

Spis treści

Przedmowa	33
Część I. Wprowadzenie	51
1. Pytania i odpowiedzi dotyczące Pythona	53
Dlaczego ludzie używają Pythona?	53
Jakość oprogramowania	54
Wydajność programistów	55
Czy Python jest językiem skryptowym?	55
Jakie są wady języka Python?	57
Kto dzisiaj używa Pythona?	59
Co mogę zrobić za pomocą Pythona?	61
Programowanie systemowe	61
Graficzne interfejsy użytkownika (GUI)	62
Skrypty internetowe	62
Integracja komponentów	63
Programowanie bazodanowe	63
Szybkie prototypowanie	64
Programowanie numeryczne i naukowe	64
I dalej: gry, przetwarzanie obrazu, wyszukiwanie danych, robotyka, Excel...	65
Jak Python jest rozwijany i wspierany?	66
Kompromisy związane z modelem open source	66
Jakie są techniczne mocne strony Pythona?	67
Jest zorientowany obiektowo i funkcyjny	67
Jest darmowy	68
Jest przenośny	68
Ma duże możliwości	69
Można go łączyć z innymi językami	70
Jest względnie łatwy w użyciu	71

Jest względnie łatwy do nauczenia się	71
Zawdzięcza swoją nazwę Monty Pythonowi	72
Jak Python wygląda na tle innych języków?	72
Podsumowanie rozdziału	74
Sprawdź swoją wiedzę — quiz	74
Sprawdź swoją wiedzę — odpowiedzi	75
2. Jak Python wykonuje programy?	79
Wprowadzenie do interpretera Pythona	79
Wykonywanie programu	81
Z punktu widzenia programisty	81
Z punktu widzenia Pythona	82
Warianty modeli wykonywania	85
Alternatywne implementacje Pythona	85
Narzędzia do optymalizacji działania programu	89
Zamrożone pliki binarne	91
Przyszłe możliwości?	92
Podsumowanie rozdziału	93
Sprawdź swoją wiedzę — quiz	93
Sprawdź swoją wiedzę — odpowiedzi	93
3. Jak wykonuje się programy?	95
Interaktywny wiersz poleceń	95
Uruchamianie sesji interaktywnej	96
Ścieżka systemowa	98
Nowe opcje systemu Windows w wersji 3.3: PATH, Launcher	98
Gdzie zapisywać programy — katalogi z kodem źródłowym	99
Czego nie wpisywać — znaki zachęty i komentarze	100
Interaktywne wykonywanie kodu	101
Do czego służy sesja interaktywna	103
Uwagi praktyczne — wykorzystywanie sesji interaktywnej	104
Systemowy wiersz poleceń i pliki źródłowe	106
Pierwszy skrypt	107
Wykonywanie plików z poziomu wiersza poleceń powłoki	108
Sposoby użycia wiersza poleceń	109
Uwagi praktyczne — wykorzystywanie wierszy poleceń i plików	110
Skrypty wykonywalne w stylu uniksowym — #!	111
Podstawy skryptów uniksowych	112
Sztuczka z wyszukiwaniem programu	
przy użyciu polecenia env w systemie Unix	112
Python 3.3 launcher — #! w systemie Windows	113

Klikanie ikon plików	114
Podstawowe zagadnienia związane z klikaniem ikon plików	114
Kliknięcie ikony w systemie Windows	115
Sztuczka z funkcją input	117
Inne ograniczenia programów uruchamianych kliknięciem ikony	119
Importowanie i przeładowywanie modułów	119
Podstawy importowania i przeładowywania modułów	119
Więcej o modułach — atrybuty	121
Uwagi praktyczne — instrukcje import i reload	124
Wykorzystywanie funkcji exec do wykonywania plików modułów	125
Interfejs użytkownika środowiska IDLE	126
Szczegóły uruchamiania środowiska IDLE	127
Podstawy środowiska IDLE	128
Wybrane funkcje środowiska IDLE	130
Zaawansowane narzędzia środowiska IDLE	130
Uwagi praktyczne — korzystanie ze środowiska IDLE	131
Inne środowiska IDE	133
Inne opcje wykonywania kodu	135
Osadzanie wywołań	135
Zamrożone binarne pliki wykonywalne	136
Uruchamianie kodu z poziomu edytora tekstu	136
Jeszcze inne możliwości uruchamiania	136
Przyszłe możliwości?	137
Jaką opcję wybrać?	137
Podsumowanie rozdziału	139
Sprawdź swoją wiedzę — quiz	139
Sprawdź swoją wiedzę — odpowiedzi	140
Sprawdź swoją wiedzę — ćwiczenia do części pierwszej	141

Część II. Typy i operacje 145

4. Wprowadzenie do typów obiektów Pythona	147
Hierarchia pojęć w Pythonie	147
Dlaczego korzystamy z typów wbudowanych	148
Najważniejsze typy danych w Pythonie	149
Liczby	151
Łańcuchy znaków	153
Operacje na sekwencjach	153
Niezmienność	155
Metody specyficzne dla typu	156
Uzyskiwanie pomocy	157

Inne sposoby kodowania łańcuchów znaków	159
Ciągi znaków w formacie Unicode	160
Dopasowywanie wzorców	162
Listy	162
Operacje na typach sekwencyjnych	163
Operacje specyficzne dla typu	163
Sprawdzanie granic	164
Zagnieżdżanie	164
Listy składane	165
Słowniki	167
Operacje na odwzorowaniach	167
Zagnieżdżanie raz jeszcze	169
Brakujące klucze — testowanie za pomocą if	170
Sortowanie kluczy — pętla for	172
Iteracja i optymalizacja	173
Krotki	175
Do czego służą krotki	176
Pliki	176
Pliki binarne	177
Pliki tekstowe Unicode	178
Inne narzędzia podobne do plików	180
Inne typy podstawowe	180
Jak zepsuć elastyczność kodu	182
Klasy definiowane przez użytkownika	183
I wszystko inne	184
Podsumowanie rozdziału	184
Sprawdź swoją wiedzę — quiz	185
Sprawdź swoją wiedzę — odpowiedzi	185

5. Typy liczbowe 187

Podstawy typów liczbowych Pythona	187
Literały liczbowe	188
Wbudowane narzędzia liczbowe	190
Operatory wyrażeń Pythona	190
Liczby w akcji	195
Zmienne i podstawowe wyrażenia	195
Formaty wyświetlania liczb	197
Porównania — zwykłe i łączone	199
Dzielenie — klasyczne, bez reszty i prawdziwe	200
Precyzja liczb całkowitych	204
Liczby zespolone	205

