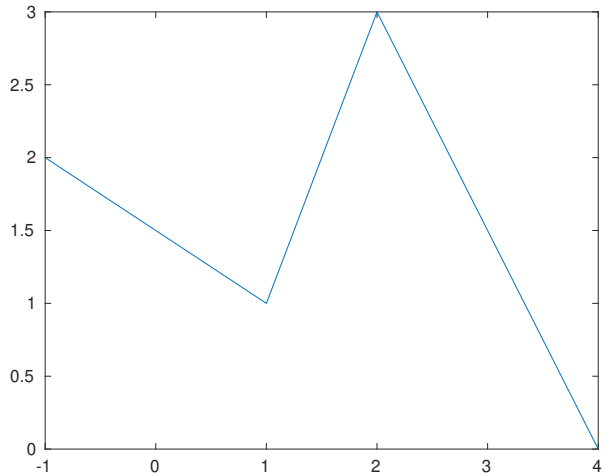
# Pontok, vonalak a síkon: `plot`

`plot(x,y)`
Töröttvonallal összekötve ábrázolja azokat a síkbeli pontokat, melyek 1. koordinátája az `x`, 2. koordinátája az `y` vektorban van felsorolva.

Rajzoltassuk ki a $(-1, 2)$, $(1, 1)$, $(2, 3)$, $(4, 0)$ pontokat összekötő töröttvonalat!

1. készítsünk egy vektort a pontok első koordinátáiból: `x=[-1,1,2,4]`
2. készítsünk egy vektort a második koordinátákból: `y=[2,1,3,0]`
3. hívjuk meg a `plot` függvényt.

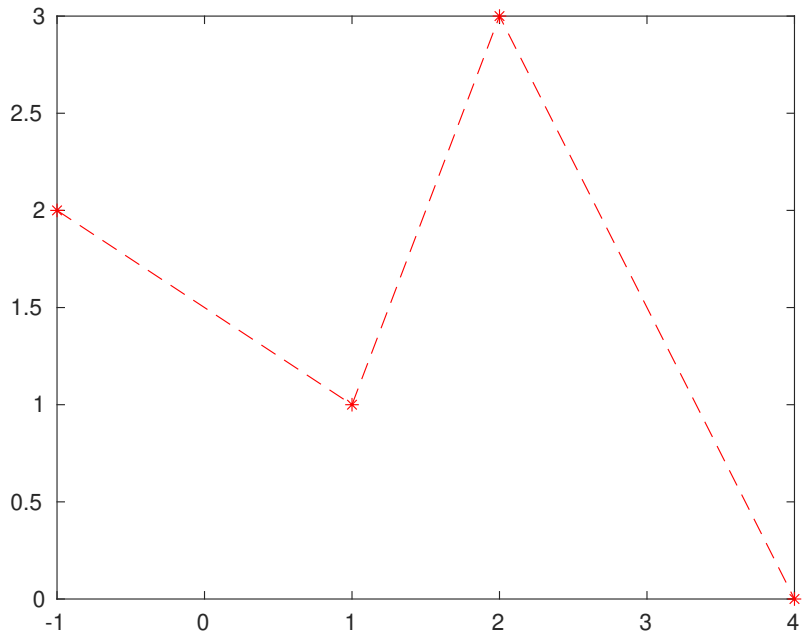
```
>> x=[-1,1,2,4];y=[2,1,3,0];
>> plot(x,y)
```



# Pontok, vonalak a síkon: `plot`

A `plot` függvény harmadik argumentumában megadhatjuk az összekötő vonal és a pontokat jelző marker típusát, színét. Pl.

```
>> x=[-1,1,2,4];y=[2,1,3,0];
>> plot(x,y,'r*--')
```




Ha csak markert adunk meg, vonaltípust nem, akkor nem köti össze a pontokat.

## Markerek

- * csillag
- o kör
- + összeadás jel
- x kereszt
- s négyzet
- d rombusz
- p ötszög
- h hatszög
- < balra mutató háromszög
- > jobbra mutató háromszög
- ∧ felfele mutató háromszög
- ∨ lefele mutató háromszög

## Vonaltípusok

- - folyamatos vonal (alapértelmezés)
- : pontozott vonal
- - - szaggatott vonal
- -. szaggatott-pontozott vonal

