

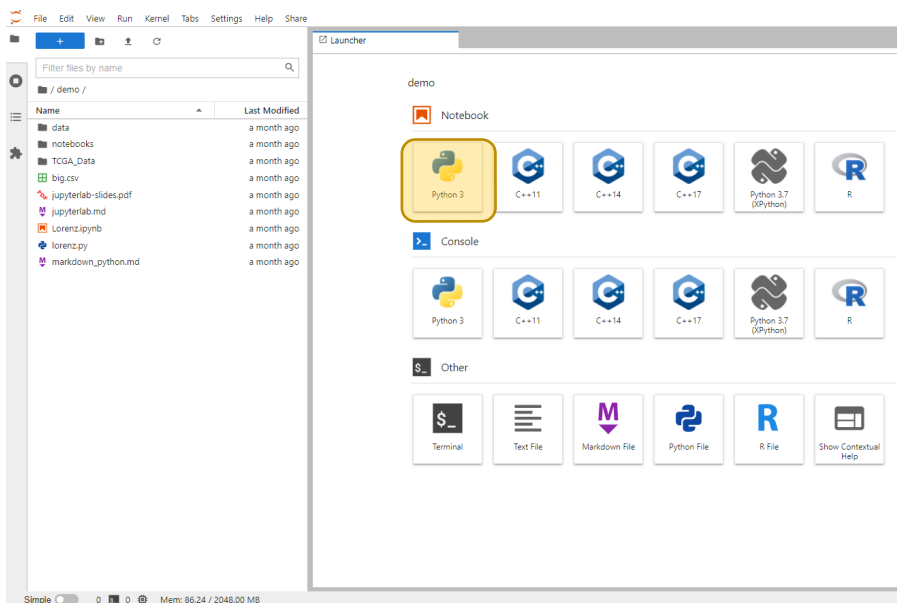
Jednoduché grafy

Návrh řešení:

V úvodu si zmíníme, že následující screenshoty jsou vytvořeny v online prostředí Jupyter notebooku, který je možné nainstalovat také jako aplikaci stolního počítače. Pro naše účely je plně dostačující online prostředí, které je možné spustit všude, kde je k dispozici internet, tedy bez nutnosti instalace a to se může často hodit. Prostorů naleznete na adrese <https://jupyter.org/try>, kde následně zvolíme možnost „JupyterLab“.



Obr. 1 – vstup do JupyterLabu



Obr. 2 - Online prostředí JupyterLab

Prostředí je samozřejmě možné zvolit dle svých priorit a používat např. PyCharm. Jupyter je zde zmíněn především jako dobře fungující alternativa.

Nyní již přejdeme k vykreslení našeho prvního grafu. V úplném počátku našeho kódu je potřebné importovat požadované knihovny, případně jejich moduly. Na obrázku níže vidíme příklad krátkého kódu, jehož výsledkem je vykreslení funkce sinus.

```
# První příklad: vykreslení průběhu funkce sin

import numpy as np
import matplotlib.pyplot as plt

# hodnoty ose x
x = np.linspace(0, 2*np.pi, 100)

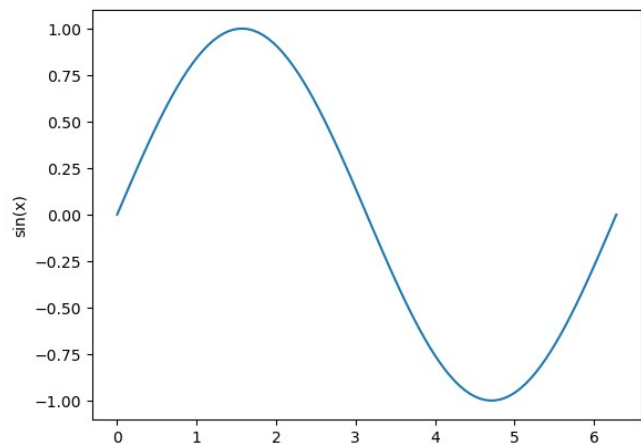
# hodnoty na ose y
y = np.sin(x)

# vykreslí průběh funkce
plt.plot(x, y)

# popis os
plt.xlabel("x")
plt.ylabel("sin(x)")

# zobrazení grafu
plt.show()
```

Obr. 3 - kód vykreslení $\sin(x)$



Obr. 4 - náš první graf

Z kódu je patrný import knihovny numpy, která slouží k práci s vektory, maticemi a vícerozměrnými poli. Dále nabízí také mnoho matematických funkcí. Dále importujeme modul pyplot ze stěžejní knihovny matplotlib. Ostatní řádky máme srozumitelně popsány v rámci těla kódu, není tedy třeba jejich další rozbor.

