

O'REILLY®

Wydanie II

# Python i Excel

Nowoczesne środowisko do automatyzacji  
i analizy danych



Helion 

Felix Zumstein

Tytuł oryginału: Python for Excel: A Modern Environment for Automation  
and Data Analysis, 2<sup>nd</sup> Edition

Tłumaczenie: Anna Mizerska, z wykorzystaniem fragmentów  
poprzedniego wydania w przekładzie Leszka Sagalary

ISBN: 978-83-289-4155-7

© 2027 Helion S.A.

Authorized Polish translation of the English edition of *Python for Excel, 2E*  
ISBN 9798341640290 © 2026 Zoomer Analytics LLC.

This translation is published and sold by permission of O'Reilly Media, Inc.,  
which owns or controls all rights to publish and sell the same.

All rights reserved. No part of this book may be reproduced or transmitted in any  
form or by any means, electronic or mechanical, including photocopying, recording  
or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości  
lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione.  
Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie  
książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie  
praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi  
bądź towarowymi ich właścicieli.

Autor oraz wydawca dołożyli wszelkich starań, by zawarte w tej książce informacje  
były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich  
wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych  
lub autorskich. Autor oraz wydawca nie ponoszą również żadnej odpowiedzialności  
za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

[helion.pl/user/opinie/pytex2](https://helion.pl/user/opinie/pytex2)

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Helion S.A.

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 230 98 63

e-mail: [helion@helion.pl](mailto:helion@helion.pl)

WWW: [helion.pl](https://helion.pl) (księgarnia internetowa, katalog książek)

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

---

# Spis treści

|                                               |           |
|-----------------------------------------------|-----------|
| <b>Wprowadzenie .....</b>                     | <b>13</b> |
| <hr/>                                         |           |
| <b>Część I. Wprowadzenie do Pythona .....</b> | <b>19</b> |
| <b>1. Dlaczego Python w Excelu? .....</b>     | <b>21</b> |
| Podstawowe zasady Pythona .....               | 22        |
| Język ogólnego przeznaczenia .....            | 22        |
| Czytelność ma znaczenie .....                 | 22        |
| Wbudowane funkcjonalności Pythona .....       | 23        |
| Kompatybilność międzyplatformowa .....        | 23        |
| Obliczenia naukowe .....                      | 24        |
| Nowoczesne cechy języka .....                 | 25        |
| Praca z tekstem .....                         | 26        |
| Obsługa tablic .....                          | 26        |
| Webowe interfejsy API .....                   | 27        |
| Obsługa błędów .....                          | 27        |
| Narzędzia .....                               | 28        |
| Menedżer pakietów .....                       | 28        |
| Testowanie .....                              | 28        |
| Kontrola wersji .....                         | 29        |
| Podsumowanie .....                            | 30        |
| <b>2. Środowisko programistyczne .....</b>    | <b>31</b> |
| Kody do pobrania dla tej książki .....        | 31        |
| Używanie terminala .....                      | 33        |
| Instalacja Pythona .....                      | 36        |
| Instalacja uv .....                           | 36        |
| Instalacja Pythona i pakietów .....           | 37        |
| REPL: interaktywna sesja Pythona .....        | 38        |

|                                                              |           |
|--------------------------------------------------------------|-----------|
| Notatniki Jupyter                                            | 39        |
| Uruchamianie notatników Jupyter                              | 40        |
| Komórki notatnika                                            | 42        |
| Tryb edycji a tryb poleceń                                   | 43        |
| Kolejność uruchamiania ma znaczenie                          | 44        |
| Visual Studio Code                                           | 45        |
| Instalacja i konfiguracja                                    | 47        |
| Uruchamianie skryptu Pythona                                 | 49        |
| Debugger                                                     | 52        |
| Uruchamianie notatników Jupyter w VS Code                    | 53        |
| Podsumowanie                                                 | 55        |
| <b>3. Wprowadzenie do Pythona .....</b>                      | <b>56</b> |
| Typy danych                                                  | 57        |
| Obiekty                                                      | 57        |
| Typy liczbowe                                                | 58        |
| Komentarze                                                   | 60        |
| Logiczny typ danych                                          | 61        |
| Łańcuchy znaków                                              | 62        |
| Indeksowanie i wycinanie                                     | 63        |
| Indeksowanie                                                 | 63        |
| Wycinanie                                                    | 64        |
| Struktury danych                                             | 65        |
| Listy                                                        | 65        |
| Słowniki                                                     | 67        |
| Krotki                                                       | 68        |
| Zbiory                                                       | 68        |
| Przepływ sterowania                                          | 69        |
| Blok kodu i instrukcja pass                                  | 69        |
| Instrukcja if i wyrażenia warunkowe                          | 70        |
| Pętle for i while                                            | 71        |
| Lista składana                                               | 74        |
| Organizacja kodu                                             | 75        |
| Funkcje                                                      | 75        |
| Moduły i instrukcja import                                   | 76        |
| Klasa datetime                                               | 78        |
| Formatowanie kodu, analiza błędów w kodzie i adnotacje typów | 80        |
| PEP 8 — przewodnik stylu kodowania w Pythonie                | 80        |
| Ruff                                                         | 82        |
| Informacje o typie                                           | 84        |
| Podsumowanie                                                 | 85        |

---

|                                                    |           |
|----------------------------------------------------|-----------|
| <b>Część II. Wprowadzenie do biblioteki pandas</b> | <b>87</b> |
| <b>4. Podstawy NumPy .....</b>                     | <b>89</b> |
| Pierwsze kroki z NumPy                             | 89        |
| Tablica NumPy                                      | 89        |
| Wektoryzacja i rozgłaszanie                        | 91        |
| Funkcje uniwersalne                                | 92        |
| Tworzenie tablic i operowanie nimi                 | 93        |
| Pobieranie i wybieranie elementów tablicy          | 93        |
| Przydatne konstruktory tablicowe                   | 94        |
| Widok a kopia                                      | 95        |
| Podsumowanie                                       | 96        |
| <b>5. Analiza danych z biblioteką pandas .....</b> | <b>97</b> |
| DataFrame i Series                                 | 97        |
| Indeks                                             | 100       |
| Kolumny                                            | 102       |
| Operowanie danymi                                  | 103       |
| Wybieranie danych                                  | 103       |
| Ustawianie danych                                  | 108       |
| Brakujące dane                                     | 111       |
| Zduplikowane dane                                  | 112       |
| Operacje arytmetyczne                              | 113       |
| Praca z kolumnami tekstowymi                       | 115       |
| Łączenie obiektów DataFrame                        | 115       |
| Konkatenacja                                       | 116       |
| Operacje join i merge                              | 117       |
| Statystyka opisowa i agregacja danych              | 119       |
| Statystyka opisowa                                 | 119       |
| Grupowanie                                         | 120       |
| Tabele przestawne                                  | 121       |
| Tworzenie wykresów                                 | 122       |
| Matplotlib                                         | 122       |
| Plotly                                             | 123       |
| Importowanie i eksportowanie obiektów DataFrame    | 126       |
| Eksportowanie plików CSV                           | 126       |
| Importowanie plików CSV                            | 127       |
| Współczesne alternatywy                            | 129       |
| Podsumowanie                                       | 130       |

---

## Część III. Python w Excelu

133

|                                                             |            |
|-------------------------------------------------------------|------------|
| <b>6. Rozpoczęcie pracy z Pythonem w Excelu .....</b>       | <b>135</b> |
| Wymagania                                                   | 135        |
| Pisanie kodu Pythona w Excelu                               | 136        |
| Komórki Pythona                                             | 136        |
| Rodzaje wyjścia                                             | 138        |
| Funkcja xl                                                  | 140        |
| Praca z obiektami DataFrame                                 | 142        |
| Dodawanie komentarzy                                        | 144        |
| Edytor Pythona                                              | 145        |
| Tworzenie wykresów                                          | 147        |
| Python w Excelu od kuchni                                   | 149        |
| Skrypt inicjalizacyjny                                      | 149        |
| Kolejność wykonywania komórek Pythona                       | 150        |
| Cykl przetwarzania obiektu DataFrame                        | 150        |
| Ustawienia                                                  | 152        |
| Skróty klawiszowe                                           | 152        |
| Ograniczenia                                                | 153        |
| Podsumowanie                                                | 155        |
| <b>7. Analiza szeregów czasowych za pomocą pandas .....</b> | <b>156</b> |
| DatetimeIndex                                               | 157        |
| Tworzenie DatetimeIndex                                     | 157        |
| Filtrowanie DatetimeIndex                                   | 160        |
| Typowe operacje na szeregach czasowych                      | 161        |
| Przesunięcia i zmiany procentowe                            | 161        |
| Zmiana podstawy i korelacja                                 | 163        |
| Resampling                                                  | 166        |
| Okna kroczące                                               | 167        |
| Wczytywanie danych z Power Query                            | 168        |
| Wczytywanie jednego pliku                                   | 168        |
| Wczytywanie plików z folderu                                | 170        |
| Podsumowanie                                                | 172        |

---

## Część IV. xlwings

173

|                                      |            |
|--------------------------------------|------------|
| <b>8. Automatyzacja Excela .....</b> | <b>175</b> |
| Pierwsze kroki z xlwings             | 175        |
| Excel jako przeglądarka danych       | 175        |
| Model obiektowy Excela               | 177        |
| Uruchamianie kodu VBA                | 183        |

|                                                                      |            |
|----------------------------------------------------------------------|------------|
| Konwertery, opcje i kolekcje                                         | 184        |
| Praca z obiektami DataFrame                                          | 184        |
| Konwertery i opcje                                                   | 185        |
| Wykresy, obrazy i zdefiniowane nazwy                                 | 187        |
| Studium przypadku (podejście nr 1): raportowanie w Excelu            | 190        |
| Zaawansowane zagadnienia związane z xlwings                          | 195        |
| Poprawa wydajności                                                   | 195        |
| Jak obejść brakującą funkcjonalność?                                 | 196        |
| Podsumowanie                                                         | 197        |
| <b>9. Narzędzia Excela działające w oparciu o język Python .....</b> | <b>199</b> |
| Wykorzystanie Excela jako frontendu                                  | 199        |
| Dodatek xlwings                                                      | 200        |
| Polecenie quickstart                                                 | 202        |
| Przycisk Run main                                                    | 202        |
| Funkcja RunPython                                                    | 203        |
| Wdrażanie                                                            | 207        |
| Zależność od Pythona                                                 | 208        |
| Autonomiczne skoroszyty                                              | 208        |
| Hierarchia konfiguracji                                              | 209        |
| Podsumowanie                                                         | 211        |
| <b>10. Tropiciel pakietów Pythona .....</b>                          | <b>212</b> |
| Co będziemy budować?                                                 | 212        |
| Podstawowa funkcjonalność                                            | 214        |
| Web API                                                              | 215        |
| Bazy danych                                                          | 218        |
| Wyjątki                                                              | 224        |
| Struktura aplikacji                                                  | 227        |
| Interfejs                                                            | 228        |
| Backend                                                              | 231        |
| Debugowanie                                                          | 235        |
| Podsumowanie                                                         | 236        |
| <b>11. Funkcje niestandardowe .....</b>                              | <b>237</b> |
| Pierwsze kroki z funkcjami niestandardowymi                          | 237        |
| Wymagania                                                            | 237        |
| Witaj, świecie                                                       | 238        |
| Praca z obiektami DataFrame                                          | 243        |
| Konwertery i opcje                                                   | 243        |
| Pobieranie danych z sieci                                            | 248        |
| Tworzenie wykresów                                                   | 250        |

|                                          |     |
|------------------------------------------|-----|
| Debugowanie                              | 252 |
| Optymalizacja wydajności                 | 253 |
| Minimalizacja wywołań między aplikacjami | 253 |
| Buforowanie                              | 255 |
| Dekorator script                         | 256 |
| Podsumowanie                             | 258 |

