TI*K*Z

KOVÁCS KRISTÓF, MAGYAR ANDRÁS, SIMON ANDRÁS

Tartalomjegyzék

1. A legegyszerűbb rajzok

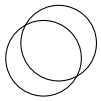
Mindenekelőtt: tegyük `\usepackage{tikz}`-t a preambulumba.

1.1. Körök, egyenesek,... Egyszerű dolgokat egyszerű csinálni, ezt a szövegközi kört pl. így lehet:

```
\begin{tikzpicture}
 \draw (0,0) circle (0.5);
\end{tikzpicture}
```

Jelentése: rajzolj $(0,0)$ középpontú, $0,5$ cm átmérőjű kört. A középpontnak így, hogy más nincs a rajzon, semmi jelentősége, mert akárhova is rajzoljuk a kört, a „papírnak” azon része jelenik meg, ahova rajzoltunk valamit: semmi sem árulja el, hogy ezt

```
\begin{tikzpicture}
 \draw (0.2,0.2) circle (0.5) ;
\end{tikzpicture}
```

rajzolta. De persze ha egy rajzon vannak,  akkor már van jelentősége a középpontok (egymáshoz képest való) elhelyezkedésének:

```
\begin{tikzpicture}
 \draw (0,0) circle (0.5) ;
 \draw (0.2,0.2) circle (0.5) ;
\end{tikzpicture}
```

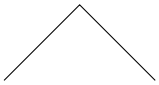
Tipikusan nem sorközi ábrákat rajzolunk, hanem (úszó) képeket, amiknek ezért így néz ki a „vázuk”:

```
\begin{figure}[h]
 \begin{center}
  \begin{tikzpicture}
    ...
  \end{tikzpicture}
 \end{center}
\caption{Ábra címe}
\end{figure}
```

Kört már tudunk rajzolni, szakaszt így lehet:

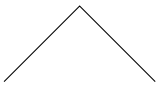
```
\begin{tikzpicture}
 \draw (0,0) -- (1,1) ;
\end{tikzpicture}
```

Töröttvonalat lehet szakaszonként rajzolni:



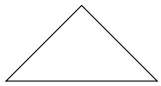
```
\begin{tikzpicture}
 \draw (0,0) -- (1,1);
 \draw (1,1) -- (2,0);
\end{tikzpicture}
```

de egy szuszra is:



```
\begin{tikzpicture}
 \draw (0,0) -- (1,1) -- (2,0);
\end{tikzpicture}
```

Ezt az egy szuszra csinált dolgot (valójában mindent ami a `\draw` és a pontosvessző között áll) úgy hívják, hogy *path*, és majd kiderül, hogy nem csak szakaszok szerepelhetnek benne. Meg az is, hogy nem kell az egészet megrajzolni, fel lehet emelni közben a tollat. És le is zárhatjuk anélkül, hogy újból megadnánk a kezdőpontját:



```
\begin{tikzpicture}
 \draw (0,0) -- (1,1) -- (2,0) -- cycle;
\end{tikzpicture}
```

Ennek egyik előnye, hogy Ti*k*Z tudja, hogy zárt, ami pl. akkor hasznos, ha be akarjuk satírozni az általa határolt tartományt:

```
\begin{tikzpicture}
 \draw[fill, color=red]
   (0,0) -- (1,1) -- (2,0) -- cycle;
\end{tikzpicture}
```

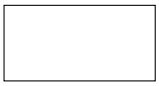
Ugyanezt írhattuk volna így is:

```
\begin{tikzpicture}
 \fill[red] (0,0) -- (1,1) -- (2,0) -- cycle;
\end{tikzpicture}
```

A *path*-ok végeire nyilat is tehetünk:

```
\begin{tikzpicture}
 \draw [<->>] (0,0) -- (1,0) -- (1,1);
\end{tikzpicture}
```

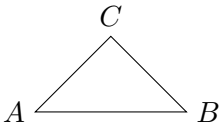
Téglalapot is könnyen rajzolhatnánk zárt töröttvonalként, de így is lehet:



```
\begin{tikzpicture}
 \draw (0,0) rectangle (2,1);
\end{tikzpicture}
```

Két szemközti csúcs koordinátáit kell megadni.

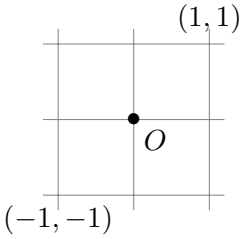
1.2. A részek megnevezése Ha meg akarjuk címkézni az ábra különböző részeit, létrehozunk egy „csomópontot” (node), ami lényegében egy doboz, amibe írhatunk:




```
\begin{tikzpicture}
 \draw (0,0) node[left] {$A$} -- (1,1)
    node[above] {$C$} -- (2,0)
    node[right] {$B$} -- cycle;
\end{tikzpicture}
```

Egy *path* konstruálása során bármikor készíthetünk ilyet.

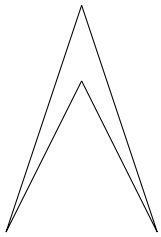
A *node*-ok elhelyezését opcionális argumentumokkal szabályozhatjuk; erre itt egy további példa, amiben az egyik ponthoz ráadásul több is tartozik:




```
\begin{tikzpicture}
 \draw[very thin, gray]
   (-1.2,-1.2) grid (1.2,1.2);
 \draw (0,0)
   node[below right] {$O$}
   node {$\bullet$};
 \draw (-1,-1) node[below] {$(-1,-1)$};
 \draw (1,1) node[above] {$(1,1)$};
\end{tikzpicture}
```

`\draw (x, y) grid (x', y')` az (x, y) bal alsó és (x', y') jobb felső csúcsú tengelypárhuzamos téglalap és az egész koordinátájú osztópontokkal rendelkező négyzetrács metszetét rajzolja meg.

