

Spis treści

Przedmowa (29)

Część I: Wprowadzenie (47)

1. Pytania i odpowiedzi dotyczące Pythona (49)

Dlaczego ludzie używają Pythona? (49)

Jakość oprogramowania (50)

Wydajność programistów (51)

Czy Python jest językiem skryptowym? (51)

Jakie są zatem wady Pythona? (53)

Kto dzisiaj używa Pythona? (53)

Co mogę zrobić za pomocą Pythona? (55)

Programowanie systemowe (55)

Graficzne interfejsy użytkownika (55)

Skrypty internetowe (56)

Integracja komponentów (56)

Programowanie bazodanowe (57)

Szybkie prototypowanie (57)

Programowanie numeryczne i naukowe (57)

Gry, grafika, porty szeregowo, XML, roboty i tym podobne (58)

Jakie wsparcie techniczne ma Python? (58)

Jakie są techniczne mocne strony Pythona? (59)

Jest zorientowany obiektowo (59)

Jest darmowy (59)

Jest przenośny (60)

Ma duże możliwości (61)

Można go łączyć z innymi językami (62)

Jest łatwy w użyciu (62)

Jest łatwy do nauczenia się (62)

Zawdzięcza swoją nazwę Monty Pythonowi (63)

Jak Python wygląda na tle innych języków? (63)

Podsumowanie rozdziału (64)

Sprawdź swoją wiedzę - quiz (65)

Sprawdź swoją wiedzę - odpowiedzi (65)

2. Jak Python wykonuje programy? (69)

Wprowadzenie do interpretera Pythona (69)

Wykonywanie programu (71)

Z punktu widzenia programisty (71)

Z punktu widzenia Pythona (72)

Warianty modeli wykonywania (74)

Alternatywne implementacje Pythona (75)

Narzędzia do optymalizacji wykonywania (76)

Zamrożone pliki binarne (78)

Inne opcje wykonywania (78)

Przyszłe możliwości? (79)

Podsumowanie rozdziału (80)

Sprawdź swoją wiedzę - quiz (80)

Sprawdź swoją wiedzę - odpowiedzi (80)

3. Jak wykonuje się programy? (81)

Interaktywny wiersz poleceń (81)

Interaktywne wykonywanie kodu (82)

Do czego służy sesja interaktywna? (83)

Wykorzystywanie sesji interaktywnej (85)

Systemowe wiersze poleceń i pliki (87)

Pierwszy skrypt (87)

Wykonywanie plików za pomocą wiersza poleceń (88)

Wykorzystywanie wierszy poleceń i plików (90)

Skrypty wykonywalne Uniksa (#!) (91)

Kliknięcie ikony pliku (92)

Kliknięcie ikony w systemie Windows (93)

Sztuczka z funkcją input (94)

Inne ograniczenia klikania ikon (95)

Importowanie i przeładowywanie modułów (96)

Więcej o modułach - atrybuty (98)

Uwagi na temat używania instrukcji import i reload (100)

Wykorzystywanie exec do wykonywania plików modułów (101)

Interfejs użytkownika IDLE (102)

Podstawy IDLE (103)

Korzystanie z IDLE (105)

Zaawansowane opcje IDLE (106)

Inne IDE (107)

Inne opcje wykonywania kodu (108)

Osadzanie wywołań (108)

Zamrożone binarne pliki wykonywalne (109)

Uruchamianie kodu w edytorze tekstowym (110)

Jeszcze inne możliwości uruchamiania (110)

Przyszłe możliwości (110)

Jaką opcję wybrać? (111)

Podsumowanie rozdziału (112)

Sprawdź swoją wiedzę - quiz (113)

Sprawdź swoją wiedzę - odpowiedzi (113)

Sprawdź swoją wiedzę - ćwiczenia do części pierwszej (114)

Część II: Typy i operacje (117)

4. Wprowadzenie do typów obiektów Pythona (119)

Po co korzysta się z typów wbudowanych? (120)

Najważniejsze typy danych w Pythonie (121)

Liczby (122)

łańcuchy znaków (124)

