0'a basiniz');
```

```
    if x==0
```

```
        flag = false; %while loop tan cikmak icin
```

```
    else
```

```
        t = t + x;
```

```
    end
```

```
end
```

```
disp(t); %toplami gosteriyoruz.
```

## İris Veri Seti

<http://ceng1.cumhuriyet.edu.tr/lectures/bil1006/>

adresinden irisEgitimX, irisEgitimY, irisTestX ve irisTestY veri setlerini current directory'e indirin.

İndirdiginiz veri setlerini aşağıdaki örnekte olduğu gibi importdata komutuyla Octave'a yükleyin.

```
>> irisEgitimX=importdata('irisEgitimX.mat');
```

**irisEgitimX** = 100 tane iris çiçeğinin fiziksel özellikleri (her bir satır bir çiçeğe denk geliyor)

**irisEgitimY**= 100 tane iris çiçeğinin sınıfı (1,2 yada 3)

**irisTestX** = 50 tane iris çiçeğinin fiziksel özellikleri (her bir satır bir çiçeğe denk geliyor). Buradaki çiçekleri sınıflandırmak istiyoruz.

**irisTestY**= Sınıflandırmak istediğimiz 50 tane iris çiçeğinin gerçek sınıfları (bu vektörü sınıflandırmamız ne kadar doğru olduğunu test ederken kullanacağız).

AMAC: **irisEgitimX** ve **irisEgitimY** den faydalanarak **irisTestX** deki çiçekleri sınıflandırmak.

Genel Strateji: **irisTestX**' deki her bir çiçek için (her bir satır) **irisEgitimX** deki en yakın çiçeği bulacağız. Bunun için ic carpım kullanacağız. Bulunan en yakın çiçeğin sınıfını **irisEgitimY** den öğreneceğiz. Bulunan sınıfı, sınıfını merak ettiğimiz çiçeğin sınıfı olarak tahmin edeceğiz.

En son olarak, 50 test çiçeği için yaptığımız tahminlerle bu çiçeklerin gerçek sınıflarının kayıtlı olduğu **irisTestY** sınıfları ile karşılaştıracamız.