TiDA 05 — Parser outputu i wyrażenia regularne

Cel zajęć

Na tych zajęciach dowiesz się:

- co znaczy „parsować” i czym jest „parser”,
- jak korzystać z **wyrażeń regularnych** (regex) do wyszukiwania danych w tekście,
- jak zapisywać dane do pliku **CSV**, który można otworzyć w Excelu,
- jak wczytać dane w Pythonie (Pandas) i narysować prosty wykres.

Część teoretyczna

Co to znaczy „parsować”?

Parsować

Słowo **„parsować”** (z ang. *to parse*) oznacza analizować tekst, aby **wydobyć z niego dane lub znaczenie**.

Przykłady:

- Z tekstu `Czas sortowania (indeksy): 0.123456 s` chcemy wyciągnąć liczbę `0.123456`.
- Przeglądarka parsuje kod HTML, żeby wiedzieć, jak wygląda strona.
- Kompilator parsuje kod źródłowy, żeby go zrozumieć.

Parser

Parser to program lub fragment kodu, który **automatycznie odczytuje dane z tekstu**.

W tej lekcji parser odczyta z pliku wyniki pomiarów i zapisze je do pliku CSV.

🔷 Wyrażenia regularne — proste wprowadzenie

Wyrażenia regularne (ang. *regular expressions* lub **regex**) to specjalne wzorce, które pozwalają wyszukiwać fragmenty tekstu.

Najczęściej używane symbole

Symbol	Znaczenie
`\d`	cyfra
`\w`	znak słowa (litera, cyfra, podkreślnik)
`\s`	spacja, tabulator
`.`	dowolny znak
`+`	jeden lub więcej znaków
`?`	zero lub jeden znak
`( )`	grupa
` `	` `

Przykład

Chcemy znaleźć w tekście taki wzorzec:

```
Czas sortowania (indeksy): 0.123456 s
```

Wzorzec w Pythonie wygląda tak:

```
r"Czas sortowania \((?P<tryb>indeksy|wskazniki)\):\s*(?P<czas>\d+(?:\.\d+)?)\s*s"
```

To oznacza:

- znajdź słowo „Czas sortowania”,
- potem w nawiasach nazwę trybu (indeksy lub wskazniki),
- a następnie liczbę (czas w sekundach).

🔷 Nasz plik z wynikami

Z poprzednich lekcji mamy plik `wynik.txt` z takimi danymi:

```
Czas sortowania (indeksy): 0.123456 s
Czas sortowania (wskazniki): 0.098765 s
Czas sortowania (indeksy) [N=100000]: 0.234 s
```

Chcemy, aby nasz skrypt odczytał te linie i zapisał dane w pliku `wyniki.csv` w postaci tabeli.

Skrypt parsera (prosta wersja)

```python
import re
import csv

pattern = re.compile(
    r"Czas sortowania \((?P<tryb>indeksy|wskazniki)\)"
    r"(?:\s*\[N=(?P<N>\d+)\])?"
    r"\s*:\s*(?P<czas>\d+(?:\.\d+)?)\s*s"
)

rows = []

with open("wynik.txt", "r", encoding="utf-8") as f:
    for line in f:
        m = pattern.search(line)
        if m:
            d = m.groupdict()
            rows.append({
                "tryb": d.get("tryb"),
                "N": d.get("N"),
                "czas_s": d.get("czas"),
            })

# zapisujemy dane do pliku CSV
with open("wyniki.csv", "w", encoding="utf-8-sig", newline="") as f:
    writer = csv.DictWriter(f, fieldnames=["tryb", "N", "czas_s"], delimiter=';')
    writer.writeheader()
    writer.writerows(rows)

print("Dane zapisano do pliku wyniki.csv")
```

Po uruchomieniu powstanie plik CSV, który możesz otworzyć w Excelu.

Otwieranie pliku CSV

1. Otwórz folder, w którym znajduje się `wyniki.csv`.
2. Kliknij dwa razy, aby otworzyć w Excelu.
   Jeśli kolumny się nie rozdzielają, wybierz **Dane → Z tekstu/CSV** i ustaw separator `;`.

Wczytanie danych w Pandas i prosta wizualizacja

```
import pandas as pd
import matplotlib.pyplot as plt

# wczytanie pliku CSV
df = pd.read_csv("wyniki.csv", sep=';')
print(df)

# zamiana kolumny 'czas_s' na typ liczbowy
df["czas_s"] = df["czas_s"].astype(float)

# wykres prosty – indeksy na osi X
plt.figure()
plt.plot(df.index, df["czas_s"], marker='o')
plt.title("Czas sortowania (sekundy)")
plt.xlabel("Pomiar")
plt.ylabel("Czas [s]")
plt.grid(True)
plt.show()
```

ZADANIA

1. **Podstawowe:** uruchom skrypt, sprawdź czy w pliku CSV znajdują się dane.
2. **Rozszerzone:** dodaj kolumnę `N` (rozmiar tablicy) i narysuj wykres `czas vs N`.
3. **Dodatkowe:** dodaj obsługę jednostki `ms` i przelicz ją na sekundy.

Co warto zapamiętać

- **Parsowanie** to wydobywanie danych z tekstu.
- **Parser** to program, który to robi automatycznie.
- **Wyrażenia regularne** pozwalają dopasowywać wzorce w tekście.
- **CSV** to prosty format pliku z tabelą danych, który można otworzyć w Excelu lub Pandas.
- **Matplotlib** pozwala w prosty sposób narysować wykresy.

Dla chętnych

Spróbuj dodać do skryptu parsera obsługę błędów — np. gdy linia nie pasuje do wzorca, wypisz komunikat:

```
else:
    print("Nie rozpoznano linii:", line.strip())
```
