Python – lekce 12

Pylab (Python Laboratory)

- Pylab je **jazyk vysoké úrovně** inspirovaný známým komerčním systémem Matlab
- je kompletně vytvořen v jazyce Python, proto lze v něm používat všechny příkazy Pythonu, respektive to lze vyjádřit i opačně – Pylab lze chápat jako nadstavbu jazyka Python **pro vědecké a technické výpočty a pro vizualizaci dat**
- jedná se o profesionální vědecký/technický nástroj pro maticové výpočty (stejně jako Matlab), navíc vybavený programovacím jazykem (to Matlab nemá), používá se i na mnohých vysokých školách
- Pylab je vlastně balíček 3 modulů (knihoven funkcí):
  - **numpy** – modul funkcí pro **num**erické výpočty v **Py**thonu
  - **scipy** – modul funkcí pro vědecké (**sci**ence) a technické výpočty v **Py**thonu, využívá i funkce z numpy
  - **matplotlib** – knihovna funkcí pro vizualizaci dat, tj. umožňuje zobrazení všemožných druhů grafů, diagramů a obrazců – vektorových i rastrových obrázků
- je platformě nezávislý (stejně jako Python) => stejný program lze spustit jak na běžném PC s Windows, tak i na nějakém lepším serveru např. s Unixem (pro složitější a rozsáhlejší výpočty)

Datový typ "vektor" (versus Pythonovský seznam)

- protože Pylab (numpy) slouží pro maticové (vektorové) výpočty, má definovány vlastní datové typy pro vektor a matici
- vektor (array) je velmi podobný seznamu z Pythonu, rozdíl je hlavně v tom, že vektor může obsahovat jen čísla a umí s ním pracovat veškeré matematické funkce Pylabu
- matice je pak de facto vektor vektorů (my však zde vystačíme s vektorem)
- vektor lze vytvořit několika způsoby:

Program 01 – vytvoření vektoru

a) z Pythonovského seznamu:

```
from pylab import *                          # importuj vše z modulu pylab
seznam = [ 5, 4, 397.1, 7.9 ]
vektor1 = array( seznam )
vektor2 = array( [ 10, 9, 12 ] )
print(seznam)
print(vektor1)
print(vektor2)
```

b) jako posloupnost – obdoba Pythonovské funkce range()

```
from pylab import *
v1 = arange(5)                               # [ 0 1 2 3 4 ]
v2 = arange(13, 18)                          # [ 13 14 15 16 17 ]
v3 = arange(2.7, 3.2, 0.1)                   # [ 2.7 2.8 2.9 3. 3.1 ]
print(v1)
print(v2)
print(v3)
```

c) jako posloupnost s počtem hodnot

```
from pylab import *
x = linspace(0,10,50)                        #vektor od 0 do 10, 50 hodnot
print(x)
```

Práce s Pylabovskými "vektory"

- s datovým typem "vektor" lze provádět všechny základní matematické operace s tím, že daná operace se automaticky provede se všemi prvky vektoru a výsledkem je opět stejně dlouhý vektor:

Program 02 – operace s vektory

```
from pylab import *
x = arange(5)                    #  [ 0  1  2  3  4 ]
print(2 * x)                     # [ 0  2  4  6  8 ]
print(x + 10)                    # [ 10  11  12  13  14 ]
print((2*x + 10)/4)              # [ 2.5  3.   3.5  4.   4.5]
```

- s vektory tedy lze pracovat podobně jednoduše jako s "proměnnými" v matematice (x)
- můžeme použít i složitější funkce – vždy však z knihovny pylab (aby uměly pracovat s vektorem):

```
print(sin(x))                    # vektor s hodnotami sinus
print(cos(x))                    # vektor s hodnotami cosinus
print(log(x+1))                  # přirozený log.
print(sqrt(x))                   # druhá odmocnina
print(x**0.5)                    # druhá odmocnina jinak
```

- srovnání s Pythonovským seznamem (neboli rozdíl mezi seznamem a vektorem):

Program 03 – seznam vs. vektor

```
from pylab import *
seznam = [ 5, 4, 7.9 ]
vektor = array( seznam )
print(seznam*3)                  # [ 5, 4, 7.9,   5, 4, 7.9,   5, 4, 7.9 ]
print(vektor*3)                  # [ 15.   12.   23.7 ]
```

Vykreslení jednoduchého grafu funkce

- chceme-li nakreslit graf nějaké funkce, musíme vytvořit tabulku hodnot pro "x" a dopočítat odpovídající funkční hodnoty:

x	0	0.1	0.2	...	6.4
y = sin x	0	0.09983	0.19867	...	0.11655

- potřebujeme tedy 2 stejně dlouhé vektory – pro hodnoty x a pro hodnoty y
- vytvořit je můžeme např. takto:

Program 04 – vykreslení jednoduchého grafu

```
from pylab import *
x = arange(0, 6.5, 0.1)          # od 0 do 6.5 krok 0.1
y = sin(x)                       # opravdu jen takto jednoduše
print(x)
print(y)
```

- graf funkce podle výše uvedené tabulky pak lze nakreslit snadno pomocí funkce plot:

```
plot(x,y)                        # vykreslí graf
```

- vlastnosti grafu můžeme nastavit pomocí následujících funkcí

```
grid(True)                          # zobrazení mřížky
title("Graf funkce sin(x)")         # titulek grafu
xlabel("Osa x")                     # popis osy x
ylabel("Osa y")
```

Program 05 – vykreslení dalšího grafu

```
from pylab import *
x=arange(-10,10,0.2)    #vektor x od -10 do 10, krok 0.5
print(x)
"""
x=linspace(-10,10,40)   #vektor x od -10 do 10, 40 hodnot
print(x)
"""
y1=2*x+1
y2=x**2
y3=x**4
grid(True)              #zobrazení mřížky
title("Graf funkce x3")
xlabel("Osa x")         # popis osy x
ylabel("Osa y")
plot(x,y1,x,y2)         #vložení údajů do grafu
plot(x,y3)
ylim(-5,50)             #limity pro osu y
legend('x')             #zobrazení legendy
```