## Színek

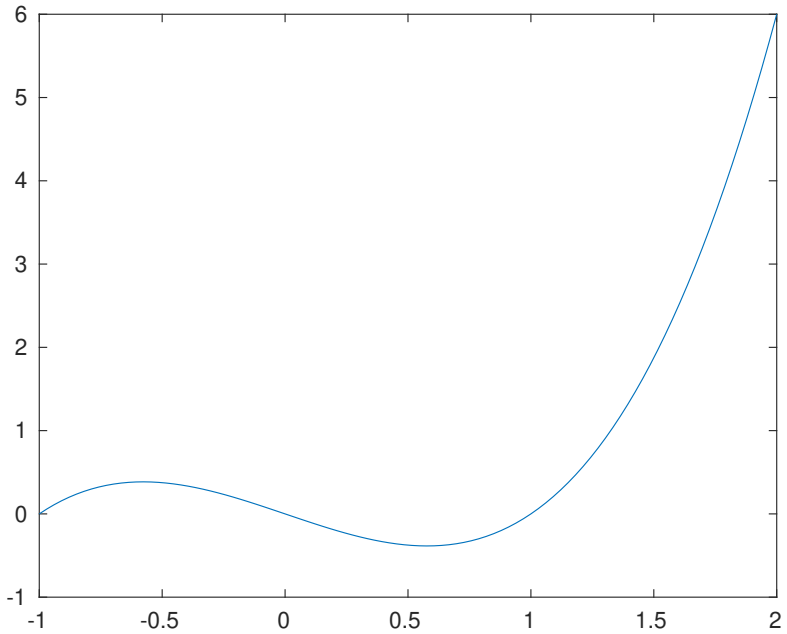
- b kék
- r piros
- g zöld
- k fekete
- w fehér
- y sárga
- m magenta
- c cián

# Pontok, vonalak a síkon: `plot`

Ha egy függvényt szeretnénk ábrázolni, akkor számítsuk ki az értékét sok pontban, és a pontpárokat ábrázoljuk.
Pl. az $f(x) = x^3 - x$ függvény a $[-1, 2]$ intervallumon:

```
x=linspace(-1,2);
y=x.^3-x;
plot(x,y)
```

`x`: 100 egyenlő lépésközű pont a $[-1, 2]$ intervallumból
`y`: a függvényértékek az `x`-beli pontokban. Elemenkénti műveletek!



# Anoním függvények, function handle

Függvényeket definiálhatunk a következő módon:

```
>> fv=@(x) x.^3-x;
```

Ilyen módon az $fv(x) = x^3 - x$ függvényt definiáltuk, hívása pl.:

```
>> fv(2)
ans =
     6
```

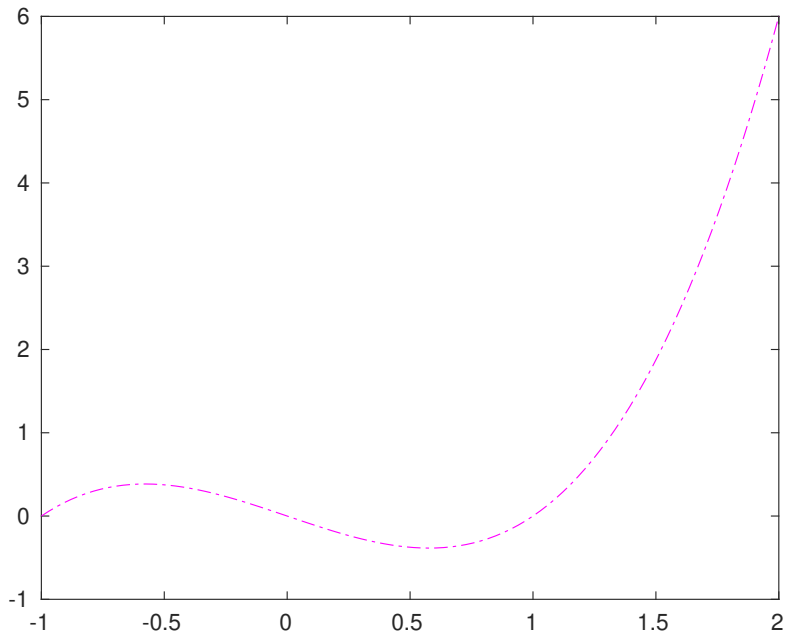
A @ szimbólum után zárójelben szerepelnek a függvény változói (most $x$), ezt követi a függvény (ez egy ú.n. anoním függvény). Az = baloldalán szereplő változó (most fv) egy ú.n. ,,function handle" típusú változó lesz.

Mivel a függvényt ,,elemenkénti műveletekkel" definiáltuk, ezért akár egy vektorral is hívhatjuk. Ebben az esetben a függvényt a vektor minden elemére kiértékeli, és a függvényértékek vektorát adja vissza.

