TiDA 04: Dokumentacja testów i raporty w Markdown

Cel zajęć

Na tych zajęciach:

- nauczysz się tworzyć raport z przeprowadzonych testów,
- poznasz strukturę raportu testowego,
- nauczysz się zapisywać wyniki testów automatycznie do pliku `.md` z poziomu Pythona,
- zobaczysz, jak z Markdown wygenerować dokumenty w innych formatach (HTML, PDF itp.).

Czym jest raport testowy?

Raport testowy to **podsumowanie wyników przeprowadzonych testów** – zawiera nie tylko dane, ale też wnioski.
Tworzy się go po zakończeniu testów wydajnościowych, funkcjonalnych etc.

Podstawowa struktura raportu testowego

Sekcja	Opis
Cel testu	Co testowaliśmy i dlaczego
Środowisko	Informacje o sprzęcie, systemie, wersji kompilatora/interpretera
Parametry i scenariusze	Jakie testy przeprowadzono, na jakich danych
Wyniki	Dane liczbowe, np. czasy, błędy, liczba testów zakończonych sukcesem
Wnioski	Co wynika z testów, co można poprawić, jakie obserwacje

---

Pobieranie informacji o systemie

```python
import platform
import datetime

print("System:", platform.system())
print("Wersja:", platform.version())
print("Procesor:", platform.processor())
print("Python:", platform.python_version())
print("Data testu:", datetime.datetime.now())
```

Markdown (MD) — lekki język do dokumentacji

Markdown to prosty format tekstowy, który można łatwo przerobić na HTML, PDF lub DOCX.
Używa się go w projektach IT (GitHub, README.md, dokumentacje API, raporty testowe).

Najważniejsze elementy składni Markdown

Nagłówki

```
# Nagłówek 1
## Nagłówek 2
### Nagłówek 3
```

Listy

```
- Element listy
- Kolejny element
  - zagnieżdżony element
```

Tabele

```
| Kolumna 1 | Kolumna 2 |
|-----------|-----------|
| Dane A    | Dane B    |
```

Formatowanie tekstu

```
**pogrubienie**, *kursywa*, ~~przekreślenie~~
```

Wstawianie kodu

````
```python
print("To jest kod Pythona")
```
````

Generowanie raportu Markdown w Pythonie

```python
import platform
import datetime

def generuj_raport(test_results):
    filename = f"raport_testu_{datetime.date.today()}.md"
    with open(filename, "w", encoding="utf-8") as f:
        f.write("# Raport testów wydajności\n\n")
        f.write(f"**Data:** {datetime.datetime.now()}\n\n")
        f.write("## Cel testu\n")
        f.write("Porównanie czasu sortowania tablic różnych rozmiarów.\n\n")

        f.write("## Środowisko\n")
        f.write(f"- System: {platform.system()} {platform.release()}\n")
        f.write(f"- Procesor: {platform.processor()}\n")
        f.write(f"- Python: {platform.python_version()}\n\n")

        f.write("## Wyniki\n")
        f.write("| Rozmiar tablicy | Czas (s) |\n")
        f.write("|-----------------|----------|\n")
        for size, time in test_results.items():
            f.write(f"| {size} | {time:.6f} |\n")

        f.write("\n## Wnioski\n")
        f.write("Czas wykonania rośnie wraz z rozmiarem tablicy, zgodnie z O(n²).\n")

    print(f"Raport zapisano jako: {filename}")

wyniki = {100: 0.01, 1000: 0.3, 5000: 2.9}
generuj_raport(wyniki)
```

🌐 Inne formaty raportów

Markdown można łatwo przekształcić na inne formaty:

- **HTML** – do publikacji w sieci,
- **PDF** – do archiwizacji lub wydruku,
- **DOCX** – do edycji w Wordzie.

🔧 Przykład konwersji Markdown → PDF

```
import pypandoc

pypandoc.convert_text(
    open("raport_testu_2025-10-24.md").read(),
    'pdf',
    format='md',
    outputfile='raport_testu.pdf',
    extra_args=['--standalone']
)
```

> ⚠️ Uwaga: wymaga zainstalowanego programu **pandoc**.

📊 Rozszerzenie – wizualizacja wyników

```
import matplotlib.pyplot as plt

sizes = [100, 1000, 5000]
times = [0.01, 0.3, 2.9]

plt.plot(sizes, times, marker='o')
plt.title("Czas sortowania vs Rozmiar tablicy")
plt.xlabel("Rozmiar tablicy")
plt.ylabel("Czas [s]")
plt.grid(True)
plt.savefig("wykres_czasu.png")
plt.show()
```

W Markdown można wstawić obrazek:

```
![Wykres wyników](wykres_czasu.png)
```

🎓 Zadania (do oddania w formie sprawozdania)

1. Utwórz własny raport `.md` z testów wydajności funkcji sortujących.
2. Dodaj sekcję **Środowisko** – pobierz automatycznie dane o systemie i wersji Pythona.
3. Zapisz wyniki w formie tabeli (rozmiar tablicy, czas wykonania).
4. Dodaj wykres i wstaw go do raportu.
5. Spróbuj przekonwertować raport Markdown do **PDF** lub **HTML**.
6. (Dla chętnych) – zaprogramuj automatyczne tworzenie raportu po każdym teście.