

# Python

Matematické metody a modelování

---

Michal Šesták

9. 1. 2019

# Úvod

- interpretovaný programovací jazyk, navržen v roce 1991, implementován v C (hlavní proud)
- dvě nekompatibilní verze: 2.x, 3.x (novější); vyvíjí se oba (2.x pomaleji); 2.x přestane být vyvíjen v roce 2020
- hybridní jazyk (objektově orientovaný a zároveň procedurální a funkcionální)
- dobře se čte a dá se velice rychle naučit
- “Python a jeho mezery”
- pomalejší jazyk (cca 400x než C++), ale lepší “development speed” a méně kódu, je k dispozici zdarma, dokáže nahradit R
- silné balíčky, ideální pro zpracovávání dat, spousta triků
- široké použití (databáze, webové programování, aplikace (hlavně linuxové), hry, věda)

# Charakteristiky

- proměnná je pojmenovaným odkazem na objekt

```
1 | a = [1, 2]
2 | b = a
3 | del b[0]
4 | a
5 | [2]
```

- Funkce se uchovává jako objekt

```
1 | def funkce():
2 |     print('Python')
3 | f = funkce
4 | p = [1, 2, 'test', f]
```

- ...

# Balíčky; indexování

- numpy, matplotlib, scipy, pandas, math, random, datetime; sys, pdb, locale, time; SymPy (symbolic computation), seaborn, lmfit

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import pandas as pd
4 from scipy.optimize import curve_fit
```

- Indexování:

```
+---+---+---+---+---+---+
| P | y | t | h | o | n |
+---+---+---+---+---+
slice position: 0   1   2   3   4   5   6
Index position:   0   1   2   3   4   5
```

# Slicing

- základní

```
1 a[start:end] # items start through end-1
2 a[start:]    # items start through the rest of the array
3 a[:end]      # items from the beginning through end-1
4 a[:]         # a copy of the whole array
5
6 a[start:end:step] # start through not past end, by step
7
8 a[-1]    # last item in the array
9 a[-2:]   # last two items in the array
10 a[:-2]  # everything except the last two items
11
12 a[::-1]  # all items in the array, reversed
13 a[1::-1] # the first two items, reversed
14 a[-3:-1] # the last two items, reversed
15 a[-3::-1] # everything except the last two items, reversed
```

- numpy.array -> spousta triků

```
1 x = np.array([[ 0,  1,  2],
2 ...          [ 3,  3,  5],
3 ...          [ 6,  1,  8],
4 ...          [ 9, 10, 11]])
5 x[:,0] #output je array([0, 3, 6, 9]), neboli první sloupec
6 x[x<5] #output je array([0, 1, 2, 3, 3, 1])
7 x[np.where(b<5)] #output je (array([0, 0, 0, 1, 1, 2]),array([0, 1, 2, 0, 1, 0]))
```

# For cyklus; příkazy in, enumerate, zip

```
1 my_list = ['apple', 'banana', 'grapes', 'pear']
2 for i in my_list:
3     print(i)
4 # Output:
5 # apple
6 # banana
7 # grapes
8 # pear
9 for i in np.arange(len(my_list)):
10    print(i)
11 # Output:
12 # 0
13 # 1
14 # 2
15 # 3
16 for i, value in enumerate(my_list, 1): #druhy argument rika ze se indexuje od jednicky
17     print(i, value)
18 # Output:
19 # 1 apple
20 # 2 banana
21 # 3 grapes
22 # 4 pear
23 alist = ['a1', 'a2', 'a3']
24 blist = ['b1', 'b2', 'b3']
25 for a, b in zip(alist, blist):
26     print(a, b)
27 # Output
28 # a1 b1
29 # a2 b2
30 # a3 b3
```

# Zpracovávání dat

- **Numpy:** maticové počítání, “náhrada matlabu”, rychlejší vektorové počítání, obsahuje matematické funkce (náhrada za balíček math)
  - *numpy.random*
  - *numpy.linalg* (lineární algebra)
- **Scipy:** postaven na Numpy; *optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers, statistics and other tasks common in science and engineering*
- **Pandas:** nadstavba Numpy a matplotlib, rychlé zpracovávání dat a dělání grafů; I/O excel, html, latex ...
- **IPython:** python prompt on steroids; it has completion, history, shell capabilities, and a lot more; případně *Jupyter Notebook/Lab*

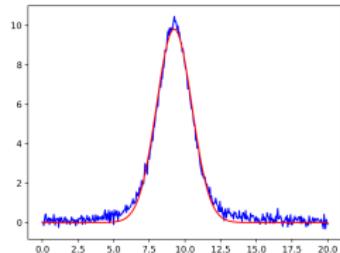


# Fitování

- np.polyfit
- scipy.optimize.curve\_fit
- lmfit