Nem csak az ábrán adhatunk nevet az érdekes részeknek, de a rajzolás során is a koordinátáknak:

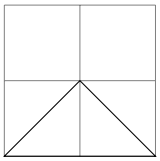


```
\begin{tikzpicture}
  \coordinate (O) at (0,0) ;
  \coordinate (O') at (2,0);
  \coordinate (A) at (1,2) ;
  \coordinate (B) at (1,3) ;
  \draw (O) -- (A) (B) -- (O);
  \draw (O') -- (A) (B) -- (O') ;
\end{tikzpicture}
```

További újdonság itt, hogy rajzolás közben „felemelhetjük a tollat”, most éppen (A) és (B) között. (Bár egyszerűbb lett volna anélkül.)

Azon kívül, hogy így olvashatóbb a kód, változtatni is könnyebb rajta, mert esetleg csak valamelyik nevet kell máshogy definiálni, nem kell végigmenni a kódon és egyenként megváltoztatni a megfelelő koordinátákat.

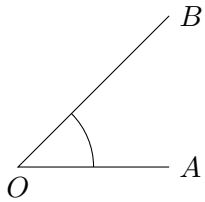
1.3. Polárkoordináták Ami már eddig kiderült: dolgozhatunk a szokásos, Descartes-féle koordinátarendszerben. De polárkoordináta-rendszerben is (ilyenkor a kezdőirányt és a kezdőpontot rögzítő félegyenes az $(1, 0)$ vektor). Polárkoordináta-rendszerben a `(szög:távolság)` szintaxissal adunk meg egy pontot, szöget pedig fokokban. Az `(1,0)` pontot polárkoordinátásan `(0:1)` adja meg.



```
\begin{tikzpicture}
 \draw[very thin, gray] (0,0) grid (2,2) ;
 \draw (0,0) -- (45:{sqrt(2)})
   -- (2,0) -- cycle ;
\end{tikzpicture}
```

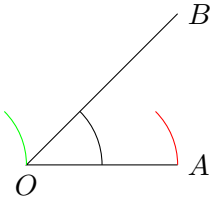
A 3.3 szakaszban van leírva, hogy milyen függvényeket használhatunk. `sqrt(2)`-t azért kellett itt bajusz-zárójelek közé tenni, mert szerepel benne (normál) zárójel.

1.4. Ívek Ív (`arc`) is lehet egy *path* része, de vigyázzunk, mert nem biztos, hogy úgy működik, ahogy az ember várná:




```
\begin{tikzpicture}
 \draw (2,0) node[right] {$A$}
   -- (0,0) node[below] {$O$}
   -- (2,2) node[right] {$B$};
 \draw (1,0) arc (0:45:1) ;
```

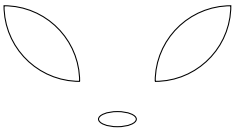
Az előtte szereplő pont nem a középpont, hanem az ív kezdőpontja.




```
\begin{tikzpicture}
 \draw (2,0) node[right] {$A$}
   -- (0,0) node[below] {$O$}
   -- (2,2) node[right] {$B$};
 \draw[green] (0,0) arc (0:45:1) ;
 \draw (1,0) arc (0:45:1) ;
 \draw[red] (2,0) arc (0:45:1) ;
\end{tikzpicture}
```

A szintaxis: `(b:e:r)`, ahol [`b`,`e`] a megrajzolandó szög-intervallum, `r` pedig a sugár.

Ív természetesen lehet egy nagyobb *path* része is[1] :



```
\begin{tikzpicture}
 \draw (0,0) arc (0:90:1) arc (180:270:1);
 \draw (1,0) arc (270:360:1) arc (90:180:1);
 \draw (0.5,-0.5)
   circle[x radius = 0.25, y radius = 0.1];
\end{tikzpicture}
```

amiben vegyesen előfordulhatnak ívek és szakaszok (és rácsok, körök,...):

```
\begin{tikzpicture}
 \draw (0,0) -- (1,0) arc (180:0:1) -- (4,0);
\end{tikzpicture}
```

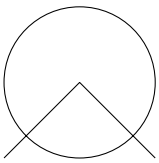
A két lehetséges félkör közül $0 - 180 < 0$ miatt rajzolódik meg a fenti, azaz a negatív irányban bejárt. Ha a másik félkört szeretnénk megrajzolni, akkor úgy kell megadni a szög-intervallumot, hogy a végpontja nagyobb legyen, mint a kezdőpontja:

```
\begin{tikzpicture}
 \draw (0,0) -- (1,0) arc (180:360:1) -- (4,0);
\end{tikzpicture}
```

vagy akár

```
\begin{tikzpicture}
 \draw (0,0) -- (1,0) arc (-180:0:1) -- (4,0);
\end{tikzpicture}
```

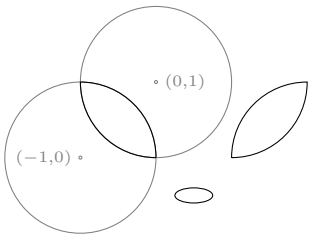
Körökről (mint összetettebb *path*-ok részeiről) esvén szó:



```
\begin{tikzpicture}
 \draw (0,0) -- (1,1) circle (1) -- (2,0);
\end{tikzpicture}
```

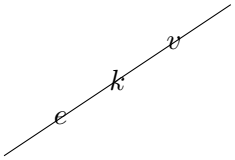
Ez a rajz mutatja, hogy az „aktuális pont” a kör rajzolása után a középpontja, szemben a szakaszokkal és az ívekkel.

