Vizualizácia dát

Matej Oravec

BAB37ZPR

10. týždeň

Dnešná prednáška

Obsah

1. Všeobecný úvod
2. Matplotlib
   - Tutoriál: Graf funkcie sínus
   - Tutoriál: Covid-19 – počet vykonaných testov v ČR
   - Tutoriál: Histogram
   - Tutoriál: Animácia
3. Záver

Dnešná prednáška

Motivácia, cieľ

- Prečo vizualizovať dáta?
  - Zisk komplexnejšej predstavy o ich obsahu
  - Prezentácia dát/výsledkov

Čo sa naučíme

- Získame prehľad o vizualizačných balíčkoch pre *Python*
- Vytvárať základné typy grafov pomocou *Matplotlib*

Čo sa nenaučíme

- Získavať a spracovávať dáta, ktoré chceme vizualizovať
  - Očakávame ich už „nachystané“

Všeobecný úvod

Možnosti vizualizácie v *Pythone*

Matplotlib

- Najbežnejší a najpoužívanejší balíček
- Jednoduchý na používanie, práca v ňom často podobná ako v *Matlabe*
- Podpora animácie
- Môže byť problém s rýchlosťou

pyqtgraph

- Cielený na rýchlosť a integráciu s *PyQt* aplikáciami
- API veľmi podobné *PyQt*
- Vhodné na veľmi rýchle animácie

Integrovaná vizualizácia v *seaborn*, *pandas*, *sympy*,...

- Vo väčšine prípadov založené na *Matplotlib*
- Zamerané hlavne na pohodlnosť pri práci so spomínanými balíčkami