V dalším kroku si ukážeme další možnosti, kterými lze graf vylepšit. Nyní si necháme vykreslit do grafu 2 funkce, doplníme graf o mřížku, legendu a popisky.

```

import numpy as np
import matplotlib.pyplot as plt

# hodnoty ose x
x = np.linspace(0, 2*np.pi, 100)

# hodnoty ose y (první funkce)
y1 = np.sin(x)

# hodnoty ose y (druhá funkce)
y2 = np.cos(x)

# vykreslí průběh obou funkcí se změnou stylu vykreslování
plt.plot(x, y1, "b-", label="sin")
plt.plot(x, y2, "r-", label="cos")

# přidání legendy
plt.legend(loc="lower left")

# nastavení rozsahů na obou osách
plt.axis([-1, 8, -1.5, 1.5])

# povolení zobrazení mřížky
plt.grid(True)

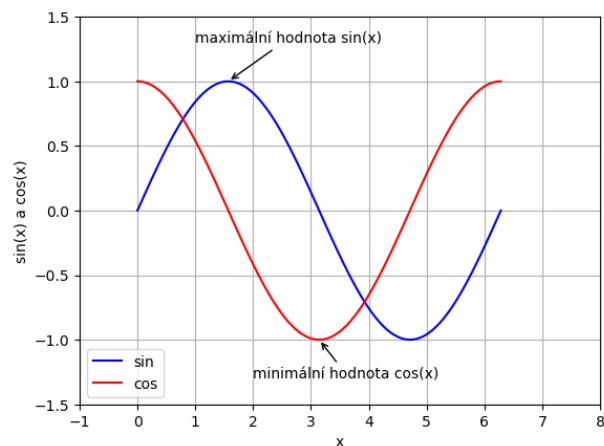
# popis os
plt.xlabel("x")
plt.ylabel("sin(x) a cos(x)")

# první popis grafu
plt.annotate("maximální hodnota sin(x)",
            xy=(np.pi/2, 1.0),
            xytext=(1, 1.3),
            arrowprops=dict(arrowstyle="->"))

# druhý popis grafu
plt.annotate("minimální hodnota cos(x)",
            xy=(np.pi, -1.0),
            xytext=(2, -1.3),
            arrowprops=dict(arrowstyle="->"))

# zobrazení grafu
plt.show()

```



Obr. 6 - druhý "vylepšený" graf

Obr. 5 - kód vykreslení dvou funkcí s popisky

Z kódu je patrné použití dalších atributů, které knihovna matplotlib nabízí. Zdaleka zde nejsou použity všechny možnosti, které se nám nabízí. Účelem je demonstrovat možnosti rozšíření výsledného grafu, ve srovnání s prvním vytvořeným grafem. V případě zájmu doporučuji navštívit odkazy použitých zdrojů, kde nalezneme podrobnější popis problematiky, který by obsahově výrazně překračoval zamýšlený rozsah tohoto protokolu. Jako další příklad si uvedeme vykreslení hojně používaného sloupcového grafu. Pro jeho zobrazení používáme funkci **matplotlib.pyplot.bar()**, případně **matplotlib.pyplot.barh()**. První uvedený vykresluje svislé sloupce, druhý pak horizontální. V rámci funkce **pyplot.bar()** je možné opět využít nastavení mnoha parametrů, které ovlivní výsledné zobrazení. Nejlépe bude vše pochopitelné opět na níže uvedeném příkladu.

```

import numpy as np
import matplotlib.pyplot as plt

# pole s cenami ropy
cena_ropy = [
    46.68, 44.68, 46.90, 47.15, 44.59, 44.00, 44.63, 45.92, 44.15, 45.94,
    46.05, 46.75, 46.25, 45.41, 49.20, 45.22, 42.56, 38.60, 39.31, 38.24,
    40.45, 41.32, 40.80, 42.62, 41.87, 42.50, 42.23, 43.30, 43.08, 44.96,
    43.87, 44.66, 45.15, 47.12, 48.52, 48.79, 47.98, 47.39, 48.14, 48.45]

# načtení počtu prvků do proměnné N
N = len(cena_ropy)

# indexy prvků
indexes = np.arange(N)

# šířka sloupců
width = 1.00

# sloupcový graf
# první parametr - osa x (počet prvků)
# druhý parametr - osa y (cena ropy > výška sloupce)
plt.bar(indexes, cena_ropy, width, color='yellow', edgecolor='black',
        label='Cena ropy')

# povolení zobrazení mřížky
plt.grid(True)

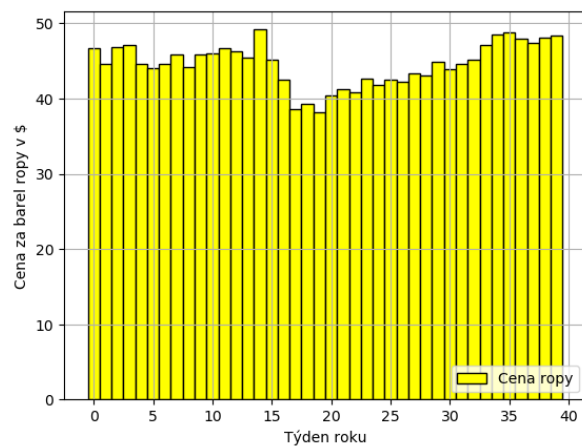
# popis os
plt.xlabel("Týden roku")
plt.ylabel("Cena za barel ropy v $")

# legenda dole vpravo
plt.legend(loc="lower right")

# zobrazení grafu
plt.show()

