1. Úloha `08_LOG`

Základní informace:

- **Účel:**
  - využití logování z **logging**
  - příprava vlastní externí konfigurace logování
  - příprava testů ověřujících správnost logování
- **Odevzdávané soubory:**

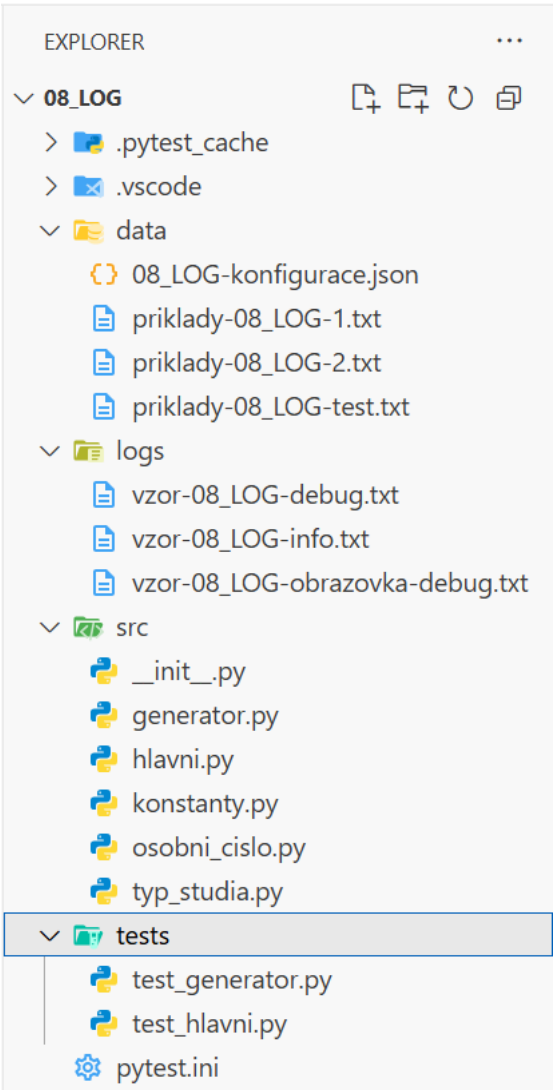
```
08_LOG-konfigurace.json
generator.py
hlavni.py
test_generator.py
test_hlavni.py
```

Zadání:

- do zdrojových souborů připravené již plně fungující aplikace doplňte logovací příkazy
- připravte JSON soubor externí konfigurace pro **logging**
- připravte testy ověřující vybrané úrovně logování

Popis vstupních dat:

- adresář úlohy



- v adresáři `data` se nachází soubor pro řízení běhu testů a pro externí konfiguraci logování
  - adresář obsahuje následující soubory:
    - `08_LOG-konfigurace.json` - kostra konfiguračního souboru logování
      - tento soubor budete doplňovat
    - `priklady-08_LOG-1.txt` - soubor s daty, která jsou velmi podobná datům z úlohy `02_MOCK`
      - tento soubor budete pouze číst a nebudete jej měnit
    - `priklady-08_LOG-2.txt` - redukovaná verze souboru `priklady-08_LOG-1.txt`
      - tento soubor budete pouze číst a nebudete jej měnit
    - `priklady-08_LOG-test.txt` - výrazně redukovaná verze souboru `priklady-08_LOG-1.txt`
      - tento soubor budete pouze číst a nebudete jej měnit
      - použije se jen pro ověření správnosti v `test_generator.py`
- v adresáři `logs` se nachází výsledky logování, které musíte dosáhnout
  - adresář obsahuje následující soubory:

- ♦ `vzor-08_LOG-obrazovka-debug.txt` - logovací soubor celé aplikace v úrovni `DEBUG` při spuštění aplikace s parametry
  - – `hlavni.py data/priklady-08_LOG-1.txt fav n`
- ♦ `vzor-08_LOG-info.txt` - logovací soubor aplikace v úrovni `Level.INFO` při postupném spuštění aplikace s parametry příkazové řádky:
  - – `hlavni.py data/neexistujici.txt fav b`
  - – `hlavni.py data/priklady-08_LOG-2.txt fav b`
  - – `hlavni.py data/priklady-08_LOG-1.txt fav n`
- ♦ `vzor-08_LOG-debug.txt` - logovací soubor aplikace v úrovni `Level.DEBUG` při spuštění aplikace s parametry
  - – `hlavni.py data/priklady-08_LOG-1.txt fav n`

  Note

  Tyto soubory (kromě `vzor-08_LOG-obrazovka-debug.txt`) jsou rozhodující - vámi upravená aplikace musí vytvořit zcela identické logovací soubory (samozřejmě bez počátečního `vzor-`) - ty budou kontrolovány validátorem.