```
1 from numpy import loadtxt
2 from lmfit.models import GaussianModel, LorentzianModel, VoigtModel
3 import matplotlib.pyplot as plt
4 data = loadtxt('test_peak.dat')
5 x = data[:, 0]
6 y = data[:, 1]
7 mod = GaussianModel()
8 pars = mod.guess(y, x=x)
9 out = mod.fit(y, pars, x=x)
10 print(out.fit_report(min_correl=0.25))
11 parametry = dict(out.params)
12 A=out.params['amplitude'].value*out.params['sigma'].value*np.sqrt(2*np.pi) #plocha piku
13 out.plot_fit ()
14 plt.show()
```

```
[[Model]]
    Model(gaussian)
[[Fit Statistics]]
# fitting method   = leastsq
# function evals   = 27
# data points     = 401
# variables       = 3
chi-square         = 29.9943157
reduced chi-square = 0.07536260
Akaike info crit   = -1033.77437
Bayesian info crit = -1021.79248
[[Variables]]
    sigma:      1.23218359 +/- 0.00737496 (0.60%) (init = 1.35)
    amplitude:  30.3135620 +/- 0.15712686 (0.52%) (init = 43.62238)
    center:     9.24277047 +/- 0.00737496 (0.08%) (init = 9.25)
    fwhm:       2.90157056 +/- 0.01736670 (0.60%) == '2.3548200*sigma'
    height:     9.81457817 +/- 0.05087283 (0.52%) == '0.3989423*amplitude/max(1.e-15, sigma)'
[[Correlations]] (unreported correlations are < 0.250)
    C(sigma, amplitude) =  0.577
```



## Příklad 1 – Manganová lázeň

V neutronové laboratoři byl pomocí metody manganové lázně charakterizován radionuklidový zdroj typu Am-Be. Ozařování bylo zahájeno 28. srpna 2017 v čase 9:48:43 a ukončeno 29. srpna 2017 v čase 10:48:38. Celková účinnost záchytu neutronů na nuklidu  $^{55}\text{Mn}$  je 0,38042 (relativní standardní nejistota údaje je 0,27 %). Měření aktivity vzniklého radionuklidu  $^{56}\text{Mn}$  započalo 29. srpna 2017 v čase 11:28:00. Detekční účinnost sestavy se rovná  $1,9113 \cdot 10^{-3}$  (relativní standardní nejistota účinnosti je 0,39 %).

