Projekt 2022 - část 1

předmět Zpracování a vizualizace dat v prostředí Python

Changelog:
27. 10. opraven obrázek u funkce

Cílem projektu je získat, zpracovat a analyzovat data dostupná na internetu. První část projektu se bude týkat ověření vašeho osvojení technik pro efektivní zpracování, vizualizaci a získání dat.

Orientační struktura projektu

- Část 1 (20 b)
  - efektivní numerické výpočty
  - jednoduchá vizualizace
  - stažení a zpracování dat
- Část 2 (20 b)
  - různé pohledy na data
  - pokročilá vizualizace výsledků
  - zpracování závěrů (porozumění datům)
- Celkový projekt (60 b)
  - znázornění dat na mapě, operace nad těmito daty
  - korelace a predikce
  - automatické vytváření částí zpráv
  - spojení do analytické zprávy

Získání a předzpracování dat (20 bodů)

Vytvořte soubor part01.py, který bude implementovat níže uvedené metody. Pro zpracování a vizualizaci dat **není povoleno** použít pokročilých knihoven jako je Pandas či Seaborn. Kromě vestavěných knihoven (os, sys, re, gzip, pickle, csv, zipfile…) byste si měli vystačit s: numpy, matplotlib, BeautifulSoup, requests. Další knihovny je možné použít po schválení opravujícím (např ve fóru IS VUT).

Úkol 1: Numerický výpočet integrálu (3 body)

Cílem této funkce je numericky vypočítat integrál tzv. lichoběžníkovou metodou. Jako vstup dostanete dva numpy vektory: *x* (seřazený) a *y*, kde *x* představuje všechny integrační body (nemusí být nutně lineárně distribuované) a *y* vyjadřuje hodnotu integrované funkce *f(x)*. Musí tedy platit, že |x| = |y| (není nutno testovat). Funkce vrátí hodnotu určitého integrálu vypočtenou podle následujícího vzorce

$$\int_{x_0}^{x_{|X|-1}} f(x)\,dx = \sum_{i=1}^{|X|-1} (x_i - x_{i-1}) \cdot \frac{y_{i-1}+y_i}{2}$$

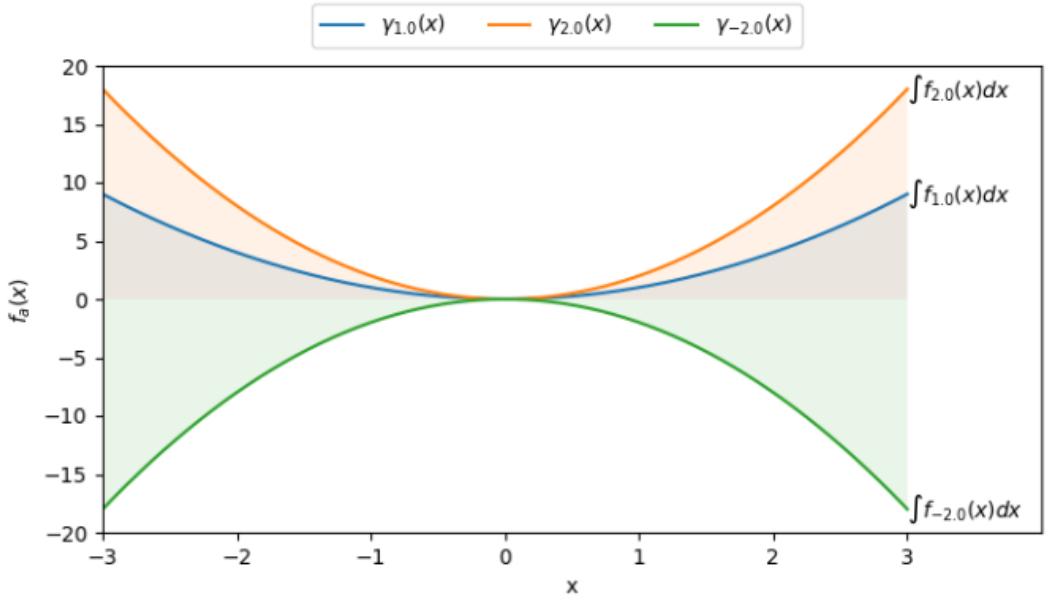
Pro získání plného hodnocení je nutné se zaměřit na efektivitu a využít možností knihovny *numpy* - t.j. vyhnout se procházení všech prvků cyklem `for`. Je zakázáno použít knihovní funkci, která přímo vrací určitý integrál.

