AKO NA

JUPYTER NOTEBOOKY

PETER GAŠPAR

3.4.2019

datalys

Jupyter notebooky

jupyter spectrogram (autosaved)

File Edit View Insert Cell Kernel Help Python 3

Markdown CellToolbar

Simple spectral analysis

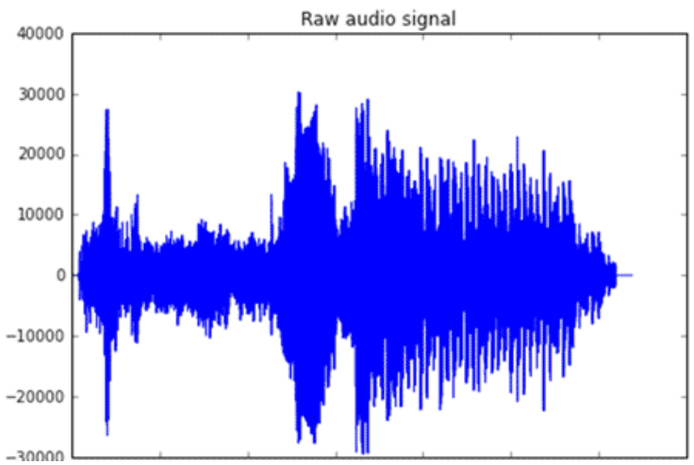
An illustration of the Discrete Fourier Transform

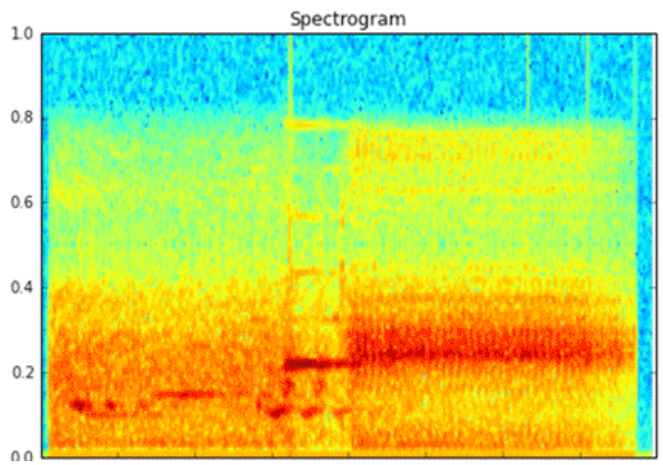
$$X_k = \sum_{n=0}^{N-1} x_n exp^{\frac{-2\pi i}{N} kn} \quad k = 0, \ldots, N-1$$

In [2]:
```
from scipy.io import wavfile
rate, x = wavfile.read('test_mono.wav')
```

And we can easily view it's spectral structure using matplotlib's builtin specgram routine:

In [5]:
```
fig, (ax1, ax2) = plt.subplots(1,2,figsize(16,5))
ax1.plot(x); ax1.set_title('Raw audio signal')
ax2.specgram(x); ax2.set_title('Spectrogram');
```





Magický IPython

Magické príkazy

```
In [1]: %lsmagic

Out[1]: Available line magics:
%alias  %alias_magic  %autocall  %automagic  %autosave  %bookmark  %ca
t  %cd  %clear  %colors  %config  %connect_info  %cp  %debug  %dhist
%dirs  %doctest_mode  %ed  %edit  %env  %gui  %hist  %history  %killbg
scripts  %ldir  %less  %lf  %lk  %ll  %load  %load_ext  %loadpy  %logo
ff  %logon  %logstart  %logstate  %logstop  %ls  %lsmagic  %lx  %macro
%magic  %man  %matplotlib  %mkdir  %more  %mv  %notebook  %page  %past
ebin  %pdb  %pdef  %pdoc  %pfile  %pinfo  %pinfo2  %popd  %pprint  %pr
ecision  %profile  %prun  %psearch  %psource  %pushd  %pwd  %pycat  %p
ylab  %qtconsole  %quickref  %recall  %rehashx  %reload_ext  %rep  %re
run  %reset  %reset_selective  %rm  %rmdir  %run  %save  %sc  %set_env
%store  %sx  %system  %tb  %time  %timeit  %unalias  %unload_ext  %who
%who_ls  %whos  %xdel  %xmode

Available cell magics:
%%!  %%HTML  %%SVG  %%bash  %%capture  %%debug  %%file  %%html  %%java
script  %%js  %%latex  %%perl  %%prun  %%pypy  %%python  %%python2  %%
python3  %%ruby  %%script  %%sh  %%svg  %%sx  %%system  %%time  %%time
it  %%writefile

Automagic is ON, % prefix IS NOT needed for line magics.
```


