

WYDANIE DRUGIE

Zaawansowany Python

*Przejrzyste, zwarte
i efektywne programowanie*

Luciano Ramalho

*przekład: Maria Chaniewska,
Jakub Niedźwiedź
Marek Włodarz*

Spis treści

Przedmowa	xxiii
-----------------	-------

Część I. Struktury danych

1. Model danych Pythona	3
Co nowego w tym rozdziale	4
Pythoniczna talia kart	5
Sposoby używania metod specjalnych	8
Emulacja typów liczbowych	9
Reprezentacja tekstowa	12
Wartość logiczna typu niestandardowego	13
API kolekcji	14
Przegląd metod specjalnych	15
Dlaczego len nie jest metodą	17
Podsumowanie rozdziału	18
Lektura uzupełniająca	18
2. Tablica sekwencji	21
Co nowego w tym rozdziale	22
Przegląd wbudowanych sekwencji	22
Wyrażenia listowe i wyrażenia generatora	25
Wyrażenia listowe a czytelność	25
Wyrażenia listowe a funkcje map i filter	27
Iloczyny kartezjańskie	28
Wyrażenia generatora	29
Krotki nie są jedynie niezmiennymi listami	31

Krotki jako rekordy	31
Krotki jako niezmiennie listy	33
Porównanie metod krotek i list	35
Rozpakowywanie sekwencji i typów iterowalnych	36
Używanie * do przechwytywania nadmiarowych elementów	37
Rozpakowywanie z * w wywołaniach funkcji i literałach sekwencji	38
Zagnieżdżone rozpakowywanie	38
Dopasowywanie wzorców w sekwencjach	40
Dopasowywanie wzorców sekwencji w interpreterze	44
Wycinanie	49
Dlaczego wycinki i zakresy wykluczają ostatni wskazany element	49
Obiekty wycinków	49
Wycinanie wielowymiarowe i wielokropki	51
Przypisywanie do wycinków	52
Używanie + i * z sekwencjami	53
Budowanie listy list	53
Przypisanie złożone w przypadku sekwencji	55
Zagadkowe przypisywanie +=	56
Metoda list.sort oraz wbudowana funkcja sorted	58
Kiedy lista nie jest rozwiązaniem	61
Tablice	61
Widoki pamięci	65
NumPy	67
Deque i inne kolejki	69
Podsumowanie rozdziału	73
Lektura uzupełniająca	74
3. Słowniki i zbiory	79
Co nowego w tym rozdziale	80
Nowoczesna składnia słowników	81
Wyrażenia słownikowe	81
Rozpakowywanie odwzorowań	82
Scalanie odwzorowań za pomocą 	82
Dopasowywanie wzorców w odwzorowaniach	83
Standardowe API typów odwzorowujących	86
Co jest haszowalne?	87
Przegląd powszechnych metod odwzorowań	88
Wstawianie lub aktualizowanie zmiennych wartości	90
Automatyczna obsługa brakujących kluczy	92

defaultdict: inne podejście do brakujących kluczy	92
Metoda <code>__missing__</code>	94
Niekonsekwentne stosowanie <code>__missing__</code> w bibliotece standardowej	97
Odmiany słowników	98
<code>collections.OrderedDict</code>	98
<code>collections.ChainMap</code>	98
<code>collections.Counter</code>	99
<code>shelve.Shelf</code>	100
Tworzenie podklas z klasy <code>UserDict</code> zamiast <code>dict</code>	100
Niezmiennie odwzorowania	102
Widoki słowników	104
Praktyczne konsekwencje tego, jak działają słowniki	105
Teoria zbiorów	106
Literały zbiorów	108
Wyrażenia zbioru	109
Jak działają zbiory – konsekwencje praktyczne	110
Operacje na zbiorach	110
Operacje zbiorów na widokach słowników	114
Podsumowanie rozdziału	116
Lektura uzupełniająca	117
4. Tekst Unicode a bajty	121
Co nowego w tym rozdziale	122
Problemy ze znakami	122
Podstawy bajtów	124
Podstawowe kodery/dekodery	127
Zrozumienie problemów kodowania/dekodowania	129
Radzenie sobie z <code>UnicodeEncodeError</code>	129
Radzenie sobie z <code>UnicodeDecodeError</code>	130
Błąd <code>SyntaxError</code> podczas ładowania modułów z nieoczekiwanym kodowaniem	132
Jak wykryć kodowanie sekwencji bajtów	132
BOM: przydatny gremlin	134
Obsługa plików tekstowych	135
Domyślne kodowanie: dom wariatów	138
Normalizacja Unicode w celu rozsądniejszego porównywania	144
Sprawdzanie do jednego rejestru	147
Funkcje narzędziowe do dopasowywania normalizowanego tekstu	148
„Normalizacja ekstremalna”: usuwanie znaków diakrytycznych	149
Sortowanie tekstu Unicode	153