[1] Egy kis segítség a rajz megértéséhez:



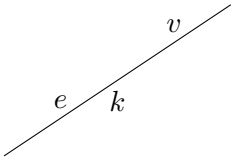
2. Második fázis

2.1. Haladó címkézés Egy *path* szegmenseit (pl. egy szakaszt) is megcímkézhetünk, csak ez egy kicsit bonyolultabb, a dolog természetéből adódóan. Először is, meg kell mondanunk (a *path*-ban a szegmens definíciója végén), hogy az eleje felé (`near start`), a közepén (`midway`), vagy vége felé (`near end`) legyen a címke:



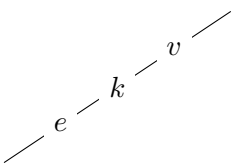

```
\begin{tikzpicture}
 \draw (0,0) -- (3,2)
   node[near start] {$e$}
   node[midway] {$k$}
   node[near end] {$v$};
\end{tikzpicture}
```

Ha nem akarunk a vonal*ra* írni, írhatunk alá vagy fölé:



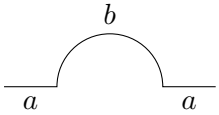

```
\begin{tikzpicture}
 \draw (0,0) -- (3,2)
  node[near start, above] {$e$}
  node[midway, below] {$k$}
  node[near end, above] {$v$};
\end{tikzpicture}
```

Ha akarunk, de azt nem szeretnénk, hogy a vonal eltakarja a feliratot, fehérre festjük a hátteret:



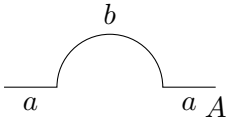

```
\begin{tikzpicture}
 \draw (0,0) -- (3,2)
   node[near start,fill=white] {$e$}
   node[midway,fill=white] {$k$}
   node[near end,fill=white] {$v$};
\end{tikzpicture}
```

Az eddigi példákban több címkével jelöltünk meg egy szegmenst; a következőben több szegmens szerepel, mindegyik egy címkével:




```
\begin{tikzpicture}
  \draw (0,0) -- (1,0) node[midway,below] {$a$}
  arc (180:0:1) node[midway,above] {$b$}
  -- (4,0) node[midway,below] {$a$};
\end{tikzpicture}
```

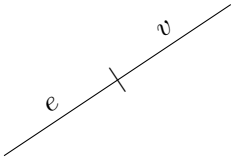
Ti*k*Z alighanem a `near start` és hasonló opcionális argumentumokból találja ki, hogy egy *node* egy szegmenshez, nem pedig a végpontjához tartozik:




```
\begin{tikzpicture}
  \draw (0,0) -- (1,0) node[midway,below] {$a$}
  arc (180:0:1) node[midway,above] {$b$}
  -- (4,0) node[midway,below] {$a$} node[below] {$A$};
\end{tikzpicture}
```

Itt a két *node* definíció az utolsó sorban felcserélhető.

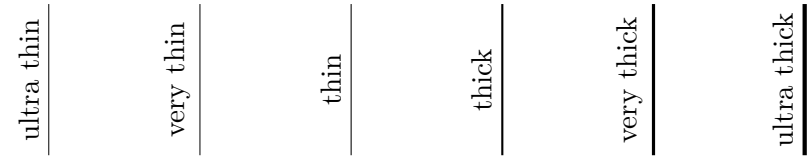
Írhatunk egy szakasszal párhuzamosan is a `sloped` opció segítségével:




```
\begin{tikzpicture}
 \draw (0,0) -- (3,2)
   node[near start, above, sloped] {$e$}
   node[midway,sloped] {$|$}
   node[near end, above, sloped] {$v$};
\end{tikzpicture}
```

2.2. Látvány

Opcionális argumentumok: vonalvastagság `\draw[vastagság]...`, ahol `vastagság` az alábbi 6 valamelyike:

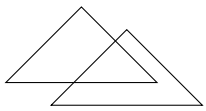


Opcionális argumentumok: vonalfajta `\draw[vonaltípus]...`, ahol `vonaltípus` az alábbi 7 valamelyike:

Opcionális argumentumok: szín `\draw[szín]...`, ahol `szín` az alábbi 9 valamelyike:

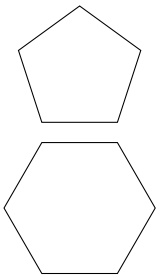
Ezek a színek keverhetőek: `green!30!blue` 30% zöld, 70% féket ( ) jelent; ha a második szín hiányzik, az olyan, mintha fehér állna ott.

2.3. Relatív koordináták „Abszolút koordináták” helyett sokszor kényelmesebb az előző ponttól való eltérést megadni. A következő rajzban például a relatív koordináták használata miatt a háromszöget megadó *path* definícióját csak egy helyen kell megváltoztatni.



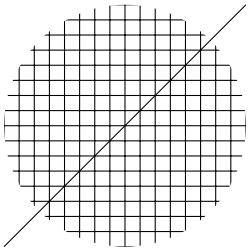
```
\begin{tikzpicture}
 \coordinate (A) at (0,0) ;
 \coordinate (A') at (0.6,-0.3) ;
 \draw (A) -- ++(1,1) -- ++(1,-1) -- cycle;
 \draw (A') -- ++(1,1) -- ++(1,-1) -- cycle;
\end{tikzpicture}
```

De szabályos sokszöget is egyszerűbb relatív (polár)koordinátákkal rajzolni:



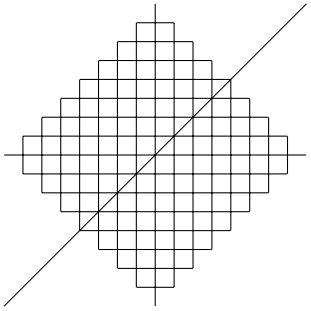
```
\begin{tikzpicture}
 \draw (0,2) -- ++(1,0) -- ++(72:1)
   -- ++(144:1) -- ++(216:1) -- cycle;
 \draw (0,0) -- ++(1,0) -- ++(60:1)
   -- ++(120:1) -- ++(180:1) --
   ++(240:1) -- ++(300:1) -- cycle;
\end{tikzpicture}
```

2.4. Képrészlet kivágása A `\clip`-pel rajzolt zárt *path* hatására a *később* rajzoltakból csak az általa határolt tartomány látszik. Itt a négyzetrácsból egy körlapnyi:



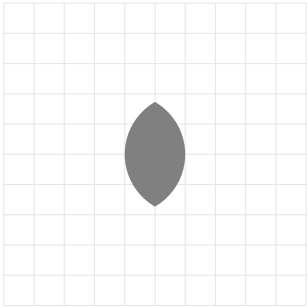
```
\begin{tikzpicture}
 \draw (-2,-2) -- (2,2);
 \clip (0,0) circle (2);
 \draw (-3,-3) grid[step=0.25] (3,3);
\end{tikzpicture}
```

