

Lepszy kod w Pythonie

Przewodnik
dla aspirujących ekspertów

David Mertz

przekład Krzysztof Kapustka

APN Promise
Warszawa 2024

Spis treści

Przedmowa xi

Wstęp xiii

Podziękowania xix

O autorze xx

Wprowadzenie 1

1 Przechodzenie w pętli po niewłaściwych rzeczach 3

- 1.1 (Rzadko) generuj listę na potrzeby iteracji 3
- 1.2 Używaj `enumerate()` zamiast przechodzić w pętli po indeksie 6
- 1.3 Nie iteruj po `dict.keys()`, jeśli chcesz `dict.items()` 8
- 1.4 Zmianianie obiektu w czasie iteracji 9
- 1.5 Pętle `for` są bardziej idiomatyczne niż pętle `while` 12
- 1.6 Operator mors dla bloków „pętli i pół” 14
- 1.7 `zip()` upraszcza korzystanie z wielu obiektów iterowalnych 15
- 1.8 `zip(strict=True)` i `itertools.zip_longest()` 17
- 1.9 Podsumowanie 20

2 Mylenie równości z tożsamością 21

- 2.1 Późne wiązanie domknięć 22
- 2.2 Nadmierne sprawdzanie wartości logicznych 25
- 2.3 Porównywanie `x == None` 29
- 2.4 Nieporozumienia związane ze zmiennymi argumentami domyślnymi 30
 - 2.4.1 Podejście pierwsze: użyj klasy 31
 - 2.4.2 Podejście drugie: użyj wartownika `None` 32
 - 2.4.3 Podejście trzecie: skorzystaj z generatorów stanowych 33
- 2.5 Kopie kontra referencje do zmiennych obiektów 34
- 2.6 Mylenie operatora `is` z `==` (w obecności internowania) 36
- 2.7 Podsumowanie 38

3 Mieszanka problemów w Pythonie 39

- 3.1 Nazywanie rzeczy 39
 - 3.1.1 Nazywanie pliku tak samo jak moduł biblioteki standardowej 40
 - 3.1.2 Unikaj używania `import *` 42
 - 3.1.3 Puste lub zbyt ogólne instrukcje `except` 46
- 3.2 Kwadratowa złożoność naiwnej konkatenacji tekstów 52
- 3.3 Użyj menedżera kontekstu do otwarcia pliku 56
 - 3.3.1 Pierwsze niebezpieczeństwo 57
 - 3.3.2 Drugie niebezpieczeństwo 58
 - 3.3.3 Poprawianie kruchości 59
- 3.4 Opcjonalny argument `key` dla `.sort()` i `sorted()` 60
- 3.5 Użyj `dict.get()` dla niepewnych kluczy 63
- 3.6 Podsumowanie 65

4 Zaawansowane użytkowanie Pythona 67

- 4.1 Porównywanie `type(x) == type(y)` 67
- 4.2 Nazywanie rzeczy (nowe spojrzenie) 71
 - 4.2.1 Nadpisywanie nazw wbudowanych 71
 - 4.2.2 Uzyskiwanie bezpośredniego dostępu do atrybutu chronionego 75
- 4.3 Pamiętaj o rzadziej używanych funkcjach 80
 - 4.3.1 Debugowanie sformatowanego tekstu 81
 - 4.3.2 Elegancka magia dekoratorów 83
 - 4.3.3 Używanie `itertools` (w odpowiednim stopniu) 91
 - 4.3.4 Biblioteka zewnętrzna `more-itertools` 97
- 4.4 Adnotacje typów nie są typami w czasie wykonywania 100
 - 4.4.1 Adnotacje typów nie są ograniczeniami w czasie wykonywania 102
 - 4.4.2 Mylenie `typing.NewType()` z typem w czasie wykonywania 104
- 4.5 Podsumowanie 107

5 To, że możesz, nie znaczy, że powinienes... 109

- 5.1 Metaklasy 109
- 5.2 Małpie łątanie 114
- 5.3 Gettery i settery 118
- 5.4 Łatwiej prosić o przebaczenie niż o pozwolenie 121
- 5.5 Strukturalne dopasowywanie wzorców 123
- 5.6 Wyrażenia regularne i katastrofalne cofanie 125
- 5.7 Podsumowanie 129