Sortowanie przy użyciu algorytmu porządku alfabetycznego Unicode	154
Baza danych Unicode	155
Wyszukiwanie znaków według nazw	156
Numeryczny sens znaku	158
Dwutrybowe interfejsy API dla typów str i bytes	159
str kontra bytes w wyrażeniach regularnych	159
str kontra bytes w funkcjach modułu os	161
Podsumowanie rozdziału	162
Lektura uzupełniająca	163
5. Budowanie klas danych	169
Co nowego w tym rozdziale	170
Przegląd elementów budujących klasy danych	170
Główne cechy	174
Klasyczne krotki nazwane	176
Typowane krotki nazwane	179
Wskazówki dla typów 101	180
Brak wpływu na wykonywanie programu	180
Składnia adnotacji zmiennych	181
Znaczenie adnotacji zmiennych	182
Więcej na temat @dataclass	186
Opcje pól	188
Przetwarzanie po zainicjowaniu	190
Typowane atrybuty klas	193
Zmienne inicjalizacyjne, które nie są polami	193
Przykład @dataclass: rekord zasobu Dublin Core	194
Klasa danych jako brzydki zapach w kodzie	197
Klasa danych jako rusztowanie	198
Klasa danych jako reprezentacja pośrednia	199
Dopasowywanie wzorców dla wystąpień klas	199
Proste wzorce klas	199
Wzorce klas według słów kluczowych	200
Pozycyjne wzorce klas	202
Podsumowanie rozdziału	203
Lektura uzupełniająca	204
6. Odwołania do obiektów, zmienność i odzyskiwanie pamięci	209
Co nowego w tym rozdziale	210
Zmienne nie są pudełkami	210

Tożsamość, równość i aliasy	212
Wybór między == a is	214
Względna niezmiennosc krotek	215
Kopie są domyślnie płytkie	216
Głębokie i płytkie kopie arbitralnych obiektów	219
Parametry funkcji jako odwołania	221
Typy zmienne jako domyślne parametry: zły pomysł	222
Programowanie defensywne ze zmiennymi parametrami	224
del i odzyskiwanie pamięci	226
Trikowe gry Pythona z niezmiennymi obiektami	229
Podsumowanie rozdziału	231
Lektura uzupełniająca	232

Część II. Funkcje jako obiekty

7. Funkcje jako obiekty pierwszej klasy	239
Co nowego w tym rozdziale	240
Traktowanie funkcji jako obiektu	240
Funkcje wyższego rzędu	242
Nowoczesne zamienniki funkcji map, filter i reduce	243
Funkcje anonimowe	245
Dziewięć odmian obiektów wywoływalnych	246
Definiowane przez użytkownika typy wywoływalne	247
Od parametrów pozycyjnych do parametrów tylko słów kluczowych	249
Parametry tylko pozycyjne	251
Pakiety do programowania funkcyjnego	251
Moduł operator	252
Zamrażanie argumentów przy użyciu funkcji functools.partial	255
Podsumowanie rozdziału	258
Lektura uzupełniająca	258
8. Wskazówki dla typów w funkcjach	263
Co nowego w tym rozdziale	264
Typowanie stopniowe	264
Typowanie stopniowe w praktyce	266
Podstawy Mypy	266
Bardziej rygorystyczne ustawienia Mypy	267
Domyślna wartość parametru	268
Użycie None jako wartości domyślnej	270

