*Mathematica* je program založený na architektuře klient - server. To co právě vidíte je klient nazývaný Front-end. Front-end vám dává k dispozici okno, v kterém je možné psát a tvořit dokument, dělat výpočet či psát program. To, co napíšete do tohoto okna, tvoří dohromady takzvaný notebook, který můžete uložit v souboru *.nb. Front-end vám kromě okna s notebookem dává k dispozici hlavní menu, v kterém najdete spoustu nastavovacích i ovládacích prvků celého prostředí *Mathematica*. Také je možné si zapnout zobrazení množtví různých pomocných sad tlačítek, tzv. palet, pro usnadnění některých postupů, jako například psaní řeckých znaků. Standartně je při zapnutí Mathematicy obvykle zobrazena paleta pojmenovaná BasicMathInput.

Nyní by všichni měli mít otevřený prázdný notebook. Notebook je oproti klasickým textovým dokumentům strukturován nikoliv ("jen") do odstavců, ale do tzv. buněk.
Zkusme si do Notebooku napsat naše dva první výpočty

```
1 + 1
2 * 3
```

V pravo se nám objevila modrá čára ohraničující buňku, kterou jsme teď vytvořili. Teď bychom ale chtěli, aby nám *Mathematica* naše výpočty provedla a vypsala výsledek. Jakékoliv výpočty v Mathematice neprování Front-end, ale servrová část systému Mathematicy zvaná Kernel. Kernel je samostatný program, který může dokonce běžet i na jiném počítači. Funguje to tak, že umístíte kurzor do buňky, kterou chcete spočítat, zmáčknete Shift-Enter (nebo též samotný pravý Enter u numerické klávesnice) a Front-end odešle obsah buňky do Kernelu. Kernel provede všechny příkazy a výpočty a vrátí Front-endu případné výsledky. Front-end vypíše výsledky do buněk, které si vytvoří pod spuštěným výpočtem.

```
1 + 1
23^35
```

```
2
```

```
457 587 614 181 485 537 342 488 537 004 525 777 796 719 632 007
```

Chcete-li nyní vytvořit další buňku, najedete s kurzorem pod buňky či mezi buňky tak, aby se místo klasického kurzoru objevila dlouhá vodorovná čára a začnete psát.
Buňky můžeme do Kernelu posílat v jakémkoliv pořadí, ale Kernel je samozřejmě spracovává postupně za sebou tak, jak k němu přišly. To znamená, že například může být nedříve odeslán a proveden výpočet, který je v notebooku v poslední buňce. Proto jsou příkazy (řádky) přicházející do Kernelu postupně číslovány a Front-end při odeslání buňky vypíše toto pořadové číslo zpracování prvního příkazu v buňce (modře vlevo od buňky). Stejnými čísly jsou pak označeny případné vypisované výsledky jednotlivých příkazů.

```
a
```

```
5
```

```
a = 5
```

```
5
```

```
a
```

```
a
```

S buňkami lze dělat spoustu dalších věcí jako : označovat je, kopírovat, mazat, měnit jejich styl písma, grafiku, zobrazování, povolovat a zakazovat jejich odeslání do Kernelu, dát více buněk do jedné "nadbuňky", sbalit "podbuňky" jedné buňky tak, aby byla vidět jen první, přiřazovat jim nastavení pomocí stylů atd... Takto lze použít notebooky v *Mathematice* nejen k výpočtům, ale i např. pro psaní textu či vytváření presentací. V Mathematice byly dokonce napsány celé knihy. Toto však nyní necháme a budeme se věnovat především možností programování a výpočtů v *Mathematice*. Udělejme si první příklad

Napišme program, který vypočítá řešení algebraické rovnice5*x^5 + 1 == Sin($\frac{2}{3}\pi$).

```
reseni = Solve[5 x^5 + 1 == Sin[2 * Pi / 3], x]
```

$$\left\{\left\{x \to \left(\frac{1}{5}\left(-1+\frac{\sqrt{3}}{2}\right)\right)^{1/5}\right\}, \left\{x \to (-1)^{2/5}\left(\frac{1}{5}\left(-1+\frac{\sqrt{3}}{2}\right)\right)^{1/5}\right\}, \left\{x \to -(-1)^{3/5}\left(\frac{1}{5}\left(-1+\frac{\sqrt{3}}{2}\right)\right)^{1/5}\right\},\right.$$
$$\left.\left\{x \to (-1)^{4/5}\left(\frac{1}{5}\left(-1+\frac{\sqrt{3}}{2}\right)\right)^{1/5}\right\}, \left\{x \to -\left(-\frac{1}{5}\right)^{1/5}\left(-1+\frac{\sqrt{3}}{2}\right)^{1/5}\right\}\right\}$$

