
Zaawansowany Python

Luciano Ramalho

*przekład: Maria Chaniewska,
Jakub Niedźwiedź*

APN Promise
Warszawa 2015

O'REILLY®

Spis treści

Przedmowa.....	xv
----------------	----

Część I Prolog

1 Model danych Pythona	3
Pythoniczna talia kart.....	4
Sposoby używania metod specjalnych	8
Emulacja typów liczbowych.....	9
Reprezentacja tekstowa.....	11
Operatory arytmetyczne.....	12
Wartość Boolean typu niestandardowego	12
Przegląd metod specjalnych.....	13
Dlaczego len nie jest metodą	14
Podsumowanie rozdziału.....	15
Lektura uzupełniająca.....	16

Część II Struktury danych

2 Sekwencje i tablice	21
Przegląd wbudowanych sekwencji	22
Wyrażenia listowe i wyrażenia generatora.....	23
Wyrażenia listowe a czytelność	23
Wyrażenia listowe a funkcje map i filter.....	25
Iloczyny kartezjańskie	26
Wyrażenia generatora	27
Krotki nie są jedynie niezmiennymi listami	29
Krotki jako rekordy	29
Rozpakowywanie krotek.....	30
Rozpakowywanie zagnieżdżonych krotek	32
Krotki nazwane.....	33
Krotki jako niezmiennicze listy.....	35
Wycinanie.....	36
Dlaczego wycinki i zakresy wykluczają ostatni element.....	37
Obiekty wycinków	37
Wycinanie wielowymiarowe i wielokropki.....	39
Przypisywanie do wycinków.....	40

Używanie + i * z sekwencjami	40
Budowanie listy list	41
Przypisanie złożone w przypadku sekwencji	43
Zagadkowe przypisywanie +=	44
Metoda list.sort oraz wbudowana funkcja sorted	46
Zarządzanie sekwencjami uporządkowanymi przy użyciu bisect	48
Wyszukiwanie za pomocą funkcji bisect	48
Wstawianie za pomocą funkcji bisect.insort	51
Kiedy lista nie jest rozwiązaniem	52
Tablice	52
Widoki pamięci	56
NumPy i SciPy	57
Deque i inne kolejki	60
Podsumowanie rozdziału	64
Lektura uzupełniająca	65
3 Słowniki i zbiory	71
Ogólne typy odwzorowujące	72
Wyrażenia słownikowe	74
Przegląd powszechnych metod odwzorowań	75
Obsługa brakujących kluczy za pomocą.setdefault	77
Odwzorowania z elastycznym przeszukiwaniem kluczy	79
defaultdict: inne podejście do brakujących kluczy	79
Metoda __missing__	81
Odmiany dict	84
Tworzenie klas podrzędnych klasy UserDict	85
Niezmienne odwzorowania	87
Teoria zbiorów	88
Literały zbiorów	90
Wyrażenia zbioru	91
Operacje na zbiorach	92
Budowa wewnętrzna typów dict i set	96
Eksperyment wydajnościowy	96
Tablice mieszające w słownikach	98
Praktyczne konsekwencje działania słownika dict	101
Jak działają zbiory – konsekwencje praktyczne	104
Podsumowanie rozdziału	105
Lektura uzupełniająca	105
4 Tekst a bajty	109
Problemy ze znakami	110
Podstawy bajtów	111
Struktury i widoki pamięci	114

Podstawowe kodery/dekodery	115
Zrozumienie problemów kodowania/dekodowania	117
Radzenie sobie z UnicodeEncodeError	117
Radzenie sobie z UnicodeDecodeError	119
Błąd SyntaxError podczas ładowania modułów z nieoczekiwanym kodowaniem	120
Jak wykryć kodowanie sekwencji bajtów	122
BOM: przydatny gremlin	122
Obsługa plików tekstowych	124
Domyślne kodowanie: dom wariatów	127
Normalizacja Unicode w celu rozsądniejszego porównywania	130
Sprowadzanie do jednego rejestru	133
Funkcje narzędziowe do dopasowywania normalizowanego tekstu ...	134
„Normalizacja ekstremalna”: usuwanie znaków diakrytycznych	135
Sortowanie tekstu Unicode	138
Sortowanie przy użyciu algorytmu porządku alfabetycznego Unicode ..	140
Baza danych Unicode	141
Dwutrybowe interfejsy API dla typów str i bytes	143
str a bytes w wyrażeniach regularnych	143
str a bytes w funkcjach modułu os	144
Podsumowanie rozdziału	147
Lektura uzupełniająca	148