http://ipython.readthedocs.io/en/stable/interactive/magics.html

Zoznam env premenných

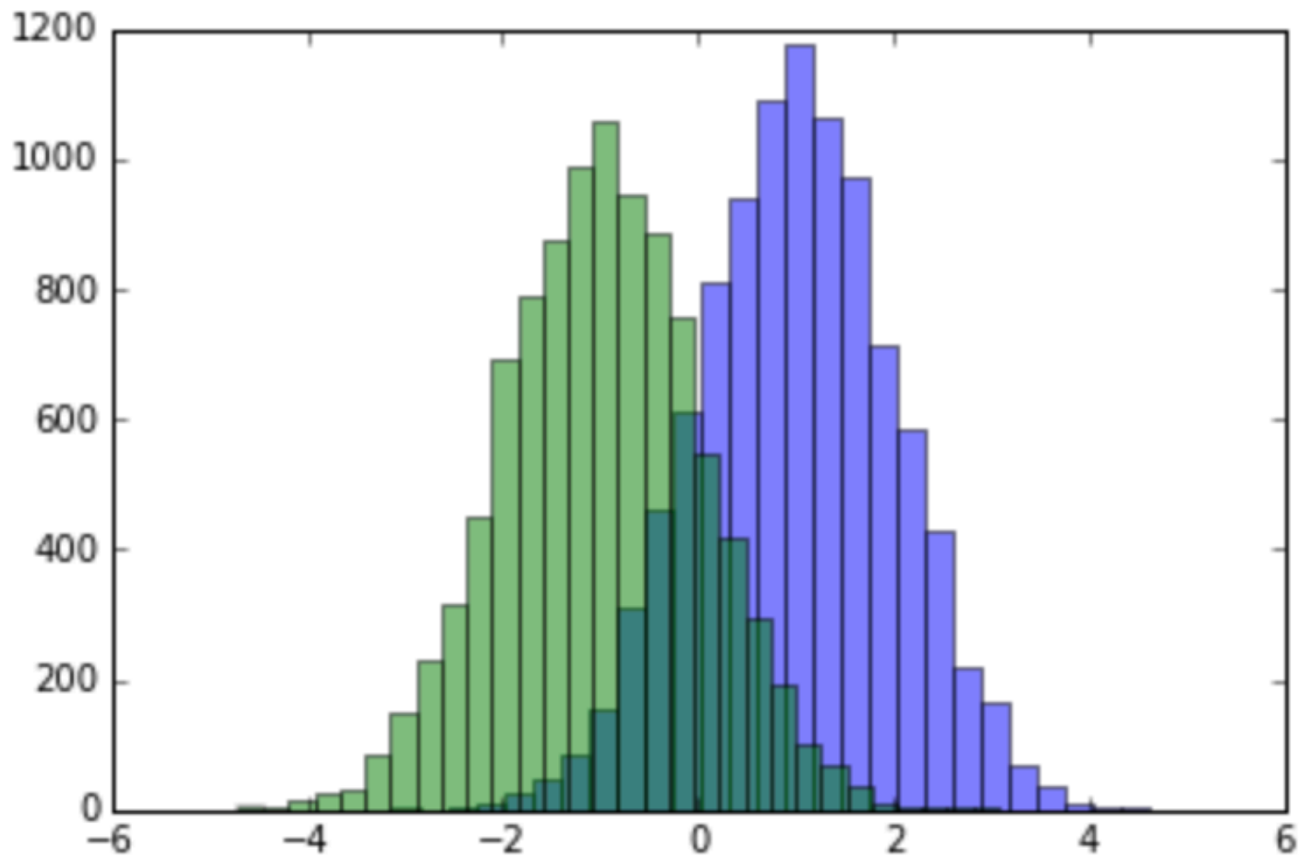
```
# Running %env without any arguments
# lists all environment variables

# The line below sets the environment
# variable
%env OMP_NUM_THREADS%env OMP_NUM_THREADS=4
```

```
env: OMP_NUM_THREADS=4
```