Nyní vidíme základy syntaxe v *Mathematice*
- *Mathematica* je case-senzitiv
- Všechny názvy vnitřních příkazů, funkcí i výrazů *Mathematicy* začínají velkým písmenem (Solve, Sin, Pi, I, LaplaceTransform )
- vaše proměnné a symboly mohou začínat pouze písmenem a nesmí obsahovat speciální znaky (nepoužívat podtržítka)
- mezera je interpretována jako krát. Pozor může být zdroj chyb

```
b1
b 2
```

```
b1
```

```
2 b
```

- ";" na konci řádku potlačí vypsání výstupu

```
a = 3;
```

- "=" značí definici (od chvíle kdy je do kernelu odesláno:

```
Petr = prase
```

```
prase
```

bude jakýkoliv Petr nahrazen

```
Petr + Zuzana
```

```
prase + Zuzana
```

Jestliže použijeme funkci, symbol, či obecně něco co *Mathematica* nezná, *Mathematica* provede všechna nahrazení, která zná a zbytek opět vrátí jako výraz

```
Petr = prase;
rande[Petr, Zuzana]
```

```
prase
```

```
rande[prase, Zuzana]
```

Zrušit definici lze například příkazem ClearAll (lze to též důkladněji příkazem Remove, který má syntaktickou zkratku "=.", ale to je jen pro zvídavé. Více viz. help)

```
ClearAll[Petr]
```

```
Petr + Zuzana
```

```
Petr + Zuzana
```

- "==" značí rovnici (patří mezi základní bilanční znaménka >, <, >=, <=, ==, != )

```
Petr = velkePrase
```

```
velkePrase
```

```
Petr == velkePrase
```

```
True
```

- "( )" jsou závorky upřesňující pořadí vyhodnocování (čili přednost operátorů)

```
5 + 3 * 7
(5 + 3) * 7
```

```
26
```

```
56
```

- "[ ]" jsou závorky pro parametry funkcí a příkazů (jednotlivé parametry se oddělují čárkou)

```
Sin[2 * Pi / 3]
```

$$\frac{\sqrt{3}}{2}$$

```
Integrate[x^5 * Sin[x], x]
```

$$-x\left(120-20\,x^2+x^4\right)\text{Cos}[x]+5\left(24-12\,x^2+x^4\right)\text{Sin}[x]$$

Pro funkce s jedním parametrem lze místo syntaxe:

```
nazevFce[parametr ]
```

```
nazevFce[parametr]
```

"nazevFce[parametr ]" použít též dva ekvivalentní způsoby zadávání parametrů fce:

```
nazevFce@parametr
parametr // nazevFce
```

```
nazevFce[parametr]
```

```
nazevFce[parametr]
```

takže funguje např.:

```
Sin@(2 * Pi / 3)
(2 * Pi / 3) // Sin
```

$$\frac{\sqrt{3}}{2}$$

$$\frac{\sqrt{3}}{2}$$

- "{ }" jsou závorky sdružující více věcí do jedne usporadane mnoziny, jednotlivé objekty se od sebe oddělují čárkou. V prostředí *Mathematica* se uspořádaná množina vytvořená pomocí "{ }" nazývá "List".

Listy používáme pro sofistikované uspořádávání svých dat

```
basaPiv = {Budwar, Prazdroj, Gambrinus}
```

```
{Budwar, Prazdroj, Gambrinus}
```

```
basaPiv2 = {Kozel, Branik}
```

```
{Kozel, Branik}
```

```
hromadaBasPiv = {basaPiv, basaPiv2}
```

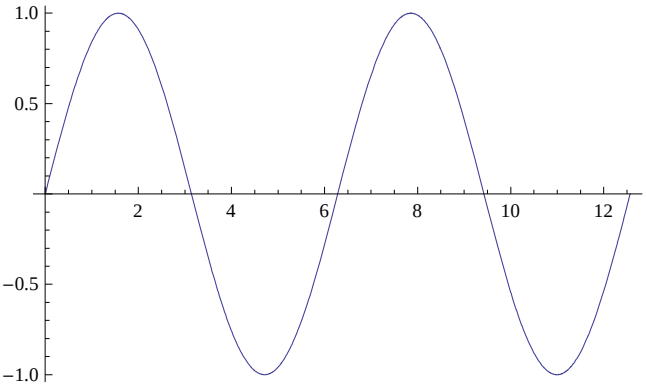
```
{{Budwar, Prazdroj, Budwar}, {Kozel, Branik}}
```

```
skladPiv = {basaPiv, basaPiv2, Bernard, CernaHora}
```