# Pontok, vonalak a síkon: `plot`

Függvény ábrázolásánál is megadhatjuk a vonaltípust és színt is, illetve a függvényt definiálhatjuk function handle-ként:

```
x=linspace(-1,2);
f=@(x) x.^3-x;
plot(x,f(x),'m-.')
```



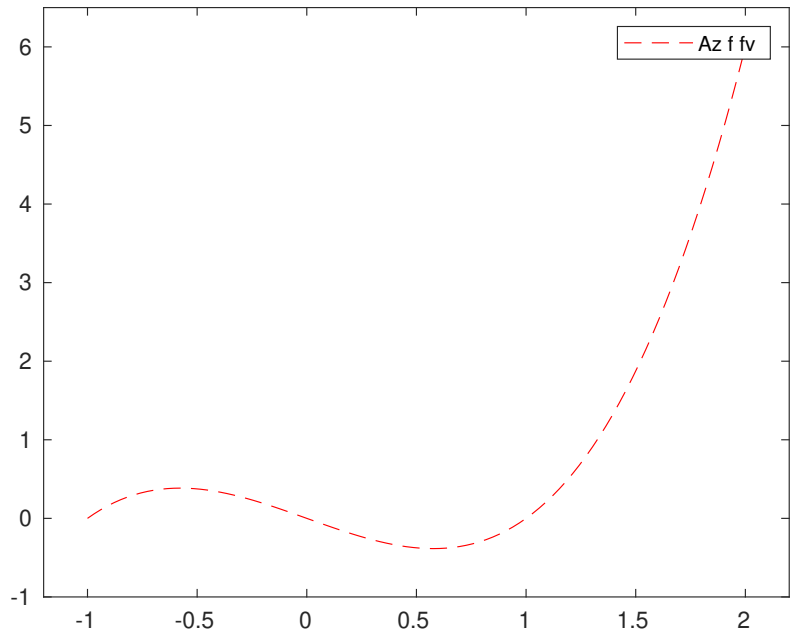
# Pontok, vonalak a síkon: figure, axis, legend

A figure utasítás hatására egy új grafikus ablak nyílik. Ennek hiányában, ha van megnyitott grafikus ablak, akkor abba készíti el az ábrát, annak korábbi tartalmát felülírva.

axis([xmin,xmax,ymin,ymax]) beállítja a tengelyek határait (ld. még az xlim és ylim függvényeket).

A legend segítségével magyarázó felíratot készíthetünk.

```
x=linspace(-1,2);
f=@(x) x.^3-x;
figure; plot(x,f(x),'r--')
axis([-1.2,2.2,-1,6.5]);
legend('Az f fv')
```



# Tengelyek tulajdonságai: gca

gca: (get current axes) az aktuális ábrán a tengelyek jellemzőit kérdezhetjük le, illetve mósosíthatjuk ezeket.

PI az utolsó ábra esetén:

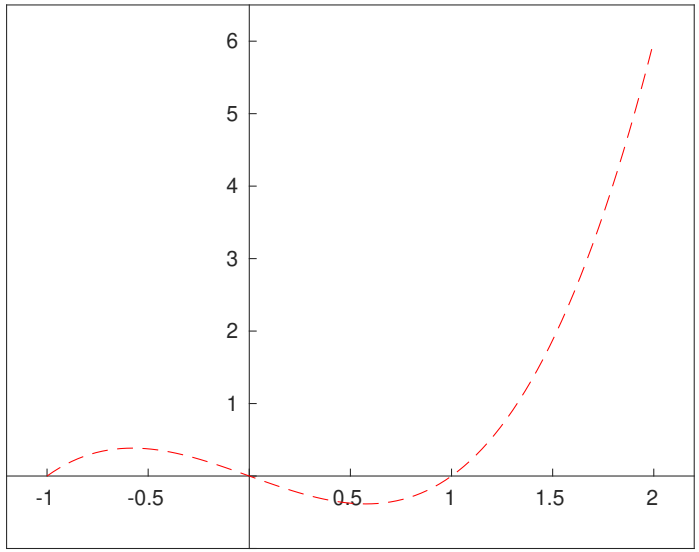
```
>> gca
ans =
  Axes with properties:
              XLim: [-1.2000 2.2000]
              YLim: [-1 6.5000]
            XScale: 'linear'
            YScale: 'linear'
    GridLineStyle: '-'
          Position: [0.1300 0.1100 0.7750 0.8150]
             Units: 'normalized'

  Show all properties
```

Csak akkor működik, ha van nyitott grafikus ablak!