■ v adresáři `src` se nachází zdrojové `.py` soubory

- adresář obsahuje soubory
  - ♦ `konstanty.py` - konstanty pro třídu `OsobniCislo` - nezměněná verze od úlohy `02_MOCK`
    - – tento soubor nebudete měnit
  - ♦ `typ_studia.py` - výčtový typ pro třídu `OsobniCislo` - nezměněná verze od úlohy `02_MOCK`
    - – tento soubor nebudete měnit
  - ♦ `osobni_cislo.py` - třída `OsobniCislo` - nezměněná verze od úlohy `02_MOCK`
    - – tento soubor nebudete měnit
  - ♦ `generator.py` - reálná verze třídy `Generator`, tj. generátoru osobních čísel
    - – ze vstupního souboru vybere jen ty studenty, kteří vyhovují konkrétní fakultě a konkrétnímu typu studia
      - tyto studenty seřadí podle abecedy a vygeneruje jim kompletní osobní čísla
      - seznam osobních čísel zapíše do výstupního souboru `<fakulta>_<typStudia>_vysledky.txt`

        např: `FAV_N_vysledky.txt`
    - – v tomto souboru budete doplňovat logovací příkazy
  - ♦ `hlavni.py` - skript, kterým lze celou aplikaci reálně spustit

  - zpracovává parametry příkazové řádky
    - spouští s nimi generátor
  - má předpřipravené parametry vstupní řádky pro jednotlivá spuštění

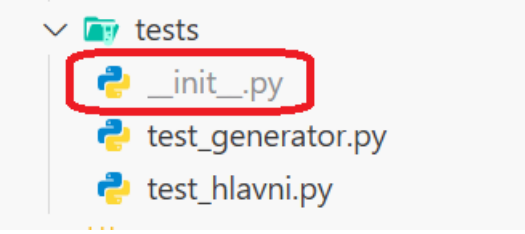
```
if __name__ == "__main__":
    main(sys.argv[1:])
    # args: list[str] = ["data/neexistujici.txt", "fav", "b"]
    # args: list[str] = ["data/priklady-08_LOG-2.txt", "fav", "b"]
    # args: list[str] = ["data/priklady-08_LOG-1.txt", "fav", "n"]
    # main(args)
```

  - v tomto souboru budete doplňovat logovací příkazy

- v adresáři `tests` se nacházejí soubory pro jednotlivé testy
  - adresář obsahuje následující soubory:
    - `test_generator.py` - kostra testů logování třídy `Generator`
      - tento soubor budete doplňovat
    - `test_hlavni.py` - kostra testů logování skriptu `hlavni`
      - tento soubor budete doplňovat

Warning

Pokud by **nešly lokálně spustit testy**, pak do adresáře `tests` zkopírujte ze `src` soubor `__init__.py`



- v kořenovém adresáři se nachází soubory:
  - `.env` - konfigurační soubor prostředí
    - tento soubor nebudete měnit
  - `pytest.ini` - konfigurační soubor spouštění **pytest**
    - tento soubor nebudete měnit

Postup řešení:

- budete řešit tři oddělené úlohy, které spolu souvisejí a musí být vykonány v tomto pořadí

1. doplnění logovacích příkazů do zdrojových souborů a nastavení formátu logování
2. příprava vlastní externí konfigurace logování

3. příprava testů ověřujících správnost logování

Doplnění logovacích příkazů a nastavení formátu logování:

- do `hlavni.py` přidejte

```
logging.config.dictConfig(json.load(open(
    "data/08_LOG-konfigurace.json"
    )))

hlavni_logger = logging.getLogger("hlavni")
```

- do `main()` přidejte

```
def main(args: list[str]):
    hlavni_logger.info("Zacatek programu")
```

- zkontrolujte importy

```
import logging
import logging.config
import json
import sys
```