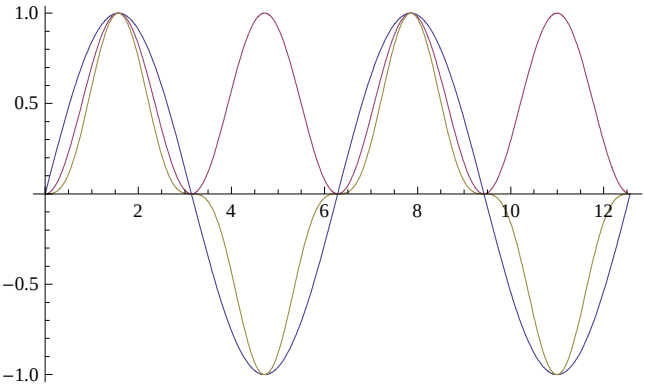
```
{{Budwar, Prazdroj, Budwar}, {Kozel, Branik}, Bernard, CernaHora}
```

Listy lze často použít tam kde funkce očekává jedinný parametr pro zadání více objektů.
Např. příkaz Plot očekává dva parametry: 1. výraz, který bude kreslit 2.za kterou proměnnou a v jakých mezích má dosazovat

```
Plot[Sin[x], {x, 0, 4 π}]
```




```
Plot[{Sin[x], Sin[x]^2, Sin[x]^3}, {x, 0, 4 π}]
```




Listy se v Mathematice používají pro reprezentaci vektorů a matic

```
ctvercovaMatice = {{1, 3, 5, 6}, {6, 5, 8, 2}, {9, 54, 56, 78}, {74, 25, 19, 3}}
```

```
{{1, 3, 5, 6}, {6, 5, 8, 2}, {9, 54, 56, 78}, {74, 25, 19, 3}}
```

```
ctvercovaMatice // MatrixForm
```

$$\begin{pmatrix} 1 & 3 & 5 & 6 \\ 6 & 5 & 8 & 2 \\ 9 & 54 & 56 & 78 \\ 74 & 25 & 19 & 3 \end{pmatrix}$$

```
Det[ctvercovaMatice]
```

```
44153
```

- "[[ ]]" jsou závorky, kterými z listu vybíráme prvek dle jeho pořadí od začátku (pomocí kladných čísel), nebo od konce (pomocí záporných čísel)

```
basaPiv
```

```
{Budwar, Prazdroj, Gambrinus}
```

```
basaPiv[[1]]
```

```
Budwar
```

```
basaPiv[[2]]
```

```
Prazdroj
```

a teď od konce

```
basaPiv[[-1]]
```

```
Gambrinus
```

nebo vícestupňový výběr

```
skladPiv
```

```
{{Budwar, Prazdroj, Budwar}, {Kozel, Branik}, Bernard, CernaHora}
```

```
skladPiv[[2]]
```

```
{Kozel, Branik}
```

```
(skladPiv[[2]])[[1]]
```

```
Kozel
```

```
skladPiv[[2, 1]]
```

```
Kozel
```

## ■ Nejdůležitější věc v *Mathematice*

Nejdůležitější věc v systému *Mathematica* : Help

Otevření nápovědy:
- Menu,
- Příkaz "?" nebo "??" vypíše základní nápovědu do notebooku

```
? Plot
```

Plot[$f$, {$x$, $x_{min}$, $x_{max}$}] generates a plot of $f$ as a function of $x$ from $x_{min}$ to $x_{max}$.
Plot[{$f_1$, $f_2$, …}, {$x$, $x_{min}$, $x_{max}$}] plots several functions $f_i$. ≫

- Nejjednodušeji klávesa F1 (tip: jestliže napíšete v notebooku příkaz, umístíte do něj kurzor a zmáčknete F1 otevře se vám přímo stránka helpu k tomuto příkazu )

otevření nápovědy k příkazu Plot

```
Plot
```

Co najdete v Helpu:

- stránku s nápovědou ke každému příkazu (například pro příkaz Plot)
  všechny helpy k jednotlivým příkazům mají stejnou strukturu:
  - nahoře je krátký popis, který se zobrazuje též v notebooku příkazem "?"
  - podrobný popis všech vlastností a možností nastavení
  - příklady použití, které si můžete vyzkoušet, klidně změnit a spouštět, nebo zkopírovat do svého notebooku a používat
  - odkazy na podobné a související příkazy
  - odkazy na tutoriály o tématu, do kterého příkaz patří
  stránku s nápovědou ke každému příkazu
  - odkazy na rozcestníky příkazů uskupené podle témat

- celá elektronická knížka o Mathematice od základů až po pokročilé věci, která je složená z tutoriálů pro jednotlivá témata. Každý tutoriál obsahuje na spoustě dalších příkladů ukázané základní postupy a možnosti. Např. tutoriál o základním vykreslování fcí. Opět si můžete beztrestně příkazy upravovat a spouštět.