Matplotlib

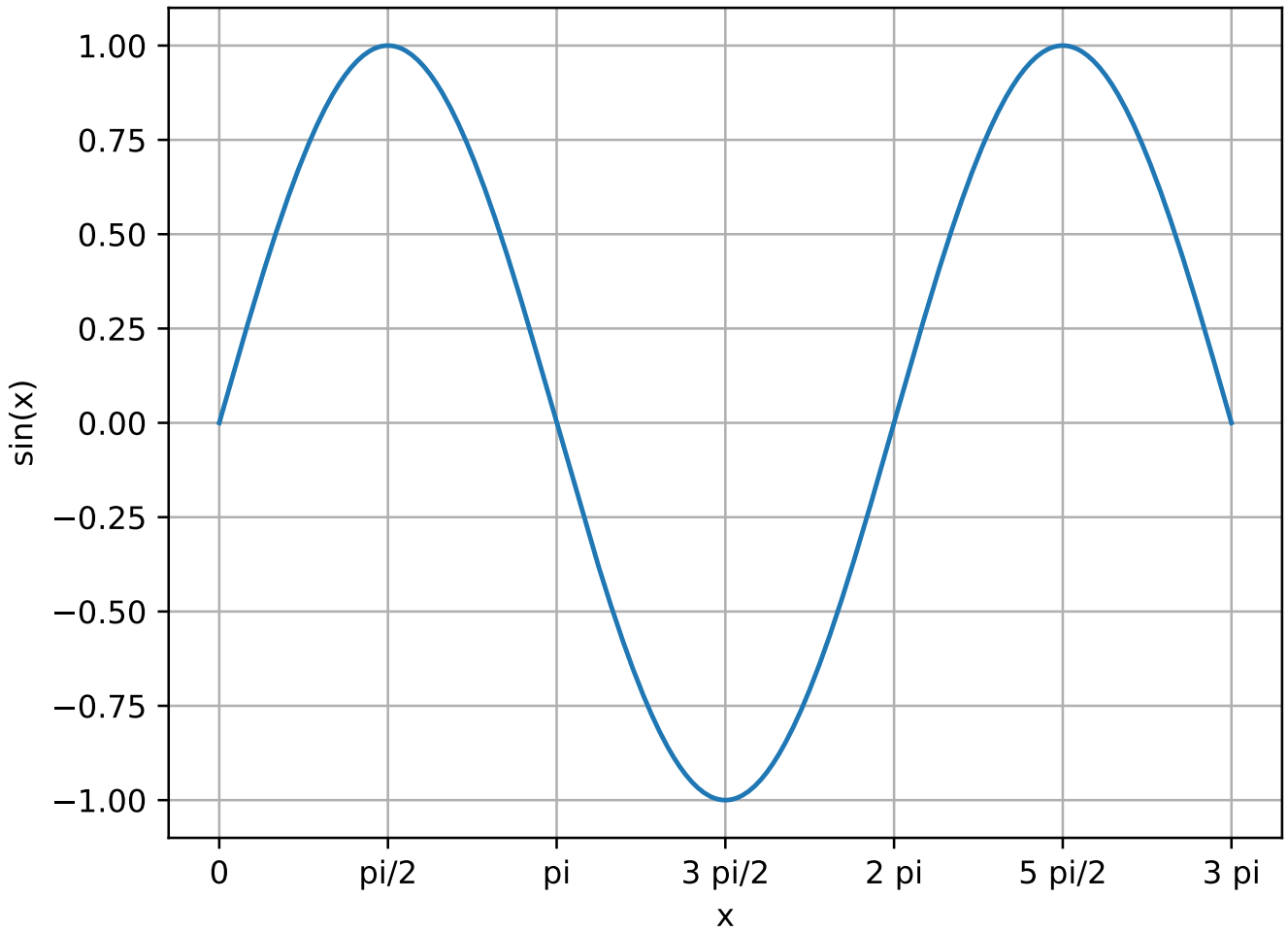
Matplotlib

Matplotlib

Tutoriál: Graf funkcie sínus

Tutoriál: Graf funkcie sínus

Cieľ



Tutoriál: Graf funkcie sínus

Štandardné načítanie balíčkov

- *Numpy* nie je nutné, ale veľmi odporúčané
- `matplotlib.pyplot` je bežne skracovaný na `plt`

```
1 import numpy as np
2 import matplotlib.pyplot as plt
```

Tutoriál: Graf funkcie sínus

Príprava dát

- *Numpy* nepracuje so symbolickou matematikou
  - V 2D grafe sú vykreslené body tvaru (x, y), kde x je hodnota nezávislej premennej a y hodnota závislej premennej
  - V našom prípade $y = \sin(x)$
- Príprava dát pre vykreslenie priebehu funkcie vo všeobecnosti prebieha v dvoch bodoch:
  1. Príprava delenia požadovaného intervalu hodnôt nezávislej premennej
  2. Výpočet funkčných hodnôt v bodoch tohoto delenia

```
3 x = np.linspace(0, 3 * np.pi, 500)
4 y = np.sin(x)
```

Tutoriál: Graf funkcie sínus

Triedy `Figure` a `Axes`

- Objekt triedy **Figure** tvorí najvyšší level ▸ Docs
  - „Okienko,“ v ktorom sa budú nachádzať osi grafu
  - Špecifikujeme jeho veľkosť, farbu, hrúbku lemu, nadpis,...
- Objekt triedy **Axes** tvorí kontajner na priamu grafickú reprezentáciu dát ▸ Docs
  - Obsahuje čiary, body, osové značky, popisy osí, mriežku,...
  - Jeden *figure* môže obsahovať viacero objektov *axes*
- Vytvorenie páru *figure* – *axes*:

```
5 fig, ax = plt.subplots()
```

Tutoriál: Graf funkcie sínus

Umiestnenie dát do *axes*

- `ax.plot()` → Spojenie bodov rovnou čiarou ▸ Docs
- `ax.set_xticks()` → Značky na vodorovnej osi ▸ Docs
- `ax.set_xticklabels()` → Popisy značiek na vodorovnej osi ▸ Docs
- `ax.set_xlabel()` → Popis horizontálnej osi ▸ Docs
- `ax.set_ylabel()` → Popis vertikálnej osi ▸ Docs
- `ax.grid()` → Mriežka ▸ Docs

```
6  ax.plot(x, y)
7  ax.set_xticks([k * np.pi / 2 for k in range(2 * 3 + 1)])
8  ax.set_xticklabels(['0', 'pi/2', 'pi', '3 pi/2', '2 pi', '5 pi/2', '
       3 pi'])
9  ax.set_xlabel('x')
10 ax.set_ylabel('sin(x)')
11 ax.grid()
```

