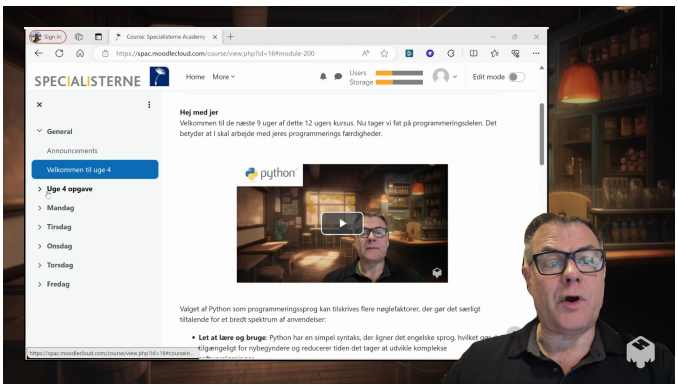

Udvikl et Python-program, der anvender generators til at filtrere og behandle data fra en meget stor fil. Programmet skal kunne læse filen linje for linje og udføre bestemte operationer baseret på dine kriterier, såsom at filtrere bestemte data eller udføre beregninger, uden nogensinde at indlæse hele filen i hukommelsen.

Tilgang til Opgaven

- **Forstå Generators**: Start med at genopfriske din forståelse af, hvad generators er, og hvordan de fungerer i Python. Husk, generators producerer elementer kun én ad gangen i stedet for at indlæse hele sekvensen i hukommelsen.
- **Læsning af Store Filer**: Brug Python's indbyggede open funktion til at åbne din store fil. Anvend en generator med en løkke til at læse filen linje for linje ved brug af yield i stedet for at returnere en hel liste.
- **Filtrering og Behandling af Data**: Bestem, hvilken type data du ønsker at filtrere eller hvilke beregninger du ønsker at udføre. Implementer din logik inden for løkken, hvor du læser filen, så hver linje behandles sekventielt.

Hints

- **Effektivitet er Nøglen**: Husk på, at hele ideen med at bruge generators er at øge effektiviteten, især når det kommer til hukommelsesforbrug.
- **Test med Mindre Dataset**: Før du kører dit script på den meget store fil, test det først med en mindre fil for at sikre, at din logik fungerer som forventet.
- **Fejlhåndtering**: Overvej at tilføje fejlhåndtering for at tage hånd om potentielle problemer, såsom filer, der ikke eksisterer, eller læse/skrive fejl.
- **Faker**: Brug evt. Faker modulet til at genere en datafil

Ressourcer

- Real Python - Functional Programming in Python: https://realpython.com/courses/functional-programming-python
- Pluralsight: Functional Programming with Python
- Faker: https://faker.readthedocs.io/en/master

Udvidet Opgave: Analyse og Visualisering af Data med Generators

Hvis du er hurtigt færdig med denne, opgave kan du få yderligere udfordringer, med følgende udvidelse af opgaven.

Opgavebeskrivelse

Byg videre på dit eksisterende program ved at tilføje funktionalitet for at analysere og visualisere de behandlede data.

Udvikl et script, der ikke kun filtrerer og behandler data fra en stor fil, men også udfører en statistisk analyse af dataene og genererer visualiseringer for at fremvise resultaterne.

Trin til Udførelse

- **Statistisk Analyse**: Efter at have filtreret og behandlet dataene, udfør en statistisk analyse for at finde trends eller mønstre.
  Dette kunne omfatte beregning af gennemsnit, median, standardafvigelse, eller andre relevante statistikker baseret på din datasæt.
- **Data Visualisering**: Anvend biblioteker som Matplotlib eller Seaborn til at skabe visualiseringer af dine analyseresultater.
  Dette kan omfatte histogrammer, scatter plots, linjediagrammer eller andre typer af visualiseringer, der bedst fremviser dine fund.

?

- **Performance Optimering**: Evaluer effektiviteten af din kode, specielt hvordan den håndterer store datamængder. Eksperimentér med forskellige måder at optimere din kode's performance på, herunder anvendelsen af mere avancerede generators, list comprehensions, eller måske multithreading/multiprocessing for at parallelisere dataforarbejdningen.

Yderligere Overvejelser

- **Udforsk Python Profilers**: Brug Python's indbyggede profilers som cProfile til at identificere flaskehalse i din kode.
- **Fejlhåndtering og Logging**: Implementer robust fejlhåndtering og logging for at gøre fejlfinding lettere og for at sikre, at dit program kører så glat som muligt, selv med store eller komplekse datasæt.

Ressourcer

- Matplotlib Dokumentation: https://matplotlib.org
- Seaborn Dokumentation: https://seaborn.pydata.org
- cProfile Dokumentation: https://docs.python.org/3/library/profile.html#module-cProfile

Last modified: Saturday, 17 February 2024, 9:13 PM