Vykonanie kódu

```
# this will execute and show the output from
# all code cells of the specified notebook
%run ./two-histograms.ipynb
```



Načítanie externého kódu

```
# Before Running
%load ./hello_world.py
```

```
# After Running
# %load ./hello_world.py
if __name__ == "__main__":
    print("Hello World!")
```

```
Hello World!
```

Prenos premenných

Notebook 1

```
data = 'this is the string I want to pass to different notebook'
%store data
del data # This has deleted the variable
```

```
Stored 'data' (str)
```

Notebook 2

```
%store -r data
print(data)
```

```
this is the string I want to pass to different notebook
```

Globálne premenné

```
one = "for the money"
two = "for the show"
three = "to get ready now go cat go"
%who str
```

```
one    three    two
```

Meranie času

```
%%time
import time
for _ in range(1000):
    time.sleep(0.01) # sleep for 0.01 seconds
```

```
CPU times: user 21.5 ms, sys: 14.8 ms, total: 36.3 ms
Wall time: 11.6 s
```

```
import numpy
%timeit numpy.random.normal(size=100)
```

```
The slowest run took 7.29 times longer than the fastest. This could mean tha
100000 loops, best of 3: 5.5 µs per loop
```

Meranie času 2

```
%prun some_useless_slow_function()
```

```
26324 function calls in 0.556 seconds

Ordered by: internal time

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
 10000    0.527    0.000    0.528    0.000 :2(append_if_not_exists)
 10000    0.022    0.000    0.022    0.000 {method 'randint' of 'mtrand.Rand
     1    0.006    0.006    0.556    0.556 :6(some_useless_slow_function)
  6320    0.001    0.000    0.001    0.000 {method 'append' of 'list' object
     1    0.000    0.000    0.556    0.556 :1()
     1    0.000    0.000    0.556    0.556 {built-in method exec}
     1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Pro
```

Debugovanie

```
%pdb

def pick_and_take():
    picked = numpy.random.randint(0, 1000)
    raise NotImplementedError()

pick_and_take()
```

```
Automatic pdb calling has been turned ON
```

Debugovanie

```
---------------------------------------------------------------------------
NotImplementedError                       Traceback (most recent call last)
 in ()
      5     raise NotImplementedError()
      6
----> 7 pick_and_take()

 in pick_and_take()
      3 def pick_and_take():
      4     picked = numpy.random.randint(0, 1000)
----> 5     raise NotImplementedError()
      6
      7 pick_and_take()

NotImplementedError:
```