Tutoriál: Graf funkcie sínus

Celý kód

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4
5  x = np.linspace(0, 3 * np.pi, 500)
6  y = np.sin(x)
7
8  fig, ax = plt.subplots()
9  ax.plot(x, y)
10 ax.set_xticks([k * np.pi / 2 for k in range(2 * 3 + 1)])
11 ax.set_xticklabels(['0', 'pi/2', 'pi', '3 pi/2', '2 pi', '5 pi/2', '
       3 pi'])
12 ax.set_xlabel('x')
13 ax.set_ylabel('sin(x)')
14 ax.grid()
15
16 plt.show()
```

Tutoriál: Graf funkcie sínus

Zopár poznámok

- Okamžité zobrazenie okna s grafom: funkcia `plt.show()`
  - V *Jupyter notebook* je možné využiť magickú funkciu `%matplotlib inline`
- Grafy nie je nutné zobrazovať, možno ich priamo ukladať ▸ Docs
  - Fukcia `plt.savefig()`
- Publikovanie grafov (napr. do bakalárky): *tikzplotlib* ▸ PyPI
  - Preklad grafu do *pgfplots*
  - Vhodné pre tých, ktorým je bližšie API *matplotlib* než vytváranie grafov priamo v *pgfplots*
  - Funkcia `tikzplotlib.save()` (výstup nutné ešte skompilovať *pdfLaTeX*-om)

Matplotlib

Tutoriál: Covid-19 – počet vykonaných testov v ČR

Tutoriál: Covid-19 – počet vykonaných testov v ČR

Dáta

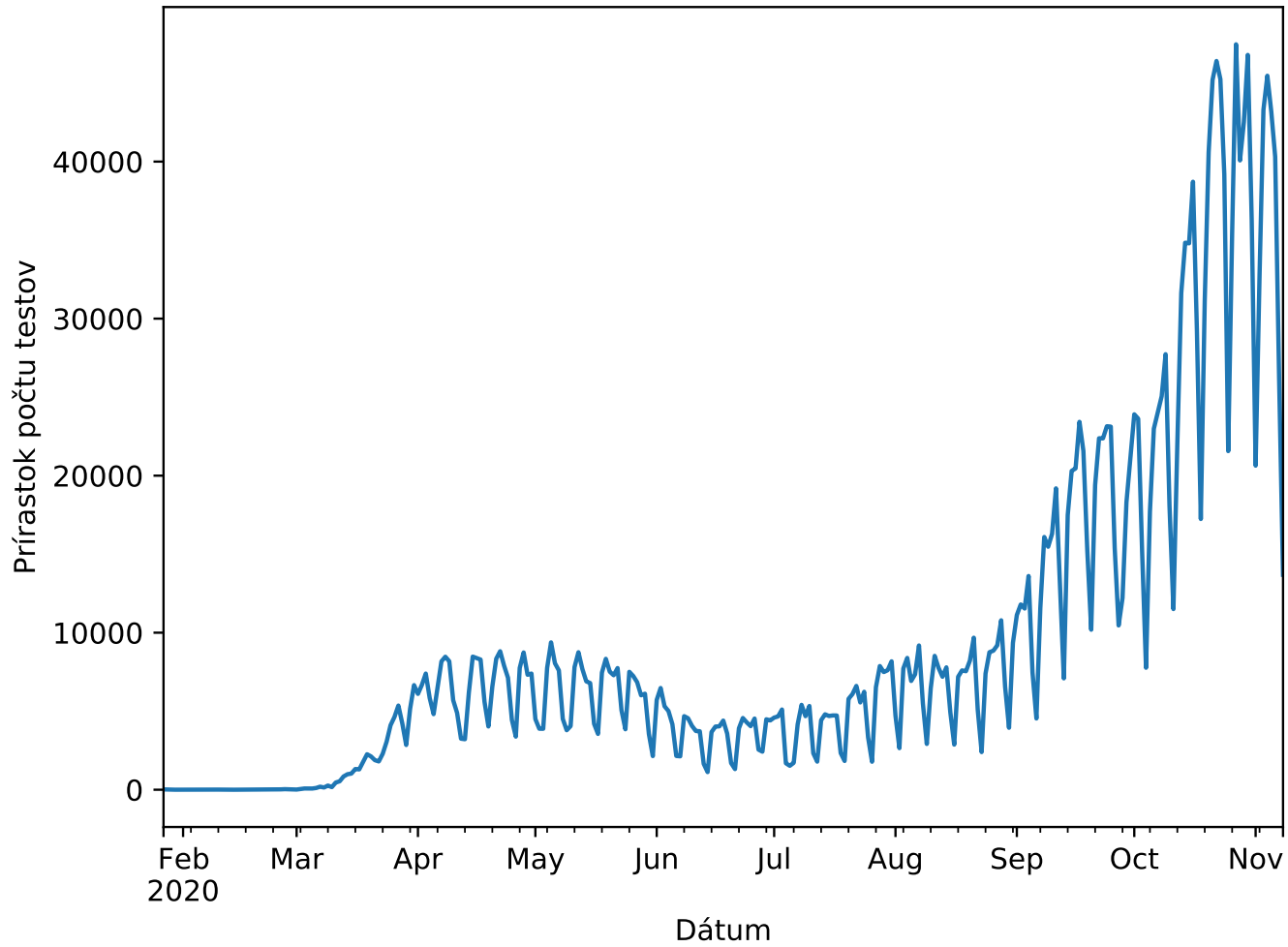
- Verejne dostupné dáta zo stránky `https://onemocneni-aktualne.mzcr.cz/api/v2/covid-19`