- Z poskytnutého záznamu měření vypočítejte emisi neutronového zdroje.

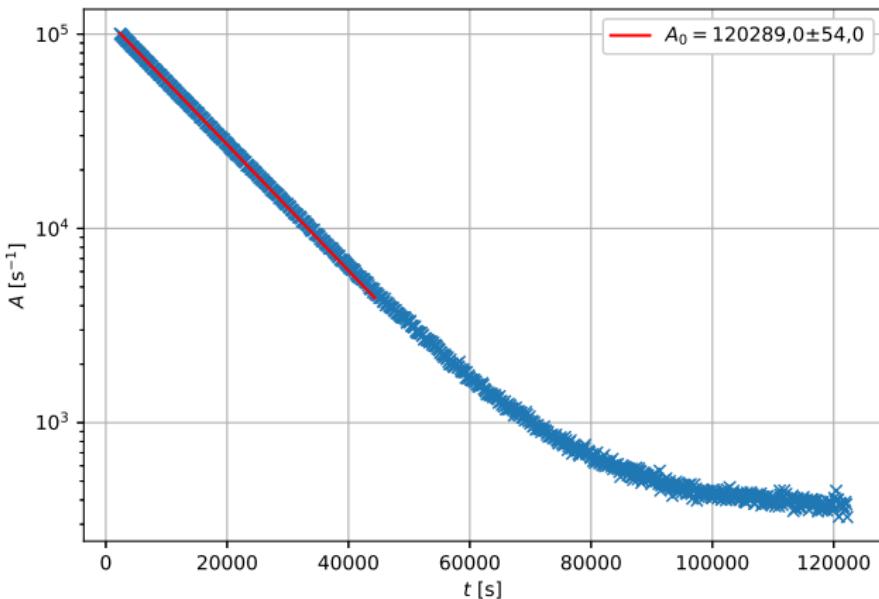
## Příklad 1 (2)

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 import datetime
4 from scipy.optimize import curve_fit
5
6 t0=datetime.datetime (2017,8,28,9,48,43) #pocatek ozarovani
7 t1=datetime.datetime (2017,8,29,10,48,38) #konec ozarovani
8 t2=datetime.datetime (2017,8,29,11,28,0) #zacatek mereni
9
10 prem_konst=np.log(2)/(2.5789*60*60) #[s^-1]
11 eps1=0.38042 #ucinnost zachytu
12 eps1_err=eps1*0.0027
13 eps2=1.9113e-3 #detekcni ucinnost
14 eps2_err=eps2*0.0039
15
16 data=np.loadtxt('Am — Be02.dat')
17
18 rozdil=int((t2-t1).total_seconds())
19 x=data[:,0]+ rozdil
20 y=data[:,1]
21
22 def exponenciela(x,A0):
23     return A0*np.exp(-prem_konst*x)
24
25 x_right=280
26 popt,pcov=curve_fit(exponenciela,x[:x_right ],y[: x_right ])
27 perr = np.sqrt(np.diag(pcov))
28 print(popt)
29 print(perr)
```

## Příklad 1 (3)

```
1 #GRAF
2 fit = '$A_0=' + str(round(popt[0],0)).replace('.','.')+' \pm ' + str(round(perr[0],0)).replace('.','.')
3 xp=np.linspace(x[0],x[x_right], num=x_right)
4 plt.semilogy(x,y,'x')
5 plt.plot(xp,exponenciela(xp,*popt),'r',label= fit )
6 plt.grid()
7 plt.xlabel('t [s]')
8 plt.ylabel('A [s^-1]')
9 plt.legend()
10 plt.show()
11
12 A0=popt[0]
13 A0_err=perr[0]
14
15 def vypocet_chyby(a,a0,eps,a0_err, eps_err):
16     return a*np.sqrt((a0_err/a0)**2+(eps_err/eps)**2)
17
18 A=A0/eps2
19 A_err=vypocet_chyby(A,A0,eps2,A0_err,eps2_err)
20
21 S=A/eps1
22 S_err=vypocet_chyby(S,A,eps1,A_err,eps1_err)
23 print('Emise neutronového zdroje=(+str(round(S,-5))+' '\pm '+str(round(S_err,-5))+') s^-1')
```