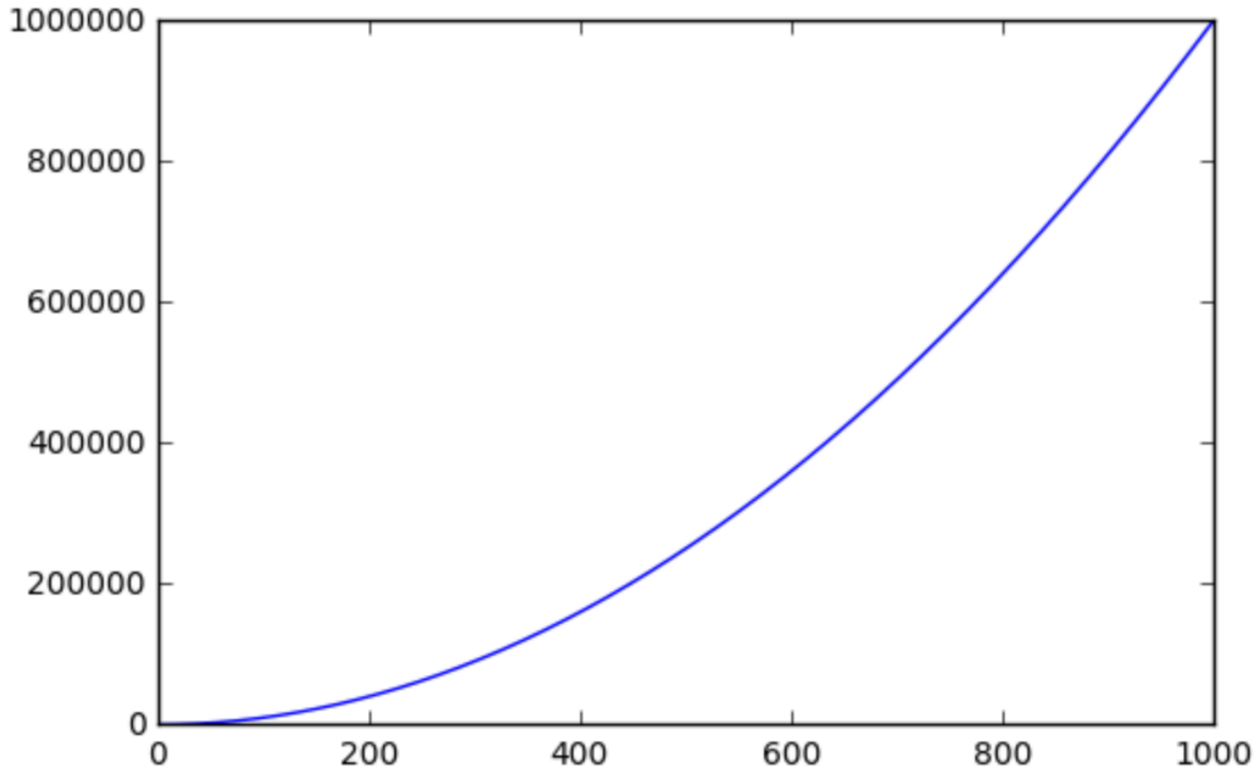
```
> (5)pick_and_take()
      3 def pick_and_take():
      4     picked = numpy.random.randint(0, 1000)
----> 5     raise NotImplementedError()
      6
      7 pick_and_take()
```