Itt meg egy rombusznyi:



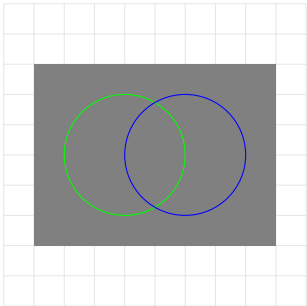
```
\begin{tikzpicture}
 \draw (-2,-2) -- (2,2);
 \clip (-2,0) -- ++(2,2) -- ++(2,-2)
   -- ++(-2,-2) -- cycle;
 \draw (-3,-3) grid[step=0.25] (3,3);
\end{tikzpicture}
```

Ha több kivágás van, a metszetük számít:



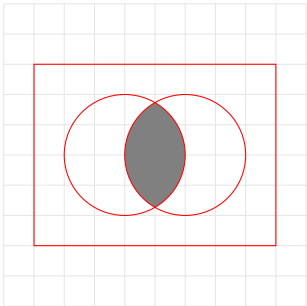
```
\begin{tikzpicture}
 \draw[gray!20,very thin]
   (-5,-5) grid (5,5);
 \clip (-1,0) circle (2);
 \clip ( 1,0) circle (2);
 \fill[gray] (-4,-3) rectangle (4,3);
\end{tikzpicture}
```

Itt vannak az előző rajz szereplői:



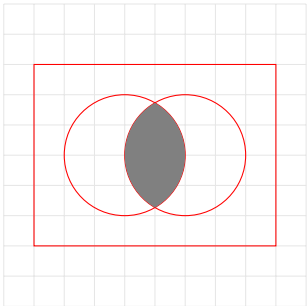
```
\begin{tikzpicture}
    \draw[gray!20,very thin] (-5,-5) grid (5,5);
    \fill[gray] (-4,-3) rectangle (4,3);
    \draw[green] (-1,0) circle (2);
    \draw[blue] ( 1,0) circle (2);
\end{tikzpicture}
```

A `scope` környezetbeli `\clip` hatása csak e környezeten belül évényesül:



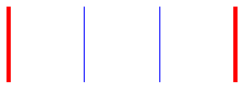
```
\begin{tikzpicture}
 \draw[gray!20,very thin]
   (-5,-5) grid (5,5);
 \begin{scope}
  \clip (-1,0) circle (2);
  \clip ( 1,0) circle (2);
  \fill[gray] (-4,-3) rectangle (4,3);
 \end{scope}
 \draw[red] (-4,-3) rectangle (4,3);
 \draw[red] (-1,0) circle (2);
 \draw[red] (1,0) circle (2);
\end{tikzpicture}
```

Itt `scope` nélkül is elboldogulnánk a sorrend variálásával:



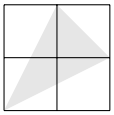
```
\begin{tikzpicture}
 \draw[gray!20,very thin]
   (-5,-5) grid (5,5);
 \draw[red] (-4,-3) rectangle (4,3);
 \draw[red] (-1,0) circle (2);
 \draw[red] (1,0) circle (2);
 \clip (-1,0) circle (2);
 \clip ( 1,0) circle (2);
 \fill[gray] (-4,-3) rectangle (4,3);
\end{tikzpicture}
```

De ez nem mindig olyan síma ügy, mert amit később rajzolunk, az elfedi azt, ami már a papíron volt. Erről a következő szakaszban lesz szó bővebben. Addig is: itt a `scope` egy másik alkalmazása:



```
\begin{tikzpicture}[ultra thick,red]
 \draw (0,0) -- (0,1);
 \begin{scope}[thin,blue]
  \draw (1,0) -- (1,1);
  \draw (2,0) -- (2,1);
 \end{scope}
 \draw (3,0) -- (3,1);
\end{tikzpicture}
```

2.5. Átlátszóság Példa arra, hogy számít a rajzolások sorrendje:



```
\begin{tikzpicture}
 \draw (-1,2) grid ++(2,2) ;
 \fill[gray!20] (-1,2) -- ++(1,2)
   -- ++(1,-1) -- cycle ;
 \fill[gray!20] (-1,-1) -- ++(1,2)
   -- ++(1,-1) -- cycle ;
 \draw (-1,-1) grid ++(2,2) ;
\end{tikzpicture}
```

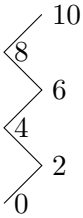
Tehát a világosszürke is eltakarta a feketét. A fehér is eltakarta volna, és ilyet kaptunk volna:  , csak a szimmetria miatt választottuk a szürkét, mert az alsó ábrán a fehér háromszöglap pont annyira lett volna látványos, mint kicsit korábban a fehér alapon fehér vonal volt.

Ha azt szeretnénk, hogy a később rajzolt dolog ne takarja el a korábban rajzoltat, az `opacity` (átlátszatlanság) opcionális paramétert kell használnunk. Ennek értéke egy 0 (teljesen átlátszó) és 1 (teljesen átlátszatlan — ez az alapértelmezés) közötti szám.


```
\begin{tikzpicture}
 \draw (-1,2) grid ++(2,2) ;
 \fill[gray!20,opacity=0.5] (-1,2)
  -- ++(1,2) -- ++(1,-1) -- cycle ;
 \fill[gray!20] (-1,-1)
  -- ++(1,2) -- ++(1,-1) -- cycle ;
 \draw[opacity=0.1] (-1,-1) grid ++(2,2) ;
\end{tikzpicture}
```

3. Haladó

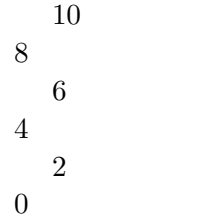
3.1. Ciklusok




```
\begin{tikzpicture}
 \draw (0,0) foreach \x in {0,2,...,10}
  { -- ({mod(\x,4)/4},\x/4) node[right] {\x}};
\end{tikzpicture}
```

`mod(\x,4)/4`-et azért kell bajusz-zárójelek közé tenni, mert szerepel benne (normál) zárójel.