Typy są definiowane przez obsługiwane operacje	271
Typy używane w adnotacjach	277
Typ Any	277
Typy proste i klasy	280
Typy Optional i Union	281
Kolekcje generyczne	282
Typy krotkowe	285
Mapowania generyczne	288
Abstrakcyjne klasy bazowe	290
Iterable	292
Sparametryzowane typy generyczne i TypeVar	294
Protokoły statyczne	298
Typ Callable	304
NoReturn	307
Adnotacje dla parametrów pozycyjnych i przyjmujących zmienną liczbę elementów	307
Niedoskonałe typowanie i silne testowanie	308
Podsumowanie rozdziału	310
Lektura uzupełniająca	311
9. Dekoratory funkcji i domknięcia	317
Co nowego w tym rozdziale	318
Dekoratory 101	318
Kiedy Python wykonuje dekoratory	320
Dekoratory rejestrujące	322
Reguły zasięgów zmiennych	322
Domknięcia	326
Deklaracja nonlocal	329
Logika wyszukiwania zmiennych	330
Implementacja prostego dekoratora	331
Jak to działa	332
Dekoratory w bibliotece standardowej	334
Memoizacja dzięki functools.cache	334
Stosowanie lru_cache	337
Funkcje generyczne z pojedynczym rozsyłaniem	338
Dekoratory parametryzowane	343
Parametryzowany dekorator rejestrujący	343
Parametryzowany dekorator Clock	346
Oparty na klasie dekorator clock	348
Podsumowanie rozdziału	349

Lektura uzupełniająca	350
10. Wzorce projektowe z funkcjami pierwszej klasy	355
Co nowego w tym rozdziale	356
Studium przypadku: refaktoryzacja wzorca Strategia	356
Klasyyczny wzorec Strategia	356
Strategia funkcyjna	360
Wybieranie najlepszej strategii: proste podejście	364
Znajdowanie strategii w module	365
Wzorec Strategia wzbogacony dekoratorem	367
Wzorec Polecenie	369
Podsumowanie rozdziału	370
Lektura uzupełniająca	371

Część III. Klasy i protokoły

11. Obiekt pythoniczny	377
Co nowego w tym rozdziale	378
Reprezentacje obiektów	378
Przypomnienie klasy Vector	379
Alternatywny konstruktor	382
classmethod kontra staticmethod	383
Formatowane wyświetlanie	384
Haszowalny obiekt Vector2d	388
Obsługiwanie pozycyjnego dopasowywania wzorców	391
Kompletny listing klasy Vector2d, wersja 3	392
'Prywatne' i 'chronione' atrybuty w Pythonie	396
Oszczędzanie miejsca dzięki atrybutowi klasy __slots__	398
Prosty pomiar oszczędności zapewnianych przez __slots__	401
Podsumowanie problemów z atrybutem __slots__	402
Przesłanianie atrybutów klasy	403
Podsumowanie rozdziału	405
Lektura uzupełniająca	406
12. Metody specjalne dla sekwencji	411
Co nowego w tym rozdziale	412
Vector: zdefiniowany przez użytkownika typ sekwencyjny	412
Vector podejście nr 1: zgodność z Vector2d	413
Protokoły i kaczce typowanie	416

Vector podejście nr 2: sekwencja z możliwością wycinania	417
Jak działa wycinanie	418
Metoda <code>__getitem__</code> świadoma wycinania	420
Vector podejście nr 3: dynamiczny dostęp do atrybutów	421
Vector podejście nr 4: haszowanie i szybsze <code>==</code>	425
Vector podejście nr 5: formatowanie	432
Podsumowanie rozdziału	440
Lektura uzupełniająca	441
13. Interfejsy, protokoły i abstrakcyjne klasy bazowe	447
Mapa typowania	448
Co nowego w tym rozdziale	449
Dwa rodzaje protokołów	450
Programowanie kaczek	452
Python lubi sekwencje	452
Małpie łatanie: implementowanie protokołu w trakcie działania programu	454
Programowanie defensywne i 'szybkie porażki'	456
Gęsie typowanie	459
Tworzenie podklasy z abstrakcyjnej klasy bazowej	464
Abstrakcyjne klasy bazowe w bibliotece standardowej	466
Definiowanie i wykorzystywanie abstrakcyjnej klasy bazowej	468
Szczegóły składni abstrakcyjnych klas bazowych	473
Tworzenie podklasy z abstrakcyjnej klasy bazowej	474
Wirtualna podklasa abstrakcyjnej klasy bazowej	477
Użycie metody <code>register</code> w praktyce	479
Typowanie strukturalne z abstrakcyjnymi klasami bazowymi	480
Protokoły statyczne	482
Typowana funkcja <code>double</code>	482
Protokoły statyczne sprawdzalne w czasie wykonywania programu	484
Ograniczenia sprawdzeń protokołów w czasie wykonywania programu	488
Obsługa protokołu statycznego	489
Projektowanie protokołu statycznego	491
Najlepsze praktyki dotyczące projektowania protokołów	493
Rozszerzanie protokołu	494
Abstrakcyjne klasy bazowe z <code>numbers</code> i protokoły liczbowe	495
Podsumowanie rozdziału	498
Lektura uzupełniająca	499