Notacja szesnastkowa, ósemkowa i dwójkowa — literały i konwersje	205
Operacje na poziomie bitów	208
Inne wbudowane narzędzia numeryczne	209
Inne typy liczbowe	211
Typ Decimal (liczby dziesiętne)	211
Typ Fraction (liczby ułamkowe)	214
Zbiory	218
Wartości Boolean	225
Rozszerzenia numeryczne	226
Podsumowanie rozdziału	226
Sprawdź swoją wiedzę — quiz	227
Sprawdź swoją wiedzę — odpowiedzi	227
6. Wprowadzenie do typów dynamicznych	229
Sprawa brakujących deklaracji typu	229
Zmienne, obiekty i referencje	229
Typy powiązane są z obiektami, a nie ze zmiennymi	231
Obiekty są uwalniane	232
Referencje współdzielone	234
Referencje współdzielone a modyfikacje w miejscu	236
Referencje współdzielone a równość	237
Typy dynamiczne są wszędzie	239
Podsumowanie rozdziału	240
Sprawdź swoją wiedzę — quiz	240
Sprawdź swoją wiedzę — odpowiedzi	240
7. Łańcuchy znaków	243
Co znajdziesz w tym rozdziale	243
Unicode — krótka historia	244
Łańcuchy znaków — podstawy	244
Literały łańcuchów znaków	246
Łańcuchy znaków w apostrofach i cudzysłowach są tym samym	247
Sekwencje ucieczki reprezentują znaki specjalne	248
Surowe łańcuchy znaków blokują sekwencje ucieczki	251
Potrójne cudzysłowy i apostrofy kodują łańcuchy znaków będące wielowierszowymi blokami	252
Łańcuchy znaków w akcji	254
Podstawowe operacje	254
Indeksowanie i wycinki	255
Narzędzia do konwersji łańcuchów znaków	259
Modyfikowanie łańcuchów znaków	262

Metody łańcuchów znaków	263
Składnia wywoływania metod	264
Metody typów znakowych	264
Przykłady metod łańcuchów znaków — modyfikowanie	265
Przykłady metod łańcuchów znaków — analiza składniowa tekstu	267
Inne często używane metody łańcuchów znaków	268
Oryginalny moduł string (usunięty w wersji 3.0)	269
Wyrażenia formatujące łańcuchy znaków	270
Formatowanie łańcuchów tekstu	
z użyciem wyrażeń formatujących — podstawy	271
Składnia zaawansowanych wyrażeń formatujących	272
Przykłady zaawansowanych wyrażeń formatujących	274
Wyrażenia formatujące oparte na słowniku	275
Formatowanie łańcuchów z użyciem metody format	276
Podstawy	276
Używanie kluczy, atrybutów i przesunięć	277
Zaawansowana składnia wywołań metody format	278
Przykłady zaawansowanego formatowania łańcuchów znaków	
z użyciem metody format	279
Porównanie metody format z wyrażeniami formatującymi	281
Dlaczego miałbyś korzystać z metody format	284
Generalne kategorie typów	289
Typy z jednej kategorii współdzielą zbiory operacji	289
Typy mutowalne można modyfikować w miejscu	290
Podsumowanie rozdziału	291
Sprawdź swoją wiedzę — quiz	291
Sprawdź swoją wiedzę — odpowiedzi	291

8. Listy oraz słowniki293

Listy	293
Listy w akcji	295
Podstawowe operacje na listach	295
Iteracje po listach i składanie list	296
Indeksowanie, wycinki i macierze	297
Modyfikacja list w miejscu	298
Słowniki	303
Słowniki w akcji	306
Podstawowe operacje na słownikach	306
Modyfikacja słowników w miejscu	307
Inne metody słowników	308
Przykład — baza danych o filmach	309

Uwagi na temat korzystania ze słowników	311
Inne sposoby tworzenia słowników	315
Zmiany dotyczące słowników w Pythonie 3.x i 2.7	317
Podsumowanie rozdziału	324
Sprawdź swoją wiedzę — quiz	325
Sprawdź swoją wiedzę — odpowiedzi	325
9. Krotki, pliki i wszystko inne	327
Krotki	328
Krotki w akcji	329
Dlaczego istnieją listy i krotki	331
Repetytorium: rekordy — krotki nazwane	332
Pliki	334
Otwieranie plików	335
Wykorzystywanie plików	336
Pliki w akcji	337
Pliki tekstowe i binarne — krótka historia	339
Przechowywanie obiektów Pythona w plikach i przetwarzanie ich	340
Przechowywanie natywnych obiektów Pythona — moduł pickle	342
Przechowywanie obiektów Pythona w formacie JSON	343
Przechowywanie spakowanych danych binarnych — moduł struct	345
Menedżery kontekstu plików	346
Inne narzędzia powiązane z plikami	346
Przegląd i podsumowanie podstawowych typów obiektów	347
Elastyczność obiektów	349
Referencje a kopie	350
Porównania, testy równości i prawda	352
Prawda czy fałsz, czyli znaczenie True i False w Pythonie	355
Hierarchie typów Pythona	357
Obiekty typów	359
Inne typy w Pythonie	360
Pułapki typów wbudowanych	360
Przypisanie tworzy referencje, nie kopie	360
Powtórzenie dodaje jeden poziom zagłębienia	361
Uwaga na cykliczne struktury danych	362
Typów niemutowalnych nie można modyfikować w miejscu	362
Podsumowanie rozdziału	363
Sprawdź swoją wiedzę — quiz	363
Sprawdź swoją wiedzę — odpowiedzi	364
Sprawdź swoją wiedzę — ćwiczenia do części drugiej	364

Część III. Instrukcje i składnia 369

10. Wprowadzenie do instrukcji Pythona 371

Raz jeszcze o hierarchii pojęciowej języka Python	371
Instrukcje Pythona	372
Historia dwóch if	374
Co dodaje Python	374
Co usuwa Python	374
Skąd bierze się składnia z użyciem wcięć	376
Kilka przypadków specjalnych	379
Szybki przykład — interaktywne pętle	381
Prosta pętla interaktywna	381
Wykonywanie obliczeń na danych wpisywanych przez użytkownika	382
Obsługa błędów poprzez sprawdzanie danych wejściowych	383
Obsługa błędów za pomocą instrukcji try	384
Kod zagnieżdżony na trzy poziomy głębokości	386
Podsumowanie rozdziału	387
Sprawdź swoją wiedzę — quiz	387
Sprawdź swoją wiedzę — odpowiedzi	387