---

## Część V. xlwings Lite 259

|                                                                   |                |
|-------------------------------------------------------------------|----------------|
| <b>12. Skrypty z xlwings Lite .....</b>                           | <b>261</b>     |
| Wprowadzenie do xlwings Lite                                      | 261            |
| Jak działa xlwings Lite?                                          | 262            |
| Instalacja                                                        | 263            |
| Interfejs użytkownika                                             | 264            |
| Uruchamianie pierwszego skryptu                                   | 265            |
| Przyciski w arkuszu                                               | 266            |
| Analiza zbioru danych taksówek w Nowym Jorku                      | 267            |
| Odkrywanie zbioru danych z DuckDB                                 | 267            |
| Analiza kursów i odległości w ujęciu godzinowym                   | 271            |
| Specyfika platformy                                               | 273            |
| Wysyłanie żądań Web API                                           | 273            |
| Instalowanie pakietów Pythona                                     | 275            |
| Ograniczenia                                                      | 276            |
| Podsumowanie                                                      | 278            |
| <br><b>13. Funkcje niestandardowe z xlwings Lite .....</b>        | <br><b>279</b> |
| Nowoczesne funkcje niestandardowe                                 | 279            |
| Witaj, świecie                                                    | 280            |
| Adnotacje typów jako konwertery                                   | 280            |
| Obsługa błędów                                                    | 284            |
| Grupowanie funkcji w przestrzeniach nazw                          | 285            |
| Budowanie funkcji AI z OpenAI                                     | 286            |
| Zarządzanie danymi dostępowymi za pomocą zmiennych środowiskowych | 286            |
| Funkcja AI                                                        | 289            |
| Analiza wydźwięku tekstu z Hugging Face                           | 292            |
| Terminologia uczenia maszynowego                                  | 292            |
| Funkcja CLASSIFY                                                  | 293            |
| Wykres danych pogodowych z Open-Meteo                             | 295            |
| Praca z datami                                                    | 295            |
| Tworzenie wykresu temperatury w czasie                            | 297            |
| Podsumowanie                                                      | 300            |

|                                                                                          |                |
|------------------------------------------------------------------------------------------|----------------|
| <b>14. Operowanie plikami Excela za pomocą pandas .....</b>                              | <b>303</b>     |
| Studium przypadku (podejście nr 2): raportowanie w Excelu                                | 303            |
| Odczytywanie i zapisywanie plików Excela za pomocą pandas                                | 306            |
| Funkcja <code>read_excel</code> i klasa <code>ExcelFile</code>                           | 306            |
| Równoległy odczyt arkuszy                                                                | 310            |
| Metoda <code>to_excel</code> i klasa <code>ExcelWriter</code>                            | 312            |
| Ograniczenia związane z używaniem pandas z plikami Excela                                | 314            |
| Podsumowanie                                                                             | 315            |
| <br><b>15. Manipulowanie plikami Excela za pomocą pakietów do odczytu i zapisu .....</b> | <br><b>316</b> |
| Pakiety do odczytu i zapisu                                                              | 316            |
| Pakiety odczytujące oparte na <code>calamine</code>                                      | 318            |
| Pakiet <code>python-calamine</code>                                                      | 318            |
| Pakiet <code>fastexcel</code>                                                            | 319            |
| Pakiet <code>XlsxWriter</code>                                                           | 320            |
| Wprowadzenie do pakietu <code>XlsxWriter</code>                                          | 320            |
| Zapis dużych plików za pomocą <code>XlsxWriter</code>                                    | 323            |
| Pakiet <code>OpenPyXL</code>                                                             | 323            |
| Odczyt za pomocą <code>OpenPyXL</code>                                                   | 323            |
| Odczyt dużych plików za pomocą <code>OpenPyXL</code>                                     | 325            |
| Zapis za pomocą <code>OpenPyXL</code>                                                    | 325            |
| Zapis dużych plików za pomocą <code>OpenPyXL</code>                                      | 327            |
| Edycja za pomocą <code>OpenPyXL</code>                                                   | 328            |
| Więcej pakietów do odczytu i zapisu                                                      | 328            |
| <code>Excelize</code>                                                                    | 329            |
| <code>pyxlsb</code>                                                                      | 330            |
| <code>xlrd</code> , <code>xlwt</code> i <code>xlutils</code>                             | 331            |
| Formatowanie obiektów <code>DataFrame</code> w Excelu                                    | 335            |
| Formatowanie indeksu i nagłówek <code>DataFrame</code>                                   | 336            |
| Formatowanie części <code>DataFrame</code> zawierającej dane                             | 337            |
| Studium przypadku (podejście nr 3): raportowanie w Excelu                                | 338            |
| Podsumowanie                                                                             | 338            |
| <br><b>A Menedżer pakietów <code>uv</code> .....</b>                                     | <br><b>341</b> |
| Tworzenie nowego projektu                                                                | 341            |
| Dodawanie i usuwanie pakietów                                                            | 343            |
| Aktualizowanie pakietów i <code>uv</code>                                                | 344            |
| Praca poza katalogiem projektu                                                           | 344            |
| Inne menedżery pakietów                                                                  | 345            |

|                                                              |            |
|--------------------------------------------------------------|------------|
| <b>B Copilot w Excelu .....</b>                              | <b>346</b> |
| Rodzina narzędzi Copilot                                     | 346        |
| Inżynieria promptów                                          | 347        |
| Pierwsze kroki z Copilotem w Excelu                          | 349        |
| Wymagania                                                    | 349        |
| Rozpoczęcie rozmowy                                          | 349        |
| <b>C Zaawansowane pojęcia związane z Pythonem .....</b>      | <b>352</b> |
| Klasy i obiekty                                              | 352        |
| Praca z obiektami datetime uwzględniającymi strefę czasową   | 353        |
| Mutowalne i niemutowalne obiekty Pythona                     | 354        |
| Wywoływanie funkcji z obiektami mutowalnymi jako argumentami | 356        |
| Funkcje z obiektami mutowalnymi jako domyślnymi argumentami  | 357        |

# Rozpoczęcie pracy z Pythonem w Excelu

Popracowałeś już trochę z obiektami DataFrame z biblioteki pandas oraz notatnikami Jupyter, co daje doskonałą bazę do odkrywania Pythona w Excelu. Dlaczego? Ponieważ Python w Excelu umożliwia przekształcanie komórek Excela w komórki notatnika Jupyter. To oznacza, że Python w Excelu głównie otwiera drzwi do lepszego silnika do przeprowadzania analiz i tworzenia wykresów w postaci obiektów DataFrame z biblioteki pandas oraz wykresów Pythona. Jednak Python w Excelu nie stanowi konkurencji dla VBA i Office Scripts, gdyż za jego pomocą nie zautomatyzujesz Excela ani nie utworzysz niestandardowej funkcji, czyli funkcji, którą można wywołać bezpośrednio z komórek Excela, na przykład `=SUMA()`.

W tym rozdziale wprowadzę podstawowe koncepcje Pythona w Excelu. Nauczysz się pisać kod Pythona w komórkach Excela i dowiesz się, jak przekształcić zakres Excela na obiekt DataFrame i z powrotem na natywne wartości Excela. Ponadto poznasz narzędzia dla Pythona w Excelu, na przykład edytor Pythona. Ale najpierw upewnij się, że masz wszystko, co potrzebne do pracy z Pythonem w Excelu.

## Wymagania

Pomimo swojej nazwy Python w Excelu działa w chmurze Microsoft Azure, a nie w samym Excelu. Za każdym razem, gdy uruchamiasz komórkę z kodem Pythona, Excel wysyła powiązane z tym kodem dane arkusza do chmury, gdzie wykonuje się ten kod i skąd wyniki są wysyłane z powrotem do Excela. Dlatego wymagania są następujące:

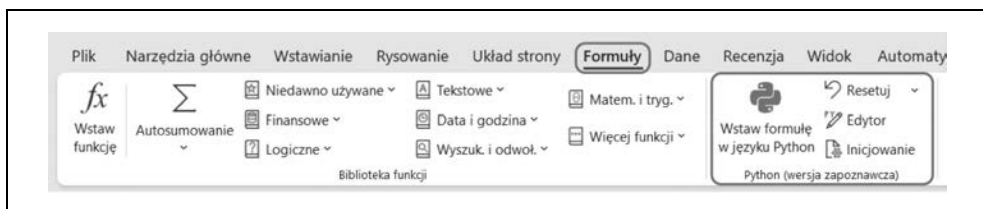
- Python w Excelu potrzebuje połączenia z internetem.
- Python w Excelu wymaga subskrypcji Microsoft 365 i nie działa w Excelu ze stałą licencją Office (np. Office 2024).



### Kod źródłowy Pythona

Choć Python w Excelu wykonuje się w chmurze, sam kod jest przechowywany w Excelu.

Aby sprawdzić, czy masz dostęp do Pythona w Excelu, przejdź do zakładki *Formuły* i zobacz, czy masz grupę *Python*, widoczną na rysunku 6.1. Jeśli nie, upewnij się, że spełniasz wymagania opisane w artykule o Pythonie w Excelu (<https://support.microsoft.com/pl-PL/Excel/python/python-in-excel-availability>), i zaktualizuj swojego Excela. Jeśli masz taką grupę w swoim menu, to świetnie, gdyż jesteś gotowy do pisania kodu.