Operacje na sekwencjach (124)

Niezmienność (126)

Metody specyficzne dla typu (126)

Otrzymanie pomocy (127)

Inne sposoby kodowania łańcuchów znaków (128)

Dopasowywanie wzorców (129)

Listy (130)

Operacje na sekwencjach (130)

Operacje specyficzne dla typu (130)

Sprawdzanie granic (131)

Zagnieżdżanie (131)

Listy składane (132)

Słowniki (133)

Operacje na odwzorowaniach (134)

Zagnieżdżanie raz jeszcze (134)

Sortowanie kluczy - pętle for (136)

Iteracja i optymalizacja (137)

Brakujące klucze - testowanie za pomocą if (138)

Krotki (139)

Czemu służą krotki? (140)

Pliki (140)

Inne narzędzia podobne do plików (142)

Inne typy podstawowe (142)

Jak zepsuć elastyczność kodu (143)

Klasy zdefiniowane przez użytkownika (144)

I wszystko inne (145)

Podsumowanie rozdziału (145)

Sprawdź swoją wiedzę - quiz (146)

Sprawdź swoją wiedzę - odpowiedzi (146)

5. Typy liczbowe (149)

Podstawy typów liczbowych Pythona (149)

Literały liczbowe (150)

Wbudowane narzędzia liczbowe (151)

Operatory wyrażeń Pythona (152)

Liczby w akcji (156)

Zmienne i podstawowe wyrażenia (157)

Formaty wyświetlania liczb (158)

Porównania - zwykłe i łączone (160)

Dzielenie - klasyczne, bez reszty i prawdziwe (161)

Precyzja liczb całkowitych (164)

Liczby zespolone (165)

Notacja szesnastkowa, ósemkowa i dwójkowa (165)

Operacje poziomego bitowego (167)

Inne wbudowane narzędzia liczbowe (168)

Inne typy liczbowe (170)

Typ liczby dziesiętnej (170)

Typ liczby ułamkowej (172)

Zbiory (176)

Wartości Boolean (181)

Dodatkowe rozszerzenia numeryczne (182)

Podsumowanie rozdziału (183)

Sprawdź swoją wiedzę - quiz (183)

Sprawdź swoją wiedzę - odpowiedzi (184)

6. Wprowadzenie do typów dynamicznych (185)

Sprawa brakujących deklaracji typu (185)

Zmienne, obiekty i referencje (186)

Typy powiązane są z obiektami, a nie ze zmiennymi (187)

Obiekty są uwalniane (188)

Referencje współdzielone (190)

Referencje współdzielone a modyfikacje w miejscu (191)

Referencje współdzielone a równość (193)

Typy dynamiczne są wszędzie (194)

Podsumowanie rozdziału (194)

Sprawdź swoją wiedzę - quiz (195)

Sprawdź swoją wiedzę - odpowiedzi (195)

7. Łańcuchy znaków (197)

Literały łańcuchów znaków (199)

Łańcuchy znaków w apostrofach i cudzysłowach są tym samym (200)

Sekwencje ucieczki reprezentują bajty specjalne (200)

Surowe łańcuchy znaków blokują sekwencje ucieczki (203)

Potrójne cudzysłowy i apostrofy kodują łańcuchy znaków będące wielowierszowymi blokami (204)

Łańcuchy znaków w akcji (205)

Podstawowe operacje (206)

Indeksowanie i wycinki (207)

Narzędzia do konwersji łańcuchów znaków (210)

Modyfikowanie łańcuchów znaków (213)

Metody łańcuchów znaków (214)

Przykłady metod łańcuchów znaków - modyfikowanie (215)

Przykłady metod łańcuchów znaków - analiza składniowa tekstu (218)

Inne znane metody łańcuchów znaków w akcji (219)

Oryginalny moduł string (usunięty w 3.0) (220)

Wyrażenia formatujące łańcuchy znaków (221)

Zaawansowane wyrażenia formatujące (222)

Wyrażenia formatujące z użyciem słownika (224)

Metoda format (225)

Podstawy (225)