11. Przypisania, wyrażenia i wyświetlanie 389

Instrukcje przypisania	389
Formy instrukcji przypisania	390
Przypisanie sekwencji	391
Rozszerzona składnia rozpakowania sekwencji w Pythonie 3.x	394
Przypisanie z wieloma celami	398
Przypisania rozszerzone	399
Reguły dotyczące nazw zmiennych	401
Instrukcje wyrażeń	404
Instrukcje wyrażeń i modyfikacje w miejscu	406
Polecenia print	406
Funkcja print z Pythona 3.x	407
Instrukcja print w Pythonie 2.x	410
Przekierowanie strumienia wyjściowego	411
Wyświetlanie niezależne od wersji	415
Podsumowanie rozdziału	418
Sprawdź swoją wiedzę — quiz	418
Sprawdź swoją wiedzę — odpowiedzi	418

12. Testy if i reguły składni	421
Instrukcje if	421
Ogólny format	421
Proste przykłady	422
Rozgałęzienia kodu	422
Reguły składni Pythona raz jeszcze	424
Ograniczniki bloków — reguły tworzenia wcięć	425
Ograniczniki instrukcji — wiersze i znaki kontynuacji	428
Kilka przypadków specjalnych	428
Testy prawdziwości i testy logiczne	430
Wyrażenie trójargumentowe if/else	432
Podsumowanie rozdziału	435
Sprawdź swoją wiedzę — quiz	435
Sprawdź swoją wiedzę — odpowiedzi	435
 13. Pętle while i for	 437
Pętle while	437
Ogólny format	438
Przykłady	438
Instrukcje break, continue, pass oraz else w pętli	439
Ogólny format pętli	439
Instrukcja pass	440
Instrukcja continue	441
Instrukcja break	441
Klauzula else pętli	442
Pętle for	444
Ogólny format	445
Przykłady	445
Techniki tworzenia pętli	451
Pętle z licznikami — range	452
Skanowanie sekwencji — pętla while z funkcją range kontra pętla for	453
Przetasowania sekwencji — funkcje range i len	454
Przechodzenie niewyczerpujące — range kontra wycinki	454
Modyfikowanie list — range kontra listy składane	455
Przechodzenie równoległe — zip oraz map	456
Generowanie wartości przesunięcia i elementów — enumerate	459
Podsumowanie rozdziału	463
Sprawdź swoją wiedzę — quiz	463
Sprawdź swoją wiedzę — odpowiedzi	463

14. Iteracje i listy składane	465
Iteracje — pierwsze spojrzenie	465
Protokół iteracyjny — iteratory plików	466
Iterowanie ręczne — iter i next	469
Inne wbudowane typy iterowalne	472
Listy składane — wprowadzenie	474
Podstawy list składanych	474
Wykorzystywanie list składanych w plikach	475
Rozszerzona składnia list składanych	476
Inne konteksty iteracyjne	478
Nowe obiekty iterowalne w Pythonie 3.x	483
Wpływ na kod w wersji 2.x — zalety i wady	483
Obiekt iterowalny range	484
Obiekty iterowalne map, zip i filter	485
Iteratory wielokrotne kontra pojedyncze	486
Obiekty iterowalne — widoki słownika	487
Inne zagadnienia związane z iteracjami	489
Podsumowanie rozdziału	489
Sprawdź swoją wiedzę — quiz	490
Sprawdź swoją wiedzę — odpowiedzi	490
 15. Wprowadzenie do dokumentacji	 491
Źródła dokumentacji Pythona	491
Komentarze ze znakami #	492
Funkcja dir	492
Notki dokumentacyjne — __doc__	494
PyDoc — funkcja help	497
PyDoc — raporty HTML	500
Nie tylko notki docstrings — pakiet Sphinx	508
Zbiór standardowej dokumentacji	509
Zasoby internetowe	509
Publikowane książki	510
Często spotykane problemy programistyczne	511
Podsumowanie rozdziału	513
Sprawdź swoją wiedzę — quiz	513
Sprawdź swoją wiedzę — odpowiedzi	513
Sprawdź swoją wiedzę — ćwiczenia do części trzeciej	514

Część IV . Funkcje i generatory 517

16. Podstawy funkcji	519
Dlaczego używamy funkcji	519
Tworzenie funkcji	521
Instrukcje def	522
Instrukcja def uruchamiana jest w czasie wykonania	523
Pierwszy przykład — definicje i wywoływanie	524
Definicja	524
Wywołanie	524
Polimorfizm w Pythonie	525
Drugi przykład — przecinające się sekwencje	526
Definicja	526
Wywołania	527
Raz jeszcze o polimorfizmie	528
Zmienne lokalne	528
Podsumowanie rozdziału	529
Sprawdź swoją wiedzę — quiz	529
Sprawdź swoją wiedzę — odpowiedzi	529
17. Zasięgi	531
Podstawy zasięgów w Pythonie	531
Reguły dotyczące zasięgów	532
Rozwiązywanie nazw — reguła LEGB	534
Przykład zasięgu	536
Zasięg wbudowany	537
Instrukcja global	540
Projektowanie programów:	
minimalizowanie stosowania zmiennych globalnych	541
Projektowanie programów:	
minimalizowanie modyfikacji dokonywanych pomiędzy plikami	543
Inne metody dostępu do zmiennych globalnych	544
Zasięgi a funkcje zagnieżdżone	545
Szczegóły dotyczące zasięgów zagnieżdżonych	546
Przykłady zasięgów zagnieżdżonych	546
Funkcje fabrykujące: domknięcia	547
Zachowywanie stanu zasięgu zawierającego	
za pomocą argumentów domyślnych	550
Instrukcja nonlocal w Pythonie 3.x	553
Podstawy instrukcji nonlocal	554
Instrukcja nonlocal w akcji	555

Czemu służą zmienne nonlocal? Opcje zachowania stanu	558
Zachowanie stanu: zmienne nonlocal (tylko w wersji 3.x)	558
Zachowanie stanu: zmienne globalne — tylko jedna kopia	559
Zachowanie stanu: klasy — jawne atrybuty (wprowadzenie)	559
Zachowanie stanu: atrybuty funkcji (w wersjach 3.x i 2.x)	561
Podsumowanie rozdziału	564
Sprawdź swoją wiedzę — quiz	565
Sprawdź swoją wiedzę — odpowiedzi	566
18. Argumenty	567
Podstawy przekazywania argumentów	567
Argumenty a współdzielone referencje	568
Unikanie modyfikacji argumentów mutowalnych	570
Symulowanie parametrów wyjścia i wielu wyników działania	571
Specjalne tryby dopasowywania argumentów	572
Podstawy dopasowywania argumentów	572
Składnia dopasowania argumentów	573
Dopasowywanie argumentów — szczegóły	575
Przykłady ze słowami kluczowymi i wartościami domyślnymi	576
Przykłady dowolnych argumentów	578
Argumenty tylko ze słowami kluczowymi (z Pythona 3.x)	582
Przykład z funkcją obliczającą minimum	585
Pełne rozwiązanie	586
Dodatkowy bonus	587
Puenta	588
Uogólnione funkcje działające na zbiorach	588
Emulacja funkcji print z Pythona 3.0	590
Wykorzystywanie argumentów ze słowami kluczowymi	592
Podsumowanie rozdziału	594
Sprawdź swoją wiedzę — quiz	594
Sprawdź swoją wiedzę — odpowiedzi	595
19. Zaawansowane zagadnienia dotyczące funkcji	597
Koncepcje projektowania funkcji	597
Funkcje rekurencyjne	599
Sumowanie z użyciem rekurencji	600
Implementacje alternatywne	600
Pętle a rekurencja	601
Obsługa dowolnych struktur	602
Obiekty funkcji — atrybuty i adnotacje	606
Pośrednie wywołania funkcji — obiekty „pierwszej klasy”	606
Introspekcja funkcji	607