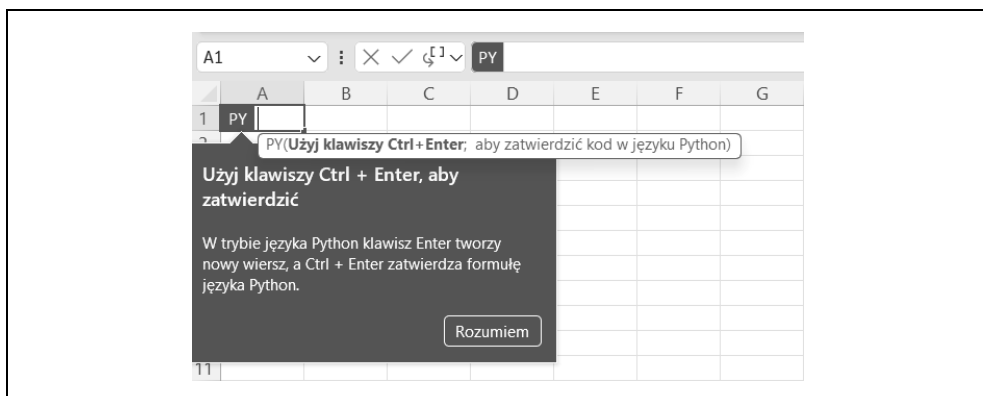
Rysunek 6.1. Menu Pythona w zakładce *Formuły* na *Wstążce* Excela

## Pisanie kodu Pythona w Excelu

W tym podrozdziale dowiesz się, że komórki Pythona mają dwa rodzaje wyjścia (*obiekt Pythona* i *wartość Excela*) i że funkcja `x1` służy do czytania wartości komórek Excela. Będziesz również tworzyć wykresy i używać edytora Pythona. Jednak zanim do tego przejdiesz, najpierw musisz dowiedzieć się, jak zamienić zwykłą komórkę Excela w komórkę Pythona.

### Komórki Pythona

Przed napisaniem kodu Pythona w komórce Excela trzeba zamienić ją w komórkę Pythona. Otwórz Excela i utwórz nowy skoroszyt, a następnie zaznacz komórkę A1. Teraz napisz `=py` i naciśnij klawisz *Tab*. Komórka A1 zmieni się w komórkę Pythona, na co wskazuje zielona ikona PY, zarówno w komórce Excela, jak i w pasku formuły (patrz rysunek 6.2).



Rysunek 6.2. Komórka Pythona

Istnieją też inne sposoby zamiany komórki Excela w komórkę Pythona:

- Zamiast naciskać klawisz *Tab*, napisz nawias otwierający: **=py(**.
- Użyj skrótu klawiaturowego *Ctrl+Alt+Shift+P* (Windows) lub *Command+Option+Shift+P* (macOS).
- Na Wstążce Excela przejdź do zakładki *Formuły* i naciśnij przycisk *Wstaw formułę w języku Python*.



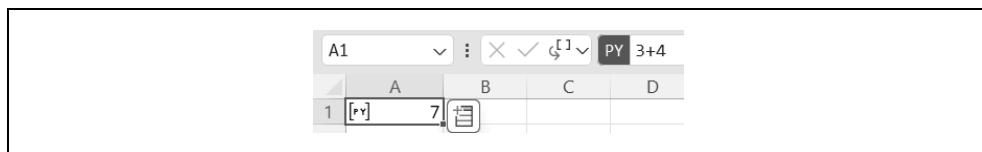
### Jak usunąć komórkę Pythona

Jeśli chcesz zamienić komórkę Pythona z powrotem w standardową komórkę Excela, zaznacz tę komórkę i naciśnij klawisz *Delete* lub po prostu wpisz coś nowego. Aby to zadziało, musisz upewnić się, że nie edytujesz tej komórki. Jeżeli ją edytujesz, najpierw naciśnij klawisz *Esc*, by przejść z edycji do zaznaczania.

Teraz komórka A1 jest komórką Pythona, więc po prostu wpisz w niej jakiś kod. Zaczniemy od tego samego przykładu co w rozdziale 2. podczas wprowadzenia notatników Jupyter:

3 + 4

Po wpisaniu **3 + 4** naciśnij *Enter*. W komórce Pythona wciśnięcie klawisza *Enter* powoduje dodanie nowej linii, a nie uruchomienie kodu. Aby uruchomić kod, należy nacisnąć *Ctrl+Enter* (Windows) lub *Command+Enter* (macOS). Teraz komórka będzie wyświetlać ikonę [PY] z wynikiem, jak widać na rysunku 6.3.



Rysunek 6.3. Python w Excelu pokazujący wynik działania 3 + 4

Podobnie do notatników Jupyter, Python w Excelu automatycznie pokazuje wynik ostatniej linii kodu w tej komórce. Jeśli jednak przypiszesz wartość do zmiennej, to w przeciwieństwie do notatników Jupyter komórka wciąż będzie pokazywać wynik. Zatem zmień komórkę A1 w taki sposób:

x = 3 + 4

Po naciśnięciu *Ctrl+Enter* (Windows) lub *Command+Enter* (macOS) znowu zobaczysz 7 w komórce. Teraz dodaj kolejną komórkę Pythona przez wpisanie **=py** w komórce A2 i naciśnięcie klawisza *Tab*. Napisz taki kod, by utworzyć listę Pythona:

```
["witaj", "świecie"]
```

Naciśnij *Ctrl+Enter* (Windows) lub *Command+Enter* (macOS), by uruchomić tę komórkę, która wyświetli [PY] list. Wynika to z tego, że Python w Excelu pokazuje wartość obiektu Pythona tylko dla prostych obiektów, takich jak liczby, łańcuchy znaków, obiekty datetime,

None i wartości logiczne. W przypadku pozostałych obiektów, w tym struktur danych, takich jak listy czy DataFrame, Python w Excelu wyświetla typ danych zamiast wartości, by zmieścić się w jednej komórce. Ale jak można sprawdzić, co zawiera ta lista? Odpowiedź znajdziesz w następnym punkcie.

## Rodzaje wyjścia

Python w Excelu ma dwa rodzaje wyjść: *obiekt Pythona* i *wartość Excela*. Przyjrzyjmy się każdemu z nich bliżej.

### Obiekt Pythona

Obiekt Pythona jest domyślnym rodzajem wyjścia dla komórek Pythona i mieści się w jednej komórce. Można go łatwo rozpoznać po ikonie `[PY]` i typie obiektu będącego wynikiem ostatniego wiersza kodu znajdującego się w tej komórce Pythona. W naszym przypadku w komórce A1 mamy 7, a w komórce A2 mamy list. Aby podejrzeć dany obiekt, naciśnij ikonę `[PY]`. Wyświetli się karta podglądu, pokazana na rysunku 6.4. Można też zaznaczyć tę komórkę i nacisnąć `Ctrl+Shift+F5` (Windows) lub `Command+Shift+F5` (macOS).



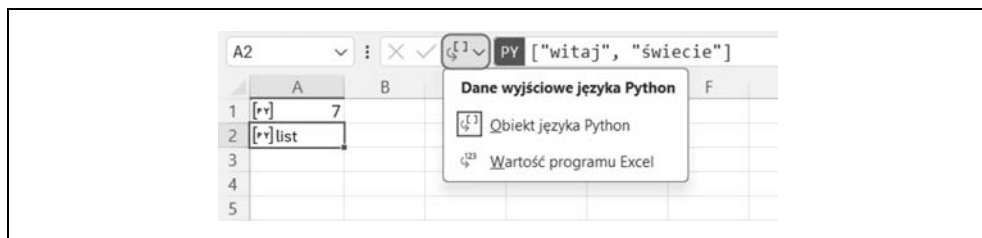
Rysunek 6.4. Karta podglądu obiektu Pythona

Aby powiększyć kartę podglądu, chwyć i przeciągnij prawy dolny róg, jak widać na rysunku 6.4. Pamiętaj, że jak sama nazwa wskazuje, to tylko podgląd, i zwykle nie będziesz widzieć wszystkich danych. Aby podejrzeć szczegóły zbioru danych, przekształć obiekt Pythona na wartości Excela, jak pokazano w następnym punkcie.

## Wartość Excela

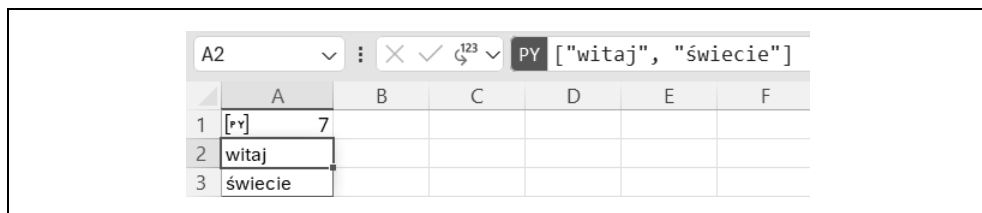
Po wykonaniu obliczeń Pythona zwykle zamienia się wyniki na wartości Excela, by przedstawić je użytkownikowi. Tak należy zamienić wyniki z obiektu Pythona na wartość Excela:

1. Zaznacz komórkę Pythona A2.
2. Naciśnij listę rozwijaną po lewej stronie paska formuły, jak pokazano na rysunku 6.5.
3. Wybierz *Wartość programu Excel*.



Rysunek 6.5. Wybór rodzaju wyjścia Pythona

Rysunek 6.6 przedstawia listę w komórce A2 po przekształceniu wyniku z obiektu Pythona na wartość Excela. Gdy komórka zawiera wartość Excela, można sprawdzić wszystkie dane, zastosować formatowanie lub dodać wykres Excela.



Rysunek 6.6. Komórka A2 po przekształceniu rodzaju wyjścia na wartość Excela

## Tablice dynamiczne

Po wybraniu wartości Excela jako rodzaju wyjścia otrzymasz tablicę dynamiczną, czyli wartości będą zapisane w wielu komórkach otoczonych niebieskim obramowaniem. Microsoft określa to mianem **rozlewania** i jeśli nie będzie wystarczającej liczby pustych komórek, do których można zapisać dane, pokaże się błąd #ROZLANIE!. Jeśli kiedykolwiek zobaczysz ten błąd, będziesz musiał usunąć niepuste komórki Excela, by zrobić miejsce dla tablicy dynamicznej. Pamiętaj, że jeśli będziesz chciał usunąć tablicę dynamiczną, musisz usunąć jej górną lewą komórkę — nie możesz usuwać pojedynczych komórek.

Poza listą rozwijaną po lewej stronie paska formuły możesz również użyć skrótu klawiszowego *Ctrl+Alt+Shift+M* (Windows) lub *Command+Option+Shift-M* (macOS), by przełączyć między rodzajami wyjścia: obiektem Pythona i wartością Excela.



## Poziome jednowymiarowe tablice dynamiczne

Tablice jednowymiarowe, takie jak listy, tablice NumPy czy obiekty Series biblioteki pandas, po zamianie na wartości Excela wyświetlają się pionowo. Aby zmienić układ na poziomy, zawiń dany obiekt w listę Pythona. Zatem aby poziomo wyświetlić listę ["witaj", "świecie"] z komórki A2, napisz `[["witaj", "świecie"]]`. To działa również dla jednowymiarowych tablic NumPy i obiektów Series z biblioteki pandas.

Tabela 6.1 pomoże Ci zrozumieć, jak obiekty Pythona są przekształcane w wartości Excela. Tabela zakłada, że bibliotekę NumPy zaimportowałeś jako np, a pandas jako pd.

