# LaTeX listák és úszó objektumok

Vadon Viktória

2022/23/I. félév

1 Listák
- Alapok
- Többszintű listák
- Listák testreszabása – `enumitem` csomag
- Egysoros listák
- Listák formázása – alapok
- Számozás folytatása
- Számlálók
- Egysoros listák formázása
- Leíró listák formázása
- Listák formázása – haladó
- Leíró lista példa
- Több lista formázása

# Alapok I

```
\begin{listakörnyezet}
\item listaelem
\item listaelem

új bekezdés, a listaelemen belül
\end{listakörnyezet}
```

- a `listakörnyezet` lehet:
  - `itemize` – felsorolás (számozatlan)
  - `enumerate` – számozás
  - `description` – leíró lista: a szám vagy felsorolásjel helyett egy szócímke
- *csak* a számozott lista elemeit tudjuk címkézni és hivatkozni rájuk

# Alapok II

- * opcionális argumentummal felülírható a listaelem jele, pl. ebben a sorban `\item[*]`
- leíró listánál a szócímke is `\item[szócímke]` módon adható meg
- a listakörnyezet nem lehet üres, kell bele legalább egy `\item` – „missing `\item`” error
- maga az `\item` sora lehet üres

# Többszintű listák I

- többszintű listák is készíthetők, a listák egymásba ágyazásával:

```
\begin{enumerate}
  \item egy
  \begin{itemize}
    \item második szint!
  \end{itemize}
  \item kettő
\end{enumerate}
```

- mint a példán is látszik, különböző típusok is egymásba ágyazhatók

# Többszintű listák II

- azonos típus esetén is különböző jelekkel számoz a különböző szinteken[1]
- legalább egy `\item` kell a következő szint megkezdése előtt
  - logikusan is, miért hagynánk ki egy szintet
  - itt is „missing `\item`” error
- a beágyazás mélysége korlátozott – „too deeply nested” error

[1] diasorban nem, mert a beamer különleges :)

# Listák testreszabása – enumitem

- „gyalogosan” is lehet, de enumitem csomaggal egyszerűbb
- \usepackage{enumitem} – dokumentáció
- formázás testreszabása (számozás, hivatkozás, térköz és elrendezés), korábbi listák folytatása, „egysoros” listák, stb.

# Egysoros listák I

- `inline` csomag opció, `\usepackage[inline]{enumitem}` engedélyezi az `inline`/„egysoros” listákat
- némi térköz kihagyásával, de egy sorban folytatja a következő elemmel*
  • egy • kettő • három
- helyet spórol, ha a felsorolás elemei rövidek – hosszú elemekre ne használjuk, átláthatatlan lesz!
- *-ozott verzióként érhetők el, pl. `\begin{itemize*}`

* közelítőleges demonstráció; beamer-rel nem kompatibilis, ezért a tényleges hatást nem tudom bemutatni

# Egysoros listák II

- formázás: `\begin{itemize*}[opciók]`, opcionális argumentum
  - `afterlabel`: címke és felsoroláselem között, alapért. `afterlabel=~`, avagy törhetetlen szóköz
  - `itemjoin`: két elem között, alapért. `itemjoin=␣`, avagy szóköz
  - utolsó elem előtt `itemjoin*`, pl. `itemjoin*={␣és␣}`
  - számozás, címke stílusa: ld. 5 szakasz
  - például

```
\begin{itemize*}[afterlabel=~,%
itemjoin=\hspace{1em},%
itemjoin*={\hspace{1em}és }]
\item ...
\end{itemize*}
```

# Listák formátuma I

- számozás formátuma `label` kulccsal:

  ```
  \begin{enumerate}[label=formátum]
  ```

  - formátumnak tartalmaznia kell egy számformátumot: `\arabic*` (arab számok), `\alph*`, `\Alph*` (kis- vagy nagybetűk), `\roman*`, `\Roman*` (kis-vagy nagybetűs római számok)
  - tartalmazhat még egyéb formázási karaktereket, pl. pont, zárójel
  - pl. `\begin{enumerate}[label=(\arabic*)]`
- `label*` kulccsal *hozzáfűzés* az előző szint számához, pl.

  ```
  \begin{enumerate}[label=\arabic*.]
  \item ...
    \begin{enumerate}[label*=\alph*.]
  ```

# Listák formátuma II

- hivatkozás formátuma
  - emlékeztető: csak számozott lista (számozott) elemére tudunk hivatkozni(!)
  - alapértelmezésben megegyezik a `label` formátumával
  - újradefiniálható `ref` kulccsal
  - például ha plusz zárójeleket szeretnénk
    `\begin{enumerate}[label=\alph*,ref=(\alph*)]`
  - sajnos `ref*` kulcs nincs!
  - ha hozzávennénk az előző szint sorszámát:

```
\begin{enumerate}[label=\arabic*)]
\item ...
  \begin{enumerate}[label=\alph*),%
    ref=\theenumi.\alph*)]
```

`\theenumi`: első szintű felsorolás számlálójának kiíratása (ld. számlálók, 7 szakasz)