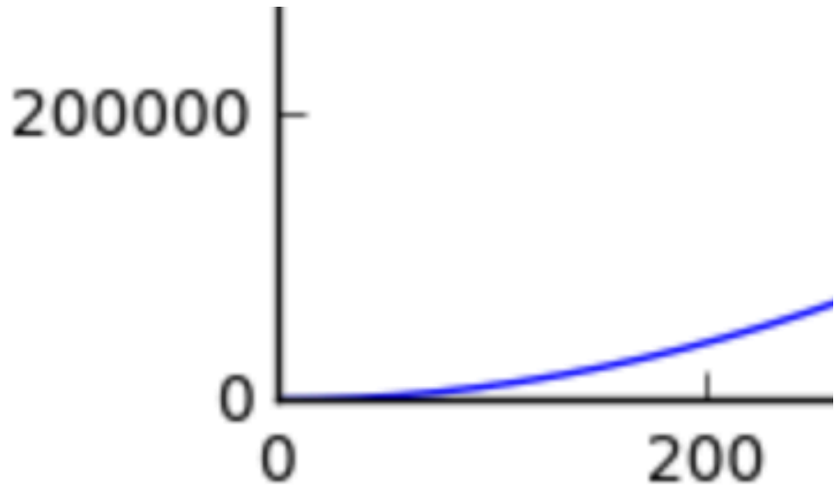
```
ipdb>
```

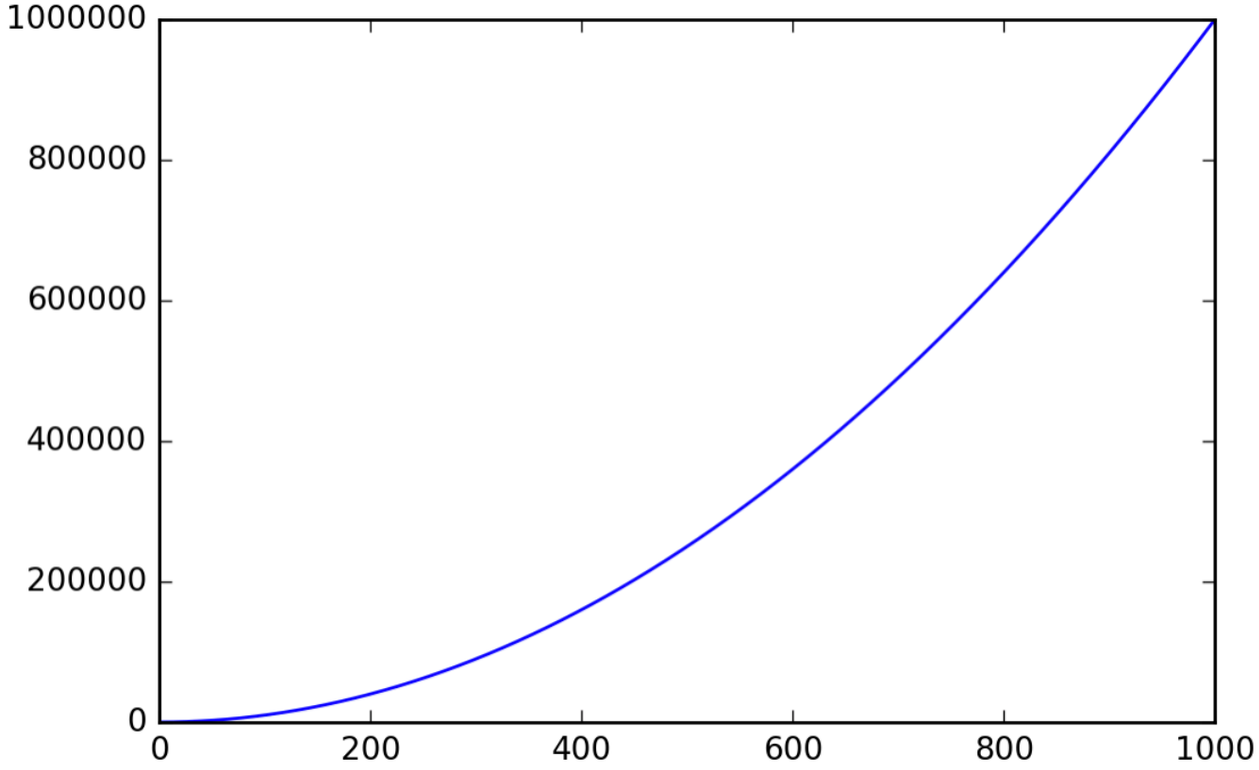
Grafy s vysokým rozlíšením

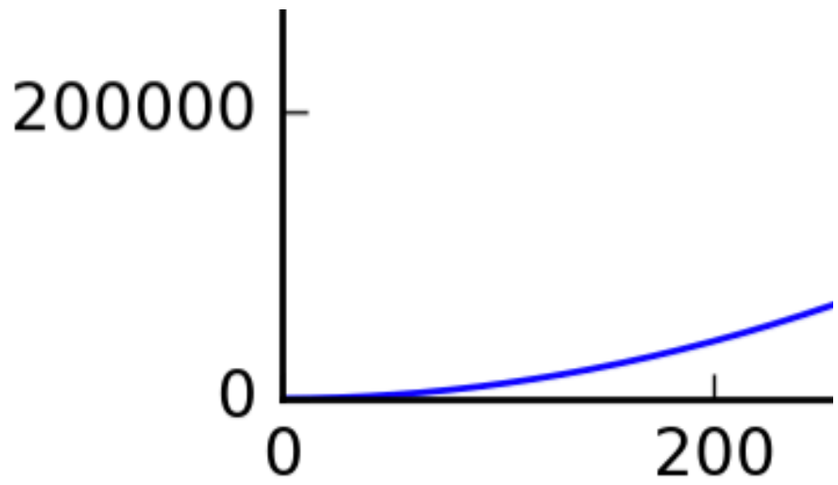
```
x = range(1000)
y = [i ** 2 for i in x]
plt.plot(x,y)
plt.show();
```

```
%config InlineBackend.figure_format ='retina'
plt.plot(x,y)
plt.show();
```