`foreach` itt pont olyan „művelet” (a hivatalos terminológia szerint: *path extension operation*) volt, mint `--` vagy `arc` vagy `rectangle`; `path`-on kívül az ilyeneknek nincs értelmük, de lehet a `\foreach` *parancsot* használni:

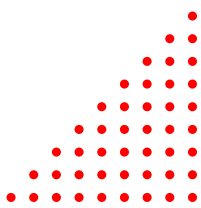



```
\begin{tikzpicture}
 \foreach \x in {0,2,...,10} \draw
  ({mod(\x,4)/4},\x/4) node[right] {\x};
\end{tikzpicture}
```

A `\foreach` parancsot viszont bárhol, még a `tikzpicture` környezeten kívül is használhatjuk. Példa ilyenre: $x_1 = 1$, $x_2 = 2$, $x_3 = 3$, ami a forrásban így fest:

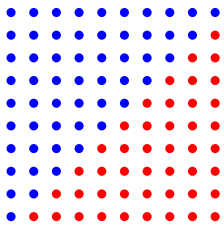
```
\foreach \x in {1,2,3} {$x_\x = \x$, }
```

A ciklusokat egymásba lehet ágyazni:



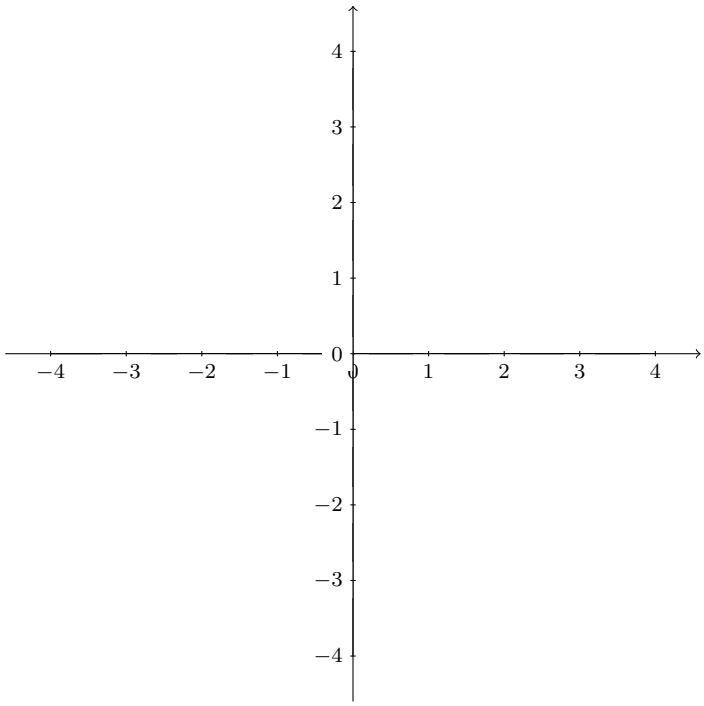
```
\begin{tikzpicture}
 \foreach \x in {1,...,10}{
  \foreach \y in {1,...,10}{
   \ifnum \x>\y
    \fill[red] (\x,\y) circle (0.2);
   \fi
  }
 }
\end{tikzpicture}
```

`\ifnum ... \fi` (teljes pompájában: `\ifnum ... \else ... \fi`) nem Ti*k*Z, hanem LaTeX, sőt, valójában TeX parancs. Példa `\else` használatára:



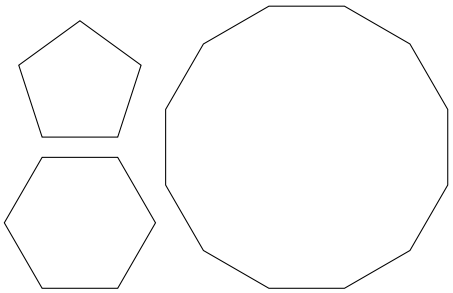
```
\begin{tikzpicture}
 \foreach \x in {1,...,10}{
  \foreach \y in {1,...,10}{
   \ifnum \x>\y
    \fill[red] (\x,\y) circle (0.2);
   \else
    \fill[blue] (\x,\y) circle (0.2);
   \fi
  }
 }
\end{tikzpicture}
```

Koordinátarendszer rajzolásánál nagyon természetes, hogy ciklusokat használjunk:




```
\draw (-4,0) node[below] {$\scriptstyle-4$}
  foreach \i in {-3,-2,...,4} {-- (\i,0)
    node[below,fill=white] {$\scriptstyle\i$}} ;
\foreach \i in {-4,-3,...,4} \draw (\i,-1pt) -- (\i,1pt);
\draw[->] (-4.6,0) -- (4.6,0);
\draw (0,-4) node[left] {$\scriptstyle-4$}
  foreach \i in {-3,-2,...,4} {-- (0,\i)
    node[left,fill=white] {$\scriptstyle\i$}} ;
\foreach \i in {-4,-3,...,4} \draw (-1pt,\i) -- (1pt,\i);
\draw[->] (0,-4.6) -- (0,4.6);
```

Ahogyan szabályos sokszögek rajzolásánál is:

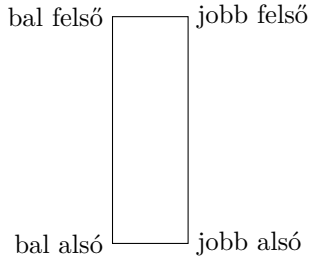


```
\begin{tikzpicture}
  \draw (0,2)
    foreach \i in {0,...,3}
     {-- ++(72*\i:1)} -- cycle;
  \draw (0,0)
    foreach \i in {0,...,4}
     {-- ++(60*\i:1)} -- cycle;
  \draw (3,0)
    foreach \i in {0,...,10}
     {-- ++(30*\i:1)} -- cycle;
\end{tikzpicture}
```

A vonaltípusokat bemutató fenti ábra is `\foreach` segítségével készült:


```
\begin{tikzpicture}
 \foreach \x / \y in
   {loosely dotted/1,densely dotted/2,dotted/3,dashed/4,
     densely dashed/5,loosely dashed/6,double/7}
     { \draw[\x] (12*\y/7,0) --  ++(0,2)
        node[midway,above,sloped] {\x};}
\end{tikzpicture}
```