# Listák formátuma III

- felsorolás szimbóluma szintén `label` kulccsal, pl. `\begin{itemize}[label=\textbullet]`
- betűstílus – leíró listánál fontos, de számozásnál is alkalmazható
  - `font` kulccsal, pl. `\begin{description}[font=\scshape]`: kiskapitális betűk
- sorszám/jel/szócímke igazítása
  - alapértelmezés: `align=right` – a címkék jobb széle van egy vonalban, akkor is, ha különböző hosszúak
  - vagy `align=left` – balra
  - `align=parleft` – balra + tördelés bekapcsolása(!), pl. ha leíró listában hosszúak a szócímkék
- függőlegesen sűrített listák:
  - `noitemsep` kapcsoló: elemek közti térközt törli
  - `nosep` kapcsoló: törli a térközt az elemek között, a lista előtt és után

# Számozás folytatása

- `resume` kapcsoló: onnan folytatja a számozást, ahol az előző számozás abbahagyta
  - ha egyedi formátumot is szeretnénk áthozni, `resume*`
- ha egy *korábbi* számozott listát szeretnénk folytatni: szériát hozhatunk létre
  - a korábbi listába `series=név`, ahol `név` tetszőleges
  - folytathatjuk `resume=név`, vagy `resume*=név`-vel – a különbség itt is a formátum másolása a *-ozott verzióban
  - akár több részletben is – széria nem feltétlen csak két felsorolás
- egy lista tetszőleges számról is indítható, pl. 3-ról: `start=3`
- egyéb (kézi) manipuláció a számozással számlálók segítségével, 7 szakasz

# Számlálók

- számozott lista minden egyes szintjéhez tartozik egy ún. counter/számláló: belső változó, ami számolja, hanyadik elemnél járunk
- nevük `enum` + szint kisbetűs római számmal: `enumi`, `enumii`, `enumiii`, `enumiv`
- kiírása: `\theenumi`, stb.
- ha meg kell változtatni az értékét...
  - pl. ha volt egy `\item[ ]`, az nem növeli a számláló értékét
  - léptetés 1-gyel: `\stepcounter{enumi}`
  - tetszőleges új érték megadása: `\setcounter{enumi}{3}`, új értéke 3
  - jó tudni: `\item` először megnöveli a számláló értékét, és utána írja ki

# Egysoros listák formázása

- ...

# Leíró listák formázása I

- leíró lista beépített „stílusai” – méretezési (ld. 10 szakasz) és igazítási opciók kombinációja (szükség szerint méretezési opciókkal együtt használandók)
  - `style=standard`: az alap documentclass-ok leíró lista stílusa
    - a szócímke dobozba van foglalva – nincs automatikus tördelés, ha nem fér ki egy sorba!
    - szöveg a szócímkével egy sorban folytatódik
    - üres szócímke esetén a szöveg balra csúszik a címke helyére – mintha függő behúzás lenne
  - `standard` az alapértelmezés, a többinél az ehhez képesti különbséget emeljük ki
  - `style=unboxed`
    - a szócímke nincs dobozba zárva – automatikus tördelés
  - `style=sameline`
    - szócímke nincs dobozban
    - üres címke esetén a szöveg nem csúszik ki balra

# Leíró listák formázása II

- `style=nextline`
  - ha a címke nem fér a margóra, a szöveg a következő sorban folytatódik
  - üres címke esetén a szöveg nem csúszik ki balra
- `style=multiline`
  - a címke egy margó szélességű dobozba kerül, de akár több sorba
  - kifejezetten ajánlott az átméretezés (ld. 10 és 11 szakaszok) – elválasztással együtt is nagyon kicsi a hely

# Listák formázása – haladó I

## Méretezés: behúzás, térközök, stb.

- ábra a dokumentáció 3. oldalán

Lista függőleges térközei

- `topsep`: térköz a lista előtt és után
- `partopsep`: ha a listát üres sor előzi meg (azaz új paragrafusba kerül), ennyi *extra* térköz kerül elé és utána is
- `parsep`: egy listaelemen belüli bekezdések közti térköz
- `itemsep`: listaelemek közé a `parsep`-en felül ennyi extra térköz kerül

Listák vízszintes elrendezése

- `rightmargin`: (teljes lista) jobb margója
  - *extra* margó a *környező szöveghez* képest
  - azaz nem kell beleszámolni az oldal margóját
  - többszintű listánál halmozódik

# Listák formázása – haladó II

## Méretezés: behúzás, térközök, stb.

- `leftmargin`: listaelem szövegtörzsének bal margója – szintén a *környező szöveghez* képest
- `itemindent`: a listaelem első sora ennyivel beljebb kerül a `leftmargin`-hoz képest
  - `listparindent`: az `itemindent` megfelelője a listaelem későbbi bekezdéseiben
- `labelindent`: a címke (sorszám/jel/szócímke) behúzása, a *környező szöveghez* képest
- `labelwidth`: címke szélessége
- `labelsep`: címke távolsága a listaelem szövegétől