# Tengelyek tulajdonságai: gca

Pl. tengelyek pozícionálása (az origóba):

```
x=linspace(-1,2);
f=@(x) x.^3-x;
figure; plot(x,f(x),'r--')
axis([-1.2,2.2,-1,6.5]);
ax=gca;
ax.XAxisLocation='origin';
ax.YAxisLocation='origin';
```




A kód utolsó 3 sora:

ax=gca; *az ax változóba menti a jelenlegi tengely-jellemzőket*

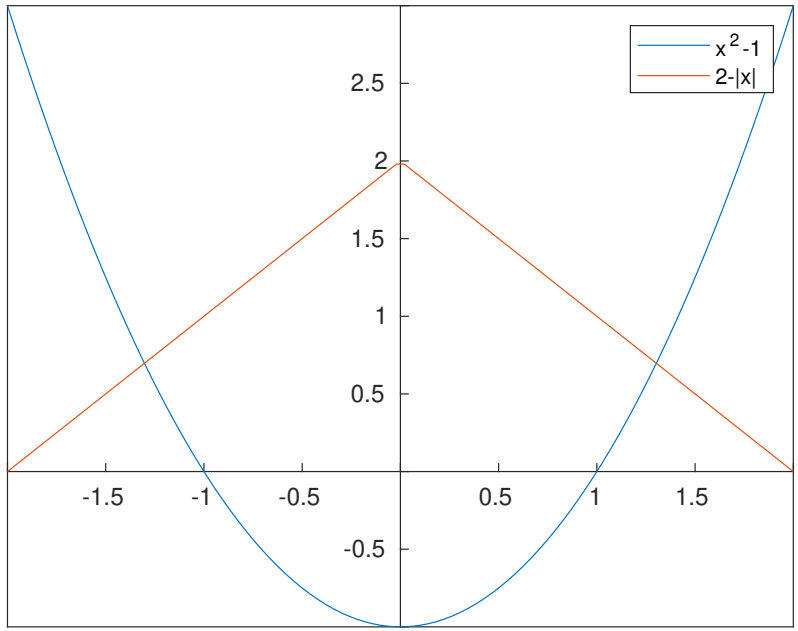
ax.XAxisLocation='origin'; *az x-tengely az origón haladjon át*

ax.YAxisLocation='origin'; *az y-tengely az origón haladjon át*

# Több függvény egy ábrán

Ha egyszer hívjuk a `plot` függvényt, több argumentummal:

```
x=linspace(-2,2);
y=x.^2-1;
z=2-abs(x);
figure; plot(x,y,x,z)
legend('x^2-1','2-|x|')
ax=gca;
ax.YAxisLocation='origin';
ax.XAxisLocation='origin';
```




`plot(x,y,x,z)`: ábrázolja az `(x,y)` és `(x,z)` párokat. Akár még több párt is megadhatunk így. Minden pár után külön megadhatjuk a színt és a vonaltípust. Pl. `plot(x,y,'r--',x,z,'m-.')`

legend: több sztringet is felsorolhatunk, ekkor a rajzolás sorrendjében rendeli hozzá a vonalakhoz a sztringeket.

# Több függvény egy ábrán

Az előző ábra a `plot` függvény többszöri hívásával:

```
x=linspace(-2,2);
y=x.^2-1;
figure; plot(x,y)
z=2-abs(x);
hold on;
plot(x,z)
legend('x^2-1','2-|x|')
ax=gca;
ax.YAxisLocation='origin';
ax.XAxisLocation='origin';
hold off;
```

`hold on` bekapcsolja a ,,rárajzoló" üzemmódot: az aktuális grafikus ablakba rajzol, az ottani eredeti ábra meghagyásával

# Több függvény egy ábrán

Az előző kód részletezve:

| | |
|---|---|
| `x=linspace(-2,2);` | *felveszünk 100 pontot a $[-2,2]$ intervallumban* |
| `y=x.^2-1;` | *kiszámítjuk $x^2-1$ értékét az x-beli pontokban* |
| `figure; plot(x,y)` | *nyitunk egy grafikus ablakot, ábrázoljuk az (x,y) párokat* |
| `z=2-abs(x);` | *kiszámítjuk $2-\|x\|$ értékét az x-beli pontokban* |
| `hold on;` | *bekapcsolja a rárajzoló módot* |
| `plot(x,z)` | *ábrázoljuk az (x,z) párokat* |
| `legend('x^2-1','2-\|x\|')` | *magyarázó szövegdoboz* |
| `ax=gca;`<br>`ax.XAxisLocation = 'origin';`<br>`ax.YAxisLocation = 'origin';` | *a tengelyek az origón haladjanak át* |
| `hold off;` | *kikapcsolja a rárajzoló módot* |