14. Dziedziczenie: na dobre czy na złe	505
Co nowego w tym rozdziale.	506
Funkcja <code>super()</code>	506
Tworzenie klas podrzędnych z typów wbudowanych jest zawiłe	509
Wielokrotne dziedziczenie i kolejność ustalania metod	512
Klasy domieszkowe	518
Odwzorowania bez rozróżniania wielkości liter	518
Wielokrotne dziedziczenie w świecie rzeczywistym	520
Abstrakcyjne klasy bazowe (ABC) również są domieszkami	521
<code>ThreadingMixIn</code> i <code>ForkingMixIn</code>	521
Domieszki w generycznych widokach Django	522
Dziedziczenie wielokrotne w pakiecie <code>Tkinter</code>	526
Radzenie sobie z wielokrotnym dziedziczeniem	528
Preferuj komponowanie obiektów przed dziedziczeniem klas	528
Zrozumieć, dlaczego w ogóle używamy dziedziczenia w danym przypadku	529
Twórz interfejsy jawne przy pomocy klas ABC	529
Korzystaj z domieszek w celu ponownego wykorzystania kodu	529
Dostarczaj użytkownikom klasy agregujące	529
Twórz podklasy tylko takich klas, które są zaprojektowane do dziedziczenia	530
Unikaj tworzenia klas potomnych z klas konkretnych	531
<code>Tkinter</code> : dobry, zły i brzydki	531
Podsumowanie rozdziału	532
Lektura uzupełniająca	534
15. Więcej na temat wskazówek dla typów	539
Co nowego w tym rozdziale	540
Przeciążone sygnatury	540
Przeciążanie funkcji <code>max</code>	542
Wnioski z przeciążeń funkcji <code>max</code>	546
<code>TypedDict</code>	547
Rzutowanie typów	555
Odczytywanie wskazówek dla typów w czasie wykonywania programu	558
Problemy z adnotacjami w czasie wykonywania programu	558
Sposoby radzenia sobie z tym problemem	561
Implementowanie klasy generycznej	563
Podstawowy żargon dotyczący typów generycznych	565
Wariancja	566
Bezwariantowy dozownik	566
Kowariantny dozownik	568

Kontrawariantny pojemnik na śmieci	569
Przegląd wariacji	570
Implementowanie generycznego protokołu statycznego	573
Podsumowanie rozdziału	575
Lektura uzupełniająca	576
16. Przeciążanie operatorów	583
Co nowego w tym rozdziale	584
Podstawy przeciążania operatorów	585
Operatory unarne	585
Przeciążanie operatora + dla dodawania wektorów	588
Przeciążanie operatora * dla mnożenia wektora przez wartość skalarną	594
Używanie @ jako operatora infiksowego	597
Podsumowanie operatorów arytmetycznych	599
Bogate operatory porównania	600
Operatory rozszerzonego przypisania	603
Podsumowanie rozdziału	608
Lektura uzupełniająca	610

Część IV. Przepływ sterowania

17. Iteratory, generatory i klasyczne współprogramy	615
Co nowego w tym rozdziale	616
Sekwencja słów	616
Dłaczego sekwencje są iterowalne: funkcja iter	618
Używanie iter wobec obiektów wywołalnych	620
Obiekty iterowalne kontra iteratory	621
Klasy Sentence z metodą __iter__	625
Klasa Sentence – podejście nr 2: klasyczny iterator	626
Nie przekształcaj obiektu iterowalnego w swój własny iterator	627
Klasa Sentence – podejście nr 3: funkcja generatora	628
Jak działa funkcja generatora	629
Leniwa klasa Sentence	633
Klasa Sentence – podejście nr 4: leniwy generator	633
Klasa Sentence – podejście nr 5: wyrażenie generatora	634
Wyrażenia generatora: kiedy ich używać	636
Generator ciągu arytmetycznego	638
Ciąg arytmetyczny wykorzystujący itertools	641
Funkcje generatora w bibliotece standardowej	642