# Listák formázása – haladó III

## Méretezés: behúzás, térközök, stb.

- nehézség: túlhatározott rendszer!

  listaelem első sorának behúzása
  = `leftmargin + itemindent`
  = `labelindent + labelwidth + labelsep`

- alapértelmezésben a `labelindent`-et számolja a többiből
  - ez az új mennyiség az `enumitem` csomagban, a többi hossz létezik alap LaTeX-ben is
  - számolandó mennyiség kijelölése: `!` vagy `*` értékként, pl. `leftmargin=*`
    - különbség: `*` esetén `labelwidth`-et (címkeszélességet) (is) automatikusan határozza meg a *címkékből*

# Listák formázása – haladó IV

## Méretezés: behúzás, térközök, stb.

- `labelwidth` számolása a címkéből
  - `enumerate`-re címke stílusa elég: ismert a legszélesebb betű/római szám
  - ha ettől keskenyebb is elég, pl. nem megyünk el `viii`-ig, lehet pl. `widest=iv`
  - `description` listához `widest=szöveg`, `szöveg` szélességét veszi a címke szélességének
  - nem feltétlen a legszélesebb címke kell – többsoros címkénél érdemesebb az előforduló leghosszabb szót vagy szótagot

# Példa leíró listára többsoros címkével

```
\begin{description}[align=parleft,%
leftmargin=*,widest={hosszabb}]
\item[címke] \hulipsum[1]
\item \hulipsum[2]
\item[ez egy hosszabb címke!] \hulipsum[3]
\end{description}
```

- `align=parleft`: többsoros címke
- `leftmargin=*`: `leftmargin` azaz bal margó számítása a többi méretből *és* `labelwidth` azaz címke(doboz) szélessége *automatikus*
  - `widest={hosszabb}`: a `hosszabb` szó szélességéhez igazít

# Több lista formázása I

- ha `resume*`-gal folytatunk egy korábbi listát, annak öröklődik a stílusa
- (helyileg felülírható) *alapbeállítások* megadása – preambulumban:

  `\setlist[típusok,szintek]{beállítások}`

  - típusok,szintek
    - minden lehetséges típus-szint kombinációra érvényes, pl.`enumerate,itemize,1,2`: `enumerate` 1-2 szintjeire és `itemize` 1-2 szintjeire vonatkozik
    - kihagyjuk a típust és/vagy szintet: mindre vonatkozik, pl. `enumerate`: `enumerate` összes szintjére, `1`: minden első szintű listára
  - beállítások: szokásos `kapcsoló,kulcs=érték` lista formájában
    - pl. `\setlist[description]{nosep,style=nextline}`

# Több lista formázása II

- „új” listatípusok definiálása – létező listatípust klónoz, így különböző alapbeállítást tudunk neki adni
- pl. (minden szinten) kisbetűs számozott lista

```
\newlist{enuma}{enumerate}{5}
\setlist[enuma]{label=\alph*)}
```

  - `enuma`: új listatípus neve, tetszőleges
  - `enumerate`: melyik listáról másoljuk (`inline`, avagy egysoros verziók is használhatók!)
  - 5: max beágyazási mélység
    - ne állítsuk fölöslegesen nagyon nagyra, mert létrehozza minden szinthez a számlálókat, példáunkban `enumai-enumav` számlálókat, ami foglalja a memóriát
  - használat:

```
\begin{enuma}
\item ...
\end{enuma}
```

# Több lista formázása III

- pl. jogszabályokhoz lista, § jellel, és egymás után fűzött arab számokkal

```
\newlist{jogi}[enumerate]{3}
\setlist[jogi]{font=,label*=.\arabic*}
\setlist[jogi,1]{label=\S\arabic*}
```

  jogi lista *első* szintjére mi lesz érvényes?
  - ha nincs ellentmondás, mindkét `\setlist` parancsai vonatkoznak rá
  - ellentmondásos kulcsok esetén a specifikusabb felülírja
  - példánkban `font=\ttfamily` és `label=\S\arabic*` lesz
- `\renewlist`: hasonló szintaxissal, létező listatípus újradefiniálása
  - miért, ha `\setlist` létezik? beágyazás mélységét is meg tudjuk változtatni!

2. Úsztatás bevezető
   - Mi az az úsztatás?
   - Miért használjunk float-okat?