Tabela 6.1. Przekształcanie typów danych Pythona w wartości Excela

| Python            | Excel                         | Komentarze                                               |
|-------------------|-------------------------------|----------------------------------------------------------|
| str               | Tekst                         |                                                          |
| int               | Liczba                        | Excel traktuje każdą liczbę jak zmiennoprzecinkową.      |
| float             | Liczba                        | Excel traktuje każdą liczbę jak zmiennoprzecinkową.      |
| bool              | Wartość logiczna              |                                                          |
| datetime.datetime | Liczba sformatowana jako data |                                                          |
| None              | 0                             | Choć wartość komórki wynosi 0, wyświetla się tekst None. |
| np.nan            | #LICZBA?                      |                                                          |
| np.inf            | #LICZBA?                      |                                                          |
| pd.NaT            | #LICZBA?                      |                                                          |
| pd.NA             | #N/D!                         |                                                          |

Takie struktury danych jak listy, tablice NumPy czy obiekty DataFrame z biblioteki pandas po zamianie obiektu Pythona na wartość Excela używają jednej komórki na element, a każdy element jest przekształcany zgodnie z tabelą 6.1. Za chwilę będziemy pracować z obiektami DataFrame.

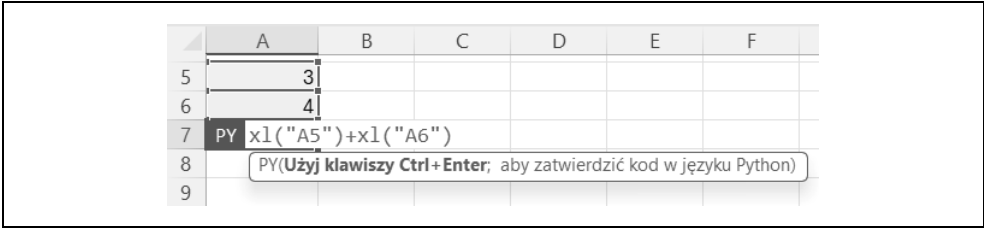
Wiesz już, jak zamieniać obiekt Pythona w wartość Excela, ale jeszcze nie wiesz, jak zamienić wartość Excela w obiekt Pythona. Tutaj przyda się funkcja `x1`.

## Funkcja `x1`

Za pomocą funkcji `x1` kod Pythona może odczytywać wartość ze zwykłej komórki Excela. Powtórzymy nasz pierwszy przykład sumujący dwie liczby, ale zamiast wpisywać to wyrażenie w komórce Pythona, składniki sumy odczytamy z komórek Excela. Wpisz **3** w komórce A5 i **4** w komórce A6. Upewnij się, że komórki A5 i A6 są zwykłymi komórkami Excela. Teraz komórkę A7 przekształć w komórkę Pythona przez wpisanie `=py` i naciśnięcie klawisza *Tab*. Podczas edytowania komórki A7 wykonaj następujące kroki:

1. Kliknij komórkę A5.
2. Wpisz znak `+`.
3. Naciśnij komórkę A6.

Kod Pythona w komórce A7 to `x1("A5")+x1("A6")`, co widać na rysunku 6.7.



Rysunek 6.7. Funkcja `x1`

Naciśnij `Ctrl+Enter` (Windows) lub `Command+Enter` (macOS), by uruchomić kod. W komórce A7 pojawi się wynik 7. Gdy zmienisz wartość w komórce A5 lub A6, komórka Pythona zostanie automatycznie obliczona ponownie. Tak naprawdę wszystkie komórki Pythona zostaną wykonane ponownie, ale o tym więcej później w tym rozdziale, w punkcie „Kolejność wykonywania komórek Pythona”.



**Bezwzględne a względne odwołania do komórek**

Jeśli chodzi o bezwzględne a względne odwoływanie się do komórek, funkcja `x1` działa tak samo jak funkcje Excela. Na przykład, jeśli komórka Pythona A1 używa `x1("A2")` i skopiujesz tę komórkę Pythona do komórki B1, to kod Pythona zmieni się w `x1("B2")`, gdyż ten kod stosuje względne odwoływanie się do komórek. Jeśli natomiast w komórce A1 napiszesz `x1("$A$2")` i skopiujesz tę komórkę do B1, to kod cały czas będzie odwoływał się do tej samej komórki, A2. Znak dolara oznacza odwołanie bezwzględne, tak samo jak w Excelu.

Zwykle nie pisze się całej funkcji `x1` ręcznie, a wybiera się odpowiednie zakresy, tak jak to zrobiliśmy przed chwilą. Funkcja `x1` obsługuje źródła danych wymienione w tabeli 6.2.

Tabela 6.2. Źródła danych akceptowane przez funkcję `x1`

| Źródło                   | Wyjaśnienie                                                                                                                                                                                                                                                                                                                |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Pojedyncze komórki       | Pojedyncze komórki są przekształcane na odpowiadający im typ danych w Pythonie, zgodnie z tabelą 6.3.                                                                                                                                                                                                                      |
| Więcej niż jedna komórka | Dwie lub więcej komórek jest przekształcanych na obiekt <code>DataFrame</code> . Więcej o tym w następnym punkcie.                                                                                                                                                                                                         |
| Zakresy nazwane          | Jeśli wybierzesz zakres nazwany, funkcja <code>x1</code> automatycznie odwoła się do tego zakresu po nazwie, np. <code>x1("ZakresNazwany")</code> .                                                                                                                                                                        |
| Tabele Excela            | Tabele Excela zwykle składają się z więcej niż jednej komórki, dlatego przekształcane są w obiekty <code>DataFrame</code> . Więcej o tym powiem w następnym punkcie.                                                                                                                                                       |
| Tabele dynamiczne        | Jeśli z poziomu komórki Pythona wybierzesz tabelę dynamiczną, funkcja <code>x1</code> będzie odwoływać się do górnej lewej komórki z dodaniem znaku <code>#</code> , np. <code>x1("A2#")</code> . Dzięki temu obiekt <code>DataFrame</code> będzie aktualizował się automatycznie przy każdej zmianie tablicy dynamicznej. |
| Power Query              | Funkcja <code>x1</code> przyjmuje również połączenia Power Query, o czym powiem więcej w następnym rozdziale.                                                                                                                                                                                                              |



## Funkcja xl nie przyjmuje zmiennych

Funkcja xl to specjalna funkcja wymagająca bezpośredniego tekstowego odwołania do źródła. Funkcja xl nie przyjmuje zmiennych.

```
data = xl("A1:C3") # To działa
ref = "A1:C3"
data = xl(ref) # To nie działa!
```

Gdy już wiesz, jakie odwołania do źródła przyjmuje funkcja xl, zobaczmy, jak funkcja ta przekształca wartości Excela, do których się odwołuje, na obiekty Pythona (patrz tabela 6.3).

Tabela 6.3. Przekształcanie typów danych z Excela na Pythona

| Excel                         | Python                  | Komentarz                                                                                 |
|-------------------------------|-------------------------|-------------------------------------------------------------------------------------------|
| Tekst                         | str                     |                                                                                           |
| Liczba                        | int lub float           | Excel traktuje każdą liczbę jak zmiennoprzecinkową.                                       |
| Wartość logiczna              | bool                    |                                                                                           |
| Liczba sformatowana jako data | datetime.datetime       |                                                                                           |
| Pusta komórka                 | None                    |                                                                                           |
| Błąd w komórce (np. #N/D!)    | Błąd komórki lub np.nan | Dla pojedynczych komórek wyświetla się błąd komórki, a obiekty DataFrame pokazują np.nan. |



## Nie używaj funkcji xl, by odwoływać się do innych komórek Pythona

Choć funkcja xl umożliwia odwoływanie się do innej komórki Pythona, którego wyjście jest obiektem Pythona, odradzam to rozwiązanie. Kod będzie kruchy i przestanie działać, gdy tylko dodasz inną linię kodu w komórce, do której się odwołujesz, lub gdy zmienisz jej rodzaj wyjścia z obiektu Pythona na wartość Excela. Lepszym rozwiązaniem jest przypisanie obiektu Pythona, np. obiektu DataFrame, do zmiennej i używanie tej zmiennej w innych komórkach Pythona.

Wszystko, co robiliśmy do tej pory, znajdziesz w pliku Excela *python\_in\_excel\_hello\_world.xlsx* w folderze *xl* w materiałach do pobrania dla tej książki.

Jak już kilkakrotnie wspominałem w tym podrozdziale, gdy w funkcji xl odwołasz się do więcej niż jednej komórki, Python w Excelu zwróci obiekt DataFrame z biblioteki pandas. Przyjrzyjmy się temu bliżej.

## Praca z obiektami DataFrame

Obiekty DataFrame z biblioteki pandas leżą u podstaw Pythona w Excelu. W tym podrozdziale będziemy ćwiczyć w pliku *fitness\_tracker.xlsx*, który znajdziesz w folderze *xl* w materiałach do pobrania dla tej książki. Znajdziesz w nim kilka kategorii danych, które zwykle są zbierane przez opaski monitorujące treningi lub aplikacje monitorujące zdrowie w smartfonach (patrz rysunek 6.8).

|   | A         | B      | C              | D       | E             | F                  |
|---|-----------|--------|----------------|---------|---------------|--------------------|
| 1 | Data      | Kroki  | Odległość (km) | Kalorie | Sen (godziny) | Komentarz          |
| 2 | 1.06.2025 | 8 200  | 6,1            | 320     | 7,2           |                    |
| 3 | 2.06.2025 | 10 500 | 7,9            | 390     | 6,5           | Odczucie zmęczenia |
| 4 | 3.06.2025 | 6 000  | 4,5            | 280     |               | Nie zapisano snu   |
| 5 | 4.06.2025 | 12 300 | 9              | 450     | 6,8           | Z grupą biegaczy   |

Rysunek 6.8. Plik *fitness\_tracker.xlsx*

W celu przeprowadzenia analizy tego miniaturowego zbioru danych zmienimy tabelę Excela w obiekt `DataFrame`. Zamień komórkę I1 w komórkę Pythona, a następnie wpisz `df =` i zaznacz zakres A1:F5. W miejscu, w którym znajduje się kursor, pojawi się to:

```
xl("DaneFitness[#Wszytsko]", headers=True)
```

Cała komórka ma zatem następującą zawartość:

```
df = xl("DaneFitness[#Wszytsko]", headers=True)
```

Nasze dane znajdują się w tabeli Excela o nazwie `DaneFitness`, a funkcja `xl` automatycznie odwołuje się do niej jako `DaneFitness[#Wszytsko]`. Składnia używana w Excelu do odwołania się do tabel Excela określana jest mianem **odwołań strukturalnych**, a `[#Wszytsko]` oznacza odwołanie do całej tabeli, wraz z nagłówkami i podsumowaniami. Zatem jeśli samodzielnie wpisujesz odwołanie, nie zapomnij dodać `[#Wszytsko]`. Jeśli chcesz dowiedzieć się więcej o składni odwołań strukturalnych dla tabeli Excela, sprawdź dokumentację Microsoftu (<https://support.microsoft.com/pl-pl/excel/using-structured-references-with-excel-tables>). Argument `headers=True` został dodany automatycznie, gdyż Excel wykrył, że pierwszy wiersz zawiera nagłówki. Dzięki temu obiekt `DataFrame` również użyje pierwszego wiersza jako nagłówków kolumn. W naszym przypadku pierwszy wiersz odpowiada nagłówkom tabeli Excela.