Atrybuty funkcji	608
Adnotacje funkcji w Pythonie 3.x	609
Funkcje anonimowe — lambda	611
Podstawy wyrażeń lambda	611
Po co używamy wyrażeń lambda	612
Jak (nie) zaciemniać kodu napisanego w Pythonie	614
Zasięgi: wyrażenia lambda również można zagnieżdżać	615
Narzędzia programowania funkcyjnego	617
Odwzorowywanie funkcji na obiekty iterowalne — map	617
Wybieranie elementów obiektów iterowalnych — funkcja filter	619
Łączenie elementów obiektów iterowalnych — funkcja reduce	619
Podsumowanie rozdziału	620
Sprawdź swoją wiedzę — quiz	621
Sprawdź swoją wiedzę — odpowiedzi	621
20. Listy składane i generatory	623
Listy składane i narzędzia funkcyjne	623
Listy składane kontra funkcja map	624
Dodajemy warunki i pętle zagnieżdżone — filter	625
Przykład — listy składane i macierze	627
Nie nadużywaj list składanych: reguła KISS	630
Funkcje i wyrażenia generatorów	631
Funkcje generatorów — yield kontra return	633
Wyrażenia generatorów — obiekty iterowalne spotykają złożenia	638
Funkcje generatorów a wyrażenia generatorów	643
Generatory są obiektami o jednoprzebiegowej iteracji	644
Generowanie wyników we wbudowanych typach, narzędziach i klasach	646
Przykład — generowanie mieszanych sekwencji	649
Nie nadużywaj generatorów: reguła EIBTI	654
Przykład — emulowanie funkcji zip i map za pomocą narzędzi iteracyjnych	656
Podsumowanie obiektów składanych	662
Zakresy i zmienne składane	662
Zrozumieć zbiory i słowniki składane	663
Rozszerzona składnia zbiorów i słowników składanych	664
Podsumowanie rozdziału	665
Sprawdź swoją wiedzę — quiz	665
Sprawdź swoją wiedzę — odpowiedzi	666
21. Wprowadzenie do pomiarów wydajności	667
Pomiary wydajności iteracji	667
Moduł pomiaru czasu domowej roboty	668
Skrypt mierzący wydajność	672

Wyniki pomiarów czasu	673
Inne rozwiązania dla modułu do pomiaru czasu	676
Inne sugestie	679
Mierzenie czasu iteracji z wykorzystaniem modułu timeit	680
Podstawowe reguły korzystania z modułu timeit	680
Moduł i skrypt testujący z użyciem modułu timeit	685
Wyniki działania skryptu testującego	686
Jeszcze trochę zabawy z mierzeniem wydajności	689
Inne zagadnienia związane z mierzeniem szybkości działania kodu — test pystone	693
Pułapki związane z funkcjami	694
Lokalne nazwy są wykrywane w sposób statyczny	694
Wartości domyślne i obiekty mutowalne	695
Funkcje, które nie zwracają wyników	697
Różne problemy związane z funkcjami	698
Podsumowanie rozdziału	698
Sprawdź swoją wiedzę — quiz	699
Sprawdź swoją wiedzę — odpowiedzi	699
Sprawdź swoją wiedzę — ćwiczenia do części czwartej	700

Część V. Moduły i pakiety 703

22. Moduły — wprowadzenie	705
Dlaczego używamy modułów	705
Architektura programu w Pythonie	706
Struktura programu	707
Importowanie i atrybuty	707
Moduły biblioteki standardowej	709
Jak działa importowanie	710
1. Odszukanie modułu	710
2. Kompilowanie (o ile jest to potrzebne)	711
3. Wykonanie	712
Pliki kodu bajtowego — __pycache__ w Pythonie 3.2+	712
Modele plików kodu bajtowego w akcji	713
Ścieżka wyszukiwania modułów	714
Konfigurowanie ścieżki wyszukiwania	717
Wariacje ścieżki wyszukiwania modułów	717
Lista sys.path	717
Wybór pliku modułu	718
Podsumowanie rozdziału	721
Sprawdź swoją wiedzę — quiz	721
Sprawdź swoją wiedzę — odpowiedzi	722

23. Podstawy tworzenia modułów	723
Tworzenie modułów	723
Nazwy modułów	723
Inne rodzaje modułów	724
Używanie modułów	724
Instrukcja import	725
Instrukcja from	725
Instrukcja from *	725
Operacja importowania jest przeprowadzana tylko raz	726
Instrukcje import oraz from są przypisaniami	727
Równoważność instrukcji import oraz from	728
Potencjalne pułapki związane z użyciem instrukcji from	729
Przestrzenie nazw modułów	730
Pliki generują przestrzenie nazw	731
Słowniki przestrzeni nazw: __dict__	732
Kwalifikowanie nazw atrybutów	733
Importowanie a zasięgi	734
Zagnieżdżanie przestrzeni nazw	734
Przeładowywanie modułów	735
Podstawy przeładowywania modułów	736
Przykład przeładowywania z użyciem reload	737
Podsumowanie rozdziału	739
Sprawdź swoją wiedzę — quiz	739
Sprawdź swoją wiedzę — odpowiedzi	740
24. Pakiety modułów	741
Podstawy importowania pakietów	741
Pakiety a ustawienia ścieżki wyszukiwania	742
Pliki pakietów __init__.py	743
Przykład importowania pakietu	745
Instrukcja from a instrukcja import w importowaniu pakietów	746
Do czego służy importowanie pakietów	747
Historia trzech systemów	748
Względne importowanie pakietów	751
Zmiany w Pythonie 3.0	751
Podstawy importowania względnego	752
Do czego służą importy względne	753
Zasięg importów względnych	755
Podsumowanie reguł wyszukiwania modułów	756
Importy względne w działaniu	757
Pułapki związane z importem względnym w pakietach: zastosowania mieszane	762

