# Tartalomjegyzék



# Tematika

- Sage (és Python) - 7 el?adás
- XHTML, CSS - 2 el?adás
- **LaTeX** - 3 el?adás

# Sage bevezet?

Miért kellenek segédprogramok a matematikához?

- számolási hibák
- gyorsabb
- négyszín-tétel

Miért a Sage?

- nyílt forráskódú
- ingyenes
- könnyen kiterjeszthet?, programozható (Python-ra épül)

# Sage - mint számológép

```
sage: 1 + 2 + 3
6

sage: 3 - 12
-9

sage: 21 * 3
63
```

```
sage: 12 / 4
3

sage: 12 / 5
12/5

sage: 5 ** 4
625

sage: 4 ^ 5
1024

sage: 53 // 4
13

sage: 35 % 8
3

sage: 35.0 / 8
4.37500000000000
```

# Sage - mint okos számológép

Változókat definiálhatunk és m?veleteket hajthatunk végre velük:

```
sage: a = 5 / 3
sage: b = 4 / 3
sage: a + b
3
```

Változó értékének törlése:

```
sage: del a
```

vagy:

```
sage: reset('a')
```

Beépített változókat is használhatunk:

```
sage: e + pi
pi + e
```

# Sage - mint szimbolikus számológép

Ha szimbólumként akarjuk használni valamelyik változót, akkor ezt a `var()` függvénnyel jelezhetjük a Sage programnak.
Egyszerre többet is de?niálhatunk:

```
sage: a,b=var('a,b')
```

Az algebrai kifejezeséket összeggé alakíthatjuk az `expand()` függvénnyel vagy `.expand()` metódussal, ill. szorzattá a `factor()`-ral:

```
sage: expand((a + b) ^2)
a^2 + 2*a*b + b^2
```

`NameError: name 'a' is not defined` hibát kapunk, ha el?tte nem adtuk meg: `a,b=var('a,b')` !!!

```
sage: 231.factor()
3 * 7 * 11
```

Egyenleteket, egyenletrendszereket is képes megoldani a Sage:

```
sage: x = var('x')
sage: solve(x^2 + 3*x +2, x)
[x == -2, x == -1]

sage: x,y = var('x,y')
sage: solve([x + y == 6, x - y == 4], x, y)
[[x == 5, y == 1]]
```

El?fordul, hogy a `solve()` nem találja meg a keresett megoldást:

```
sage: x=var('x')
sage: solve(sin(x) == cos(), x)
[sin(x) == cos(x)]
```

A `find_root()` numerikus megoldást keres adott intervallumon (legyen pl: 0 < y < /2):

```
sage: find_root(sin(x) == cos(x), 0, pi/2)
0.7853981633974484
```

# Sage

## Sage - relációk

```
sage: 3 < 5
True

sage: 4 == 4
True

sage: 3 + 4 <= 6
False

sage: a > b
a > b

sage: a = 5; b = 7
sage: a > b
False
sage: a != b
True
```

## Sage - Boole algebra

```
sage: True and False
False
sage: True and True
True
sage: True or False
True
```

```
sage: not False
True
sage: True and not False
True

sage: a = True; b = False
sage: (a and not b) or b
True
```

# Sage - függvények

Gyakran használt matematikai függvények:

```
sqrt(), sin(), cos(), tan(), cot()

is_prime(), factor(), expand(), solve(), simplify()

plot(), parametric_plot()
```

stb.

Saját függvény definiálása (a `def` kulcsszó a Python nyelv része):

```
sage: def paros(x):
....:     return x%2 == 0
....:
sage: paros(9)
False
sage: paros(16)
True

sage: a = True; b = False
sage: (a and not b) or b
True
```

# Sage - függvényábrázolás

A `plot()` függvénnyel ábrákat készíthetünk.
Általában függvény ábrázolására használjuk, de egyebeket is rajzolhatunk (pl. köröket, sokszögeket).
A plot els? paramétere az ábrázolandó dolog, utána még sok paraméterrel módosíthatjuk a képet (intervallum, színek stb).

```
sage: plot(x^2, (-3,3), aspect_ratio=1)

sage: plot(sin(x), (-pi, 2*pi), rgbcolor='red')
```

Több függvényt is megjeleníthetünk közös koordinátarendszerben. Ehhez el?ször elmentjük az egyes gra?kus objektumokat egy-egy valtozóba, majd a `show()` metódust használjuk:

```
sage: g1 = plot(sin, (-2*pi, 2*pi))
sage: g2 = plot(cos, (-2*pi, 2*pi), rgbcolor='red')
sage: show(g1 + g2)
```

# Python - Sage különbségek

Hatványozás m?veleti jele Python-ban **, a ^ mást jelent (XOR)!

```
>>> 3^3
0
>>> 1^3
2
```

Osztás (/) operátor Pythonban egész-osztást végez.

```
>>> 3/5
0
```