Teraz naciśnij *Enter*, by dodać drugą linię kodu, w której ustawimy indeks obiektu `DataFrame`. Bardzo często kolumna daty służy jako indeks `DataFrame`, o czym przekonasz się w rozdziale 7.

```
df = xl("DaneFitness[#Wszytsko]", headers=True)
df = df.set_index("Data")
```



### True a TRUE

Podczas pracy z Excelem mogłeś przywyknąć do pisania wartości `TRUE` z wielkich liter, gdyż tak ta wartość jest zapisywana w formułach Excela. Jednak w Pythonie trzeba używać odpowiedniego zapisu, w którym tylko pierwsza litera jest wielka: `True`.

Uruchom kod za pomocą *Ctrl+Enter* (Windows) lub *Command+Enter* (macOS). Twój arkusz powinien wyglądać tak jak na rysunku 6.9.

Zmiana rozmiaru

|   | A         | B      | C              | D       | E             | F                  | G | H | I             |
|---|-----------|--------|----------------|---------|---------------|--------------------|---|---|---------------|
|   | Data      | Kroki  | Odległość (km) | Kalorie | Sen (godziny) | Komentarz          |   |   |               |
| 1 |           |        |                |         |               |                    |   |   | [r] DataFrame |
| 2 | 1.06.2025 | 8 200  | 6,1            | 320     | 7,2           |                    |   |   |               |
| 3 | 2.06.2025 | 10 500 | 7,9            | 390     | 6,5           | Odczucie zmęczenia |   |   |               |
| 4 | 3.06.2025 | 6 000  | 4,5            | 280     |               | Nie zapisano snu   |   |   |               |
| 5 | 4.06.2025 | 12 300 | 9              | 450     | 6,8           | Z grupą biegaczy   |   |   |               |

Rysunek 6.9. Przekształcenie tabeli Excela w obiekt DataFrame



## Poszerzanie paska formuły

Jeśli chcesz odczytać kod Pythona w pasku formuły, możesz przeciągnąć jego dolną krawędź w dół, by rozciągnął się na kilka wierszy, tak jak na rysunku 6.9.

Teraz zamień komórkę I2 w komórkę Pythona (jak wcześniej: wpisz `=py` i naciśnij klawisz *Tab*). Aby uzyskać statystyki dla naszych danych, dodaj ten kod, który widziałeś już w rozdziale 5.:

```
df.describe()
```

Naciśnij *Ctrl+Enter* (Windows) lub *Command+Enter* (macOS), by uruchomić kod. Teraz zmień rodzaj wyjścia komórki I2 na wartości Excela. W tym celu zaznacz tę komórkę i kliknij przycisk znajdujący się po lewej stronie paska formuły, jak pokazano wcześniej na rysunku 6.5. W ten sposób zobaczysz opisowe statystyki zwrócone przez metodę `describe`, jak widać na rysunku 6.10.

|  | H | I             | J          | K             | L          | M             |
|--|---|---------------|------------|---------------|------------|---------------|
|  |   | [r] DataFrame |            |               |            |               |
|  |   |               | Kroki      | Odległość (kr | Kalorie    | Sen (godziny) |
|  |   | count         | 4          | 4             | 4          | 3             |
|  |   | mean          | 9250       | 6,875         | 360        | 6,83333333    |
|  |   | std           | 2740,43792 | 1,98389348    | 75,2772653 | 0,35118846    |
|  |   | min           | 6000       | 4,5           | 280        | 6,5           |
|  |   | 25%           | 7650       | 5,7           | 310        | 6,65          |
|  |   | 50%           | 9350       | 7             | 355        | 6,8           |
|  |   | 75%           | 10950      | 8,175         | 405        | 7             |
|  |   | max           | 12300      | 9             | 450        | 7,2           |

Rysunek 6.10. Statystyki opisowe

Przed dodaniem kolejnych komórek Pythona zróbmy w nich porządek!

## Dodawanie komentarzy

Jak się dowiedziałeś w rozdziale 2., w klasycznym notatniku Jupyter można używać albo komórek Markdown, albo komentarzy, by opisać, co robi dany fragment kodu. Jednak Python w Excelu nie oferuje komórek Markdown, a komentarze Pythona są ukryte, jeśli nie klikniesz komórki. Na szczęście istnieją dwa sposoby dodawania komentarzy dla komórek Pythona:

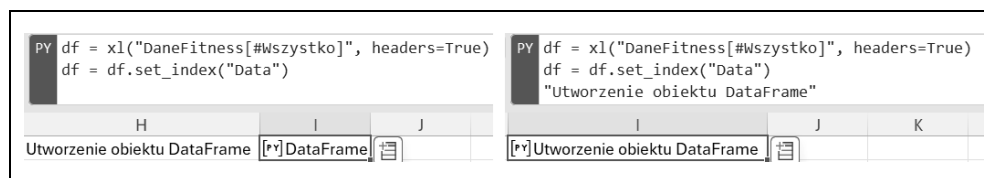
## Zwykłe komórki Excela

Prosty sposób na dodanie komentarza do komórki Pythona polega na wpisaniu tego komentarza do zwykłej komórki Excela, zwykle do tej, która sąsiaduje z lewej strony z komórką Pythona.

## Obiekt Pythona

Gdy używasz obiektu Pythona jako wyjścia, możesz też posłużyć się łańcuchem znaków w ostatniej linii kodu Pythona w danej komórce. Jak już widziałeś wcześniej w tym rozdziale, Python w Excelu wyświetli ten łańcuch znaków bezpośrednio w komórce.

Rysunek 6.11 przedstawia porównanie tych dwóch podejść.



Rysunek 6.11. Użycie zwykłej komórki Excela (po lewej) oraz obiektu Pythona (po prawej) jako sposób na dodanie komentarza

W tej książce będę używał zwykłych komórek Pythona, gdyż to podejście działa z dwoma rodzajami wyjścia: obiektami Pythona i wartościami Excela.

Poza dodawaniem komentarzy dla komórek Pythona Python w Excelu oferuje inny sposób na lepszą kontrolę nad komórkami w Pythonie: edytor Pythona.

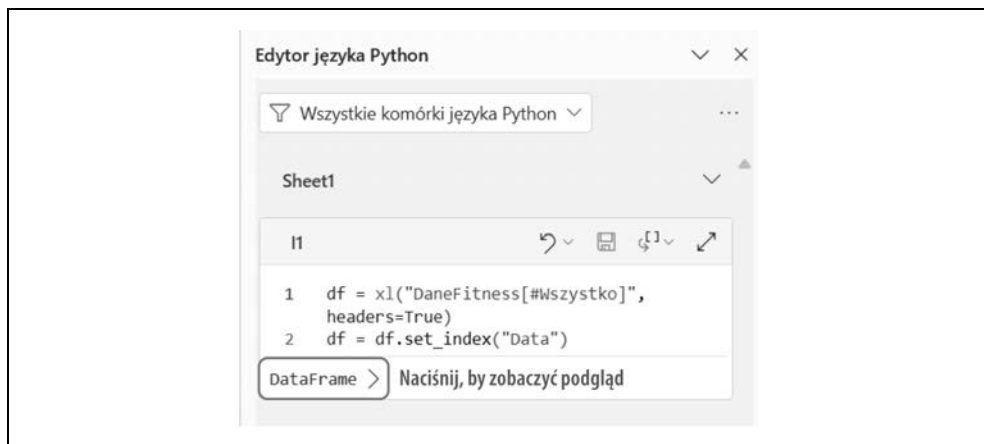
## Edytor Pythona

W porównaniu z tym, co oferują komórki Excela i pasek formuły, edytor Pythona zapewnia wygodniejszy sposób edytowania kodu Pythona. Aby go otworzyć, na Wstążce Excela przejdź do zakładki *Formuły* i naciśnij znajdujący się w grupie *Python* przycisk *Edytor* (patrz rysunek 6.12).



Rysunek 6.12. Uruchamianie edytora Pythona z poziomu Wstążki Excela

Możesz też nacisnąć prawym przyciskiem myszy komórkę Pythona i wybrać *Pokaż edytor* albo użyć skrótu klawiszowego *Ctrl+Alt+Shift+F2* (Windows) lub *Command+Option+Shift+F2* (macOS). Po prawej stronie Excela otworzy się edytor. Jak widać na rysunku 6.13, edytor Pythona wygląda podobnie jak notatnik Jupyter.



Rysunek 6.13. Edytor Pythona z kodem z komórki I1

Oto kilka zalet używania edytora zamiast pisania kodu bezpośrednio w komórkach lub pasku formuły:

### Całościowy widok

Jeśli masz wiele komórek Pythona porozrzucanych po różnych arkuszach, edytor pokaże je w kolejności wykonywania, od góry do dołu. Więcej o kolejności powiem później w tym rozdziale, w punkcie „Kolejność wykonywania komórek Pythona”.

### IntelliSense

Automatyczne uzupełnianie kodu, nazywane przez firmę Microsoft *IntelliSense*, umożliwia wybór atrybutów obiektów z listy, więc nie trzeba ich wszystkich pamiętać. Edytor Pythona oferuje dużo lepsze uzupełnianie kodu w porównaniu z pisanem kodu samodzielnie w komórce lub pasku formuły.

### Podgląd wyników

Edytor pokaże wynik pod komórką kodu, podobnie jak notatniki Jupyter. Zwróć uwagę, że wiele obiektów, między innymi DataFrame, jest domyślnie zwiniętych, więc musisz kliknąć link poniżej komórki, by zobaczyć pełny wynik, co widać na rysunku 6.13.

### Wyświetlanie wyniku i komunikatów o błędach

W edytorze wyświetlają się wyniki i informacje o błędach. Pojawiają się automatycznie po użyciu funkcji print lub gdy wystąpi błąd. Jeśli Ci się to nie podoba, można to zmienić w ustawieniach (wyjaśnienie znajdziesz w następnym podrozdziale, w punkcie „Ustawienia”).

### Samodzielny edytor

Jeśli chcesz, by edytor Pythona działał w osobnym oknie, przytrzymaj edytor za mały szary pasek i przeciągnij go na zewnątrz Excela. To umożliwi pełnoekranowy tryb edytora. Ale nawet jeśli zostawisz edytor w Excelu, możesz go powiększyć przez przeciągnięcie lewej krawędzi w stronę środka okna Excela.

## Skróty klawiszowe

Edytor obsługuje wiele skrótów klawiszowych, które możesz znać z edytora Visual Studio Code, na przykład *Ctrl*+*/* (Windows) lub *Command*+*/* (macOS) do dodawania/usuwania znacznika komentarza z wybranej linii kodu.



### Funkcja xl w edytorze Pythona

Gdy piszę tę książkę, po zaznaczeniu zakresu w Excelu edytor Pythona nie wstawia automatycznie funkcji *xl*. Musisz napisać ją samodzielnie lub (co załecam) musisz edytować komórkę Pythona bezpośrednio w siatce Excela lub pasku formuły za każdym razem, gdy chcesz dodać odwołanie za pomocą funkcji *xl*.