# Úsztatás

- mi az, hogy úsztatás?
  - pl. képet szeretnénk beszúrni, de fordításkor már nem fér ki az oldal aljára
  - opció 1: kép és utána jövő szöveg is következő oldalra kerül – sok üres hely maradhat az oldal alján!
  - opció 2: képlékenyen kezeljük a kép helyét a környező szöveghez képest, pl. az oldal alját kitöltjük a kép utáni szöveggel, a kép pedig a következő oldal tetejére kerül
- opció 2 = úsztatás: kivesszük a képet a szöveg folyamából, hogy optimális(abb) helyre kerülhessen
- *mit* úsztatunk? nagyobb objektumokat, amiket nem lehet vagy nem akarunk tördelni, pl. kép, táblázat, kódrészlet
- úszó objektum = angol nevén float

# Mit tudnak még a float-ok?

- az úszó környezeten belül elhelyezhető egy felirat + sorszám
  - ha van sorszám, akkor lehet `\label`-t elhelyezni és hivatkozni is rá!
- LaTeX nyilvántartja őket, automatikusan listázhatók az úszó objektumok, tartalomjegyzékhez hasonlóan
- számozás és listázás *típusonként/kategóriánként*
  - alapértelmezésben ábrák és táblázatok
  - `float` csomaggal újak is definiálhatók, vagy bizonyos csomagok automatikusan definiálnak típusokat (pl. programkód, később tanuljuk)
- pontos használatot az adott típusoknál részletezzük

3 Képek

- Képek beszúrása – úsztatás nélkül
- Képek úsztatása
- Float elhelyezése
- Képfelirat
- Részábrák

# Képek beszúrása I

- `\usepackage{graphicx}` kell hozzá!
- ismert formátumok: .jpg, .png, .pdf
- `\includegraphics[opciók,méretezés]%`
  `{relatív/elérési/út/képnév}`
- *nem* úsztat, „szöveggel egy sorba” szúr be – akár önmagában is használható, ha úgy látjuk jónak
- `relatív/elérési/út/fájlnév`
  - ha `TEX/3ora/3ora.tex` fájlban dolgozunk
  - és `TEX/forras/kep.png`-t szeretnénk beilleszteni
  - relatív elérés: 1 mappát föl (`..`), be a `forras` mappába: `\includegraphics{../forras/kep}`
  - fájl kiterjesztés (sokszor) elhagyható

# Képek beszúrása II

- `méretezés` – opcionális
  - `scale=0.5`: kép eredeti méretének 0.5-szöröse
    - lehet nagyítani, ill. negatív scale-lel 180°-is forgatás
    - nehézkes lehet használni, ha az eredeti méret nem ismert
  - `width=5cm` vagy `width=0.8\linewidth`: szélesség beállítása abszolút hosszra (cm, mm, pt), vagy sorszélesség (tört)részére
    - hivatkozáshoz hasznos lehet még `\columnwidth`: oszlop szélessége
  - hasonlóképp `height=...`
    - ha *forgatjuk*, használjuk helyette a `totalheight` kulcsot
  - ha adunk meg `width`-et *és* `height`-et is, nyújtja a képet

# Képek beszúrása III

- `opciók` – opcionális
  - `keepaspectratio`: méretarány megtartása – nem nyújt, csak nagyít/kicsinyít
    - ha `width`-et és `height`-et is megadunk, *maximumnak* veszi őket; az egyiket veszi csak fel pontosan, hogy ne kelljen nyújtani a képet
  - forgatás szöge `angle=90`, +90° (óramutatóval ellentétesen)
  - forgatás középpontja `origin=..`, max két betűs kombináció:
    - `l`, `r`: bal vagy jobb él
    - `t`, `b`: felső vagy alsó él, `B` alapvonal
    - `c`: közép – önmagában használva a kép közepe függőlegesen *és* vízszintesen is; valamivel kombinációban az adott él közepe
    - alapértelmezés `origin=lB`
- keretezés, egyéb transzformációk (pl. tükrözés, nyújtás) első óráról: `\framebox{\includegraphics{kep}}`, stb.

# Képek úsztatása I

```
\begin{figure}[bt]
\centering
\caption{Felirat/képaláírás}
\label{fig:kepek}
\includegraphics[width=.4\linewidth]{kep1}
\hspace{1em} % vízszintes helykihagyás
\includegraphics[width=.4\linewidth]{kep2}
\end{figure}
```

- a `figure` környezet az úsztatásról gondoskodik
- a teljes oldalszélességet kitölti, képmérettől függetlenül – csomagokkal érhető el, hogy szöveg kerülhessen mellé
- a `figure` nem foglalkozik a tartalmával – kerülhet bele sima szöveg, vagy akár több `\includegraphics` parancs is!

# Képek úsztatása II

- zölddel: működnek benne az igazítás és helykitöltés parancsai
- kékkel: `figure` által biztosított extra opciók/parancsok
- `[bt]`: opcionális argumentumban *elhelyezési opciók*, ld. 3 szakasz
- `\caption`: kép felirata, *számozása*
  - kereszthivatkozásnak ez a horgonya(!)
  - listába is csak akkor kerül, ha van `\caption`-je
  - opcionális argumentum: `\caption[Rövid]{Kép hosszú felirata}`, a képpel a hosszú képfelirat jelenik meg, listázásra a rövid név kerül
  - bővebben: 4 szakasz
- ábrák listázása: `\listoffigures`, tartalomjegyzékhez hasonlóan működik
  - `babel`-lel magyarul írja ki, kézzel átnevezhető: `\renewcommand{\listfigurename}{Ábralista új neve}`

# Float elhelyezése I

- `figure` opcionális argumentumával befolyásolható a float elhelyezése, pl.