Funkcje redukujące obiekty iterowalne	653
Podgeneratory wykorzystujące yield from	656
Ponowne wynalezienie generatora chain	657
Przechodzenie przez drzewo	658
Generyczne typy iterowalne	663
Klasyczne współprogramy	665
Przykład: Współprogram obliczający średnią krocząca	667
Zwracanie wartości ze współprogramu	670
Generyczne wskazówki typów dla klasycznych współprogramów	674
Podsumowanie rozdziału	676
Lektura uzupełniająca	677
18. Bloki with, match i else	683
Co nowego w tym rozdziale	684
Zarządzanie kontekstem i bloki with	684
Narzędzia contextlib	689
Korzystanie z @contextmanager	690
Dopasowywanie wzorców w lis.py: studium przypadku	694
Składnia Scheme	695
Importy oraz typy	696
Parser	697
Klasa Environment	699
REPL	701
Ewaluator	702
Procedure: Klasa implementująca domknięcie	711
Używanie wzorców alternatywy	712
Zrób to, a potem tamto: bloki else poza instrukcją if	713
Podsumowanie rozdziału	716
Lektura uzupełniająca	716
19. Modele współbieżności w Pythonie	723
Co nowego w tym rozdziale	724
Całościowy obraz	724
Szczypta żargonu	725
Procesy, wątki i niesławna globalna blokada GIL w Pythonie	727
Współbieżny program Hello World	730
Bączek z wątkami	730
Bączek z procesami	733
Bączek ze współprogramami	735

Porównanie funkcji supervisor	739
Prawdziwa siła oddziaływania blokady GIL	741
Szybki quiz	742
Własna pula procesów	744
Rozwiązanie oparte na procesach	746
Zrozumienie czasów przetwarzania	747
Kod dla wielordzeniowego programu sprawdzającego pierwszośc	748
Eksperymentowanie z większą lub mniejszą liczbą procesów	752
Słabe rozwiązanie oparte na wątkach	753
Python w świecie wielordzeniowym	754
Administracja systemami	755
Nauka o danych	756
Programowanie aplikacji webowych/mobilnych po stronie serwera	757
Serwery aplikacji WSGI	759
Rozproszone kolejki zadań	761
Podsumowanie rozdziału	763
Lektura uzupełniająca	763
Współbieżność przy użyciu wątków i procesów	763
Blokada GIL	765
Współbieżność poza biblioteką standardową	766
Współbieżność i skalowalność poza językiem Python	768
20. Współbieżni wykonawcy	773
Co nowego w tym rozdziale	773
Współbieżne pobieranie z sieci web	774
Skrypt pobierania sekwencyjnego	776
Pobieranie przy pomocy concurrent.futures	779
Gdzie są obiekty future?	781
Uruchamianie procesów przy użyciu concurrent.futures	784
Redukcja wielordzeniowego programu sprawdzającego pierwszośc	784
Eksperymentowanie z Executor.map	788
Pobieranie flag z wyświetlaniem postępów i obsługą błędów	791
Obsługa błędów w przykładach flags2	796
Korzystanie z futures.as_completed	799
Podsumowanie rozdziału	802
Lektura uzupełniająca	802
21. Programowanie asynchroniczne	805
Co nowego w tym rozdziale	806

Kilka definicji807
Przykład z asyncio: sondowanie domen808
Sztuczka Guido przy czytaniu kodu asynchronicznego.....	.810
Nowe pojęcie: obiekty oczekiwalne811
Pobieranie obrazów przy pomocy asyncio i HTTPX812
Tajemnica natywnych współprogramów: skromne generatory815
Problem „wszystko albo nic”816
Asynchroniczne menedżery kontekstu816
Ulepszanie skryptu pobierającego obrazy wykorzystującego asyncio818
Użycie asyncio.as_completed i wątku819
Tłumienie żądań przy pomocy semafora821
Wykonywanie wielu żądań dla każdego pobierania825
Delegowanie zadań do elementów wykonawczych828
Pisanie serwerów wykorzystujących asyncio830
Usługa web wykorzystująca FastAPI.....	.832
Serwer TCP wykorzystujący asyncio835
Asynchroniczna iteracja i asynchroniczne elementy iterowalne842
Asynchroniczne funkcje generatorów842
Wyrażenia asynchroniczne i wyrażenia generatorów asynchronicznych849
async poza asyncio: Curio.....	.852
Wskazówki dotyczące typów dla obiektów asynchronicznych855
Jak działa programowanie asynchroniczne, a jak nie.....	.856
Bieganie w kółko wokół wywołań blokujących.....	.856
Mit systemów ograniczonych wejściem/wyjściem857
Unikanie pułapek związanych z procesorem858
Podsumowanie rozdziału.....	.858
Lektura uzupełniająca859