## Příklad 1 (výsledek)



$$S = (165.4 \pm 0.8) \text{ Ms}^{-1} \quad (1)$$

## Příklad 2 – Animace

```
1 import numpy as np
2 import scipy
3 import scipy.sparse.linalg
4 import matplotlib.pyplot as plt
5 import matplotlib.animation as animation
6 import time
7 import math
8 import pdb
9 import sys
10 #zde jsou definovany konstanty (polomer koule, pocatecni koncentrace vne koule, difuzni soucinitele, premenova konstanta
11
12 d=float(sys.argv[1]) # tloustka steny (0;0.1]
13 k_u=float(sys.argv[2]) #koeficient prestupu radonu na vnitri okrajove podmínce
14 k_v=float(sys.argv[3]) #koeficient prestupu radonu na vnejsi okrajove podmince
15
16 def vypocet_CN (..., pp,D,c_u,c_v,tolerance =0.01,...)
17     #Crank—Nicolsonova metoda pro numericky vypocet difuzni rovnice
```

## Příklad 2 (2)

```
1 def animace(vysledek,c_u, c_v,tau,k,prostredi, interval =1, ulozit=False):
2     """
3         Inputs:
4             vysledek(array)
5             c_u(array)
6             c_v(float)
7             tau(float)
8         """
9     #zde se definuji objekty
10    def animate(i):
11        #zde se nastavuji objekty pro krok i
12        return line, line_u, text_min, text_sek
13
14    def init():
15        #zde se nastavuji pocatecni hodnoty objektu
16        return line, line_u, text_min, text_sek
17
18    anim = animation.FuncAnimation(fig, animate, np.arange(0, k), init_func=init,
19                                    interval=interval, blit=True, repeat=False)
20
21    if ulozit:
22        anim.save('animace.mp4', fps=30, extra_args=['-vcodec', 'libx264'])
plt.show()
```

## Příklad 2 (3)

```
1 #nejake dalsi funkce vypocitavajici napr. nove pocatecni podminky
2
3 start=time.time()
4
5 #Prvni cyklus=NAPLNOVANI
6 k_t1,tau_u_t1,c_u_t1, vysledek_t1=vypocet_CN(T_t,pp,D_t,tolerance=0.01)
7 animace(vysledek_t1,c_u_t1,c0,tau_t1,k_t1,'tekute', interval=1)
8 #Urceni nove vnitri okrajove podminky a pocatecni podminky
9 c_u_t0,pp_t=urcit_nove_pp(k_t1,c_u_t1,vysledek_t1)
10 #Druhy cyklus=VYPRAZDOVANI
11 k_t2,tau_u_t2,c_u_t2, vysledek_t2=vypocet_CN(T_t,pp_t,D_t,c_u=c_u_t0,c_v=0,tolerance=0.01)
12 animace(vysledek_t2,c_u_t2,0,tau_t2,k_t2,'tekute', interval=1)
13 #Vypocet hledanych dob
14 t_t=k_t1*tau_u_t1+k_t2*tau_u_t2
15 print("\nDoba experimentu pro tekute prostredi: "+str(t_t)+" s; "+str(round(t_t/60,3))+" min; "+str(round(t_t/3600,2))+" hod")
16
17 stop=time.time()
18 print('\nSpotrebovany cas celkove: '+str(stop-start))
```

## Příklad 2 (výsledek)



# Užitečné odkazy

- [https://medium.freecodecamp.org/  
an-a-z-of-useful-python-tricks-b467524ee747](https://medium.freecodecamp.org/an-a-z-of-useful-python-tricks-b467524ee747)
- [https://en.wikipedia.org/wiki/Zen\\_of\\_Python](https://en.wikipedia.org/wiki/Zen_of_Python)
- [https://docs.scipy.org/doc/numpy/user/  
numpy-for-matlab-users.html](https://docs.scipy.org/doc/numpy/user/numpy-for-matlab-users.html)

## Reference

-  <https://cs.wikipedia.org/wiki/Python>
-  <https://stackoverflow.com/questions/509211/understanding-pythons-slice-notation>
-  <https://python.cz/>
-  <https://stackoverflow.com/questions/509211/understanding-pythons-slice-notation>
- 