1000000
800000
600000
400000
200000
0
0
200
400
600
800
1000


200000
0
0
200


1000000
800000
600000
400000
200000
0
0
200
400
600
800
1000


200000
0
0
200

Zákaz výpisu poslednej premennej

```
a = 1

a
```

```
Out[1]: 1
```

```
a = 1

a;
```

```
(žiaden výstup)
```

Prístup na konzolu

```
!ls *.csv
```

```
nba_2016.csv                titanic.csv
pixar_movies.csv            whitehouse_employees.csv
```

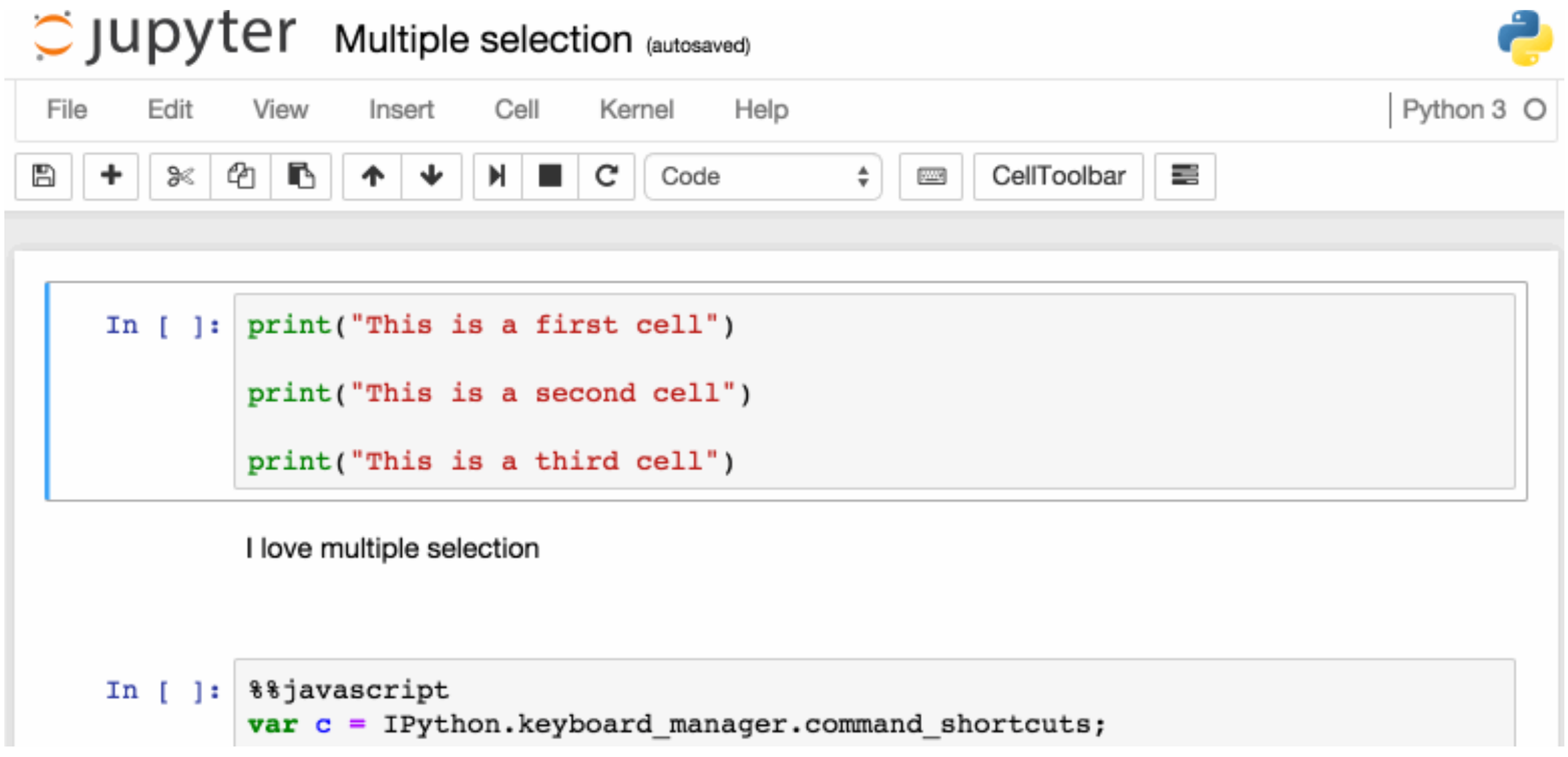
```
!pip install numpy
!pip list | grep pandas
```

```
Requirement already satisfied (use --upgrade to upgrade): numpy in /Library/
pandas (0.18.1)
```

Tipy a triky

datalys

Klávesové skratky



Klávesové skratky

ESC Prepnutie do *Command módu*.