Pakiety przestrzeni nazw w Pythonie 3.3	767
Semantyka pakietów przestrzeni nazw	768
Wpływ na zwykłe pakiety: opcjonalne pliki <code>__init__.py</code>	769
Pakiety przestrzeni nazw w akcji	770
Zagnieżdżanie pakietów przestrzeni nazw	771
Pliki nadal mają pierwszeństwo przed katalogami	772
Podsumowanie rozdziału	774
Sprawdź swoją wiedzę — quiz	775
Sprawdź swoją wiedzę — odpowiedzi	775
25. Zaawansowane zagadnienia związane z modułami	777
Koncepty związane z projektowaniem modułów	777
Ukrywanie danych w modułach	779
Minimalizacja niebezpieczeństw użycia <code>from *</code> — <code>__X</code> oraz <code>__all__</code>	779
Włączanie opcji z przyszłych wersji Pythona: <code>__future__</code>	780
Mieszane tryby użycia — <code>__name__</code> oraz <code>__main__</code>	781
Testy jednostkowe z wykorzystaniem atrybutu <code>__name__</code>	782
Przykład — kod działający w dwóch trybach	783
Symbole walut: Unicode w akcji	786
Notki dokumentacyjne: dokumentacja modułu w działaniu	787
Modyfikacja ścieżki wyszukiwania modułów	789
Rozszerzenie <code>as</code> dla instrukcji <code>import</code> oraz <code>from</code>	790
Przykład — moduły są obiektami	791
Importowanie modułów z użyciem nazwy w postaci ciągu znaków	793
Uruchamianie ciągów znaków zawierających kod	793
Bezpośrednie wywołania: dwie opcje	794
Przykład — przechodnie przeładowywanie modułów	795
Przeładowywanie rekurencyjne	795
Rozwiązania alternatywne	798
Pułapki związane z modułami	802
Kolizje nazw modułów: pakiety i importowanie względne w pakietach	802
W kodzie najwyższego poziomu kolejność instrukcji ma znaczenie	803
Instrukcja <code>from</code> kopiuje nazwy, jednak łączy już nie	803
Instrukcja <code>from *</code> może zaciemnić znaczenie zmiennych	804
Funkcja <code>reload</code> może nie mieć wpływu na obiekty importowane za pomocą <code>from</code>	805
Funkcja <code>reload</code> i instrukcja <code>from</code> a testowanie interaktywne	805
Rekurencyjne importowanie za pomocą <code>from</code> może nie działać	806
Podsumowanie rozdziału	808
Sprawdź swoją wiedzę — quiz	808
Sprawdź swoją wiedzę — odpowiedzi	808
Sprawdź swoją wiedzę — ćwiczenia do części piątej	809

Część VI. Klasy i programowanie zorientowane obiektowo 813

26. Programowanie zorientowane obiektowo — wprowadzenie 815

Po co używa się klas	816
Programowanie zorientowane obiektowo z dystansu	817
Wyszukiwanie atrybutów dziedziczonych	817
Klasy a instancje	820
Wywołania metod klasy	820
Tworzenie drzew klas	821
Przeciążanie operatorów	823
Programowanie zorientowane obiektowo oparte jest na ponownym wykorzystaniu kodu	824
Podsumowanie rozdziału	827
Sprawdź swoją wiedzę — quiz	827
Sprawdź swoją wiedzę — odpowiedzi	828

27. Podstawy tworzenia klas 829

Klasy generują wiele obiektów instancji	829
Obiekty klas udostępniają zachowania domyślne	830
Obiekty instancji są rzeczywistymi elementami	830
Pierwszy przykład	831
Klasy dostosowujemy do własnych potrzeb przez dziedziczenie	833
Drugi przykład	834
Klasy są atrybutami w modułach	836
Klasy mogą przechwytywać operatory Pythona	837
Trzeci przykład	838
Po co przeciążamy operatory	840
Najprostsza klasa Pythona na świecie	841
Jeszcze kilka słów o rekordach: klasy kontra słowniki	844
Podsumowanie rozdziału	846
Sprawdź swoją wiedzę — quiz	846
Sprawdź swoją wiedzę — odpowiedzi	847

28. Bardziej realistyczny przykład 849

Krok 1. — tworzenie instancji	850
Tworzenie konstruktorów	850
Testowanie w miarę pracy	851
Wykorzystywanie kodu na dwa sposoby	853
Krok 2. — dodawanie metod	854
Tworzenie kodu metod	856
Krok 3. — przeciążanie operatorów	858
Udostępnienie sposobów wyświetlania	858

Krok 4. — dostosowywanie zachowania za pomocą klas podrzędnych	860
Tworzenie klas podrzędnych	861
Rozszerzanie metod — niepoprawny sposób	861
Rozszerzanie metod — poprawny sposób	862
Polimorfizm w akcji	863
Dziedziczenie, dostosowanie do własnych potrzeb i rozszerzenie	865
Programowanie zorientowane obiektowo — idea	866
Krok 5. — dostosowanie do własnych potrzeb także konstruktorów	866
Programowanie zorientowane obiektowo jest prostsze, niż się wydaje	868
Inne sposoby łączenia klas	869
Krok 6. — wykorzystywanie narzędzi do introspekcji	872
Specjalne atrybuty klas	873
Uniwersalne narzędzie do wyświetlania	874
Atrybuty instancji a atrybuty klas	875
Nazwy w klasach narzędziowych	876
Ostateczna postać naszych klas	877
Krok 7. i ostatni — przechowywanie obiektów w bazie danych	879
Obiekty pickle i shelve	879
Przechowywanie obiektów w bazie danych za pomocą shelve	880
Interaktywna eksploracja obiektów shelve	882
Uaktualnianie obiektów w pliku shelve	884
Przyszłe kierunki rozwoju	885
Podsumowanie rozdziału	887
Sprawdź swoją wiedzę — quiz	887
Sprawdź swoją wiedzę — odpowiedzi	888

29. Szczegóły kodowania klas 891

Instrukcja class	891
Ogólna forma	892
Przykład	892
Metody	894
Przykład metody	895
Wywoływanie konstruktorów klas nadrzędnych	896
Inne możliwości wywoływania metod	896
Dziedziczenie	897
Tworzenie drzewa atrybutów	897
Specjalizacja odziedziczonych metod	898
Techniki interfejsów klas	899
Abstrakcyjne klasy nadrzędne	900
Przestrzenie nazw — cała historia	903
Proste nazwy — globalne, o ile nie są przypisane	903
Nazwy atrybutów — przestrzenie nazw obiektów	903