- po spuštění skriptu jako

```
if __name__ == "__main__":
    main(sys.argv[1:])
```

  byste měli vidět na konzoli

```
INFO | hlavni | Zacatek programu
```

  - pokud tento výpis nevidíte, udělali jste v postupu chybu

- editujte soubor `08_LOG-konfigurace.json` a opravujte jeho formát výpisu tak dlouho, dokud se nebude shodovat s požadovaným formátem, který zjistíte ze souboru `vzor-08_LOG-obrazovka-debug.txt`
  - výpis začíná úrovní
  - pak je jméno třídy
  - následuje jméno metody včetně prázdných závorek `()`
  - dále je vypisována logovací zpráva
- po spuštění skriptu byste měli vidět na konzoli

```
[INFO - hlavni.main()] - Zacatek programu
```

- z logovacího souboru `vzor-08_LOG-obrazovka-debug.txt` tedy vidíte, do jaké metody máte umístit volání logovací funkce konkrétní úrovně a s jakou konkrétní zprávou

- doplňování logů je tak víceméně mechanická práce
- postupujte tak, že nejdříve doplníte (a zkontrolujete) všechny logy do modulu `Hlavni`
- pak ve třídě `Generator` definujte statický logger `logger`
  - budete jej dále používat např. jako `Generator.logger.info()`
- do třídy `Generator` doplňte všechny logy

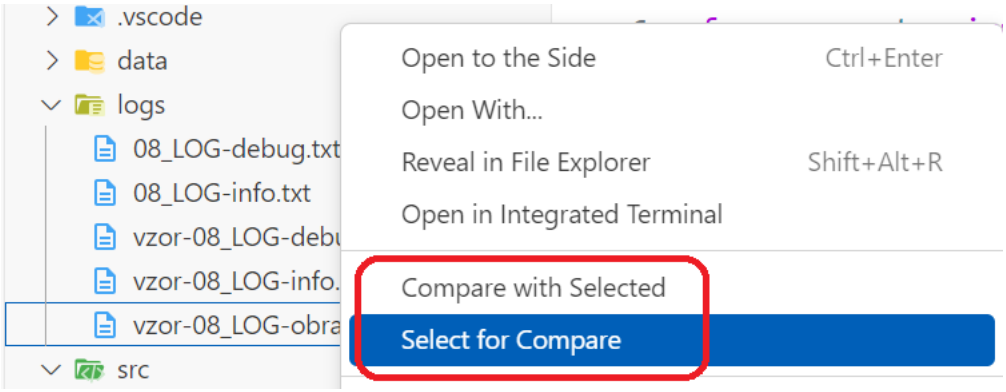
- spuštění skriptu bude nyní:

```
if __name__ == "__main__":
    args: list[str] = ["data/priklady-08_LOG-1.txt", "fav", "n"]
    main(args)
```

- díky nastavení v `08_LOG-konfigurace.json` jsou všechny logy všech loggerů na obrazovku
  - v doplňování logovacích příkazů postupujte tak dlouho, dokud nedostanete na obrazovce výpis totožný s obsahem souboru `vzor-08_LOG-obrazovka-debug.txt`

Note

Pro porovnání obsahu souborů můžete využít ve VSC služby **Select for Compare** a **Compare With Selected**



Příprava vlastní externí konfigurace logování:

- další činnost budete provádět editováním souboru `08_LOG-konfigurace.json`
  - postupujte iterativně, nesnažte se napsat celou konfiguraci najednou
- najednou se bude logovat do dvou souborů - jejich jména:
  - `08_LOG-info.txt` - sem se zapisuje v režimu *append* a jen do úrovně `INFO`
    - prakticky to znamená, že sem zapisují současně loggery z `Hlavni` a z `Generator`
    - protože však `Generator` pouští až do úrovně `DEBUG` (viz dále), je třeba tuto úroveň odfiltrovat
  - `08_LOG-debug.txt` - soubor se při každém běhu přemazává (je jen pro ladění) a přijímá do úrovně `DEBUG` - tj. jen logger z `Generator`
- možná budete mít problémy s dvojím výpisem stejného logu - pak se jedná o `propagate`