A Pridanie novej bunky **nad** aktuálnu.

B Pridanie novej bunky **pod** aktuálnu.

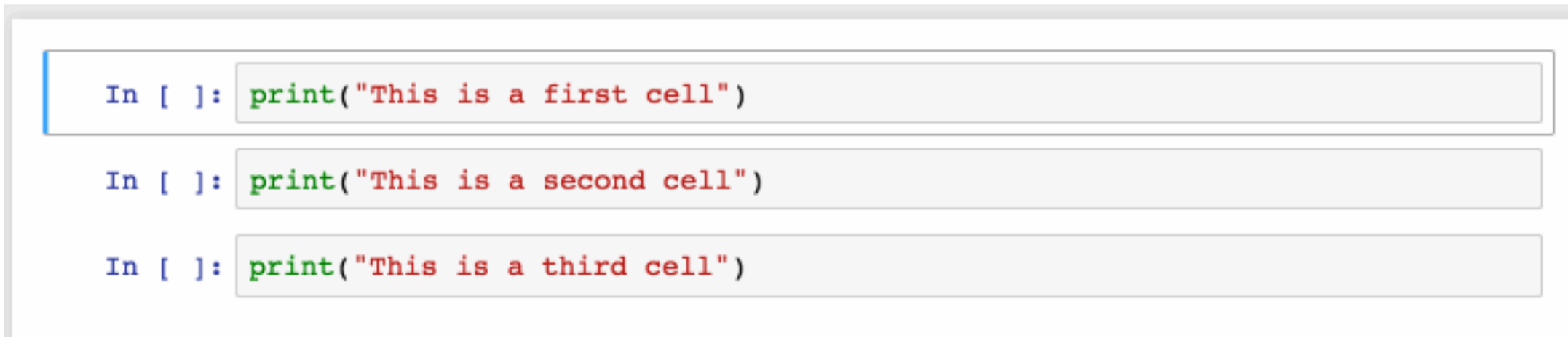
M Prepnutie bunky do režimu *Markdown*.

Y Prepnutie bunky do režimu *Kód*.

ENTER Prepnutie do *Edit módu*.

ESC + **O** Zobrazenie/skrytie výsledku.

Klávesové skratky



SHIFT + J

SHIFT + ▼

Vyberie nasledujúcu nižšiu bunku.

SHIFT + K

SHIFT + ▲

Vyberie nasledujúcu vyššiu bunku.

Pekný výpis pre viac premenných

```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

```
from pydataset import data
quakes = data('quakes')
quakes.head()
quakes.tail()
```

	lat	long	depth	mag	stations
1	-20.42	181.62	562	4.8	41
2	-20.62	181.03	650	4.2	15
3	-26.00	184.10	42	5.4	43
4	-17.97	181.66	626	4.1	19
5	-20.42	181.96	649	4.0	11

Pre trvalé zapnutie:

```
c = get_config()

# Run all nodes interactively
c.InteractiveShell.ast_node_interactivity = "all"
```

	lat	long	depth	mag	stations
996	-25.93	179.54	470	4.4	22
997	-12.28	167.06	248	4.7	35
998	-20.13	184.20	244	4.5	34
999	-17.40	187.80	40	4.5	14
1000	-21.59	170.56	165	6.0	119

Rýchla pomoc

V *Help* menu nájdete dokumentáciu pre mnohé známe knižnice.

alebo

```
?str.replace()
```

```
Docstring:
S.replace(old, new[, count]) -> str

Return a copy of S with all occurrences of substringold replaced by new.  If
Type:      method_descriptor
```

Kreslenie v notebookoch

matplotlib - (`%matplotlib inline`), tutoriál

Seaborn - postavené na Matplotlib, “krajšie”

bokeh - interaktívne grafy

plot.ly - pokročilé grafy

Altair - slabšia prispôsobiteľnosť

mpld3

Cachovanie