Część III Funkcje jako obiekty

5 Funkcje pierwszej klasy	155
Traktowanie funkcji jako obiektu	156
Funkcje wyższego rzędu	157
Nowoczesne odpowiedniki funkcji map, filter i reduce	158
Funkcje anonimowe	160
Siedem odmian obiektów wywołalnych	161
Definiowane przez użytkownika typy wywołalne	162
Introspekcja funkcji	163
Od parametrów pozycyjnych do parametrów tylko słów kluczowych	165
Pozyskiwanie informacji o parametrach	167
Adnotacje do funkcji	172
Pakiety do programowania funkcyjnego	174
Moduł operator	174
Zamrażanie argumentów przy użyciu funkcji functools.partial	178
Podsumowanie rozdziału	180
Lektura uzupełniająca	181

6	Wzorce projektowe z funkcjami pierwszej klasy	185
	Studium przypadku: refaktoryzacja wzorca Strategia	186
	Klasyczny wzorzec Strategia	186
	Strategia zorientowana funkcyjnie	190
	Wybieranie najlepszej strategii: proste podejście	193
	Znajdowanie Strategii w module	194
	Polecenie	196
	Podsumowanie rozdziału	197
	Lektura uzupełniająca	198
7	Dekoratory funkcji i domknięcia	201
	Dekoratory 101	202
	Kiedy Python wykonuje dekoratory	203
	Wzorzec Strategia wzbogacony dekoratorem	205
	Reguły zasięgów zmiennych	207
	Domknięcia	210
	Deklaracja nonlocal	213
	Implementacja prostego dekoratora	215
	Sposób działania	216
	Dekoratory w bibliotece standardowej	218
	Memoizacja dzięki functools.lru_cache	219
	Funkcje generyczne z pojedynczym rozsyłaniem	221
	Zagnieżdżanie dekoratorów	224
	Dekoratory parametryzowane	225
	Parametryzowany dekorator rejestrujący	226
	Parametryzowany dekorator Clock	228
	Podsumowanie rozdziału	230
	Lektura uzupełniająca	231

Część IV Idiomy zorientowane obiektowo

8	Odwołania do obiektów, zmienność i odzyskiwanie pamięci	237
	Zmienne nie są pudełkami	238
	Tożsamość, równość i aliasy	240
	Wybór między == a is	241
	Względna niezmienność krotek	242
	Kopie są domyślnie płytkie	243
	Głębokie i płytkie kopie arbitralnych obiektów	246
	Parametry funkcji jako odwołania	247
	Typy zmienne jako domyślne parametry: zły pomysł	249
	Programowanie obronne ze zmiennymi parametrami	251

del i odzyskiwanie pamięci	253
Słabe odwołania	255
Skecz WeakValueDictionary	256
Ograniczenia słabych odwołań	258
Trikowe gry Pythona z niezmiennymi obiektami	259
Podsumowanie rozdziału	261
Lektura uzupełniająca	262
9 Obiekt pythonowy	267
Reprezentacje obiektów	268
Przypomnienie klasy Vector	268
Alternatywny konstruktor	271
classmethod a staticmethod	272
Formatowane wyświetlanie	274
Haszowalny obiekt Vector2d	277
Prywatne i „chronione” atrybuty w Pythonie	283
Oszczędzanie miejsca dzięki atrybutowi klasy __slots__	285
Problemy z atrybutem __slots__	288
Przesłanie atrybutów klasy	288
Podsumowanie rozdziału	291
Lektura uzupełniająca	292
10 Kodowanie, haszowanie i wycinanie sekwencji	297
Vector: definiowany przez użytkownika typ sekwencyjny	298
Vector podejście nr 1: zgodność z Vector2d	298
Protokoły i kacze typowanie	301
Vector podejście nr 2: sekwencja z możliwością wycinania	302
Działanie wycinania	303
Metoda __getitem__ świadoma wycinania	305
Vector podejście nr 3: dynamiczny dostęp do atrybutów	307
Vector podejście nr 4: haszowanie i szybsze ==	311
Vector podejście nr 5: formatowanie	316
Podsumowanie rozdziału	324
Lektura uzupełniająca	325
11 Interfejsy: od protokołów do abstrakcyjnych klas bazowych	331
Interfejsy i protokoły w kulturze języka Python	332
Python lubi sekwencje	334
Małpie łatanie w celu zaimplementowania protokołu w trakcie	
działania programu	336
Wodne ptactwo Alexa Martelli	338
Tworzenie podklasy z abstrakcyjnej klasy bazowej	344
Abstrakcyjne klasy bazowe w bibliotece standardowej	346