Prototyp funkce:

```
def integrate(x : np.array, y : np.array) -> float
```

Úkol 2: Generování grafu s různými koeficienty (6 bodů)

Navrhněte funkci `generate_graph`, která bude vizualizovat funkci $f_a(x) = a * x^2$, definovanou na rozsahu <-3, 3>. Na začátku kódu vygenerujte v jednom kroku výsledky funkce *f* pro všechny hodnoty *a* (t.j. bez cyklů, do dvourozměrné matice) zadané ve vstupním argumentu (reprezentované jako seznam čísel s plovoucí desetinnou čárkou). Je tedy nutné (pro plné hodnocení) využít broadcasting.
Následně tuto matici po jednotlivých řádcích vizualizujte tak, aby výsledný graf vypadal následovně. Pro nastavení rozsahů zobrazení, umístění popisků a podobně můžete počítat s tím, že `a=[1.0, 2.0, -2.0]`. Je nutné dodržet Latex styl sazby popisků os a jednotlivých čar. Zachovejte následující vzhled (opraveno 27. 10.):



Funkce `generate_graph` má další dva argumenty - boolean hodnotu `show_figure`, která určuje, zda se má graf zobrazit pomocí funkce `show()` a `save_path`, která (pokud je nastavena), určuje, kam se má graf uložit pomocí funkce savefig().

Prototyp funkce

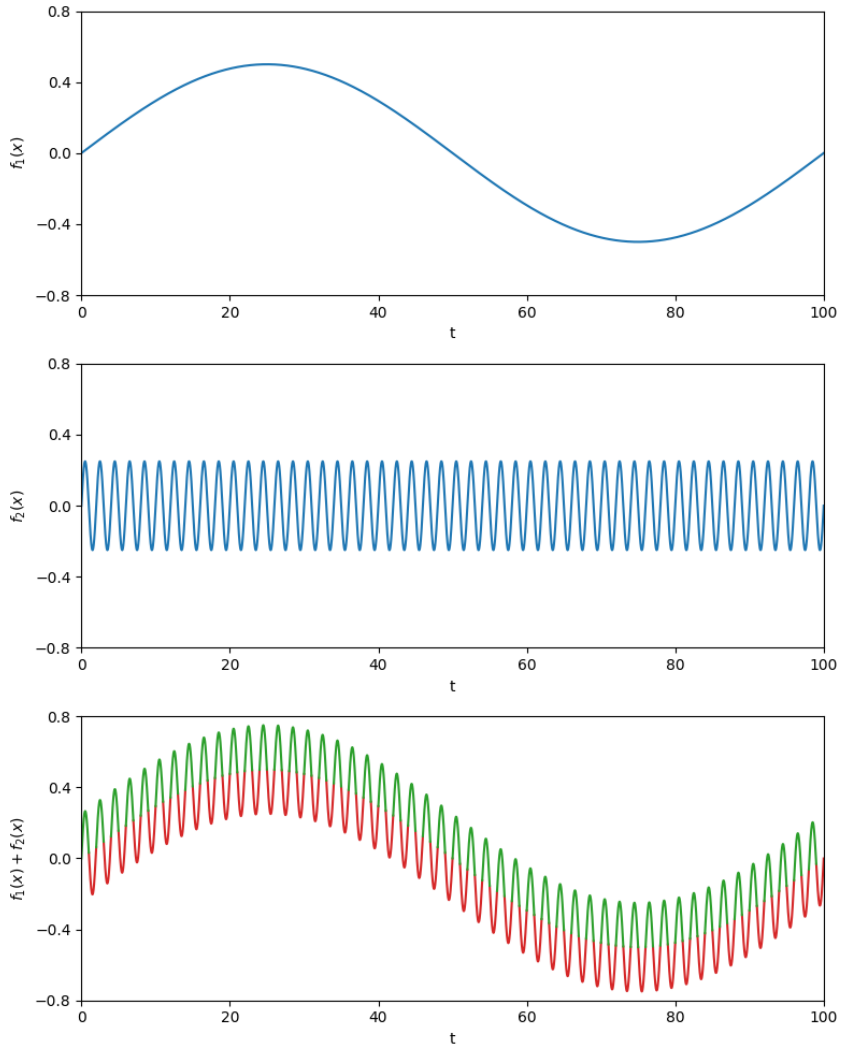
```
def generate_graph(a: List[float],
                   show_figure: bool = False,
                   save_path: str | None = None):
```