  `\begin{figure}[bt]`

  - akkor teljesíti, ha „elég jól néz ki” – a LaTeX tipográfiai elvárásai szerint
  - betűk tetszőleges kombinációja használható (sorrend *nem* számít, az összeset kipróbálja az algoritmus):
  - `h`: „itt” – minél közelebb ahhoz, ahol a szövegben előfordult, akár az oldal közepén is
  - `t`: oldal tetején
  - `b`: oldal alján
  - `p`: külön float oldalon
  - `!`: „erősebben kérjük”
  - alapértelmezés: `tbp`

# Float elhelyezése II

- kéthasábos dokumentum esetén `figure*` változat lap tetejére kerül és *mindkét* hasábot elfoglalja
- `\clearpage` parancs nem csak oldaltörést tesz, kényszeríti hogy a felhalmozott (az eddig beolvasott, de még ki nem írt) float-okat kiírja
- csomagokkal még több kontroll:
- `float` csomaggal elhelyezéshez `H` betű: mindenképpen oda kerül, ahol a kódban van – avagy nem úszik, de elérhető `\caption`, `\label`, listázás
- `placeins` csomaggal új `\FloatBarrier` parancs[2]: ezen nem tud „átúszni” a float – ami ez előtt van a kódban, ez előtt lesz a dokumentumban is
  `\usepackage[section]{placeins}`: automatikusan minden `\section` címsorral együtt `\FloatBarrier`

[2] figyeljünk a nagybetűkre!

# Képfelirat

`\caption{Kép címe}`

- megformázza a képfeliratot + sorszámoz
- a `\caption` *után* (vagy bele) kerüljön a `\label`! – hogy jól működjön a kereszthivatkozás
- `\caption` és `\includegraphics` sorrendjétől függően a kép alá vagy fölé kerül a felirat
  - ízlés kérdése, de egy dokumentumon belül legyen konzisztens
- csak egyetlen `\caption` legyen egy `figure` környezeten belül!
  - gyakorlatilag lehet több is, de akkor több sorszámot kap
  - ha külön szeretnénk feliratozni (és/vagy hivatkozni) a képeket egy `figure`-ön belül, használjunk *részábrákat*, ld. 5 szakasz

# Részábrák I

- `subcaption` csomag, *részábrák*
- cél: egy `figure`-ön belül több képnek külön felirat
- részábra: `figure`-ön *belül* használható a `subfigure` környezet
- használat: `\begin{subfigure}{5cm}` – kötelező argumentum a szélessége, mint a `minipage`-nél
  - tulajdonképpen `minipage`-re épül. örökli az igazítási opciókat, pl. `\begin{subfigure}[c]{5cm}`; lehet `c,t,b`: a `subfigure` közepe, legfelső vagy legalsó *alapvonala* illeszkedik a vele egy sorban lévő tartalomhoz; pluszban `T,B`, `subfigure` teteje vagy alja igazodik
- `subfigure`-ön belül (újra) használható a `\caption`
  - sorszám helyett betűzi a részábrákat
  - másképp formázza a feliratot
  - részábra kaphat külön `\label`-t!

# Részábrák II

- hivatkozhatunk rá `\ref` vagy `\subref` paranccsal – `\subref` csak a `subfigure` betűjele, `\ref`-ben a tartalmazó `figure` száma is benne van

- egy `figure`-ön belül több `subfigure` is lehet
- sőt, `subfigure`-ök egymásba is ágyazhatók
- `subfigure`-ök elrendezése:
  - alapértelmezésben soronként tölti fel, amennyi kifér az adott szélességével
  - sortörés, helykitöltés parancsai, pl. `\vspace`, `\hspace`, `\hfill` használhatók
  - `subfigure`-ök egymásba ágyazásával oszlopok pl.
  - a beágyazás mélysége nem számít, a `\caption`-jeiket folytonosan betűzi és azonosan formázza – azaz mind azonos szintűnek számít

# Részábrák: egy példa

```
\begin{figure}[hb]
\centering
\caption{ábra felirata}
  \begin{subfigure}{5cm}
  \centering
  \caption{első részábra}
  \includegraphics{kep1}
  \end{subfigure}
\\
  \begin{subfigure}{5cm}
  \centering
  \caption{második részábra}
  \includegraphics{kep2}
  \end{subfigure}
\end{figure}
```

4. Táblázatok
   - Táblázatok – úsztatás nélkül
   - Táblázatok úsztatása
   - Táblázatok testreszabása – `array` csomag
   - Cellák egyesítése

# Táblázatok – úsztatás nélkül I

- tabular környezet, például:

```
\begin{tabular}{r|ll}
objektum & beillesztés & úsztatás \\ \hline
kép & \verb|\includegraphics| & figure \\
táblázat & tabular & table \\
\end{tabular}
```

| objektum | beillesztés | úsztatás |
|---|---|---|
| kép | \includegraphics | figure |
| táblázat | tabular | table |