```
datum,prirustkovy_pocet_testu,kumulativni_pocet_testu
2020-01-27,20,20
2020-01-28,8,28
2020-01-29,5,33
2020-01-30,1,34
2020-01-31,3,37
2020-02-01,1,38
2020-02-02,5,43
2020-02-03,5,48
2020-02-04,5,53
2020-02-05,0,53
2020-02-06,3,56
2020-02-07,6,62
...
```

Tutoriál: Covid-19 – počet vykonaných testov v ČR

Cieľ

- Vizualizácia časovej postupnosti z textového súboru



Tutoriál: Covid-19 – počet vykonaných testov v ČR

Štandardné načítanie balíčkov

- Dáta pripravíme v *pandas*
- Na vytvorenie grafu použijeme `pandas.Series.plot()` ▸ Docs

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
```

Tutoriál: Covid-19 – počet vykonaných testov v ČR

Načítanie dát

- Použijeme funkciu `pd.read_csv()` ▸ Docs
  - Netreba špecifikovať separátor ani hlavičku – dáta už sú v očakávanom formáte (oddelené čiarkou, v prvom riadku názvy stĺpcov, od druhého riadku dáta)
- Dáta obsahujú časové údaje → využijeme *časový index*

```
4 data = pd.read_csv('testy.csv')
5 data.index = pd.to_datetime(data['datum'])
6 data.drop(columns=['datum'], inplace=True)
```

Tutoriál: Covid-19 – počet vykonaných testov v ČR

Vytvorenie grafu

- `pandas.DataFrame` i `pandas.Series` majú implementovanú metódu `plot()`
  - Metóda vracia objekt triedy `plt.Axes`
  - Umožňuje pohodlne vykresľovať časovo indexované dáta

```
7 ax = data['prirustkovy_pocet_testu'].plot(figsize=(7, 5))
8 ax.set_xlabel('Dátum')
9 ax.set_ylabel('Prírastok počtu testov')
```

Tutoriál: Covid-19 – počet vykonaných testov v ČR

Celý kód

```
1  import numpy as np
2  import pandas as pd
3  import matplotlib.pyplot as plt
4
5
6  data = pd.read_csv('testy.csv')
7  data.index = pd.to_datetime(data['datum'])
8  data.drop(columns=['datum'], inplace=True)
9
10 ax = data['prirustkovy_pocet_testu'].plot(figsize=(7, 5))
11 ax.set_xlabel('Dátum')
12 ax.set_ylabel('Prírastok počtu testov')
13
14 plt.show()
```