Zen przestrzeni nazw Pythona — przypisanie klasyfikują zmienne	904
Klasy zagnieżdżone — jeszcze kilka słów o regule LEGB	906
Słowniki przestrzeni nazw — przegląd	908
Łącza przestrzeni nazw — przechodzenie w górę drzewa klas	910
Raz jeszcze o notkach dokumentacyjnych	912
Klasy a moduły	914
Podsumowanie rozdziału	914
Sprawdź swoją wiedzę — quiz	915
Sprawdź swoją wiedzę — odpowiedzi	915
30. Przeciążanie operatorów	917
Podstawy	917
Konstruktory i wyrażenia — <code>__init__</code> i <code>__sub__</code>	918
Często spotykane metody przeciążania operatorów	918
Indeksowanie i wycinanie — <code>__getitem__</code> i <code>__setitem__</code>	920
Wycinki	921
Wycinanie i indeksowanie w Pythonie 2.x	923
Metoda <code>__index__</code> w wersji 3.x nie służy do indeksowania!	923
Iteracja po indeksie — <code>__getitem__</code>	924
Obiekty iteratorów — <code>__iter__</code> i <code>__next__</code>	925
Iteratory zdefiniowane przez użytkownika	925
Wiele iteracji po jednym obiekcie	928
Alternatywa: metoda <code>__iter__</code> i instrukcja <code>yield</code>	930
Test przynależności — <code>__contains__</code> , <code>__iter__</code> i <code>__getitem__</code>	935
Dostęp do atrybutów — <code>__getattr__</code> oraz <code>__setattr__</code>	937
Odwołania do atrybutów	937
Przypisywanie wartości i usuwanie atrybutów	938
Inne narzędzia do zarządzania atrybutami	940
Emulowanie prywatności w atrybutach instancji	940
Reprezentacje łańcuchów — <code>__repr__</code> oraz <code>__str__</code>	941
Po co nam dwie metody wyświetlania?	942
Uwagi dotyczące wyświetlania	944
Dodawanie prawostronne i miejscowa modyfikacja: metody <code>__radd__</code> i <code>__iadd__</code>	945
Dodawanie prawostronne	945
Dodawanie w miejscu	948
Wywołania — <code>__call__</code>	949
Interfejsy funkcji i kod oparty na wywołaniach zwrotnych	950
Porównania — <code>__lt__</code> , <code>__gt__</code> i inne	952
Metoda <code>__cmp__</code> w 2.x	953
Testy logiczne — <code>__bool__</code> i <code>__len__</code>	954
Metody logiczne w Pythonie 2.x	955

Destrukcja obiektu — __del__	956
Uwagi dotyczące stosowania destruktorów	957
Podsumowanie rozdziału	958
Sprawdź swoją wiedzę — quiz	958
Sprawdź swoją wiedzę — odpowiedzi	958
31. Projektowanie z użyciem klas	961
Python a programowanie zorientowane obiektowo	961
Polimorfizm to interfejsy, a nie sygnatury wywołań	962
Programowanie zorientowane obiektowo i dziedziczenie — związek „jest”	963
Programowanie zorientowane obiektowo i kompozycja — związki typu „ma”	964
Raz jeszcze procesor strumienia danych	966
Programowanie zorientowane obiektowo a delegacja — obiekty „opakowujące”	969
Pseudoprywatne atrybuty klas	971
Przegląd zniekształcania nazw zmiennych	972
Po co używa się atrybutów pseudoprywatnych	973
Metody są obiektami — z wiązaniem i bez wiązania	975
W wersji 3.x metody niezwiązane są funkcjami	977
Metody związane i inne obiekty wywoływane	978
Klasy są obiektami — uniwersalne fabryki obiektów	981
Do czego służą fabryki	982
Dziedziczenie wielokrotne — klasy mieszane	983
Tworzenie klas mieszanych	984
Inne zagadnienia związane z projektowaniem	1002
Podsumowanie rozdziału	1003
Sprawdź swoją wiedzę — quiz	1003
Sprawdź swoją wiedzę — odpowiedzi	1003
32. Zaawansowane zagadnienia związane z klasami	1005
Rozszerzanie typów wbudowanych	1006
Rozszerzanie typów za pomocą osadzania	1006
Rozszerzanie typów za pomocą klas podrzędnych	1007
Klasy w nowym stylu	1009
Jak nowy jest nowy styl	1010
Nowości w klasach w nowym stylu	1011
Pomijanie instancji we wbudowanych operacjach przy pobieraniu atrybutów	1012
Zmiany w modelu typów	1017
Wszystkie obiekty dziedziczą po klasie object	1020
Zmiany w dziedziczeniu diamentowym	1022
Więcej o kolejności odwzorowywania nazw	1026
Przykład — wiązanie atrybutów ze źródłami dziedziczenia	1028

Nowości w klasach w nowym stylu	1034
Sloty: deklaracje atrybutów	1034
Właściwości klas: dostęp do atrybutów	1043
Narzędzia atrybutów: <code>__getattr__</code> i deskryptory	1045
Inne zmiany i rozszerzenia klas	1046
Metody statyczne oraz metody klasy	1047
Do czego potrzebujemy metod specjalnych	1047
Metody statyczne w 2.x i 3.x	1048
Alternatywy dla metod statycznych	1049
Używanie metod statycznych i metod klas	1051
Zliczanie instancji z użyciem metod statycznych	1052
Zliczanie instancji z metodami klas	1053
Dekoratory i metaklasy — część 1.	1056
Podstawowe informacje o dekoratorach funkcji	1056
Pierwsze spojrzenie na funkcję dekoratora zdefiniowaną przez użytkownika	1058
Pierwsze spojrzenie na dekoratory klas i metaklasy	1059
Dalsza lektura	1061
Wbudowana funkcja <code>super</code> : zmiana na lepsze czy na gorsze?	1062
Wielka debata o funkcji <code>super</code>	1062
Tradycyjny, uniwersalny i ogólny sposób wywoływania klasy nadrzędnej	1063
Podstawy i kompromisy użycia funkcji <code>super</code>	1064
Zalety funkcji <code>super</code> : zmiany drzewa i kierowania metod	1069
Zmiana klasy w trakcie działania programu a funkcja <code>super</code>	1070
Kooperatywne kierowanie metod w drzewie wielokrotnego dziedziczenia	1071
Podsumowanie funkcji <code>super</code>	1082
Pułapki związane z klasami	1083
Modyfikacja atrybutów klas może mieć efekty uboczne	1084
Modyfikowanie mutowalnych atrybutów klas również może mieć efekty uboczne	1085
Dziedziczenie wielokrotne — kolejność ma znaczenie	1086
Zakresy w metodach i klasach	1087
Różne pułapki związane z klasami	1088
Przesadne opakowywanie	1089
Podsumowanie rozdziału	1089
Sprawdź swoją wiedzę — quiz	1090
Sprawdź swoją wiedzę — odpowiedzi	1090
Sprawdź swoją wiedzę — ćwiczenia do części szóstej	1091