- &: soron belül cellák elválasztása
- \\: sor lezárása – használható az opcionális argumentum extra térközhöz az adott sor után, pl. \\[.5ex]
- \hline: vízszintes rácsvonal – elhagyható, duplázható

# Táblázatok – úsztatás nélkül II

- részleges vízszintes rácsvonal: `\cline{1-2}`, 1-2 oszlopokban
  - mindenképp tartomány kell! csak az 1. oszlopra: `\cline{1-1}`
  - nem dupláz – nincs error, de nem rajzolja duplán
  - több is használható, ha nem összefüggő vonal kell, pl. `\cline{1-1} \cline{3-3}`
- kötelező argumentum: pl. `{r|ll}` oszlopok és igazításuk + függőleges rácsvonalak
  - `l`: balra zárt oszlop
  - `r`: jobbra zárt oszlop
  - `c`: középre zárt oszlop
  - `|`: függőleges rácsvonal – elhagyható, duplázható
  - `l, r, c` oszlopok méretezése: automatikus, a legszélesebb cellához – nem tördel, kilóghat az oldalról a táblázat!
  - `p{3cm}`: 3cm széles paragrafus oszlop – ebben tördel is (`\parbox` a cella); cellán belül sortörés `\newline` paranccsal

# Táblázatok – úsztatás nélkül III

- kicsit haladóbb tudnivaló az oszlopokról...
  - oszlopok közé `@{kód}`: oszlopelválasztó helyi felülírása
    - pl. `@{\hspace{1em}}`, nagyobb térköz, választóvonal nélkül
    - | helyett használjuk a `\vline` parancsot! különben „szaggatott” lesz a vonal. pl. `@{\hspace{1em}\vline\hspace{1em}}`, függőleges vonal, de nagyobb térköz
  - oszloptávolság átállítása: `\setlength{\tablcolsep}{6pt}` (alapértelmezés)
  - oszlopelválasztó &-k
    - ha nincs köztük semmi: üres cella
    - ideális esetben oszlopok-1 darab (pl. 3 oszlop közt 2 elválasztó)
    - ha túl kevés, nincs error, üres cellák a sor végén
    - ha túl sok, error
  - függőleges rácsvonal ki/vagy bekapcsolása csak adott sorban/cellában: ld. cellák egyesítése, 4 szakasz

# Táblázatok úsztatása

- `table` környezet – `figure`-höz hasonlóképp, `table`-be ágyazzuk be a `tabular`-t
- `table`-ön belül él a `\caption` parancs
  - szokásos módon felirat + számozás
  - kerülhet a táblázat fölé vagy alá (ha korábban ill. ha később van a kódban mint a `tabular`)
  - szükséges a `\label` elhelyezéséhez és listázáshoz
  - opcionális argumentumban megadható rövidített verzió, az kerül a listába
- listázás `\listoftables` paranccsal
  - `babel`-lel lefordítja a lista nevét
  - kézzel átnevezhető:

    ```
    \renewcommand{\listtablename}%
    {Táblázatlista új neve}
    ```
- `subcaption` csomaggal létezik `subtable` is, `subfigure`-höz hasonlóan

# Táblázatok testreszabása

- array csomaggal
- sormagasság növelése *általánosan* pl. 2pt-tal: `\setlength{\extrarowheight}{2pt}`
- új oszloptípusok
  - `m{szélesség}` függőlegesen középre igazított bekezdés
  - `b{szélesség}` függőlegesen alulra igazított bekezdés
- kód beszúrása oszlop elé: pl. vastagbetűs (középre zárt) oszlop `>{\bfseries}c`
  - mögé: pl. % jel beillesztése (jobbra zárt számok után) `r<{\%}`
  - saját oszloptípusként is definiálható: `\newcolumntype{b}{>{\bfseries}c}`
- `!{}`: `@{}`-hez hasonló, | lecserélhető egyéb szimbólumra, automatikus térközzel

# Cellák egyesítése I

- többoszlopos cella (egy sorban több cella egyesítése): alap LaTeX \multicolumn parancsa

```
\begin{tabular}{lll}
\multicolumn{2}{|c|}{1-2} & 3 \\
1 & 2 & 3 \\
\end{tabular}
```

| 1-2 | 3
1 2 3

- 2: hány oszlopot foglal el – kevesebb oszlopelválasztót is kell beírni!
- |c|: az új, összevont cella igazítása, ill. rácsvonalak mellette – felülírja az alapértelmezett függőleges rácsozást
- 1-2: az összevont cella tartalma

# Cellák egyesítése II

- többsoros cella (egy oszlopban több cella egyesítése): `multirow` csomag `\multirow` parancsa
- más a szintaxis!

```
\begin{tabular}{lll}
1 & \multirow{2}{2em}{1-2} & 1 \\
2 &  & 2 \\
\end{tabular}
```

| | | |
|---|---|---|
| 1 | 1-2 | 1 |
| 2 | | 2 |

- 2: hány sort tölt ki – minden sorban hagyjuk ki neki az üres cellát, és legyen meg az összes oszlopelválasztó!
- 2em: a cella szélessége – * használható automatikus méretezéshez
- 1-2: az összevont cella tartalma