Użycie kluczy, atrybutów i przesunięć (226)

Formatowanie specjalizowane (227)

Porównanie z wyrażeniami formatującymi (229)

Po co nam kolejny mechanizm formatujący? (232)

Generalne kategorie typów (235)

Typy z jednej kategorii współdzielą zbiory operacji (235)

Typy zmienne można modyfikować w miejscu (236)

Podsumowanie rozdziału (236)

Sprawdź swoją wiedzę - quiz (236)

Sprawdź swoją wiedzę - odpowiedzi (237)

8. Listy oraz słowniki (239)

Listy (239)

Listy w akcji (241)

Podstawowe operacje na listach (241)

Iteracje po listach i składanie list (242)

Indeksowanie, wycinki i macierze (243)

Modyfikacja list w miejscu (244)

Słowniki (248)

Słowniki w akcji (249)

Podstawowe operacje na słownikach (250)

Modyfikacja słowników w miejscu (251)

Inne metody słowników (252)

Przykład z tabelą języków programowania (253)

Uwagi na temat korzystania ze słowników (254)

Inne sposoby tworzenia słowników (257)

Zmiany dotyczące słowników w 3.0 (258)

Podsumowanie rozdziału (264)

Sprawdź swoją wiedzę - quiz (264)

Sprawdź swoją wiedzę - odpowiedzi (264)

9. Krotki, pliki i pozostałe (267)

Krotki (267)

Krotki w akcji (268)

Dlaczego istnieją listy i krotki? (271)

Pliki (271)

Otwieranie plików (272)

Wykorzystywanie plików (273)

Pliki w akcji (274)

Inne narzędzia powiązane z plikami (280)

Raz jeszcze o kategoriach typów (281)

Elastyczność obiektów (282)

Referencje a kopie (283)

Porównania, równość i prawda (285)

Porównywanie słowników w Pythonie 3.0 (287)

Znaczenie True i False w Pythonie (288)

Hierarchie typów Pythona (290)

Obiekty typów (291)

Inne typy w Pythonie (291)

Pułapki typów wbudowanych (292)

Przypisanie tworzy referencje, nie kopie (292)

Powtórzenie dodaje jeden poziom zagłębienia (293)

Uwaga na cykliczne struktury danych (293)

Typów niezmiennych nie można modyfikować w miejscu (294)

Podsumowanie rozdziału (294)

Sprawdź swoją wiedzę - quiz (294)

Sprawdź swoją wiedzę - odpowiedzi (295)

Sprawdź swoją wiedzę - ćwiczenia do części drugiej (295)

Część III: Instrukcje i składnia (299)

10. Wprowadzenie do instrukcji Pythona (301)

Raz jeszcze o strukturze programu Pythona (301)

Instrukcje Pythona (301)

Historia dwóch if (303)

Co dodaje Python (304)

Co usuwa Python (304)

Skąd bierze się składnia indentacji? (306)

Kilka przypadków specjalnych (308)

Szybki przykład - interaktywne pętle (310)

Prosta pętla interaktywna (310)

Wykonywanie obliczeń na danych użytkownika (311)

Obsługa błędów za pomocą sprawdzania danych wejściowych (312)

Obsługa błędów za pomocą instrukcji try (313)

Kod zagnieżdżony na trzy poziomy głębokości (314)

Podsumowanie rozdziału (315)

Sprawdź swoją wiedzę - quiz (315)

Sprawdź swoją wiedzę - odpowiedzi (315)

11. Przypisania, wyrażenia i wyświetlanie (317)

Instrukcje przypisania (317)

Formy instrukcji przypisania (318)

Przypisanie sekwencji (319)

Rozszerzona składnia rozpakowania sekwencji w 3.0 (322)

Przypisanie z wieloma celami (325)

Przypisania rozszerzone (326)

Reguły dotyczące nazw zmiennych (329)

Instrukcje wyrażień (332)

Instrukcje wyrażień i modyfikacje w miejscu (333)

Polecenia print (334)

Funkcja print Pythona 3.0 (334)

Instrukcja print w Pythonie 2.6 (337)