```
>>> from joblib import Memory
>>> cachedir = 'your_cache_dir_goes_here'
>>> mem = Memory(cachedir)
>>> import numpy as np
>>> a = np.vander(np.arange(3)).astype(np.float)
>>> square = mem.cache(np.square)
>>> b = square(a)
________________________________________________________________________________
[Memory] Calling square...
square(array([[0., 0., 1.],
       [1., 1., 1.],
       [4., 2., 1.]]))
_____________________________________________________________square - 0...s,

>>> c = square(a)
>>> # The above call did not trigger an evaluation
```

Experimentovanie

```
from random import shuffle
import numpy as np

sample = np.random.rand(1000)
sample2 = list(sample) + list(np.random.rand(1000))
shuffle(sample2)
```

```
%%timeit
x = np.intersect1d(sample, sample2)
```

```
6.71 ms ± 1.38 ms per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

```
sample_s = frozenset(sample)
sample2_s = frozenset(sample2)
```

```
%%timeit
x = [a for a in sample_s if a in sample2_s]
```

```
2.43 ms ± 413 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

Speciality

https://docs.google.com/document/d/1QK-WjyrRvwyidQeivRCWClBF2JSll1xlxNZe14zUHak/show

datalys

Jupyter notebooky

Ak ich začnete používať na zložitejšie výpočty, narazíte na viacero problémov:

- pre zobrazenie výsledku musíte (v niektorých prípadoch) ponechať notebook otvorený v prehliadači,
- náročnejšie výpočty môžu zatuhnúť a z notebooku neviete monitorovať ich priebeh,
- ak po dokončení práce s notebookom notebook neukončíte (Shutdown), zostane zaberať miesto v pamäti (RAM).

Spúšťanie výpočtov na pozadí

```
nohup python36 model_train.py &
```

alebo nástroj **Screen**

Začatie screen sedenia: `screen -S <login>`

Získanie pomoci: CTRL + A ?

Opustenie session: CTRL + A D

Obnovenie screen sedenia: `screen -r <login>`

Paralelizmus

```
from multiprocessing import Pool

def f(x):
    return x*x

if __name__ == '__main__':
    with Pool(5) as p:
        print(p.map(f, [1, 2, 3]))
```

```
# Výsledok
[1, 4, 9]
```

Upozornenia

```
import json
import requests

SLACK_URL = 'VASA_SLACK_WEBHOOK_URL'
SLACK_USERNAME = 'Môj skvelý skript'
SLACK_EMOJI = ':tada:'

def slack(msg):
    names = len([i for i, letter in enumerate(msg) if letter == '@'])
    slack_data = {'text': msg, 'username': SLACK_USERNAME, 'icon_emoji':
SLACK_EMOJI, "link_names": names }
    requests.post(SLACK_URL, data=json.dumps(slack_data), headers={'Content-Type':
'application/json'})
```

```
presnost = 1.0
slack(f'Vypocet sa skoncit. Priemerna presnost: {presnost}')
```

Ukladanie medzivýsledkov

pickle - https://docs.python.org/3.7/library/pickle.html

```
import pickle
import numpy as np

moje_pole = np.array([1, 2, 3, 4, 5])
moje_pole2 = moje_pole * 2

# Uloženie medzivýsledku.
pickle.dump(moje_pole2, open('subor.pickle', 'wb'))


# Načítanie medzivýsledku
moje_pole3 = pickle.load(open('subor.pickle', 'rb'))
print(moje_pole3)
```

Rady na záver

Analyzujte dáta v Jupyter notebookoch. Tie sú priam na to stvorené.

Implementujte svoju metódu, skúšajte ju s rôznymi parametrami, ideálne na podmnožine datasetu.

Keď si budete istí, že je všekto v poriadku, vykonajte trénovanie/predikciu/testovanie tak, ako ste si pôvodne zaumienili, ale použite na to **samostatné súbory**, ktoré **budete spúšťať z konzoly** (SSH) a nie z Jupyter Notebooku.

Referencie

Spracované podľa:

https://www.dataquest.io/blog/jupyter-notebook-tips-tricks-shortcuts/