# Cellák egyesítése III

- egy cella több sorban *és* több oszlopban – lehetséges
- `\multicolumn` és `\multirow` egymásba ágyazásával
- csak akkor működik, ha a `\multicolumn` van *kívül*
- de a `\multirow` szerint kell kihagyni az üres cellákat(!)
- rácsvonalak miatt szükség lehet a kihagyott sor(ok)ba üres `\multicolumn`-ö(ke)t tenni

5 Kiterjesztések

- Szöveg körbefuttatása
- Float-ok testreszabása és saját float-ok – `float` csomag
- Felirat a float mellett – `sidecap`
- Táblázatok színezése
  - Oszlopok
  - Sorok
  - Cellák és rácsvonalak
- Törhető táblázat – `longtable`

# Szöveg körbefuttatása – wrapfig I

- keskeny ábra ne foglalja el a teljes oldalszélességet, helyette mellé szöveg rövidebb sorokkal
- `wrapfig` csomag `wrapfigure` és `wraptable` környezetei
- `\begin{wrapfigure}[12]{r}[\marginparwidth]{5cm}`
  `\caption{felirat}`
  `\includegraphics[width=4.5cm]{kep}`
  `\end{wrapfigure}`
- `{r}`: kötelező, elhelyezés
  - `l,L` bal vagy `r,R` jobb (egyoldalas dokumentum)
  - `i,I` belső vagy `o,O` külső oldal (kétoldalas dokumentum)
  - kisbetű: „ide”, nagybetű: úszhat (azaz automatikusan áthelyezhető, ha szükséges)
  - sokszor érdemes kisbetűsen hagyni és kézzel áthelyezni a kódban

# Szöveg körbefuttatása – wrapfig II

- {5cm}: kötelező, ábra szélessége
  - ha 0pt-t adunk meg, automatikusan méretez
  - de ha van \caption, érdemes kézzel méretezni
- [12]: opcionális, hány sor magas lesz a kép
  - érdemes próbálkozni vele, ha túl sok whitespace-t hagy magától
  - vagy negatív \vspace-eket rakni
- [\marginparwidth]: opcionális, túllógás a margóra (alapértelmezés 0pt)
- alias/alternatív használat, szöveg körbefuttatása bármilyen float típusra, pl. table esetén: \begin{wrapfloat}{table}{r}{5cm}
- float csomaggal együtt használva (ld. 2 szakasz) automatikusan definiál wrapmyfloat verziót minden saját myfloat típushoz

# Float-ok testreszabása és saját float-ok – float csomag I

- float csomag: létező float-ok testreszabása és új float típusok definiálása
- új float definiálása

  \newfloat{program}{hbt}{lop}[section]
- {program}: float típus neve, tetszőleges
- {hbt}: alapértelmezett elhelyezés
- {lop}: listázáshoz külső fájlt használ (mint a tartalomjegyzék .toc fájlja), ennek a kiterjesztése, tetszőleges (ha nem ütközik létezővel)
- [section]: opcionális, ha megadjuk, section-ön belüli számozást kapcsolja be (lehetne chapter stb.)

# Float-ok testreszabása és saját float-ok – float csomag II

- float megjelenítendő neve (a \caption-ben):

  \floatname{program}{Programkód}

  {program}: float típus, {Programkód}: megjelenítendő név
- listázás:

  \listof{program}{Programkódok listája}

  {program}: listázandó típus, {Programkódok listája}: címsor hozzá
- float típus alapértelmezett elhelyezése:

  \floatplacement{figure}{hbt}

  {figure}: típus, {hbt}: új alapértelmezés

# Float-ok testreszabása és saját float-ok – float csomag III

- float stílusok: \floatstyle
  - pl. \floatstyle{plain}: eredeti stílus, de \caption mindig alul
  - plaintop: hasonló, de \caption mindig felül
  - boxed: keretezett float, \caption alatta
  - ruled: \caption felül, vízszintes vonalak körülötte és a float alatt
  - az *utána definiált* float-okra érvényes(!)
  - alkalmazás beépített típusokra: pl. \restylefloat{figure}
- szöveg körbefuttatása
  - együtt tud működni a wrapfig csomaggal (ld. 1 szakasz)
  - a float csomagot töltsük be előbb, és a wrapfig csomagot utána
  - az új float definíciók *mindkét* csomag betöltése után jöjjenek!
  - akkor a program típus mellett automatikusan definiálja a wrapprogram típust is

# Felirat a float mellett – sidecap

- sidecap csomag SCfigure, SCtable környezetei
- csillagozott verziók: két hasábot kitöltő
- picit más szintaxis:

  \begin{SCfigure}[1.0][hbt]

  ...
- [1.0]: \caption és float szélességének aránya – alapértelmezésben 1.0
- [hbt]: elhelyezés – szokásos módon, de csak akkor használható, ha az előző opcionális argumentum (is) szerepel!
- \caption elhelyezése: csomag opciók
  - pl. \usepackage[outercaption]{sidecap} (alapértelmezett): külső oldalon
  - innercaption: belső oldalon
  - rightcaption: *mindig* jobb oldalon
  - leftcaption: *mindig* bal oldalon