W następnym punkcie będziemy dodawać wykresy za pomocą edytora Pythona.

## Tworzenie wykresów

Python w Excelu obsługuje kilka bibliotek do tworzenia bibliotek, w tym Matplotlib i seaborn. Bibliotekę Matplotlib poznałeś już w rozdziale 5., w którym używaliśmy metody *plot* obiektu *DataFrame*. W tym punkcie będziemy robić to samo, co pozwoli Ci zrozumieć, jak Python w Excelu działa z wykresami.

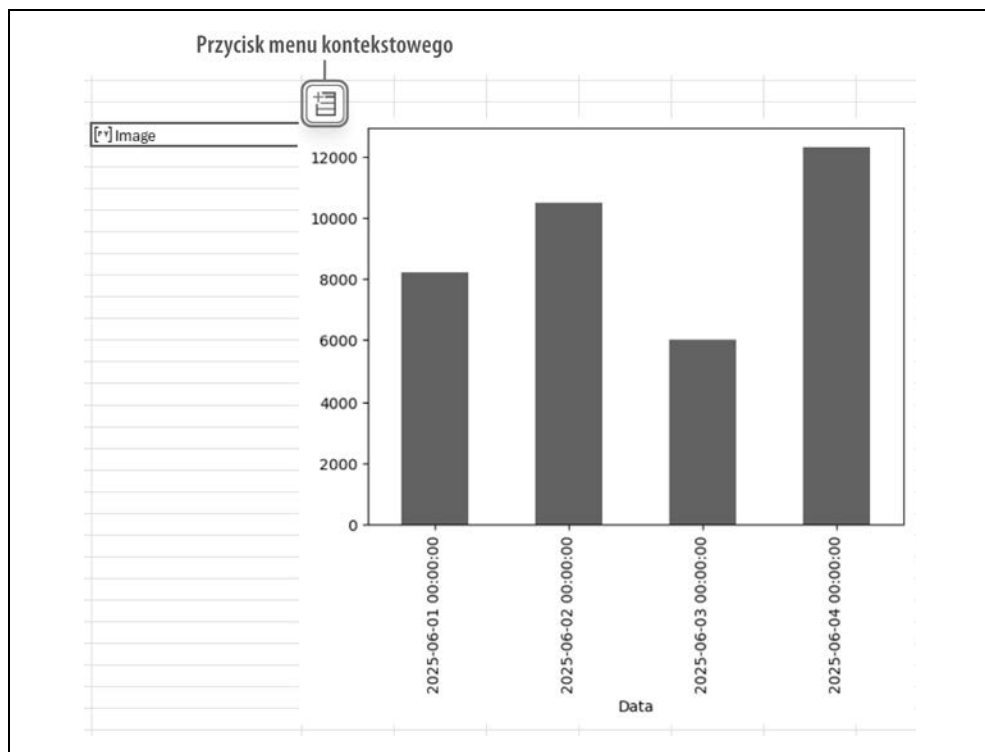
Kliknij komórkę I12, a następnie otwórz edytor Pythona (patrz poprzedni punkt) i naciśnij przycisk *Dodaj komórkę języka Python*, znajdujący się na dole edytora. W ten sposób komórka I12 zmieni się w komórkę Pythona. By utworzyć wykres z liczbą kroków, wpisz w tej komórce taki kod:

```
df["Kroki"].plot.bar()
```

Uruchom kod za pomocą *Ctrl*+*Enter* (Windows) lub *Command*+*Enter* (macOS). W komórce I12 wyświetli się [PY] Image. Aby zobaczyć wykres, w edytorze Pythona naciśnij przycisk *PngImageFile* znajdujący się pod tą komórką lub naciśnij ikonę [PY] bezpośrednio w komórce I12, by wyświetlić kartę podglądu.

Jednak w większości przypadków będziesz chciał pokazywać wykresy tak samo, jak robiłeś to w Excelu. W tym celu naciśnij prawym przyciskiem myszy komórkę Pythona z obrazem i wybierz *Wyświetl wykres nad komórkami*. W arkuszu zostanie dodany wykres jako obraz. Możesz śmiało zmieniać jego rozmiar i przesuwać go w inne miejsce arkusza. Możesz też kliknąć przycisk menu kontekstowego i z tego poziomu wybrać *Wyświetl wykres nad komórkami*, jak pokazano na rysunku 6.14.

Biblioteka Matplotlib działa dobrze, ale wykresy przez nią domyślnie zwracane nie zawsze są doskonałe. Na przykład oś daty pokazuje godzinę, nawet jeśli to zawsze jest 00:00:00. Możesz zastosować różne sztuczki, by sformatować oś daty, ale to dobra okazja, by wypróbować pakiet seaborn. Choć seaborn bazuje na Matplotlib, domyślnie rysuje schludniejsze wykresy, a poza tym oferuje wykresy statystyczne.



Rysunek 6.14. Wyświetlanie wykresu Pythona nad komórkami Excela



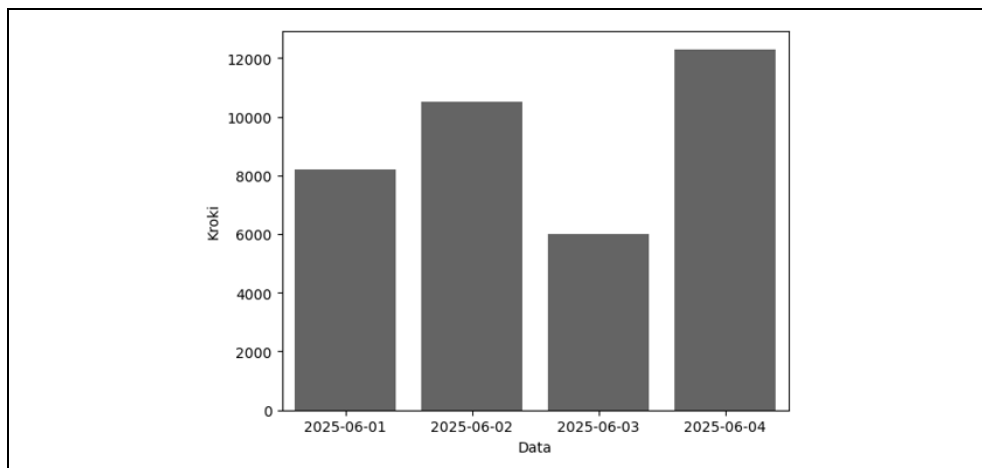
### Wskazówki dotyczące tworzenia wykresu

- Jeśli próbujesz zmienić ustawienia swojego wykresu lub sprawdzasz inne rodzaje wykresów, to Twój wykres może zostać zaburzony. Aby to naprawić, kliknij prawym przyciskiem myszy swój wykres i wybierz *Resetuj obraz i rozmiar*.
- Jeśli wolisz wyświetlić swój wykres w postaci obrazu w komórce, zaznacz odpowiednią komórkę Pythona z [PY] Image i zmień rodzaj wyjścia z obiektu Pythona na wartość Excela.
- Poza Matplotlib i seaborn Python w Excelu obsługuje bibliotekę Plotnine (<https://plotnine.org/>), której składnia jest podobna do składni pakietu ggplot2 dla języka R.

Zacznij od zamiany komórki I13 w komórkę Pythona, a następnie napisz taki kod, by utworzyć wykres kolumnowy pakietu seaborn:

```
sns.barplot(df.reset_index(), x="Data", y="Kroki")
```

Uruchom tę komórkę Pythona, a następnie kliknij ją prawym przyciskiem myszy i wybierz *Wyświetl wykres nad komórkami*. Teraz wykres wygląda bezsprzecznie lepiej, co widać na rysunku 6.15.



Rysunek 6.15. Wykres seaborn

Ale jak to? Nie zaimportowaliśmy pakietu seaborn jako sns (`import seaborn as sns`), a nasz kod zadziałał! Dowiedzmy się, jak to możliwe.

## Python w Excelu od kuchni

W tym podrozdziale zajrzemy głębiej, by zrozumieć kilka narzędzi i koncepcji używanych przez Pythona w Excelu. Omówimy między innymi skrypt inicjalizacyjny, kolejność wykonywania komórek Pythona i ustawienia. Zaczniemy od skryptu inicjalizującego.