Część V. Metaprogramowanie

22. Atrybuty i właściwości dynamiczne	867
Co nowego w tym rozdziale.....	.868
Przekształcanie danych przy pomocy atrybutów dynamicznych868
Badanie danych przypominających JSON za pomocą atrybutów dynamicznych870
Problem z nieprawidłowymi nazwami atrybutów874
Elastyczne tworzenie obiektów przy pomocy <code>__new__</code>875
Właściwości obliczane878
Krok 1: Sterowane danymi tworzenie atrybutów879

Krok 2: Właściwość pobierająca połączony rekord	881
Krok 3: Właściwość przesłaniająca istniejący atrybut	885
Krok 4: Pamięć podręczna właściwości na zamówienie	886
Krok 5: Buforowanie właściwości za pomocą functors	887
Użycie właściwości do sprawdzania poprawności atrybutów	890
Lineltem – podejście nr 1: klasa dla elementu zamówienia	890
Lineltem – podejście nr 2: właściwość sprawdzająca poprawność	891
Właściwe spojrzenie na właściwości	892
Właściwości przesłaniają atrybuty instancji	894
Dokumentacja właściwości	896
Kodowanie fabryki właściwości	897
Obsługiwanie usuwania atrybutów	900
Podstawowe atrybuty i funkcje obsługujące atrybuty	902
Atrybuty specjalne, które wpływają na obsługę atrybutów	902
Funkcje wbudowane do obsługi atrybutów	903
Metody specjalne do obsługi atrybutów	904
Podsumowanie rozdziału	906
Lektura uzupełniająca	906
23. Deskryptory atrybutów	911
Co nowego w tym rozdziale	912
Przykład deskryptora: sprawdzanie poprawności atrybutu	912
Lineltem podejście nr 3: prosty deskryptor	912
Lineltem podejście nr 4: automatyczne nazwy atrybutów przechowywania	919
Lineltem podejście nr 5: nowy typ deskryptora	921
Deskryptory przesłaniające a nieprzesłaniające	924
Deskryptory przesłaniające	926
Deskryptor przesłaniający bez __get__	927
Deskryptor nieprzesłaniający	928
Nadpisywanie deskryptora w klasie	929
Metody są deskryptorami	930
Wskazówki dotyczące użycia deskryptorów	932
Dokumentacja docstring deskryptora i przesłanianie usuwania	934
Podsumowanie rozdziału	935
Lektura uzupełniająca	936
24. Metaprogramowanie klas	939
Co nowego w tym rozdziale	940
Klasy jako obiekty	940

type: wbudowana fabryka klas941
Funkcja fabryki klas943
Wprowadzenie do <code>__init_subclass__</code>946
Dlaczego metoda <code>__init_subclass__</code> nie może skonfigurować <code>__slots__</code>954
Ulepszanie klas za pomocą dekoratorów klasy954
Kiedy co się wydarza: czas importu kontra czas działania957
Ewaluacja eksperymentów czasowych958
Metaklasy 101963
Jak metaklasa dostosowuje klasę965
Ładny przykład metaklasy967
Ewaluacja eksperymentów czasowych dla metaklasy970
Rozwiązanie Checked oparte na metaklasach975
Metaklasy w świecie rzeczywistym980
Nowoczesne funkcjonalności upraszczają albo zastępują metaklasy980
Metaklasy są stabilnymi funkcjonalnościami języka981
Klasa może mieć tylko jedną metaklasę981
Metaklasy powinny być szczegółami implementacyjnymi982
Metaklasowa sztuczka z <code>__prepare__</code>983
Podsumowanie985
Podsumowanie rozdziału986
Lektura uzupełniająca987
Posłowie991
Indeks995
O autorze	1018