for-ciklus

```
for ciklusvaltozo=vektor
  utasitasok
end
```

- A `for` és az `end` kulcsszavak határolják a for-ciklust.
- A törzsben szereplő utasítások általában függenek a ciklusváltozó értékétől.
- A ciklusváltozó lehetséges értékei a ciklus fejében szerepelnek, a ,,vektor"-ban felsorolva.
- Az utasításokat a ciklusváltozó minden lehetséges értékére végrehajtja.

```
for k=1:3
    disp(['A ciklusvaltozo erteke most:', num2str(k)]);
end
```

- A ciklusváltozó neve ebben az esetben `k`
- A ciklusváltozó lehetséges értékei az `1:3`, azaz az `[1,2,3]` vektorban vannak felsorolva
- a törzsben egyetlen utasítás szerepel. A disp egy kiiratő (display) függvény, az argumentumában egy vektor áll (ld. szögletes zárójelek!), melynek két eleme van, ezek most sztring (szöveg) típusúak. Az első elem a ,,A ciklusvaltozo erteke most:" szöveg, a második pedig a `k` aktuális számértéke szöveggé alakítva.

Az eredmény:

```
A ciklusvaltozo erteke most:1
A ciklusvaltozo erteke most:2
A ciklusvaltozo erteke most:3
```

```
s=1;
for i=1:2:9
    s=s*i;
    disp(['s erteke:',num2str(s)]);
end
```

- Itt az `i` ciklusváltozó az 1,3,5,7,9 értékeket veszi fel (1-től 2-es lépésközzel lépünk 9-ig).
- A ciklus minden lépésében az `s` értékét szorozza az `i` aktuális értékével (`s` kezdőértéke 1 volt) és kiírja az `s` pillanatnyi értékét.

Az eredmény:

```
s erteke:1
s erteke:3
s erteke:15
s erteke:105
s erteke:945
```

←Az `s` kezdőértékét megszorozta `i`=1-gyel
←Az `s` előző értékét megszorozta `i`=3-mal
←Az `s` előző értékét megszorozta `i`=5-tel
←Az `s` előző értékét megszorozta `i`=7-tel
←Az `s` előző értékét megszorozta `i`=9-cel

Függvények írása

A Matlab-függvények szerkezete:

```
function [kimenovaltozok]=fvneve(bemenovaltozok)
  utasitasok
end
```

- A függvény a `function` és `end` kulcsszavak között helyezkedik el
- a függvény bemenő változói a kerek zárójelek között vanank felsorolva, vesszővel elválasztva
- a függvény által visszaadott értékek a szögletes zárójelben vannak felsorolva, vesszővel elválasztva
- a bemenő értékekből az `utasitasok`-ban megadott módon határozza meg a visszaadott értékeket

Ha a fenti függvényt egy külön m-fájlban írtuk meg, akkor fvneve.m néven kell elmenteni. Ha a függvényt egy kód részeként hoztuk létre, akkor az m-fájl végén kell állnia (ekkor akár több ilyen függvény is lehet a fájl végén). A függvény ilyenkor csak az adott m-fájlon belül hívható (ú.n. lokális függvény).

Példák

```
function  y=fv1(x)
   y=x.^2-1;
end
```

Ekkor a `y=fv1(x)` utasítás eredménye a $x^2 - 1$ kifejezés értéke, ahol x akár vektor is lehet. Utóbbi esetben a függvény elemenként hajtódik végre és y is vektor (ezt az teszi lehetővé, hogy a függvényben minden művelet végrehajtható vektorokra is, mivel a négyzetreemelés jele elé kitettük a . jelet)
Például `y=fv1(-3)` esetén:

```
y =
     8
```

`y=fv1([1,2,3])` esetén:

```
y =
     0     3     8
```

Példák

```
function  [s,p]=fv2(x,y,z)
    s=x+y+z;
    p=x*y*z;
end
```

Ennek a függvénynek 3 bemenő paramétere van, ezeket vesszővel elválasztva soroljuk (akkor is, ha nem azonos típusúak, pl. egy skalár egy vektor és egy mátrix lenne).
A függvény 2 értékkel tér vissza: a bemenetként megadott 3 szám összegével és szorzatával.
Ha több visszatérési érték van, akkor ezeket a függvény létrehozásakor és hívásakor szögletes zárójelben soroljuk (akkor is, ha nem azonos típusúak).

```
>>[s,p]=fv2(-5,1,4)
s =
     0
p =
   -20
```

Ha a függvény hívásakor nem adunk meg kimenő változókat (vagy csak egyet adunk), akkor a két visszatérési érték közül az elsőt adja:

```
>>fv2(-5,1,4)
ans =
     0
```

Ha csak a második visszatérési értékre vagyunk kiváncsiak, akkor

```
>>[~,p]=fv2(-5,1,4)
p =
     -20
```

function handle

A function handle egy függvényekkel kapcsolatos adattípus.
Szükségünk lehet rá, pl. ha

- egy függvényt át akarunk adni egy másik függvénynek (pl a Matlab `integral` függvényének az integrálandó függvényt)
- parancssorban szeretnénk függvényt definiálni

Létrehozása: a @ szimbólum után következik

- egy már létező függvény neve, pl. az előbb létrehozott függvényekkel `af1=@fv1` vagy `af2=@fv2`
- egy ú.n. anoním függvény, pl. `af3=@(x) x.^3-x` vagy `af4=@(x,y) x+y-x.*y`

Hívása:

- `af1(-3)` vagy `af1([1,2,3])` vagy `[s,p]=af2(-5,1,4)`
- `af3(-3)` vagy `af3([1,2,3])` vagy `af4(6,1)`

Az előző példában miért

`af1([1,2,3])` és `af2(-5,1,4)`?

Az egyik esetben miért használtunk szögletes zárójelet, a másikban miért nem?

Az `af1` (illetve az `fv1`) egy egyváltozós függvény. Ez az egy változó lehet akár vektor is (`fv1`-ben elemenkénti műveleteket használtunk!), de a függvényt csak egyetlen változóval hívhatjuk. Ez most az `[1,2,3]` vektor. Ha elhagynánk a szögletes zárójelet, akkor 3 skalárral próbálnánk hívni a függvényt.

Az `af2` (illetve az `fv2`) egy háromváltozós függvény, 3 értékkel kell hívnunk. Ezek most a -5, 1 és 4. Ha szögletes zárójelbe tennénk a 3 értéket, akkor az egyetlen változó lenne (egy vektor).

Ha

`af3=@(x) x.^3-x` és `af4=@(x,y) x+y-x.*y`

akkor `af3` egy egyváltozós függvény (ahol a változó akár vektor is lehet, mert a hatványozás előtt használtuk a pontot). Hívhatjuk egyetlen számmal, ekkor egy számot ad vissza, vagy egy vektorral, ekkor a vektor minden elemére kiértékeli a függvényt (és az értékek vektorát adja vissza).

`af4` egy kétváltozós függvény. Hívhatjuk 2 számmal:

```
>> af4(6,1)
ans =
     1
```

de két (egyforma méretű) vektorral is, mert a szorzást elemenként definiáltuk. Ekkor a visszaadott érték is vektor:

```
>> af4([6,-1],[1,2])
ans =
     1     3
```

Kiértékelte a függvényt a $(6, 1)$ és $(-1, 2)$ párokra.