6 Wybieranie odpowiedniej struktury danych 131

- 6.1 `collections.defaultdict` 131
- 6.2 `collections.Counter` 134
 - 6.2.1 Rozwiązanie 134
 - 6.2.2 Błąd 136
- 6.3 `collections.deque` 137
 - 6.3.1 Rozwiązanie 138
 - 6.3.2 Błąd 139
- 6.4 `collections.ChainMap` 140
 - 6.4.1 Rozwiązanie 141
 - 6.4.2 Błąd 142
- 6.5 Klasy danych i krotki nazwane 144
 - 6.5.1 Korzystanie z krotek nazwanych 145
 - 6.5.2 Statyczne kontra dynamiczne 146
 - 6.5.3 Klasy danych 147
- 6.6 Wydajne konkretne sekwencje 149
- 6.7 Podsumowanie 153

7 Niewłaściwe wykorzystywanie struktur danych 155

- 7.1 Kwadratowa złożoność wielokrotnego wyszukiwania na liście 155
- 7.2 Usuwanie lub dodawanie elementów do środka listy 159
 - 7.2.1 Bardziej wydajne struktury danych 164
- 7.3 Teksty są obiektami iterowalnymi tekstów 166
- 7.4 (Często) używaj `enum` zamiast `STAŁEJ` 169
- 7.5 Poznaj mniej popularne metody słownika 172
 - 7.5.1 Słowniki definiujące obiekty 172
 - 7.5.2 Powrót do naszego planowanego błędu 174
- 7.6 JSON nie obsługuje w Pythonie konwersji w obie strony 178
 - 7.6.1 Informacje podstawowe na temat formatu JSON 178
 - 7.6.2 Dane, które nie są konwertowane w obie strony 179
- 7.7 Tworzenie własnych struktur danych 182
 - 7.7.1 Kiedy tworzenie własnej struktury danych jest złym pomysłem 183
 - 7.7.2 Kiedy tworzenie własnej struktury danych jest dobrym pomysłem 185
 - 7.7.3 Najważniejsze wnioski 190
- 7.8 Podsumowanie 191

8 Bezpieczeństwo 193

- 8.1 Rodzaje losowości 194
 - 8.1.1 Używaj modułu `secrets` na potrzeby losowości kryptograficznej 195
 - 8.1.2 Odtwarzalne rozkłady losowe 196
- 8.2 Umieszczanie haseł lub innych sekretów w „bezpiecznym” kodzie źródłowym 200
- 8.3 Tworzenie własnych mechanizmów bezpieczeństwa 203
- 8.4 Używaj SSL/TLS na potrzeby mikrousług 206
- 8.5 Korzystanie z zewnętrznej biblioteki `requests` 211
- 8.6 Ataki SQL Injection, gdy nie korzysta się z DB-API 213
- 8.7 Nie używaj `assert` do sprawdzania założeń bezpieczeństwa 218
- 8.8 Podsumowanie 221

9 Obliczenia numeryczne w Pythonie 223

- 9.1 Liczby zmiennoprzecinkowe IEEE-754 224
 - 9.1.1 Porównywanie wartości NaN (i innych liczb zmiennoprzecinkowych) 224
 - 9.1.2 Wartości NaN i `statistics.median()` 228
 - 9.1.3 Naiwne użycie liczb zmiennoprzecinkowych: łączność i rozdzielność 230
 - 9.1.4 Naiwne użycie liczb zmiennoprzecinkowych: szczegółowość 232
- 9.2 Numeryczne typy danych 235
 - 9.2.1 Unikaj liczb zmiennoprzecinkowych w obliczeniach finansowych 235
 - 9.2.2 Nieoczywiste zachowania numerycznych typów danych 239
- 9.3 Podsumowanie 246

A Tematy na inne książki 247

- A.1 Test-Driven Development (TDD) 247
- A.2 Współbieżność 248
- A.3 Tworzenie pakietów 249
- A.4 Sprawdzanie typów 250
- A.5 Biblioteki numeryczne i biblioteki ramek danych 250

Indeks 253