Praca z plikami Excel w Pythonie

Biblioteki i ich zastosowania

Cel prezentacji

- - Omówienie głównych bibliotek Pythona używanych do pracy z Excel
- - Wyjaśnienie różnic między starszymi i nowoczesnymi rozwiązaniami
- - Przegląd kluczowych funkcji i zastosowań w praktyce

Dlaczego Python i Excel?

- - Popularność Excela w analizie danych
- - Python jako narzędzie do automatyzacji i przetwarzania dużych zbiorów danych
- - Kluczowe korzyści:
- - Szybkość przetwarzania
- - Możliwość automatyzacji zadań
- - Rozszerzalność (praca z różnymi formatami)

Przegląd bibliotek Python

Biblioteka	Główne zastosowanie	Obsługiwane formaty
pandas	Analiza danych, odczyt/zapis Excel	.xls, .xlsx
openpyxl	Praca z nowoczesnymi plikami Excel `.xlsx`	.xlsx
xlrd	Odczyt starszych plików Excel `.xls`	.xls
xlwt	Tworzenie starszych plików Excel `.xls`	.xls

Praca z `pandas`

- - Główne funkcje:
- - `read_excel()` – odczyt plików Excel
- - `to_excel()` – zapis danych do Excela
- - Obsługa arkuszy:
- - Parametr `sheet_name` do określenia, który arkusz wczytać
- - Przykład:
- ```python
- import pandas as pd
- df = pd.read_excel('plik.xlsx', sheet_name='Arkusz1')
- df.to_excel('wyniki.xlsx', index=False)
- ```

Praca z `openpyxl`

- - Zastosowanie:
- - Manipulacja komórkami i stylami
- - Tworzenie wykresów i formatowanie arkuszy
- - Obsługiwane formaty: `.xlsx`
- - Przykład:
- ```python
- from openpyxl import Workbook
- wb = Workbook()
- ws = wb.active
- ws.title = 'Dane'
- ws.append(['Kolumna1', 'Kolumna2'])
- wb.save('plik.xlsx')
- ```

Praca z `xlrd` i `xlwt`

- - `xlrd`:
- - Odczyt plików `.xls`
- - Obsługuje tylko starsze formaty Excela
- - `xlwt`:
- - Tworzenie plików `.xls`
- - Ograniczenia:
- - Brak wsparcia dla `.xlsx`
- - Nie obsługują stylizacji w nowoczesnych plikach Excel

Porównanie bibliotek

Porównanie bibliotek

Biblioteka	Odczyt	Zapis	Formatowanie	Wykresy	Obsługa starszych plików .xls
Pandas	✅	✅	❌	❌	❌
Openpyxl	✅	✅	✅	✅	❌
XlsxWriter	❌	✅	✅	✅	❌
Xlrd	✅	❌	❌	❌	✅
Xlwt	❌	✅	✅	❌	✅
Xlutils	✅	✅	❌	❌	✅
Pyexcel	✅	✅	❌	❌	✅

Najczęściej używane funkcje

- 1. Odczyt pliku Excel:
- ```python
- pd.read_excel('plik.xlsx', sheet_name='Arkusz1')
- ```
- 2. Zapis pliku Excel:
- ```python
- df.to_excel('wyniki.xlsx', index=False)
- ```
- 3. Tworzenie wykresu w `openpyxl`:
- ```python
- from openpyxl import Workbook
- from openpyxl.chart import BarChart, Reference
- wb = Workbook()
- ws = wb.active
- ws.append(['A', 'B'])
- chart = BarChart()
- chart.add_data(Reference(ws, min_col=1, max_col=2), titles_from_data=True)
- ws.add_chart(chart, 'E5')
- wb.save('wykres.xlsx')
- ```

Ograniczenia i wyzwania

- - `xlrd` i `xlwt`:
- - Brak wsparcia dla nowoczesnych plików Excel
- - `pandas`:
- - Brak obsługi zaawansowanego formatowania
- - `openpyxl`:
- - Większa złożoność kodu dla stylizacji i wykresów

Podsumowanie

- - Python oferuje różnorodne narzędzia do pracy z plikami Excel
- - Wybór biblioteki zależy od wymagań:
- - Analiza danych → `pandas`
- - Stylizacja → `openpyxl`
- - Starsze formaty → `xlrd`/`xlwt`
- - Zastosowanie tych bibliotek pozwala na automatyzację i efektywność w pracy z danymi