- v `root` vynechte připojení `konzole`, čímž zrušíte výpisy na konzoli
  - to znamená, že při spuštění se budou vytvářet jen logovací soubory a soubor výsledků `*_vysledky.txt`
- postupné spouštění pro vytvoření `08_LOG-info.txt` zajistíte postupným odkomentováváním:

```
if __name__ == "__main__":
    args: list[str] = ["data/neexistujici.txt", "fav", "b"]
    # args: list[str] = ["data/priklady-08_LOG-2.txt", "fav", "b"]
    # args: list[str] = ["data/priklady-08_LOG-1.txt", "fav", "n"]
    main(args)
```

Příprava testů ověřujících správnost logování:

Warning

Před psaním testů upravte konec souboru `hlavni.py` takto:

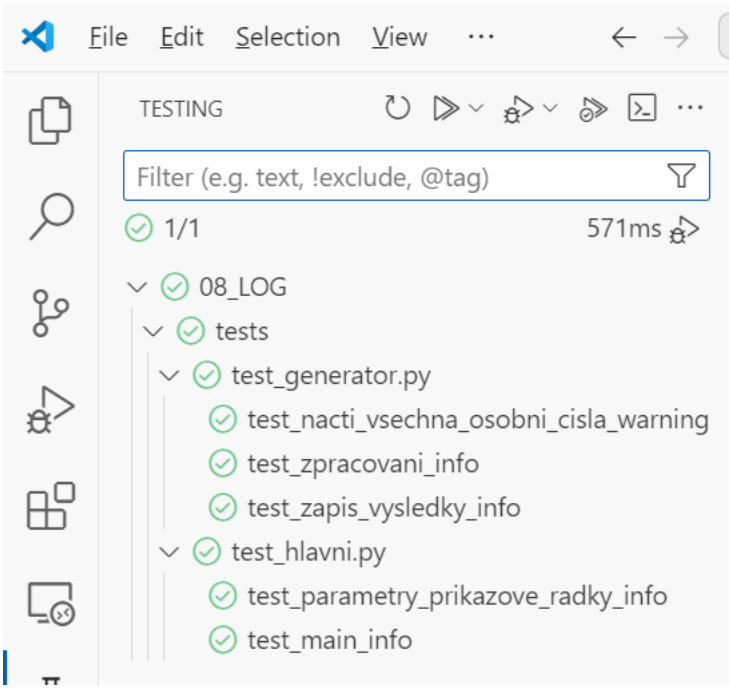
```
if __name__ == "__main__":
    main(sys.argv[1:])
    # args: list[str] = ["data/neexistujici.txt", "fav", "b"]
    # args: list[str] = ["data/priklady-08_LOG-2.txt", "fav", "b"]
    # args: list[str] = ["data/priklady-08_LOG-1.txt", "fav", "n"]
    # main(args)
```

V této podobě pak bude soubor `hlavni.py` commitován.

- po dokončení úprav konfiguračního souboru `08_LOG-konfigurace.json` budete v adresáři `tests` psát jednotkové testy
  - zaměříte se pouze na na správnost logování, nikoliv na funkcionalitu aplikace
- do připravených souborů `test_hlavni.py` a `test_generator.py` budete postupně doplňovat těla jednotlivých funkcí tak, aby ověřily správnost logovacích výpisů daného loggeru a dané úrovně
  - v názvu funkce je jak název testované funkce, tak i název úrovně, např.:

    ```
    def test_parametry_prikazove_radky_info(caplog):
    ```

  - otestujete všechny logy úrovně `INFO` a `WARNING`
    - úroveň `ERROR` lze obtížně nasimulovat
    - úroveň `DEBUG` je příliš nízká
  - v `test_zpracovani_info(caplog):` použijete pro testování soubor `priklady-08_LOG-test.txt`
  - v `test_zapis_vysledky_info(caplog):` použijete jako skutečný parametr prázdný seznam `[]`
- všechny testy se spouštějí zcela standardním způsobem a všechny musí projít (i po zápisu asertů)



Commitované soubory:

- ```
  08_LOG-konfigurace.json
  generator.py
  hlavni.py
  test_generator.py
  test_hlavni.py
  ```

Časová náročnost úlohy:

- po úspěšném odevzdání, prosím, vyplňte v tabulce časovou náročnost této úlohy

  https://docs.google.com/spreadsheets/d/1brjDkZ4pBkAD1WG7tLan89zFHkOcoaQMQXtcTq1JtU0
  ```