Itt újdonság, hogy a lista rendezett párokból áll, ahol a pár tagjai és a fölöttük futó változók is /-el vannak elválasztva. Hasonló módon rendezett n-esek listáján is végigmehetünk, itt most például négyeseken:

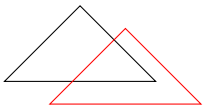



```
\begin{tikzpicture}
\draw  (0,0) rectangle (1,3) ;
 \foreach \x / \y / \felirat / \hol in
  {0/0/bal alsó/left, 1/0/jobb alsó/right,
  1/3/jobb felső/right,0/3/bal felső/left}
  {\draw (\x,\y) node[\hol]{\felirat}; }
\end{tikzpicture}
```

Ugyanez `\foreach` helyett a `foreach` *path extension operation*-nel:

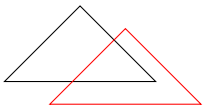
```
\begin{tikzpicture}
  \draw (0,0) foreach \x / \y / \felirat / \hol in
        {0/0/bal alsó/left, 1/0/jobb alsó/right,
          1/3/jobb felső/right,0/3/bal felső/left}
        {-- (\x,\y) node[\hol]{\felirat} } -- cycle;
\end{tikzpicture}
```

3.2. Transzformációk Korábban a relatív koordináták illusztrálására szolgált egy ehhez hasonló ábra:



```
\begin{tikzpicture}
 \draw (0,0) -- ++(1,1)
   -- ++(1,-1) -- cycle;
 \draw[red] (0.6,-0.3) -- ++(1,1)
   -- ++(1,-1) -- cycle;
\end{tikzpicture}
```

Ezt eltolással is lehet:



```
\begin{tikzpicture}
\draw (0,0) -- ++(1,1) -- ++(1,-1) -- cycle ;
\draw[red,shift={(0.6,-0.3)}] (0,0) --
  ++(1,1) -- ++(1,-1) -- cycle;
\end{tikzpicture}
```

És így is lehetett volna:

```
\draw (0,0) -- ++(1,1) -- ++(1,-1) -- cycle
  [shift={(0.6,-0.3)}] (0,0) -- ++(1,1) -- ++(1,-1) -- cycle;
```

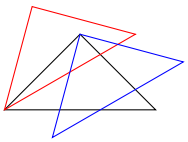
azaz egy path-on belül is tologathatunk (ill. általánosabban: transzformálgathatunk).

Technikai megjegyzés: az eredeti példában a bal alsó sarkok koordinátáinak nevet adtunk, de itt

```
\begin{tikzpicture}
\coordinate (A) at (0,0);
\draw (A) -- ++(1,1) -- ++(1,-1) -- cycle ;
\draw[red,shift={(0.6,-0.3)}] (A) --
  ++(1,1) -- ++(1,-1) -- cycle;
\end{tikzpicture}
```

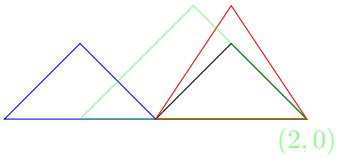
nem működne, mert `(A)` nem a `(0,0)` koordinátát, hanem az általa megjelölt pontot tárolja, vagyis hiába tologatjuk a koordinátarendszert, `(A)` helyben marad. Próbáljuk ki!

Az eltoláshoz hasonlóan van forgatás (`rotate`):



```
\begin{tikzpicture}
\draw (0,0) -- ++(1,1) -- ++(1,-1) -- cycle ;
\draw[red,rotate=30] (0,0) --
  ++(1,1) -- ++(1,-1) -- cycle;
\draw[blue,rotate around={30:(1,1)}]
  (0,0) -- ++(1,1) -- ++(1,-1) -- cycle;
\end{tikzpicture}
```

nagyítás (`scale`), külön-külön a koordinátatengelyek mentén (`xscale`, `yscale`) is:

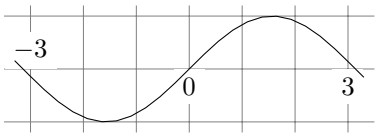



```
\begin{tikzpicture}
 \draw (0,0) -- ++(1,1)
   -- ++(1,-1) -- cycle ;
 \draw[red,yscale=1.5] (0,0) --
   ++(1,1) -- ++(1,-1) -- cycle;
 \draw[blue,xscale=-1] (0,0) --
   ++(1,1) -- ++(1,-1) -- cycle;
 \draw [opacity=0.5,green,
   scale around={1.5:(2,0)}]
   (0,0) -- ++(1,1) -- ++(1,-1)
   node[below] {$(2,0)$} -- cycle;
\end{tikzpicture}
```

3.3. Függvényábrázolás

A warning before we get started: If you are looking for an easy way to create a normal plot of a function with scientific axes, ignore this section and instead look at the pgfplots package...

https://tikz.dev/tikz-plots




```
\begin{tikzpicture}
 \draw[very thin,gray]
  (-3.5,-1.2) grid (3.5,1.2) ;
 \draw (0,0)
  node [below,fill=white] {$0$};
 \draw (3,0)
  node [below,fill=white] {$3$};
 \draw (-3,0)
  node [above,fill=white] {$-3$};
 \draw plot[domain=-3.3:3.3]
  (\x,{sin(deg(\x))});
\end{tikzpicture}
```

A használható függvények:

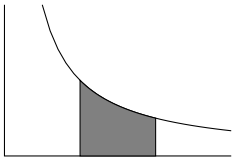
- `+`,`-`,`*`,`/`
- `mod`,`min`,`max`
- `abs`,`exp`,`ln`,`sqrt`
- `round`,`floor`,`ceil`
- `sin`,`cos`,`tan`
- `asin`,`acos`,`atan`
- `pi`,`deg`,`rad`
- `rnd` (0 és 1 közti véletlen szám), `rand` (−1 és 1 közti véletlen szám)
- `==`, `<`, `>` (az eredmény 0 ha hamis, 1 ha igaz)

A trigonometrikus függvények fokban várják az argumentumukat, ezért van a fenti rajzban `sin(deg(\x))`. Ez annyira gyakran felmerülő probléma, hogy ugyanezt így is lehet írni: `sin(\x r)`.