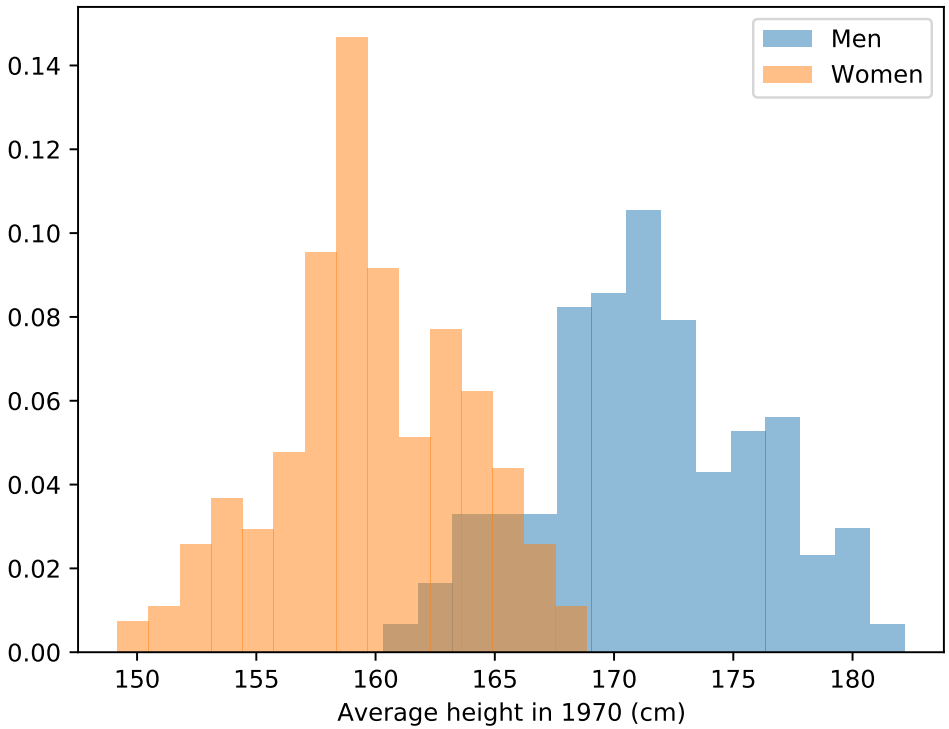
Matplotlib

Tutoriál: Histogram

Tutoriál: Histogram

Cieľ

- Na základe dát z `https://ourworldindata.org/human-height` vytvoriť graf, obsahujúci:
  - Histogram výšky priemerného muža v roku 1970
  - Histogram výšky priemernej ženy v roku 1970



Tutoriál: Histogram

Štruktúra dát

- Dva `.csv` súbory – jeden pre mužov, druhý pre ženy
- Pre každú krajinu a rok obsahujú priemernú výšku entity v centimetroch

```
Entity,Code,Year,Mean male height (cm)
Afghanistan,AFG,1896,161.1640953
Afghanistan,AFG,1897,161.196286
Afghanistan,AFG,1898,161.2282966
Afghanistan,AFG,1899,161.2607274
Afghanistan,AFG,1900,161.2930682
Afghanistan,AFG,1901,161.325492
Afghanistan,AFG,1902,161.3583553
Afghanistan,AFG,1903,161.3912147
...
```

Tutoriál: Histogram

Štandardné načítanie balíčkov, príprava dát

- Ako v predchádzajúcom prípade – dáta načítame a pripravíme pomocou *pandas*, vykresľujeme pomocou *matplotlib.pyplot*

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
```

- Oba súbory načítame pomocou `pd.read_csv()`
- Výber užitočných dát pomocou logického indexovania

```
4 data_m = pd.read_csv('average-height-of-men.csv')
5 data_f = pd.read_csv('average-height-of-women.csv')
6
7 data_m = data_m['Mean male height (cm)'][data_m['Year'] == 1970]
8 data_f = data_f['Mean female height (cm)'][data_f['Year'] == 1970]
```

Tutoriál: Histogram

Vykreslenie grafu

- Počet binov:
  a) Odhadom podľa dát (vyžaduje bližšie preskúmanie)
  b) Výpočtom podľa jednej z odporúčaných metód
     - ⋆ Odmocninová → bežná, dobrá pre menšie datasety s distribúciou blízkou normálnej
     - ⋆ Scottova → vhodná pre väčšie datasety s distribúciou blízkou normálnej
- Využijeme funkciu `plt.Axes.hist()` ▸ Docs

```
9  fig, ax = plt.subplots()
10 ax.hist(data_m, bins=15, density=True, alpha=.5, label='Men')
11 ax.hist(data_f, bins=15, density=True, alpha=.5, label='Women')
12 ax.set_xlabel('Average height in 1970 (cm)')
13 plt.legend()
```

Tutoriál: Histogram

Celý kód

```
1  import numpy as np
2  import pandas as pd
3  import matplotlib.pyplot as plt
4
5
6  data_m = pd.read_csv('average-height-of-men.csv')
7  data_f = pd.read_csv('average-height-of-women.csv')
8
9  data_m = data_m['Mean male height (cm)'][data_m['Year'] == 1970]
10 data_f = data_f['Mean female height (cm)'][data_f['Year'] == 1970]
11
12 fig, ax = plt.subplots()
13 ax.hist(data_m, bins=15, density=True, alpha=.5, label='Men')
14 ax.hist(data_f, bins=15, density=True, alpha=.5, label='Women')
15 ax.set_xlabel('Average height in 1970 (cm)')
16 plt.legend()
17
18 plt.show()
```

Matplotlib

Tutoriál: Animácia

Tutoriál: Animácia

Cieľ

Prečo animovať

- Často je časový vývoj dát kľúčom k ich lepšiemu pochopeniu
- Ak je potrebné sledovať vývoj viacrozmerného grafu v čase, môže pomôcť animácia
  - Časové rezy („dlaždice“) sú v mnohých prípadoch neprehľadné

- Vytvoríme animovanú guľôčku, putujúcu po grafe funkcie sínus *(ukážka na prednáške)*

Tutoriál: Animácia

Štandardné načítanie balíčkov, inicializácia

- API pre tvorbu animácií je zahrnuté v balíčku `matplotlib.animation`

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import matplotlib.animation as animation
```