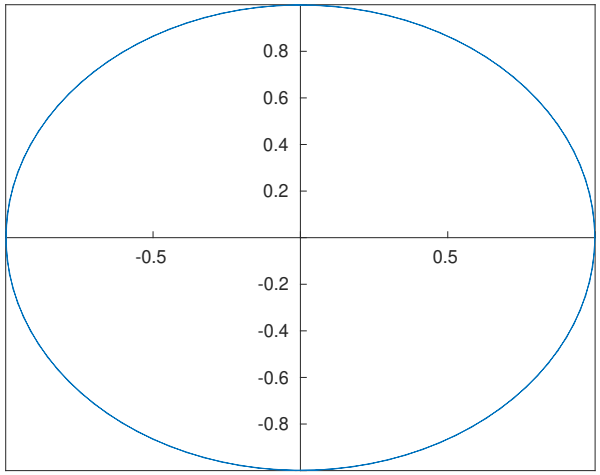
# axis equal

A tengelyeken egyforma hosszú egységet választ.

Példa: rajzoljunk egy kört!

Miközben $\alpha$ befutja a $[0, 2\pi]$ intervallumot a $(\cos\alpha, \sin\alpha)$ pontpárok az origó középpontú egység sugarú körön futnak végig:

```
alfa=linspace(0,2*pi);
figure;
plot(cos(alfa),sin(alfa))
ax=gca;
ax.YAxisLocation='origin';
ax.XAxisLocation='origin';
```

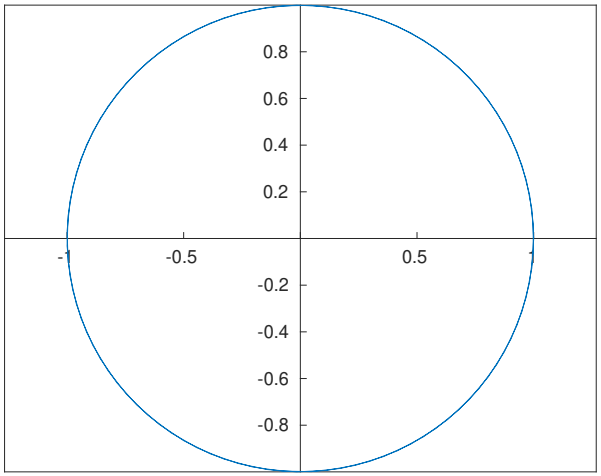



Mivel nem egyforma hosszú az egység a két tengelyen, így az alakzat nem látszik körnek.

# axis equal

Adjunk hozzá az előző kódhoz még egy parancsot:

```
alfa=linspace(0,2*pi);
figure;
plot(cos(alfa),sin(alfa))
ax=gca;
ax.YAxisLocation='origin';
ax.XAxisLocation='origin';
axis equal;
```




`axis equal`: egyforma hosszú egységet választ minden tengelyen.

## 1. feladat

Olvassa be a `salary.xlsx` állományt. Ennek első oszlopában a szakmai tapasztalat (években), másodikban az megfelelő éves fizetések szerepelnek, valamilyen valutában. (Forrás: Kaggle) Ábrázolja az adatokat (diszkrét pontokként): a vízszintes tengelyen az éveket, a függőlegesen a fizetéseket.

Az adatok beolvasásához használhatja a `readtable` függvényt.

Ekkor a beolvasott értékek egy táblázat (`table`) típusú változóba kerülnek. Mivel a táblázat most csak numerikus adatokat tartalmaz, így a `table2array` függvénnyel numerikus tömbbé alakítható.

Az ábra alapján milyen jellegű kapcsolat látszik a szakmai tapasztalat és a fizetések között?

## 2. feladat

Olvassa be a `baleset.xlsx` állományt. Ebben a magyarországi lakosság számának alakulását találja 1990 és 2022 között, illetve ugyanezen években a személyi sérüléssel járó közúti közlekedési balesetek számát (forrás: KSH). Ábrázolja mindkét adatsort, majd ábrázolja az évek függvényében a 100000 főre jutó balesetek számát is.

## 3. feladat

Olvassa be a `japan_h_w_man.xlsx` állományt. Ebben többek között 5-17 éves japán fiúk átlagos magassága (cm) és testsúlya (kg) szerepel, városonkénti bontásban. (Forrás: Kaggle) Ábrázolja a testsúlyokat a magasság függvényében! Ábrázolja ugyanezt csak a 6 éves fiúk esetén.