Część VII. Wyjątki oraz narzędzia1099

33. Podstawy wyjątków	1101
Po co używa się wyjątków	1101
Role wyjątków	1102
Wyjątki w skrócie	1103
Domyślny program obsługi wyjątków	1103
Przechwytywanie wyjątków	1104
Zgłaszanie wyjątków	1105
Wyjątki zdefiniowane przez użytkownika	1106
Działania końcowe	1106
Podsumowanie rozdziału	1109
Sprawdź swoją wiedzę — quiz	1109
Sprawdź swoją wiedzę — odpowiedzi	1110
 34. Szczegółowe informacje dotyczące wyjątków	 1111
Instrukcja try/except/else	1111
Jak działa instrukcja try	1112
Części instrukcji try	1113
Część try/else	1115
Przykład — zachowanie domyślne	1116
Przykład — przechwytywanie wbudowanych wyjątków	1117
Instrukcja try/finally	1118
Przykład — działania kończące kod z użyciem try/finally	1119
Połączona instrukcja try/except/finally	1120
Składnia połączonej instrukcji try	1121
Łączenie finally oraz except za pomocą zagnieżdżenia	1121
Przykład połączonego try	1122
Instrukcja raise	1123
Zgłaszanie wyjątków	1124
Zakresy widoczności zmiennych w instrukcjach try i except	1125
Przekazywanie wyjątków za pomocą raise	1126
Łańcuchy wyjątków w Pythonie 3.x — raise from	1127
Instrukcja assert	1128
Przykład — wyłapywanie ograniczeń (ale nie błędów!)	1129
Menedżery kontekstu with/as	1130
Podstawowe zastosowanie	1130
Protokół zarządzania kontekstem	1132
Kilka menedżerów kontekstu w wersjach 3.1, 2.7 i nowszych	1133
Podsumowanie rozdziału	1135
Sprawdź swoją wiedzę — quiz	1135
Sprawdź swoją wiedzę — odpowiedzi	1136

35. Obiekty wyjątków	1137
Wyjątki — powrót do przeszłości	1138
Wyjątki oparte na łańcuchach znaków znikają	1138
Wyjątki oparte na klasach	1139
Tworzenie klas wyjątków	1140
Do czego służą hierarchie wyjątków	1142
Wbudowane klasy wyjątków	1144
Kategorie wbudowanych wyjątków	1146
Domyślne wyświetlanie oraz stan	1147
Własne sposoby wyświetlania	1148
Własne dane oraz zachowania	1149
Udostępnianie szczegółów wyjątku	1150
Udostępnianie metod wyjątków	1150
Podsumowanie rozdziału	1152
Sprawdź swoją wiedzę — quiz	1152
Sprawdź swoją wiedzę — odpowiedzi	1152
36. Projektowanie z wykorzystaniem wyjątków	1155
Zagnieżdżanie programów obsługi wyjątków	1155
Przykład — zagnieżdżanie przebiegu sterowania	1157
Przykład — zagnieżdżanie składniowe	1157
Zastosowanie wyjątków	1159
Wychodzenie z głęboko zagnieżdżonych pętli: instrukcja go to	1159
Wyjątki nie zawsze są błędami	1160
Funkcje mogą sygnalizować warunki za pomocą raise	1160
Zamykanie plików oraz połączeń z serwerem	1161
Debugowanie z wykorzystaniem zewnętrznych instrukcji try	1162
Testowanie kodu wewnątrz tego samego procesu	1163
Więcej informacji na temat funkcji sys.exc_info	1164
Wyświetlanie błędów i śladów stosu	1164
Wskazówki i pułapki dotyczące projektowania wyjątków	1165
Co powinniśmy opakować w try	1166
Jak nie przechwytywać zbyt wiele — unikanie pustych except i wyjątków	1166
Jak nie przechwytywać zbyt mało — korzystanie z kategorii opartych na klasach	1168
Podsumowanie podstaw języka Python	1169
Zbiór narzędzi Pythona	1169
Narzędzia programistyczne przeznaczone do większych projektów	1170
Podsumowanie rozdziału	1174
Sprawdź swoją wiedzę — quiz	1174
Sprawdź swoją wiedzę — odpowiedzi	1175
Sprawdź swoją wiedzę — ćwiczenia do części siódmej	1175

Część VIII. Zagadnienia zaawansowane 1177

37. Łańcuchy znaków Unicode oraz łańcuchy bajtowe 1179

Zmiany w łańcuchach znaków w Pythonie 3.x	1180
Podstawy łańcuchów znaków	1181
Kodowanie znaków	1181
Jak Python zapisuje ciągi znaków w pamięci	1183
Typy łańcuchów znaków Pythona	1185
Pliki binarne i tekstowe	1186
Podstawy kodowania ciągów znaków	1188
Literały tekstowe w Pythonie 3.x	1188
Literały tekstowe w Pythonie 2.x	1190
Konwersje typów ciągów	1190
Kod łańcuchów znaków Unicode	1192
Kod tekstu z zakresu ASCII	1192
Kod tekstu spoza zakresu ASCII	1193
Kodowanie i dekodowanie tekstu spoza zakresu ASCII	1194
Inne techniki kodowania łańcuchów Unicode	1194
Literały bajtowe	1196
Konwersja kodowania	1197
Łańcuchy znaków Unicode w Pythonie 2.x	1198
Deklaracje typu kodowania znaków pliku źródłowego	1200
Wykorzystywanie obiektów bytes z Pythona 3.x	1201
Wywołania metod	1202
Operacje na sekwencjach	1203
Inne sposoby tworzenia obiektów bytes	1203
Mieszanie typów łańcuchów znaków	1204
Obiekt bytearray w wersji 3.x (oraz 2.6 lub nowszej)	1205
Typ bytearray w akcji	1205
Podsumowanie typów ciągów znaków w Pythonie 3.x	1207
Wykorzystywanie plików tekstowych i binarnych	1208
Podstawy plików tekstowych	1208
Tryby tekstowy i binarny w Pythonie 2.x i 3.x	1209
Brak dopasowania typu i zawartości w Pythonie 3.x	1210
Wykorzystywanie plików Unicode	1212
Odczyt i zapis Unicode w Pythonie 3.x	1212
Obsługa BOM w Pythonie 3.x	1213
Pliki Unicode w Pythonie 2.x	1216
Unicode w nazwach plików i w strumieniach	1217