Abstrakcyjne klasy bazowe w collections.abc	346
Wieża liczbowa klas ABC	348
Definiowanie i wykorzystywanie abstrakcyjnej klasy bazowej	349
Szczegóły składni abstrakcyjnych klas bazowych	354
Tworzenie podklas dla abstrakcyjnej klasy bazowej Tombola	355
Wirtualna podklasa klasy Tombola	357
Jak testowano podklasy klasy Tombola	360
Użycie metody register w praktyce	363
Gęsi mogą zachowywać się jak kaczki	364
Podsumowanie rozdziału	365
Lektura uzupełniająca	368
12 Dziedziczenie: na dobre czy na złe	375
Tworzenie klas podrzędnych z typów wbudowanych jest zawiłe	376
Wielokrotne dziedziczenie i kolejność ustalania metod	379
Wielokrotne dziedziczenie w świecie rzeczywistym	384
Radzenie sobie z wielokrotnym dziedziczeniem	387
1. Rozróżniać dziedziczenie interfejsów od dziedziczenia implementacji	387
2. Tworzyć jawne interfejsy przy pomocy klas ABC	387
3. Korzystać z domieszek w celu ponownego wykorzystania kodu	387
4. Jawnie deklarować domieszki dzięki nazewnictwu	388
5. Klasa ABC może być też domieszką, ale nie na odwrót	388
6. Nie tworzyć podklas dziedziczącej z więcej niż jednej klasy konkretnej	388
7. Dostarczać użytkownikom klasy łączone	389
8. „Preferować komponowanie obiektów przed dziedziczeniem klas”	389
Tkinter: dobry, zły i brzydki	390
Nowoczesny przykład: domieszki w ogólnych widokach Django	391
Podsumowanie rozdziału	394
Lektura uzupełniająca	395
13 Przeciążanie operatorów: rób to poprawnie	399
Podstawy przeciążania operatorów	400
Operatory unarne	400
Przeciążanie operatora + w celu zaimplementowania dodawania wektorów	403
Przeciążanie operatora * dla mnożenia wektora przez wartość skalarną	409
Bogate operatory porównania	413
Operatory rozszerzonego przypisania	418
Podsumowanie rozdziału	423
Lektura uzupełniająca	424

Część V Przepływ sterowania

14	Iterowalność, iteratory i generatory	431
	Klasa Sentence – podejście nr 1: sekwencja słów	432
	Dlaczego sekwencje są iterowalne: funkcja iter	434
	Obiekty iterowalne a iteratory	436
	Klasa Sentence – podejście nr 2: klasyczne wnętrze	440
	Klasa Sentence jako iterator: zły pomysł	441
	Klasa Sentence – podejście nr 3: funkcja generatora	442
	Jak działa funkcja generatora	443
	Klasa Sentence – podejście nr 4: leniwa implementacja	447
	Klasa Sentence – podejście nr 5: wyrażenie generatora	448
	Wyrażenia generatora: kiedy ich używać	450
	Inny przykład: generator ciągu arytmetycznego	451
	Ciąg arytmetyczny wykorzystujący itertools	453
	Funkcje generatora w bibliotece standardowej	455
	Nowa składnia w wersji Python 3.3: yield from	467
	Funkcje redukujące obiekty iterowalne	468
	Bliższe przyjrzenie się funkcji iter	470
	Studium przypadku: generatory w narzędziu do konwersji baz danych	471
	Generatory jako współprogramy	473
	Podsumowanie rozdziału	474
	Lektura uzupełniająca	474
15	Zarządzanie kontekstem i bloki else	481
	Zrób to, potem tamto: bloki else poza instrukcją if	482
	Zarządzanie kontekstem i bloki with	484
	Narzędzia contextlib	489
	Korzystanie z @contextmanager	489
	Podsumowanie rozdziału	493
	Lektura uzupełniająca	494
16	Współprogramy	497
	Jak współprogramy wyewoluowały z generatorów	498
	Podstawowe zachowanie generatora zastosowane jako współprogram	499
	Przykład: współprogram obliczający średnią kroczącą	503
	Dekoratory przygotowujące współprogram	504
	Kończenie współprogramów i obsługa wyjątków	506
	Zwracanie wartości ze współprogramu	510
	Korzystanie z yield from	512
	Znaczenie konstrukcji yield from	519
	Przypadek użycia: współprogramy dla dyskretnego symulowania zdarzeń	525