```

Obr. 7 - kód pro tvorbu sloupcového grafu



Obr. 8 - sloupcový graf s cenami ropy

Jak je patrné z výše uvedeného kódu, není příliš komplikované pomocí námi používané knihovny ze zadaných dat vytvořit obrazové vyjádření hodnot. Opět pouze zmíním, že typů grafů nabízí matplotlib nepřeberné množství a jistě nalezneme vždy vhodný typ pro požadované použití.

Výsledky a fakta:

Dosud byly použity k vykreslení matematické funkce ve spolupráci s knihovnou numpy, případně manuálně zadaná data v rámci pole. Výsledkem tohoto protokolu bude v následující části graf, který bude vytvořen ze vzdáleného souboru, který bude stažen a následně vybraná obsažená data zobrazena do grafu.

```

import matplotlib.pyplot as plt
import os
import pandas as pd
#from urllib.request import urlretrieve

# Seznam souborů (viz níže)
zdroje = [
    # Covid data
    "https://onemocneni-aktualne.mzcr.cz/api/v2/covid-19/incidence-7-14-cr.csv",
]

for zdroj in zdroje:
    # Pouze poslední část cesty adresy datového zdroje je jeho jméno
    jmeno = zdroj.rsplit("/")[-1]

    if not os.path.exists(jmeno):
        print(f"Soubor {jmeno} ještě není stažen, jdeme na to...")
        urlretrieve(url=zdroj, filename=jmeno)
        print(f"Soubor {jmeno} úspěšně stažen.")
    else:
        print(f"Soubor {jmeno} už byl stažen, použijeme místní kopii.")
print("Všechny soubory jsou staženy.")

covid_data = pd.read_csv("incidence-7-14-cr.csv")

x = covid_data["datum"]
y = covid_data["incidence_7"]

plt.plot(x, y, label="Covid data CZE")

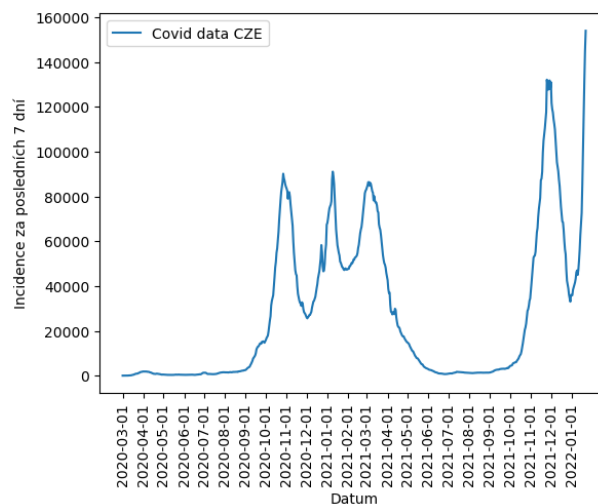
# popis os
plt.xlabel("Datum")
plt.ylabel("Incidence za posledních 7 dní")

# hodnoty osy x
plt.xticks([0, 31, 61, 92, 122, 153, 184, 214, 245, 275, 306, 337, 365,
            396, 426, 457, 487, 518, 549, 579, 610, 640, 671], rotation=90)

# Legenda dole vpravo
plt.legend(loc="upper left")

# zobrazení grafu
plt.show()

```



Obr. 10 - výsledný graf

Obr. 9 - kód pro tvorbu grafu ze vzdáleného souboru

Závěr:

V průběhu získávání informací v rámci problematiky „vizualizace v pythonu“ napříč tímto projektem, se nám podařilo nabýt velmi praktické dovednosti. V mnohých situacích přijde vhod automatická tvorba vizuálních dat, před jejich manuálním získáváním např. pomocí některého z tabulkových procesorů. Tyto metody se obzvláště hodí v situacích s určitou periodicitou, kdy nám stačí proces tvorby vytvořit pouze jednou a následně probíhá automaticky a my tak máme k dispozici rovnou obrazová data.

Použité zdroje:

- <https://www.root.cz/clanky/tvorba-grafu-v-jupyter-notebooku-s-vyuzitim-knihovny-matplotlib/>
- https://naucse.python.cz/2020/pydata-praha-jaro/pydata/visualization_basics/