11. Programování - knihovna Pylab – tvorba grafů

knihovna pylab

Pylab je knihovna pro vědecké a technické výpočty a pro vizualizaci dat inspirovaná nástrojem MATLAB. Pylab je vlastně balíček 3 modulů:

- numpy – pro numerické výpočty (vektory, matice) v Pythonu
- scipy – pro vědecké výpočty v Pythonu
- matplotlib – pro vizualizaci dat, tj. umožňuje zobrazení všemožných druhů grafů, diagramů a obrazců

Nepatří mezi základní knihovny, před importem ji musíme stáhnout přes balíčkovací systém pip.

```
import pylab as pl  # Nedoporučuje se importovat * - bije se např. s fcemi z math
```

vytvoření vektoru, operace s vektorem

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 (numpy).

Vektor lze **vytvořit** několika způsoby:

1. z Pythonovského seznamu

   ```
   vektor = pl.array( [ 10, 9, 12 ] )
   ```

2. pomocí speciálních funkcí (rand, zeros, ones, ...)

   ```
   vektor1 = pl.rand(3)    # [ 0.76798 0.86647 0.40247]
   vektor2 = pl.zeros(5)   # [ 0 0 0 0 0 ]
   vektor3 = pl.ones(5)    # [ 1 1 1 1 1 ]
   ```

3. jako posloupnost – obdoba Pythonovské funkce range(); většinou menší krok pro plynulejší graf

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

4. jako rozsah s počtem hodnot s lineárním krokem/rozdílem mezi hodnotami

   ```
   x = pl.linspace(0,10,50)      # 50 hodnot od 0 po 10 včetně
   ```

S vektory 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.

```
x = pl.arange(5)      # [ 0 1 2 3 4 ]
print(2 * x)          # [ 0 2 4 6 8 ]
print(x + 10)         # [ 10 11 12 13 14 ]
```

Můžeme použít i **složitější funkce** – vždy však z knihovny pylab (numpy), aby uměly pracovat s vektorem.

```
print(pl.sin(x))      # vektor s hodnotami sinus
print(pl.cos(x))      # vektor s hodnotami cosinus
print(pl.sqrt(x))     # druhá odmocnina
```

vykreslení grafu

Chceme-li nakreslit graf nějaké funkce, musíme vytvořit tabulku hodnot pro "x" a dopočítat odpovídající funkční hodnoty. Potřebujeme tedy **2 stejně dlouhé vektory** – pro hodnoty x a pro hodnoty y. Vykreslíme je funkcí **plot** a zobrazíme GUI funkcí **show**.

```
x = pl.arange(0, 6.5, 0.1) # od 0 do 6.5 (6.4 včetně) krok 0.1
y = pl.sin(x)
pl.plot(x, y)      # vykreslí čáru
pl.plot(x, y*2)    # vykreslí další čáru do stejného grafu
pl.show()          # zobrazí okno s grafem na obrazovce a resetuje všechna předchozí nastavení
```

parametry čáry

Do fce plot předáváme řetězec ve tvaru “<barva><styl čáry><tvar bodu>”.

Barvy: r g b c m y k

Styly: - . : -- -.

Body: x o s ^

Tloušťka čáry - parametr linewidth

```
plot(x,y1,"g.", linewidth=4)    #zelená, tečkovaná, tlustá
plot(x,y2,"b:")                 #modrá, jemněji tečkovaná
plot(x,y3,"k^")                 #jen samotné body – trojúhelníky
```

Nebo lze použít parametr **color** pro hexadecimální zápis.

vytvoření legendy, parametry grafu

```
pl.grid()                 # zobrazení mřížky - funguje jako přepínač
pl.grid(True)             # zobrazení mřížky - nastaví vždy
pl.title("Graf funkce x3")  # titulek grafu
```

Popisy os

```
pl.xlabel("Osa x")
pl.ylabel("Osa y")
```

Změna výchozího zobrazení grafu na určité **rozsahy os**. (zbytek grafu si může uživatel vždy prohlédnout posunem v okně)

```
pl.xlim(-3,3)
pl.ylim(-1,10)
```

Legenda

```
pl.legend(loc="center")
```

Možnosti umístění legendy řetězcem:

- best *(default)*
- center right/left
- upper right/left
- lower right/left
- center
- upper center
- lower center

Vložení údajů s **popisy pro legendu (syntax jazyka Latex)**

```
pl.plot(x, y1, label="$y=2x+1$")
pl.plot(x, y2, label="$y=\sqrt{x^2}$")
pl.plot(x, y3, label="$y=x^3$")
```

více grafů

Zobrazení více grafů, každý **do zvláštního okna** → **figure**(číslo_okna)

```
x=pl.arange(0,10,0.2)
y=x**2
pl.figure(1)      # nyní budeme kreslit do okna "1"
pl.plot(x,y)
pl.figure(2)      # nyní budeme kreslit do okna "2"
pl.plot(y,x)
pl.show()         # zobrazí všechna okna s grafy
```

Zobrazení více grafů **v jednom okně** → **subplot**(počet_řádků, počet_sloupců, číslo_grafu)

Určuje část okna, do které se budou vykreslovat následující volání "plot( )"; logika je taková, že dojde jakoby k rozdělení okna na daný počet řádků a sloupců, buňky se číslují od levého horního rohu od 1. Překryje-li daný podgraf třeba jen částí jiný již vykreslený podgraf, pak ten dříve vykreslený podgraf z okna zmizí.

```
pl.subplot(2,2,1)       # kresli do levé horní čtvrtiny
pl.plot(...)
pl.subplot(2,2,2)       # do pravé horní čtvrtiny
pl.plot(...)
pl.subplot(2,1,2)       # do dolní poloviny
pl.plot(...)
pl.show()
```

plocha pod grafem a mezi grafy

fill(x, y1, barva) – vyplní plochu pod grafem zadanou barvou

```
x=pl.arange(1,3*pi,0.01)
y1=pl.sin(x)
pl.fill(x,y1,"#00ff42")
pl.plot(x,y1)
pl.show()
```

fill_between(x, y1, y2, where=None, facecolor=barva) – vyplní plochu mezi dvěma čarami, 2. vektor (y2) není povinný, bez něj se chová jako **fill()** (ale můžeme použít parametr where)

where = podmínka, která určuje, kde má být plocha vyplněna; není-li uvedeno, plocha je vyplněna všude

```
x=pl.arange(-3,3,0.01)
y1=x**2-1
y2=2*x+1
pl.fill_between(x,y1,y2,where=y1>0,facecolor="gray")
```

pl.plot(x,y1,"black")

pl.plot(x,y2,"black")

pl.show()

vložení textu do grafu

text(x, y, řetězec, **parametry) - umístí řetězec na zvolené souřadnice v grafu, kotevní bod textu je vlevo dole, ale způsob zarovnání můžeme změnit pomocí nepovinných parametrů.

pl.text(0.3, 1, “Text”, color="r", fontsize="16", horizontalalignment='right', verticalalignment='top')

annotate(řetězec,xy=(x,y),xytext=(x,y),arrowprops=dict(facecolor=color,shrink=x))

- poznámka s šipkou
- šipka ukazuje od poznámky na bod xy
- poznámka je umístěna na souřadnicích xytext
- vlastnosti šipky se definují v arrowprops, shrink znamená zkrácení šipky jako desetinné číslo (0.25 = čtvrtina)

pl.annotate('Vybarvená plocha', xy=(1,2), xytext=(-1.5,5), arrowprops=dict(facecolor='red'))