	Symulacje zdarzeń dyskretnych	525
	Symulacja floty taksówek	526
	Podsumowanie rozdziału	535
	Lektura uzupełniająca	536
17	Współbieżność z futures	543
	Przykład: pobieranie stron WWW na trzy sposoby	544
	Skrypt pobierania sekwencyjnego	546
	Pobieranie przy pomocy concurrent.futures	548
	Gdzie są obiekty future?	549
	Blokowanie wejścia/wyjścia a GIL	553
	Uruchamianie procesów przy pomocy concurrent.futures	554
	Eksperymentowanie z Executor.map	556
	Pobierania flag z wyświetlaniem postępów i obsługą błędów	559
	Obsługa błędów w przykładach flags2	564
	Korzystanie z futures.as_completed	566
	Alternatywy dla przetwarzania wielowątkowego	569
	Podsumowanie rozdziału	570
	Lektura uzupełniająca	571
18	Współbieżność z asyncio	577
	Wątek kontra współprogram: porównanie	579
	Klasa asyncio.Future: nieblokująca z założenia	585
	Instrukcja yield from a obiekty future, zadania i współprogramy	586
	Pobieranie obrazów przy pomocy asyncio i aiohttp	588
	Bieganie w kółko wokół wywołań blokujących	593
	Ulepszanie skryptu pobierającego obrazu wykorzystującego asyncio	595
	Wykorzystanie asyncio.as_completed	596
	Korzystanie z obiektu wykonawczego w celu uniknięcia zablokowania pętli zdarzeń	601
	Od procedur zwrotnych do obiektów future i współprogramów	603
	Wykonywanie wielu żądań dla każdego pobierania	605
	Pisanie serwerów wykorzystujących asyncio	608
	Serwer TCP wykorzystujący asyncio	609
	Serwer WWW wykorzystujący aiohttp	614
	Inteligentniejsi klienci a lepsza współbieżność	617
	Podsumowanie rozdziału	618
	Lektura uzupełniająca	619

Część VI Metaprogramowanie

19	Atrybuty i właściwości dynamiczne	627
	Przekształcanie danych przy pomocy atrybutów dynamicznych	628
	Badanie danych przypominających JSON przy pomocy atrybutów dynamicznych	631
	Problem z nieprawidłowymi nazwami atrybutów	634
	Elastyczne tworzenie obiektów przy pomocy <code>__new__</code>	635
	Restrukturyzacja źródła danych OSCON przy pomocy <code>shelve</code>	637
	Pobieranie połączonych rekordów przy pomocy właściwości	641
	Użycie właściwości do sprawdzania poprawności atrybutów	647
	LineItem – podejście nr 1: klasa dla elementu zamówienia	647
	LineItem – podejście nr 2: właściwość sprawdzająca swoją poprawność	648
	Właściwe spojrzenie na właściwości	650
	Właściwości przesłaniają atrybuty instancji	651
	Dokumentacja właściwości	654
	Kodowanie fabryki właściwości	655
	Obsługiwanie usuwania atrybutów	658
	Podstawowe atrybuty i funkcje obsługujące atrybuty	659
	Atrybuty specjalne, które wpływają na obsługę atrybutów	659
	Funkcje wbudowane do obsługi atrybutów	660
	Metody specjalne do obsługi atrybutów	661
	Podsumowanie rozdziału	662
	Lektura uzupełniająca	663
20	Deskryptory atrybutów	669
	Przykład deskryptora: sprawdzanie poprawności atrybutu	669
	LineItem podejście nr 3: prosty deskryptor	670
	LineItem podejście nr 4: automatyczne nazwy atrybutów przechowywania	675
	LineItem podejście nr 5: nowy typ deskryptora	681
	Deskryptory przesłaniające a nieprzesłaniające	684
	Deskryptor przesłaniający	686
	Deskryptor przesłaniający bez <code>__get__</code>	687
	Deskryptor nieprzesłaniający	688
	Nadpisywanie deskryptora w klasie	689
	Metody są deskryptorami	690
	Wskazówki dotyczące użycia deskryptorów	693
	Dokumentacja docstring deskryptora i przesłanianie usuwania	694
	Podsumowanie rozdziału	695
	Lektura uzupełniająca	696

21	Metaprogramowanie klas	699
	Fabryka klas	700
	Dekorator klasy służący do dostosowywania dekryptorów	703
	Co dzieje się kiedy: czas importu a czas działania	706
	Ćwiczenia dotyczące czasu przetwarzania	707
	Metaklasy 101.	710
	Ćwiczenie dotyczące czasu przetwarzania metaklasy	713
	Metaklasa do dostosowywania deskryptorów	717
	Metoda specjalna <code>__prepare__</code> metaklasy	719
	Klasy jako obiekty	721
	Podsumowanie rozdziału	722
	Lektura uzupełniająca	723
	<i>Posłowie</i>	727
	<i>Dodatek: Skrypty pomocnicze</i>	731
	<i>Żargon społeczności Pythona</i>	759
	<i>Indeks</i>	769
	<i>O autorze</i>	788