`plot`, mint minden *path extension operation* lehet része egy *path*-nak (persze a fenti példában is az volt, csak ott épp *egyetlen* része):

```
\fill[gray] (1,0) -- (1,1) --
  plot [domain=1:2] (\x,1/\x) -- (2,0) -- cycle;
```

Ez hasznos, pl. egy ilyen ábrában:



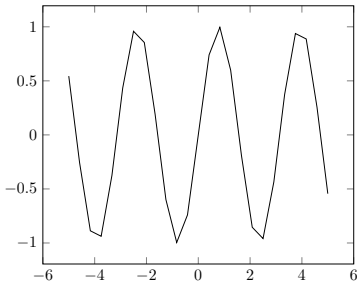
```
\draw (3,0) -- (0,0) -- (0,2);
\draw plot[domain=0.5:3] (\x,1/\x);
\fill[gray,draw=black] (1,0) -- (1,1) --
  plot[domain=1:2] (\x,1/\x) -- (2,0) -- cycle;
```

FÜGGELÉK A. HOGYAN KELL *VALÓJÁBAN* FÜGGVÉNYEKET ÁBRÁZOLNI

Azt már láttuk, hogyan lehet egy függvény grafikonja egy *path* része. Legtöbbször azonban nem ezt akarjuk, hanem valami koordináta-rendszerben akarunk ábrázolni egy függvényt, beleértve, hogy nem mi szeretnénk megrajzolni a koordinátatengelyeket. Ebben az esetben a `pgfplots` csomagot érdemes használni. Ehhez tegyük a következőket a preambulumba:

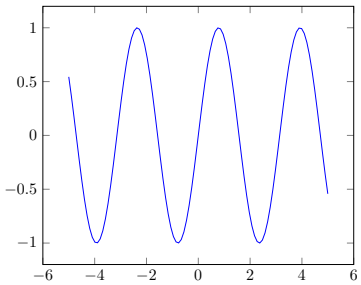
```
\usepackage{pgfplots}
\pgfplotsset{compat = newest}%\pgfplotsset{width=7cm,compat=1.18}
```

Ábrázoljuk a $\sin 2x$ függvényt a $[-5, 5]$ intervallumon!



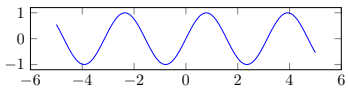

```
\begin{tikzpicture}
 \begin{axis}[domain=-5:5]
 \addplot []
   {sin(deg(2*x))};
 \end{axis}
\end{tikzpicture}
```

Ez így egy kicsit szögletes, de ezen segíthetünk a `samples` paraméterrel, ami azt mondja meg, hogy az intervallumon hány osztóponton nézze meg a függvényértékeket, amiket azután összeköt.



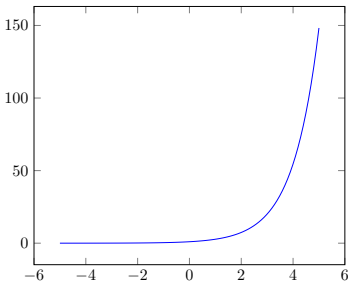

```
\begin{tikzpicture}
 \begin{axis}[]
 \addplot [blue, domain=-5:5, samples=100]
   {sin(deg(2*x))};
 \end{axis}
\end{tikzpicture}
```

Ha zavar, hogy a két tengelyen különbözik az egység (az *aspect ratio* nem 1), ezt tehetjük:



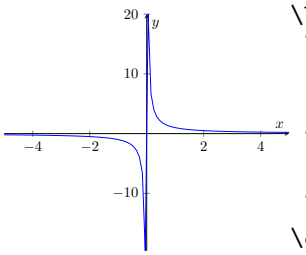

```
\begin{tikzpicture}
 \begin{axis}[axis equal image=true]
 %            ^^^ aspect ratio = 1
 \addplot [blue, domain=-5:5, samples=100]
   {sin(deg(2*x))};
 \end{axis}
\end{tikzpicture}
```

Az automatikus *aspect ratio* jól tud jönni, pl. itt:



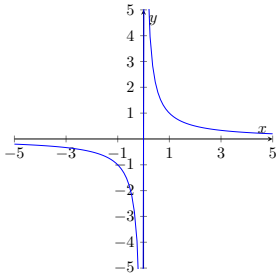

```
\begin{tikzpicture}
 \begin{axis}[]
 \addplot [blue, domain=-5:5, samples=100]
   {exp(x)};
 \end{axis}
\end{tikzpicture}
```

De az sem mindig segít:



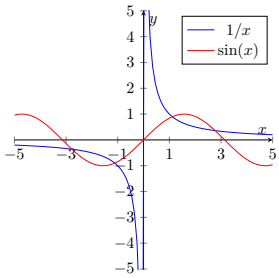

```
\begin{tikzpicture}
 \begin{axis}[axis x line=center,
  axis y line=center,
  xlabel={$x$}, ylabel={$y$}]
 \addplot [blue, domain=-5:5, samples=100]
   {1/x))};
 \end{axis}
\end{tikzpicture}
```

(Mellesleg áttértünk a szokásos módon rajzolt koordinátatengelyekre.) Itt jobban járunk, ha lecsípjük a kilógó értékeket: erre szolgál `ymin` és `ymax`.




```
\begin{tikzpicture}
 \begin{axis}[axis equal image=true,
 ymax=5, ymin=-5, axis x line=center,
 axis y line=center, xtick={-5,-3,...,5},
 ytick={-5,-4,...,5}, xlabel={$x$}, ylabel={$y$}]
 \addplot [blue, domain=-5:5, samples=100]
   {1/x))};
 \end{axis}
\end{tikzpicture}
```