# Táblázatok színezése

- `\usepackage{colortbl}`, vagy `\usepackage[table]{xcolor}`
- szükséges hozzá `xcolor` és `array` csomag is
- használhatók az `xcolor` színei
  - legegyszerűbb név szerint
  - gyors színkeverés: `\textcolor{red!60!black}{60\% piros, 40\% fekete}` 60% piros, 40% fekete
  - ha a második szín hiányzik, fehéret kever bele: `\colorbox{red!20}{20\%-os piros}` 20%-os piros
  - szín definiálása RGB modellben: pl. `\definecolor{red}{RGB}{255,0,0}`, piros, zöld, kék ebben a sorrendben, 0-255 közti értékek
  - szín definiálása HTML modellben (ugyanez hexadecimálisan), pl. `\definecolor{red}{HTML}{FF0000}`

# Oszlopok

- oszlop háttérszíne: `\columncolor{}` – az oszlopspecifikációval kell megadni, oszlop elé illesztendő kódként (ld. `array` csomag):

```
\begin{tabular}{>{\columncolor{cyan}}l%
>{\columncolor{red}\color{white}}l}
világoskék & piros \\
a &  \\
b \\
\end{tabular}
```

| | |
|---|---|
| világoskék | piros |
| a | |
| b | |

  - átláthatóság kedvéért érdemes új oszloptípust definiálni rá!
  - figyeljük meg, mi történik az üresen hagyott és a kihagyott cellával!

# Sorok I

- sor háttérszíne: `\rowcolor{}` – sor elejére:

```
\begin{tabular}{ll}
\rowcolor{cyan}
világoskék & a \\
\rowcolor{red}\color{white}
piros & b \\
\end{tabular}
```

| | |
|---|---|
| világoskék | a |
| piros | b |

  - vegyük észre: szöveg színe sajnos nem alkalmazható egész sorra
    (majd programozós megoldást nézünk rá)
- sor színe felülírja az oszlop színét(!)

# Sorok II

- sorok váltakozó színezése: `\rowcolors` parancs – a táblázat *előtt* használandó!

```
\rowcolors[\hline]{2}{red!20}{blue!20}
\begin{tabular}{ll}
szám & betű \\
a & 1 \\
b & 2 \\
c & 3 \\
\end{tabular}
```

| szám | betű |
|---|---|
| a | 1 |
| b | 2 |
| c | 3 |

# Sorok III

- [\hline]: opcionális argumentumba tetszőleges, minden sorban végrehajtandó parancsok kerülnek
- {2}: hanyadik sortól érvényes a váltakozó színezés
- \rowcolors* verzióval a parancsok is csak a fenti sortól érvényesek
- {red!20}: páratlan sorok háttérszíne
- {blue!20}: páros sorok háttérszíne
- üresen is hagyhatók, akkor fehér háttér

# Cellák és rácsvonalak I

- egyedi cella háttérszíne: `\cellcolor{}` – bárhová a cellába elhelyezve működik, felülírja az oszlop- és sorspecifikációt
- függőleges rácsvonal helyi színezése pl. `r!{\color{red}\vline}l`
- rácsvonalak „alapértelmezett” színe: `\arrayrulecolor{}`
  - `\arrayrulecolor` használható táblázat előtt vagy közben
  - a *parancs után definiált* rácsvonalakra érvényes – vízszintes rácsvonalakra jól működik
  - a függőleges rácsvonalakat a táblázat elején definiáljuk, ezért a táblázat/vonal közepén nem változtatható meg a színük! (kivéve `\multicolumn`-os trükközést)

# Törhető táblázat – longtable I

- ha oldalak közt megtörhető táblázatra van szükségünk, mert olyan hosszú: `longtable` csomag `longtable` környezete
- kb. a `tabular` egy verziójának felel meg, de ezt *ne* csomagoljuk `table` környezetbe, mert akkor megint törhetetlen lesz!
- önmagában foglalja a feliratozást és számozást, `\caption` használható, *de* sortörés kell utána(!)
- `\kill` parancs: adott sort nem jeleníti meg, de az elrendezésbe, sorszélességekbe beleszámolja
- elhelyezhető benne `\footnote`!
- készíthető vele táblázatfejléc és -lábléc, ami az összes oldalon a táblázat tetejére ill. aljára kerül – avagy, címsorok automatikus ismétlése, akár `\caption`-nel együtt!

# Törhető táblázat – longtable II

```
\begin{longtable}{ll}
% első oldal fejléce
\caption{longtable}\\
oszlop1 & oszlop2 \endfirsthead
% többi oldal fejléce
oszlop1 & oszlop2 \endhead
% utolsó sortörés helyett \endhead(!)
% táblázat lábléce - opcionális
\hline \endfoot
%% hasonlóképp utolsó oldal lábléce \endlastfoot-ig
% rendes sorok...
\end{longtable}
```