Przekierowanie strumienia wyjściowego (338)

Wyświetlanie niezależne od wersji (341)

Podsumowanie rozdziału (343)

Sprawdź swoją wiedzę - quiz (344)

Sprawdź swoją wiedzę - odpowiedzi (344)

12. Testy if i reguły składni (345)

Instrukcje if (345)

Ogólny format (345)

Proste przykłady (346)

Rozgałęzienia kodu (346)

Reguły składni Pythona (348)

Ograniczniki bloków - reguły indentacji (349)

Ograniczniki instrukcji - wiersze i kontynuacje (351)

Kilka przypadków specjalnych (352)

Testy prawdziwości (353)

Wyrażenie trójargumentowe if/else (355)

Podsumowanie rozdziału (356)

Sprawdź swoją wiedzę - quiz (357)

Sprawdź swoją wiedzę - odpowiedzi (358)

13. Pętle while i for (359)

Pętla while (359)

Ogólny format (360)

Przykłady (360)

Instrukcje break, continue, pass oraz else w pętli (361)

Ogólny format pętli (361)

Instrukcja pass (361)

Instrukcja continue (363)

Instrukcja break (363)

Instrukcja else (364)

Pętla for (365)

Ogólny format (365)

Przykłady (367)

Techniki tworzenia pętli (372)

Pętla liczników - while i range (373)

Przechodzenie niewyczerpujące - range i wycinki (374)

Modyfikacja list - range (375)

Przechodzenie równoległe - zip oraz map (376)

Generowanie wartości przesunięcia i elementów - enumerate (379)

Podsumowanie rozdziału (380)

Sprawdź swoją wiedzę - quiz (380)

Sprawdź swoją wiedzę - odpowiedzi (380)

14. Iteracje i składanie list - część 1. (383)

Pierwsze spojrzenie na iteratory (383)

Protokół iteracyjny, iteratory plików (384)

Kontrola iteracji - iter i next (386)

Inne iteratory typów wbudowanych (388)

Listy składane - wprowadzenie (390)

Podstawy list składanych (390)

Wykorzystywanie list składanych w plikach (391)

Rozszerzona składnia list składanych (392)

Inne konteksty iteracyjne (393)

Nowe obiekty iterowane w Pythonie 3.0 (397)

Iterator range() (397)

Iteratory map(), zip() i filter() (398)

Kilka iteratorów na tym samym obiekcie (399)

Iteratory widoku słownika (400)

Inne zagadnienia związane z iteratorami (402)

Podsumowanie rozdziału (402)

Sprawdź swoją wiedzę - quiz (402)

Sprawdź swoją wiedzę - odpowiedzi (403)

15. Wprowadzenie do dokumentacji (405)

Źródła dokumentacji Pythona (405)

Komentarze ze znakami # (406)

Funkcja dir (406)

łańcuchy znaków dokumentacji - `__doc__` (407)

PyDoc - funkcja help (410)

PyDoc - raporty HTML (412)

Zbiór standardowej dokumentacji (415)

Zasoby internetowe (415)

Publikowane książki (416)

Często spotykane problemy programistyczne (417)

Podsumowanie rozdziału (419)

Sprawdź swoją wiedzę - quiz (419)

Sprawdź swoją wiedzę - odpowiedzi (419)

Ćwiczenia do części trzeciej (420)

Część IV: Funkcje (423)

16. Podstawy funkcji (425)

Po co używa się funkcji? (426)

Tworzenie funkcji (426)

Instrukcje def (428)

Instrukcja def uruchamiana jest w czasie wykonania (428)

Pierwszy przykład - definicje i wywoływanie (429)

Definicja (429)

Wywołanie (430)

Polimorfizm w Pythonie (430)

Drugi przykład - przecinające się sekwencje (431)

Definicja (432)

Wywołania (432)

Raz jeszcze o polimorfizmie (433)

Zmienne lokalne (433)

Podsumowanie rozdziału (434)

Sprawdź swoją wiedzę - quiz (434)

Sprawdź swoją wiedzę - odpowiedzi (434)