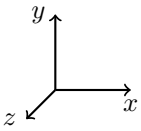
Végül: ábrázolhatunk több függvényt is egyszerre. Ilyenkor érdemes több színt használni, és a `\legend` parancs segítségével megmondani, hogy melyik szín melyik függvényhez tartozik.




```
\begin{tikzpicture}
 \begin{axis}[axis equal image=true,
 ymax=5, ymin=-5, axis x line=center,
 axis y line=center, xtick={-5,-3,...,5},
 ytick={-5,-4,...,5}, xlabel={$x$}, ylabel={$y$}]
 \addplot [blue, domain=-5:5, samples=100]
   {1/x))};
 \addplot[red, domain=-5:5, samples=100]
   {sin(deg(x))};
\legend{$1/x$,$\sin(x)$}
 \end{axis}
\end{tikzpicture}
```

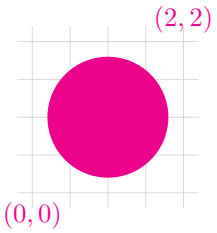
FÜGGELÉK B. KITEKINTÉS A HARMADIK DIMENZIÓBA

Ha térben akarunk rajzolni, elég rendezett hármasokat megadni koordinátákként. Az alapértelmezett koordináta-rendszer így néz ki:



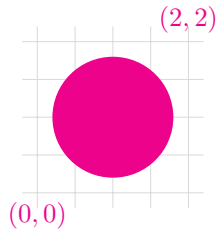

```
\begin{tikzpicture}
  \coordinate (O) at (0,0,0);
  \draw[thick,->] (O) -- (1,0,0) node[below]{$x$};
  \draw[thick,->] (O) -- (0,1,0) node[left]{$y$};
  \draw[thick,->] (O) -- (0,0,1) node[left]{$z$};
\end{tikzpicture}
```

De a kétdimenziós rajzainkat is térbeliekké alakíthatjuk úgy, hogy különböző síkokra rajzoljuk őket. (Ehhez írjuk a preambulumba `\usetikzlibrary{3d}`-t valahova `\usepackage{tikz}` után.) Például ezt a rajzot:



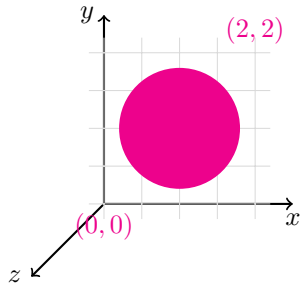

```
\begin{tikzpicture}[magenta]
    \draw[very thin, gray!30] (-.2,-.2)
              grid[step = 0.5] (2.2,2.2);
    \fill (1,1) node[below right] {$O$}
              node{$\bullet$} circle (0.8);
    \draw (0,0) node[below] {$(0,0)$};
    \draw (2,2) node[above] {$(2,2)$};
\end{tikzpicture}
```

nem lehet megkülönböztetni a következőtől, amit az xy-síkba rajzoltunk:




```
\begin{tikzpicture}[magenta,canvas is xy plane at z=0]
    \draw[very thin, gray!30] (-.2,-.2)
              grid[step = 0.5] (2.2,2.2);
    \fill (1,1) node[below right] {$O$}
              node{$\bullet$} circle (0.8);
    \draw (0,0) node[below] {$(0,0)$};
    \draw (2,2) node[above] {$(2,2)$};
\end{tikzpicture}
```

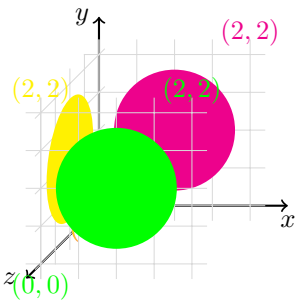
Itt látszik, hogy miről van szó:




```
\begin{tikzpicture}
    \coordinate (O) at (0,0,0);
    \draw[thick,->] (O) -- (2.5,0,0) node[below]{$x$};
    \draw[thick,->] (O) -- (0,2.5,0) node[left]{$y$};
    \draw[thick,->] (O) -- (0,0,2.5) node[left]{$z$};
    \begin{scope}[magenta,canvas is xy plane at z=0]
    \draw[very thin, gray!30] (-.2,-.2)
              grid[step = 0.5] (2.2,2.2);
    \fill (1,1) node[below right] {$O$}
              node{$\bullet$} circle (0.8);
    \draw (0,0) node[below] {$(0,0)$};
    \draw (2,2) node[above] {$(2,2)$};
    \end{scope}
\end{tikzpicture}
```

Azért volt szükség `scope`-ra, mert a tengelyeket még a sík megadása előtt meg kellett rajzolni.

És persze amiért ez az egész érdekes, az az, hogy ugyanezt az ábrát más síkokra is rajzolhatjuk. Itt például rögtön háromra (de a forrásban kihagytam a tengelyek megadását):




```
\begin{tikzpicture}
  \begin{scope}[magenta,canvas is xy plane at z=0]
    \draw[very thin, gray!30] (-.2,-.2)
              grid[step = 0.5] (2.2,2.2);
    \fill (1,1) node[below right] {$O$}
              node{$\bullet$} circle (0.8);
    \draw (0,0) node[below] {$(0,0)$};
    \draw (2,2) node[above] {$(2,2)$};
  \end{scope}
  \begin{scope}[yellow,canvas is yz plane at x=0]
    \draw[very thin, gray!30] (-.2,-.2)
              grid[step = 0.5] (2.2,2.2);
    \fill (1,1) node[below right] {$O$}
              node{$\bullet$} circle (0.8);
    \draw (0,0) node[below] {$(0,0)$};
    \draw (2,2) node[above] {$(2,2)$};
  \end{scope}
  \begin{scope}[green,canvas is xy plane at z=2]
    \draw[very thin, gray!30] (-.2,-.2)
              grid[step = 0.5] (2.2,2.2);
    \fill (1,1) node[below right] {$O$}
              node{$\bullet$} circle (0.8);
    \draw (0,0) node[below] {$(0,0)$};
    \draw (2,2) node[above] {$(2,2)$};
  \end{scope}
\end{tikzpicture}
```

Itt, a színtől eltekintve, a három rajz ugyanaz, csak más-más síkokon jelennek meg.