- Pri inicializácii si pripravíme objekty *figure* a *axes*
- Do nich vykreslíme počiatočný stav (možno vykresliť prázdne dáta)

```
4 x = np.linspace(0, 4 * np.pi, 500)
5 y = np.sin(x)
6
7 fig, ax = plt.subplots()
8 ax.plot(x, y)
9 dot, = ax.plot(0, 0, '.', c='r', ms=20)
```

- Objekty, ktoré budú animované, si uložíme do premenných

Tutoriál: Animácia

Vytvorenie generátoru dát

- Animácia je zložená z množstva za sebou nasledujúcich statických obrázkov. Základná slučka animácie je
  1. Získať nové dáta
  2. Aktualizovať na základe nich obrázok (volanie *update* funkcie)
- Nové dáta ako argument *update* funkcie môžu byť špecifikované
  - Iterovateľným objektom
  - Generátorom
- V našom príklade použijeme generátor

```
10 def data_generator():
11     for i in range(1, np.size(x)):
12         yield x[i], y[i]
```

Tutoriál: Animácia

Definícia update funkcie, vytvorenie animácie

- Definuje nový obrázok s použitím nových dát
- Vracia *iterovateľný objekt*, zahŕňajúci všetky prvky grafu, ktoré sú pri animácii pozmeňované
  - V našom prípade ide o jediný objekt – `dot` triedy `matplotlib.lines.Line2D`

```
13 def update(data):
14     x, y = data
15     dot.set_data(x, y)
16 
17     return dot,
```

Tutoriál: Animácia

Definícia update funkcie, vytvorenie animácie

- Definuje nový obrázok s použitím nových dát
- Vracia *iterovateľný objekt*, zahŕňajúci všetky prvky grafu, ktoré sú pri animácii pozmeňované
  - V našom prípade ide o jediný objekt – `dot` triedy `matplotlib.lines.Line2D`

```
13 def update(data):
14     x, y = data
15     dot.set_data(x, y)
16 
17     return dot,
```

- `matplotlib.animation` ponúka niekoľko typov animácie, v našom prípade použijeme `animation.FuncAnimation`
  - Definovaná objektom *figure*, a update funkciou (+ voliteľné parametre ako napr. zdroj dát)

```
18 ani = animation.FuncAnimation(fig, update, data_generator, interval
       =1, blit=True)
```

Tutoriál: Animácia

Celý kód

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  import matplotlib.animation as animation
4
5
6  x = np.linspace(0, 4 * np.pi, 500)
7  y = np.sin(x)
8
9  fig, ax = plt.subplots()
10 ax.plot(x, y)
11 dot, = ax.plot(0, 0, '.', c='r', ms=20)
```

Tutoriál: Animácia

Celý kód

```
12 def data_generator():
13     for i in range(1, np.size(x)):
14         yield x[i], y[i]
15 
16 def update(data):
17     x, y = data
18     dot.set_data(x, y)
19 
20     return dot,
21 
22 ani = animation.FuncAnimation(fig, update, data_generator, interval
       =1, blit=True)
23 
24 plt.show()
```

Záver

Záverečné poznámky

3D grafy

- Možné pomocou rozšírenia `mpl_toolkits.mplot3d` ▸ Docs
- API zachováva princípy a nomenklatúru platné pre 2D grafy

Pyqtgraph

- Zamerané na rýchlosť a integráciu s *PyQt* aplikáciami
- V prípade záujmu alebo potreby mrknite na: ▸ Domovská stránka projektu

Seaborn

- Balíček na štatistickú analýzu dát ▸ Domovská stránka projektu
- Ponúka rôzne zaujímavé datasety a nástroje na vizualizáciu. Grafy sú založené na *matplotlib*
  - Violin plot, beeswarm plot,...
- Okrem iného zahŕňa niekoľko grafických „tém“

Ďakujem za pozornosť.