### Skrypt inicjalizacyjny

W poprzednim podrozdziale użyliśmy wyrażenia `sns.barplot` i nie musieliśmy go poprzedzać instrukcją `import seaborn as sns`. To działa dzięki **skryptowi inicjalizującemu**. Jest to skrypt, który uruchamia się automatycznie po uruchomieniu Pythona w Excelu. Aby go otworzyć, naciśnij przycisk *Inicjowanie* w zakładce *Formuły* na Wstążce Excela. Skrypt inicjalizujący importuje kilka najpopularniejszych bibliotek:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels as sm
import seaborn as sns
```

Wszystkich tych aliasów można używać w komórkach Pythona bez konieczności importowania pakietu. Zawartość tego pliku może się z czasem zmieniać, zatem od czasu do czasu warto go sprawdzać.



#### Skrypt inicjalizacyjny jest zapisany w skoroszycie

Podobnie jak kod, który wpisujesz w komórkach Pythona, kod skryptu inicjalizującego jest zapisany w samym skoroszycie.

W skrypcie inicjalizującym możesz importować inne biblioteki i dodawać w nim definicje funkcji. Jeśli używasz starszej wersji Pythona w Excelu, to możliwe, że edycja tego skryptu nie będzie możliwa. W takim przypadku możesz zamienić komórkę A1 w pierwszym arkuszu swojego skoroszytu w komórkę Pythona i wpisać w niej dodatkowe instrukcje importowania. Dlaczego akurat w tym miejscu, w pierwszym arkuszu, w komórce A1? Ponieważ Python wykonuje kod w określonej kolejności, o czym powiem zaraz.

## Kolejność wykonywania komórek Pythona

Python w Excelu wykonuje kod od lewej do prawej, a następnie od góry do dołu. W przeciwieństwie do zwykłych komórek Excela, gdzie kolejność wykonywania obliczeń jest definiowana w oparciu o zależności między komórkami, komórki Pythona w każdym arkuszu są po prostu wykonywane od lewej do prawej i z góry na dół. Ta logika uwzględnia arkusze, które są przetwarzane od lewej do prawej.

Wprowadzony wcześniej w tym rozdziale edytor Pythona pokazuje sposób wykonywania komórek, wyświetlając je w kolejności, w jakiej są uruchamiane, podobnie jak w notatnikach Jupyter. I tak samo jak w notatnikach Jupyter wszystkie komórki Pythona wykonują się w tej samej globalnej przestrzeni nazw, co oznacza, że zmienne zdefiniowane w jednej komórce są dostępne dla pozostałych komórek Pythona.

Aby zobaczyć kolejność wykonywania się komórek, wróć do pliku *fitness\_tracker.xlsx*, zmień komórkę H1 w komórkę Pythona i wpisz w niej **df** (komentarz *Utworzenie obiektu DataFrame* zostanie nadpisany). Po uruchomieniu kodu za pomocą *Ctrl+Enter* (Windows) lub *Command+Enter* (macOS) otworzy się edytor Pythona i wyświetli się taki błąd: `NameError: name 'df' is not defined` (nazwa „df” nie została zdefiniowana). Jak można łatwo zauważyć w edytorze, komórka H1 jest pierwszą komórką Pythona i dlatego wykonuje się przed komórką I1, w której zdefiniowaliśmy **df**.

Przenieś kod z komórki H1 do A7 pod naszą tabelę Excela. Jak widać w edytorze, komórka A7 wykonuje się po I1, gdzie zdefiniowany jest obiekt **df**, zatem teraz Python nie zwraca błędu. Zmień rodzaj wyjścia komórki A7 z obiektu Pythona na wartość Excela. Przypominam, że możesz to zrobić za pomocą listy rozwijanej znajdującej się po lewej stronie paska formuły albo za pomocą skrótu klawiszowego *Ctrl+Alt+Shift+M* (Windows) lub *Command+Option+Shift+M* (macOS). Jeśli porównasz wartości z wartościami pierwotnymi w Excelu, zauważysz kilka różnic, które przeanalizujemy w następnym punkcie.

## Cykl przetwarzania obiektu DataFrame

Mając pierwotną tabelę Excela i wartości naszego obiektu **DataFrame** jedna nad drugą (patrz rysunek 6.16), będzie Ci łatwiej zrozumieć, co się dzieje podczas zamiany danych Excela w obiekt **DataFrame** i z powrotem na wartości Excela.

|    |           |        |                |         |               |                    |
|----|-----------|--------|----------------|---------|---------------|--------------------|
| A7 |           |        |                |         |               |                    |
|    | A         | B      | C              | D       | E             | F                  |
| 1  | Data      | Kroki  | Odległość (km) | Kalorie | Sen (godziny) | Komentarz          |
| 2  | 1.06.2025 | 8 200  | 6,1            | 320     | 7,2           |                    |
| 3  | 2.06.2025 | 10 500 | 7,9            | 390     | 6,5           | Odczucie zmęczenia |
| 4  | 3.06.2025 | 6 000  | 4,5            | 280     |               | Nie zapisano snu   |
| 5  | 4.06.2025 | 12 300 | 9              | 450     | 6,8           | Z grupą biegaczy   |
| 6  |           |        |                |         |               |                    |
| 7  |           | Kroki  | Odległość (km) | Kalorie | Sen (godziny) | Komentarz          |
| 8  | Data      |        |                |         |               |                    |
| 9  | 1.06.2025 | 8200   | 6,1            | 320     | 7,2           | None               |
| 10 | 2.06.2025 | 10500  | 7,9            | 390     | 6,5           | Odczucie zmęczenia |
| 11 | 3.06.2025 | 6000   | 4,5            | 280     | #LICZBA!      | Nie zapisano snu   |
| 12 | 4.06.2025 | 12300  | 9              | 450     | 6,8           | Z grupą biegaczy   |

Rysunek 6.16. Cykl przetwarzania obiektu DataFrame

W naszym przypadku podczas zamiany rodzaju wyjścia zostało wprowadzonych kilka zmian:

### Pusty wiersz po wierszu tytułowym

Python w Excelu dodaje pusty wiersz między nagłówkami i pierwszym wierszem z danymi, gdy obiekt DataFrame ma indeks inny niż domyślny (indeks ustawiliśmy na kolumnę Data w komórce I1). Python w Excelu próbuje naśladować sposób, w jaki notatniki Jupyter wyświetlają obiekty DataFrame. Jeśli chcesz się pozbyć tego pustego wiersza, zresetuj indeks:

```
df.reset_index()
```

Możliwe, że pamiętasz z rozdziału 5., że resetowanie indeksu zastępuje bieżący indeks domyślnym indeksem w postaci liczb całkowitych. Jednak nasz indeks Data był wyświetlany, a indeks w postaci liczb całkowitych nie. Python w Excelu pokazuje tylko niedomyślne indeksy. Zresetowanie indeksu oznacza, że bieżący indeks jest zamieniany w zwykłą kolumnę obiektu DataFrame.

### Brak formatowania

Przez zmianę rodzaju wyjścia na wartości Excela otrzymujesz niesformatowaną tablicę dynamiczną. Choć możesz zastosować formatowanie, na przykład pogrubić czcionkę nagłówków czy zmienić kolor tła komórek, nie możesz zamienić tablicy dynamicznej na tabelę Excela, gdyż tablice dynamiczne i tabele Excela nie są ze sobą kompatybilne.

### Puste komórki

Jak już widzieliśmy wcześniej w tabeli 6.3, przechodzące z Excela do Pythona puste komórki są przekształcane na wartość None. Ponieważ funkcja xl niezwłocznie przekształca dane w obiekt DataFrame, None zmienia się w np.nan. To z kolei ma wpływ na to, jak pierwotne puste komórki zostaną zwrócone do Excela. Wartość np.nan zostanie zamieniona na błąd komórki #LICZBA!, a None na 0, ale będzie wyświetlana jako None, zgodnie z tabelą 6.1.

Jeśli chcesz, by puste komórki pozostały puste, możesz zastąpić brakujące wartości znakami cudzysłowu, df.fillna(""). Pamiętaj jednak, że to spowoduje przekształcenie kolumn liczbowych z brakującymi wartościami na typ object. Dlatego stosuj tę sztuczkę tylko na potrzeby wyświetlania, a nie dla obliczeń w Pythonie.

Poznałeś już wszystkie podstawowe funkcjonalności Pythona w Excelu. W materiałach do pobrania dla tej książki, w folderze *xl*, znajdziesz plik Excela o nazwie *fitness\_tracker\_end.xlsx* z wszystkim, co robiliśmy w tym rozdziale.

Czasami trzeba zmienić sposób działania Pythona w Excelu, dlatego teraz omówimy ustawienia, jakie oferuje Python w Excelu.

## Ustawienia

Aby przejść do ustawień Pythona w Excelu, idź do *Plik/Opcje/Zaawansowane* (Windows) lub *Excel/Preferencje/Obliczenia* (macOS). W sekcji *Podczas obliczania w tym skoroszybie* znajdziesz takie dwie opcje:

### *Automatycznie pokazuj edytor języka Python*

Ta opcja jest zaznaczona domyślnie, co powoduje automatyczne otwieranie się edytora Pythona, jeśli kod w komórce Pythona zwraca błąd lub używa funkcji `print`. Ponieważ metoda `info` obiektu `DataFrame` używa funkcji `print`, po uruchomieniu `df.info()` również otworzy edytor Pythona.

### *Limit czasu formuły języka Python*

Jak się dowiedziałeś na początku tego rozdziału, Python w Excelu wykonuje kod w chmurze. Domyślnie każda formuła, która działa dłużej niż 30 sekund, zostanie przerwana i zwróci błąd `#TIMEOUT!`. Aby zmienić ten limit, dostosuj wartość przy opcji *Limit czasu formuły języka Python (w sekundach)*.

Jeśli chcesz wyczyścić wszystkie istniejące zmienne i ponownie uruchomić wszystkie obliczenia, użyj przycisku *Resetuj*. Nie znajdziesz go w opcji Excela, ale na Wstążce, w zakładce *Formuły*. Ten przycisk oferuje kilka opcji:

### *Resetuj*

Przycisk *Resetuj* usuwa wszystkie globalne zmienne Pythona i ponownie wykonuje obliczenia. Jest to szczególnie przydatne przy wykonywaniu częściowych obliczeń, które wymagają dodatku *Język Python w programie Excel* (patrz „Dodatek Język Python w programie Excel” później w tym rozdziale).

### *Resetuj środowisko uruchomieniowe*

Opcja *Resetuj środowisko uruchomieniowe* restartuje środowisko uruchomieniowe Pythona i powoduje, że wszystkie obliczenia są wykonywane ponownie. Opcja ta przydaje się, gdy występują błędy `#ZAJĘTY!` i `#POŁĄCZ!`.

Gdy wiesz już, jak zmieniać ustawienia i do czego używać przycisku *Resetuj*, przypomnijmy sobie najważniejsze skróty klawiszowe.

## Skróty klawiszowe

Ten punkt zbiera wszystkie skróty klawiszowe (patrz tabela 6.4) poznane w tym rozdziale. Jeśli pracujesz w systemie macOS, zastąp *Ctrl* klawiszem *Command*, a *Alt* klawiszem *Option*.

Tabela 6.4. Skróty klawiszowe

| Skrót klawiszowy         | Wyjaśnienie                                                              |
|--------------------------|--------------------------------------------------------------------------|
| <i>Ctrl+Enter</i>        | Uruchamia komórki Pythona.                                               |
| <i>Ctrl+Alt+Shift+M</i>  | Przełącza między rodzajami wyjścia: obiektem Pythona a wartością Excela. |
| <i>Ctrl+Alt+Shift+F2</i> | Otwiera edytora Pythona.                                                 |
| <i>Ctrl+Alt+Shift+C</i>  | Użyty na komórkach [PY] Image pokazuje obraz nad komórkami Excela.       |
| <i>Ctrl+Shift+F5</i>     | Otwiera kartę podglądu (wymaga, by komórka zwracała obiekt Pythona).     |

Na koniec tego rozdziału omówimy ograniczenia Pythona w Excelu.

## Ograniczenia

Python w Excelu skupia się na wąskich zastosowaniach: poprzez obiekt biblioteki pandas zapewnia narzędzia do analizy o bardzo wielkiej mocy oraz wizualizacje. Ma sporo ograniczeń. Jednak pamiętaj, że gdy piszę tę książkę, Python w Excelu ma tylko rok i niektóre z tych ograniczeń mogą być z czasem zniesione:

### Ograniczenia użycia

Python w Excelu ma zarówno dzienny, jak i miesięczny limit użycia. Wprowadźcie Microsoft nie podaje, ile wynosi ten limit, ale gdy go przekroczysz, pojawi się błąd #BLOCKED!.

### Wykonuje się w chmurze

Kod Pythona w Excelu wykonuje się w chmurze, co wpływa na prywatność i uniemożliwia używanie tej funkcjonalności bez internetu.

### Stały zestaw pakietów

Python w Excelu ma stały zestaw pakietów i nie można zainstalować dodatkowych. Aby sprawdzić, jakie pakiety są zainstalowane, uruchom ten kod w komórce Pythona i zmień rodzaj wyjścia na wartości Excela:

```
import pkg_resources
sorted([[pkg.key, pkg.version] for pkg in pkg_resources.working_set])
```

Jednak to, że pakiet jest zainstalowany, nie oznacza, że jest wspierany przez Pythona w Excelu.

### Brak bezpośredniego dostępu do zewnętrznych plików i systemów

Python w Excelu nie pozwala na pobieranie danych z internetu i nie może łączyć się z bazami danych i zewnętrznymi plikami. Jeśli musisz pobrać dane z zewnętrznych źródeł, użyj Power Query, a za pomocą Pythona w Excelu odwołuj się do połączenia Power Query za pomocą funkcji `x1`. W następnym rozdziale zobaczymy, jak to działa.

### Stałe powiązanie z obiektami DataFrame

Jeśli funkcja `x1` odwołuje się do więcej niż jednej komórki, to zwraca obiekt DataFrame. Nie można w łatwy sposób zwrócić listy Pythona czy tablicy NumPy. Jeśli częściej pracujesz z tablicami NumPy i listami niż z obiektami DataFrame, dodaj taki kod na końcu swojego skryptu inicjalizującego:

```
def custom_converter(x, headers=False, convert=None, **kwargs):
    # Zostaw domyślne bez zmian
    if convert is None or convert == pd.DataFrame:
        return excel.convert_to_dataframe(x, headers=headers, **kwargs)
    # Spłaszcz jednowymiarowe tablice kolumnowe i wierszowe
    if isinstance(x[0], list) and len(x[0]) == 1:
        x = [row[0] for row in x]
    elif len(x) == 1 and isinstance(x[0], list):
        x = x[0]
    # Obsługa tablic NumPy i list
    if convert in (np.array, np.ndarray):
        return np.array(x)
    elif convert == list:
        return x

# Nadpisz sposób konwertowania tablic, by używany był ten konwerter niestandardowy
excel.set_xl_array_conversion(custom_converter)
```

Dzięki temu domyślne działanie funkcji `xl` pozostanie bez zmian, ale zyskasz łatwy i wydajny sposób na uzyskanie tablicy NumPy lub (zagnieżdżonej) listy:

```
# Pobieranie obiektu DataFrame działa jak wcześniej
xl("...", headers=True)
# Zwrócenie jedno- lub dwuwymiarowej tablicy NumPy
xl("...", convert=np.ndarray)
# Zwrócenie listy lub listy zagnieżdżonej
xl("...", convert=list)
```

Możesz też użyć wyrażenia `df.to_numpy()` lub `df.to_numpy().tolist()`, by zmienić obiekt `DataFrame` odpowiednio w tablicę NumPy lub zagnieżdżoną listę (to zakłada, że obiekt `DataFrame` nazywa się `df`). Jednak to rozwiązanie jest mniej wydajne, gdyż dane są najpierw przekształcane z listy na obiekt `DataFrame`, a następnie z powrotem, co może spowodować nieoczekiwane rezultaty konwersji, na przykład puste komórki mogą być zmienne na `#LICZBA!` zamiast na `None`.

## Brak automatyzacji

Python w Excelu nie ma dostępu do modelu obiektów Excela. To znaczy, że nie możesz na przykład wstawiać nowych arkuszy czy pogrubiać czcionki. Co więcej, zamiast tworzyć pętlę w celu pobrania wszystkich danych z wszystkich arkuszy i zebrania ich w jeden obiekt `DataFrame`, musisz użyć funkcji `xl` w każdym arkuszu osobno.

## Maksymalny rozmiar zbioru danych

Python w Excelu może pracować ze zbiorami danych zajmującymi co najwyżej 100 MB. W przeciwnym razie zostanie zwrócony błąd `#OBL!`. Za pomocą `df.info()` możesz się dowiedzieć, ile pamięci zużywa Twój obiekt `DataFrame` (zakładając, że Twój obiekt nazywa się `df`). Po uruchomieniu tej metody edytor Pythona otworzy się i pod wynikiem wyświetli zużycie pamięci (by zobaczyć wynik, w edytorze Pythona naciśnij `DataFrame` pod komórką).

## Komórek nie można uruchamiać w izolacji

W przeciwieństwie do notatników Jupyter Python w Excelu zawsze ponownie oblicza wszystkie komórki Pythona, nawet jeśli uruchamiasz tylko jedną z nich. Ma to swoją zaletę, gdyż nigdy nie będziesz miał niesynchronizowanych komórek. Jednak musisz za każdym

razem czekać, aż wykona się każda komórka, co spowalnia pracę. Co więcej, takie ponowne obliczenia zużywają Twój dzienny i miesięczny limit użycia. Istnieje rozwiązanie tego problemu, ale to zawsze wymaga zakupu dodatku Język Python w programie Excel (patrz „Dodatek Język Python w programie Excel”). Dodatek ten umożliwia ustawienie trybu obliczeń na częściowe lub ręczne, co z kolei pozwala na uruchamianie pojedynczych komórek, tak jak w notatnikach Jupyter.

### Brak możliwości ponownego użycia kodu

Nie da się ponownie użyć kodu zapisanego w jednym ze skoroszytów w innym bez kopiowania i wklejania go do odpowiednich komórek Pythona.

### Maksymalna długość komórki

W komórce Excela można zapisać co najwyżej 32 767 znaków, więc jeśli Twój kod jest dłuższy, musisz go podzielić na kilka komórek.

## Dodatek Język Python w programie Excel

Choć podstawowa wersja Pythona w Excelu jest dołączona do subskrypcji Microsoft 365, Microsoft oferuje również płatny dodatek. Dodatek ten daje lepszą moc obliczeniową i poza automatycznym trybem obliczania oferuje możliwość użycia częściowego lub ręcznego trybu. Myślę, że z czasem ta oferta będzie się poszerzać, dlatego warto odwiedzać stronę Python w Excelu (<https://www.microsoft.com/pl-pl/microsoft-365/python-in-excel>), na której znajdziesz aktualne informacje.

## Podsumowanie

W tym rozdziale poznałeś podstawy pracy z Pythonem w Excelu. Nauczyłeś się używać funkcji `x1`, by przekształcać dane Excela w obiekt `DataFrame` biblioteki `pandas`. Widziałeś również, jak wykonać odwrotną operację, czyli zamienić obiekt `DataFrame` z powrotem na wartości Excela. Wystarczy zmienić rodzaj wyjścia z obiektu Pythona na wartości Excela.

Python w Excelu oferuje możliwość przeprowadzania analizy na całkiem nowym poziomie dzięki wykorzystaniu obiektu `DateFrame` biblioteki `pandas`. Subskrybenci Excela mogą zacząć używać Pythona w Excelu od razu, bez żadnej dodatkowej konfiguracji. Moim zdaniem mogłoby być mniej ograniczeń. Na przykład byłoby dobrze, gdyby Python w Excelu był dostępny ze stałą licencją pakietu Office. Ponadto informacja o wykorzystanym limicie użycia byłaby pomocna, by nie być nagle zaskoczonym błędem `#BLOCKED!`. Jednak z uwagi na fakt, że Python w Excelu jest stosunkowo nowym narzędziem, są szanse, że w przyszłości to będzie możliwe.

W następnym rozdziale będziemy mówić o najczęstszym przypadku użycia Pythona w Excelu: analizie szeregów czasowych z pakietem `pandas`. Analiza szeregów czasowych jest jednym z głównych powodów tak częstego używania pakietu `pandas` w branży finansowej. Zobaczmy zatem, dlaczego ta biblioteka ma tak dużą przewagę nad Excelem.



# PROGRAM PARTNERSKI

— GRUPY HELION —



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW w działający bankomat!

**Dowiedz się więcej i dołącz już dzisiaj!**

<http://program-partnerski.helion.pl>

GRUPA  
**Helion** 

# Excel na sterydach: Python robi to, czego VBA nie potrafi

Python od lat utrzymuje się w czołówce najpopularniejszych języków programowania, a Excel pozostaje niezastąpionym narzędziem w środowiskach biznesowych na całym świecie. Połączenie tych dwóch technologii to jeden z gorętszych trendów w analityce danych — szczególnie odkąd w 2003 roku Microsoft oficjalnie wprowadził Pythona do Excela. Rosnące zapotrzebowanie na automatyzację procesów, eliminację ręcznych błędów i zaawansowaną analizę danych sprawia, że umiejętność integracji Pythona z Excelem staje się kluczową kompetencją analityków, inżynierów i specjalistów do spraw danych.

Książka prowadzi czytelnika od podstaw Pythona i biblioteki pandas, przez oficjalną funkcję Python w Excelu, aż po zaawansowaną automatyzację z pakietem xlwings i xlwings Lite. Autor szczegółowo omawia pobieranie, oczyszczanie i analizowanie danych, tworzenie interaktywnych raportów, pisanie niestandardowych funkcji wywoływanych bezpośrednio z komórek arkusza i budowanie pełnoprawnych aplikacji biznesowych z bazą danych. Drugie wydanie zostało rozszerzone o nowe rozdziały poświęcone Pythonowi w Excelu, xlwings Lite, funkcjom AI opartym na OpenAI i Hugging Face, a także integracji z nowoczesnymi technologiami, takimi jak DuckDB i pliki Parquet.

**Python i Excel to najbardziej zrozumiały i najbardziej praktyczny przewodnik po Excelu z Pythonem. Nie tylko pokazuje, jak działają te narzędzia, ale również dlaczego tak bardzo poszerzają możliwości Excela.**

**George Mount**, MVP Microsoft Excel, autor książki *Nowoczesna analiza danych w Excelu*

**Felix tłumaczy materiał niesamowicie dobrze i prosto. Zwięzłe przykłady pomogą Ci bardzo szybko robić postępy. To przyjemna, świetnie napisana książka.**

**Cody Paustell**, programista, współtwórca Rubberduck-VBA

## Najważniejsze zagadnienia

- Analiza szeregów czasowych, tabele przestawne i wizualizacje z biblioteką pandas
- Automatyzacja Excela za pomocą xlwings — zastępowanie VBA kodem Pythona
- Tworzenie niestandardowych funkcji Excela w Pythonie (xlwings i xlwings Lite)
- Budowanie aplikacji biznesowych z Web API, bazami danych SQL i obsługą wyjątków
- Analiza dużych zbiorów danych z DuckDB i plikami Parquet bezpośrednio w Excelu
- Funkcje AI z OpenAI i analiza wydźwięku tekstu z modelami Hugging Face
- Odczytywanie i zapisywanie plików Excela bez instalacji aplikacji (XlsxWriter, OpenPyXL)

**Felix Zumstein** jest niezależnym programistą i założycielem Zommer Analytics GmbH. Twórca projektu open source xlwings — pionierskiego rozwiązania integrującego Pythona z Excelem, które wyprzedziło oficjalną funkcję Microsoftu o dekadę. Prowadzi kursy na LinkedIn Learning i organizuje spotkania społeczności xlwings w Londynie.

|                                                                                                                                                                                        |                                                                                     |                                                                                     |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
| <b>Helion</b>                                                                                                                                                                          | <b>KOD KORZYŚCI</b><br>Sięgnij po więcej!                                           |  |
|  <b>helion.pl</b>                                                                                   | <b>ISBN 978-83-289-4155-7</b>                                                       |                                                                                     |
|  <b>HELION S.A.</b><br>ul. Kościuski 1c<br>44-100 Gliwice<br>tel.: 32 230 98 63<br>helion@helion.pl |  |                                                                                     |
| <b>Cena: 99,00 zł</b>                                                                                                                                                                  |                                                                                     |                                                                                     |