Inne zmiany w narzędziach do przetwarzania łańcuchów znaków w Pythonie 3.x	1218
Moduł dopasowywania wzorców re	1218
Moduł danych binarnych struct	1219
Moduł serializacji obiektów pickle	1221
Narzędzia do analizy składniowej XML	1223
Podsumowanie rozdziału	1227
Sprawdź swoją wiedzę — quiz	1227
Sprawdź swoją wiedzę — odpowiedzi	1228
38. Zarządzane atrybuty	1231
Po co zarządza się atrybutami	1231
Wstawianie kodu wykonywanego w momencie dostępu do atrybutów	1232
Właściwości	1233
Podstawy	1233
Pierwszy przykład	1234
Obliczanie atrybutów	1235
Zapisywanie właściwości w kodzie za pomocą dekoratorów	1236
Deskryptory	1237
Podstawy	1238
Pierwszy przykład	1241
Obliczone atrybuty	1242
Wykorzystywanie informacji o stanie w deskryptorach	1243
Powiązania pomiędzy właściwościami a deskryptorami	1246
Metody <code>__getattr__</code> oraz <code>__getattribute__</code>	1248
Podstawy	1249
Pierwszy przykład	1252
Obliczanie atrybutów	1253
Porównanie metod <code>__getattr__</code> oraz <code>__getattribute__</code>	1255
Porównanie technik zarządzania atrybutami	1256
Przechwytywanie atrybutów wbudowanych operacji	1258
Przykład — sprawdzanie poprawności atrybutów	1266
Wykorzystywanie właściwości do sprawdzania poprawności	1266
Wykorzystywanie deskryptorów do sprawdzania poprawności	1268
Wykorzystywanie metody <code>__getattr__</code> do sprawdzania poprawności	1272
Wykorzystywanie metody <code>__getattribute__</code> do sprawdzania poprawności	1273
Podsumowanie rozdziału	1275
Sprawdź swoją wiedzę — quiz	1275
Sprawdź swoją wiedzę — odpowiedzi	1275

39. Dekoratory	1277
Czym jest dekorator	1277
Zarządzanie wywołaniami oraz instancjami	1278
Zarządzanie funkcjami oraz klasami	1278
Wykorzystywanie i definiowanie dekoratorów	1279
Do czego służą dekoratory	1279
Podstawy	1281
Dekoratory funkcji	1281
Dekoratory klas	1284
Zagnieżdżanie dekoratorów	1287
Argumenty dekoratorów	1288
Dekoratory zarządzają także funkcjami oraz klasami	1289
Kod dekoratorów funkcji	1290
Śledzenie wywołań	1290
Możliwości w zakresie zachowania informacji o stanie	1292
Uwagi na temat klas I — dekorowanie metod klas	1296
Mierzenie czasu wywołania	1301
Dodawanie argumentów dekoratora	1304
Kod dekoratorów klas	1307
Klasy singletona	1307
Śledzenie interfejsów obiektów	1309
Uwagi na temat klas II — zachowanie większej liczby instancji	1312
Dekoratory a funkcje zarządzające	1314
Do czego służą dekoratory (raz jeszcze)	1315
Bezpośrednie zarządzanie funkcjami oraz klasami	1316
Przykład — atrybuty „prywatne” i „publiczne”	1319
Implementacja atrybutów prywatnych	1319
Szczegóły implementacji I	1321
Uogólnienie kodu pod kątem deklaracji atrybutów jako publicznych	1322
Szczegóły implementacji II	1325
Znane problemy	1326
W Pythonie nie chodzi o kontrolę	1333
Przykład — sprawdzanie poprawności argumentów funkcji	1333
Cel	1334
Prosty dekorator sprawdzający przedziały dla argumentów pozycyjnych	1335
Uogólnienie kodu pod kątem słów kluczowych i wartości domyślnych	1337
Szczegóły implementacji	1339
Znane problemy	1342
Argumenty dekoratora a adnotacje funkcji	1344
Inne zastosowania — sprawdzanie typów (skoro nalegamy!)	1346

Podsumowanie rozdziału	1347
Sprawdź swoją wiedzę — quiz	1347
Sprawdź swoją wiedzę — odpowiedzi	1348
40. Metaklasy	1357
Tworzyć metaklasy czy tego nie robić?	1358
Zwiększające się poziomy magii	1359
Język pełen haczyków	1360
Wady funkcji pomocniczych	1361
Metaklasy a dekoratory klas — runda 1.	1363
Model metaklasy	1365
Klasy są instancjami obiektu type	1365
Metaklasy są klasami podrzędnymi klasy type	1368
Protokół instrukcji class	1368
Deklarowanie metaklas	1369
Deklarowanie w wersji 3.x	1370
Deklarowanie w wersji 2.x	1370
Kierowanie metaklas w wersjach 3.x i 2.x	1371
Tworzenie metaklas	1371
Prosta metaklasa	1372
Dostosowywanie tworzenia do własnych potrzeb oraz inicjalizacja	1373
Pozostałe sposoby tworzenia metaklas	1374
Instancje a dziedziczenie	1379
Metaklasa a klasa nadrzędna	1380
Dziedziczenie: pełna historia	1382
Metody metaklas	1388
Metody metaklasy a metody klasy	1388
Przeciążanie operatorów w metodach metaklasy	1389
Przykład — dodawanie metod do klas	1390
Ręczne rozszerzanie	1390
Rozszerzanie oparte na metaklasie	1391
Metaklasy a dekoratory klas — runda 2.	1393
Przykład — zastosowanie dekoratorów do metod	1398
Ręczne śledzenie za pomocą dekoracji	1398
Śledzenie za pomocą metaklas oraz dekoratorów	1399
Zastosowanie dowolnego dekoratora do metod	1400
Metaklasy a dekoratory klas — runda 3. (i ostatnia)	1402
Podsumowanie rozdziału	1404
Sprawdź swoją wiedzę — quiz	1404
Sprawdź swoją wiedzę — odpowiedzi	1405



41. Wszystko, co najlepsze	1407
Paradoks Pythona	1407
„Opcjonalne” cechy języka	1408
Przeciwko niepokojącym usprawnieniom	1408
Złożoność a siła	1409
Prostota a elitarność	1410
Końcowe wnioski	1411
Dokąd dalej?	1411
Na bis: wydrukuj swój certyfikat!	1412
 Dodatki	 1415
A Instalacja i konfiguracja	1417
B Uruchamianie Pythona 3.x w systemie Windows	1431
C Zmiany w języku Python a niniejsza książka	1445
D Rozwiązania ćwiczeń podsumowujących poszczególne części książki	1457