Úkol 3: Pokročilá vizualizace sinusového signálu (5 bodů)

Vytvořte graf se třemi podgrafy, zobrazující funkci f_1, f_2 a součet f_1+f_2 v rozsahu $t \in$ <0,100>. Funkce jsou definovány následovně:

$$f_1(t) = 0.5 \cdot sin(\frac{1}{50}\pi t), f_2(t) = 0.25 \cdot sin(\pi t)$$

V třetím podgrafu bude část, kdy se hodnota součtu obou funkcí dostává nad hodnotu samotné funkce f_1 zeleně, v opačném případě červeně. Je nutné, aby na grafu nevznikly žádné další artefakty (např. nějaké spojovací čáry a podobně). Upravte také body na ose tak, aby vypadaly, jako na tomto vzorovém obrázku. Pro argumenty `show_figure` a `save_path` platí stejné podmínky, jako v druhém úkolu.



Prototyp funkce

```
def generate_sinus(show_figure: bool = False,
                   save_path: str | None = None)
```

Úkol 4: Stažení tabulky (3 body)

Ze stránek https://ehw.fit.vutbr.cz/izv/temp.html stáhněte tabulku průměrných teplot v Brně. Každý řádek tabulky reprezentuje jeden měsíc, první buňka značí rok, druhá měsíc a následují jednotlivé dny. Pro každý řádek tabulky vytvořte záznam typu slovník s následující strukturou:

`{"year": 2018, "month": 1, "temp": np.array([0.9, 1.5, ...])}`

kde klíče `year` a `month` budou ukládat hodnotu typu integer, `temp` bude obsahovat pole float hodnot datového typu np.array. Výstupem funkce bude seznam (list) obsahující záznamy pro

jednotlivé řádky. Validita výstupního formátu je základním způsobem testována i v přiloženém *unittestu*.

Prototyp funkce

```
def download_data(url="https://ehw.fit.vutbr.cz/izv/temp.html")
```

Úkol 5: Procházení dat (2 body)

Napište funkci, která bere za vstup výsledky z úkolu 4 a počítá průměrnou teplotu podle specifikace - volbou argumentů můžeme filtrovat měsíc, rok či obojí. Pokud argumenty nejsou specifikovány, berou se všechny roky či měsíce.

Prototyp funkce

```
def get_avg_temp(data, year=None, month=None) -> float
```

Testování

K otestování funkčnosti můžete přistoupit dvěma základními způsoby. Buď do části, která se spouští pouze pokud je zavolán celý skript (`if __name__=="__main__"`), vložíte vaši testovací sekvenci, nebo můžete použít předpřipravený unittest v souboru *test_part01.py* a knihovnu pytest. Pokud máte tuto knihovnu nainstalovanou, můžete spustit příkaz `pytest` či `python3 -m pytest`. Pokud projdete testovacím skriptem, je to podmínka nutná (nikoliv dostačující) k dobrému hodnocení.

Dokumentace všech částí (souborů, funkcí) bude přímo v odevzdaném souboru. Snažte se dodržovat konvenci PEP 257 [https://www.python.org/dev/peps/pep-0257] a PEP 8 [https://www.python.org/dev/peps/pep-0008/]. Pozor, Python má přímo definovaný kódovací styl (narozdíl od C), a proto kontrola bude součástí hodnocení (1 bod).

Odevzdávání a hodnocení

Do 11. 11. 2022 odevzdejte jeden soubor *part01.py*.

Hodnotit se bude zejména:

- správnost výsledků
- vizuální dojem z grafů
- kvalita kódu
    - efektivitu implementace a reprezentace (i rychlost v porovnání s ostatními řešeními), využívání efektivních funkcí (např. NumPy)
    - přehlednost kódu
    - dodržení standardů a zvyklostí pro práci s jazykem Python - dokumentační řetězce, splnění PEP8
    - dokumentace funkcí a souborů
    - znovupoužitelnost kódů - správná izolace potřebných částí do funkcí

Celkem za první část můžete získat až 20 bodů, přičemž je k zápočtu nutné získat z této části minimálně 1 bod.

Dotazy a připomínky

Na fóru IS VUT případně na mailu mrazek@fit.vutbr.cz.