- v Mathematice 7 jsou v helpu nahoře, kromě vyhledávače, tlačítka pro otevření rozcestníku fcí a pro otevření rozcestníku tutoriálů

Základní práce s Helpem:
- zkusit odhadnout jméno příkazu nebo alespoň podobného příkazu
- najít podobný příkaz a projít související příkazy nebo ještě lépe rozcestník se souvisejícími příkazy
- najít vhodný příklad, ten zkopírovat do svého notebooku a upravit

Např najděte příkaz pro otočení pořadí prvků v listu (List - List Manipulation)

```
List
```

```
Reverse[{a, b, c, d}]
```

```
{d, c, b, a}
```

## ■ Některé základní příkazy v *Mathematice*

Ukažme si nyní některé základní příkazy, které byste si měli projít

```
Plot
```

```
Solve
```

```
Table
Range
```

```
% (Out)
```

## ■ Programování v *Mathematice*

### ■ Pravidla nahrazování

pravidlo = coNahradit → cimNahradit

```
pravidlo = x → y^2
```

$x \to y^2$

necoUpravene = necoPredUpravou /. pravidlo

```
x + 3 /. pravidlo
```

$3 + y^2$

Další příklady:

```
x + y + z /. {x → z, y → z}
```

$3\,z$

```
x /. x → 25
```

$25$

Výsledkem řešení rovnic pomocí Solve je list pravidel. Vyřešte rovnici $x^4 + x^2+1==0$ a získejte první řešení ve formě výrazu

```
rovnice = x^4 + x^2 + 1 == 0
vsechnaReseniPravidla = Solve[rovnice, x]
prvniReseniPravidlo = vsechnaReseniPravidla[[1]]
prvniReseniVyraz = x /. prvniReseniPravidlo
```

$1 + x^2 + x^4 == 0$

$\{\{x \to -(-1)^{1/3}\}, \{x \to (-1)^{1/3}\}, \{x \to -(-1)^{2/3}\}, \{x \to (-1)^{2/3}\}\}$

$\{x \to -(-1)^{1/3}\}$

$-(-1)^{1/3}$

získejte vsechna řešení ve formě výrazů

```
vsechnaReseniVyraz = x /. vsechnaReseniPravidla
```

$\{-(-1)^{1/3}, (-1)^{1/3}, -(-1)^{2/3}, (-1)^{2/3}\}$

- **Základy o funkcích v Mathematice**

- Tvoření vlastních funkcí

základní syntaxe:
nazevFce[parametry s podtrzitkem] := coVracet

```
mojeFce[par1_, par2_] := par1 + par2
```

```
mojeFce[2, 5]
mojeFce[Manka, Rumcajz]
```

$7$

$Manka + Rumcajz$

- Aplikace funkce na každý prvek listu

příkaz Map aplikuje funkci na každý prvek z listu

Map[f, {x1,x2,...}] = {f[x1], f[x2], ...}
nebo jiným zápisem
f/@{x1,x2,...} = {f[x1], f[x2], ...}

```
Map[g, {x1, x2, x3}]
```

```
{g[x1], g[x2], g[x3]}
```

```
g /@ {x1, x2, x3}
```

```
{g[x1], g[x2], g[x3]}
```

Příklad na rozmnožování havěti:

```
f[x_] := 10 * x
havet = {blecha, ves, mys}
Map[f, havet]
f /@ havet
```

```
{blecha, ves, mys}
```

```
{10 blecha, 10 ves, 10 mys}
```

```
{10 blecha, 10 ves, 10 mys}
```

- Aplikace funkce na celý list

Apply z prvků listu udělá argumenty zadané funkce

Apply[f, {x1,x2,...}] = f[x1, x2, ...]
nebo jiným zápisem
f@@ {x1,x2,...} = f[x1, x2, ...]

```
Apply[g, {x1, x2, x3}]
```

```
g[x1, x2, x3]
```

```
g @@ {x1, x2, x3}
```

```
g[x1, x2, x3]
```

Příklad:
sečtěte prvky listu, když víte, že sčítání je v Mathematice reprezentováno fcí Plus

```
Plus[a, b, c]
```

```
a + b + c
```

```
Plus @@ {a, b, c}
```

```
a + b + c
```

- Vícenásobná aplikace funkce na jeden parametr

Příkaz Nest aplikuje fci na parametr několikrát

Nest[g, x, n] = g[g[..g[x]]..]

```
Nest[g, x, 3]
```

```
g[g[g[x]]]
```

Příkaz NestList také aplikuje fci na parametr několikrát, ale vrátí i mezivýsledky

NestList[g, x, n] = {x,g[x],g[g[x]], ..., g[g[..g[x]]..]}

```
NestList[g, x, 3]
```

```
{x, g[x], g[g[x]], g[g[g[x]]]}
```