17. Zakresy (437)

Podstawy zakresów w Pythonie (437)

Reguły dotyczące zakresów (438)

Rozwiązywanie konfliktów w zakresie nazw - reguła LEGB (440)

Przykład zakresu (441)

Zakres wbudowany (442)

Instrukcja global (443)

Minimalizowanie stosowania zmiennych globalnych (445)

Minimalizacja modyfikacji dokonywanych pomiędzy plikami (446)

Inne metody dostępu do zmiennych globalnych (447)

Zakresy a funkcje zagnieżdżone (448)

Szczegóły dotyczące zakresów zagnieżdżonych (449)

Przykład zakresu zagnieżdżonego (449)

Instrukcja nonlocal (455)

Podstawy instrukcji nonlocal (455)

Instrukcja nonlocal w akcji (456)

Czemu służą zmienne nielocalne? (458)

Podsumowanie rozdziału (462)

Sprawdź swoją wiedzę - quiz (462)

Sprawdź swoją wiedzę - odpowiedzi (463)

18. Argumenty (465)

Podstawy przekazywania argumentów (465)

Argumenty a współdzielone referencje (466)

Unikanie modyfikacji zmiennych argumentów (468)

Symulowanie parametrów wyjścia (469)

Specjalne tryby dopasowania argumentów (470)

Podstawy (470)

Składnia dopasowania (471)

Dopasowywanie argumentów - szczegóły (472)

Przykłady ze słowami kluczowymi i wartościami domyślnymi (473)

Przykład dowolnych argumentów (475)

Argumenty mogące być tylko słowami kluczowymi z Pythona 3.0 (479)

Przykład z funkcją obliczającą minimum (482)

Pełne rozwiązanie (483)

Dodatkowy bonus (484)

Puenta (485)

Uogólnione funkcje działające na zbiorach (485)

Emulacja funkcji print z Pythona 3.0 (486)

Wykorzystywanie argumentów mogących być tylko słowami kluczowymi (487)

Podsumowanie rozdziału (488)

Sprawdź swoją wiedzę - quiz (489)

Sprawdź swoją wiedzę - odpowiedzi (490)

19. Zaawansowane zagadnienia dotyczące funkcji (491)

Koncepcje projektowania funkcji (491)

Funkcje rekurencyjne (493)

Sumowanie z użyciem rekurencji (493)

Implementacje alternatywne (494)

Pętle a rekurencja (495)

Obsługa dowolnych struktur (496)

Obiekty funkcji - atrybuty i adnotacje (497)

Pośrednie wywołania funkcji (497)

Introspekcja funkcji (498)

Atrybuty funkcji (499)

Adnotacje funkcji w Pythonie 3.0 (499)

Funkcje anonimowe - lambda (501)

Wyrażenia lambda (501)

Po co używa się wyrażenia lambda? (503)

Jak łatwo zaciemnić kod napisany w Pythonie (504)

Zagnieżdżone wyrażenia lambda a zakresy (505)

Odwzorowywanie funkcji na sekwencje - map (507)

Narzędzia programowania funkcyjnego - filter i reduce (508)

Podsumowanie rozdziału (510)

Sprawdź swoją wiedzę - quiz (510)

Sprawdź swoją wiedzę - odpowiedzi (510)

20. Iteracje i składanie list - część 2. (513)

Listy składane, podejście drugie - narzędzia funkcyjne (513)

Listy składane kontra map (514)

Dodajemy warunki i pętle zagnieżdżone - filter (515)

Listy składane i macierze (517)

Zrozumieć listy składane (518)

Iteratorów ciąg dalszy - generatory (520)

Funkcje generatorów - yield kontra return (520)

Wyrażenia generatorów - iteratory spotykają złożenia (524)

Funkcje generatorów kontra wyrażenia generatorów (525)

Generatory są jednorazowymi iteratorami (526)

Emulacja funkcji zip i map za pomocą narzędzi iteracyjnych (527)

Generowanie wyników we wbudowanych typach i klasach (531)

Podsumowanie obiektów składanych w 3.0 (533)

Zrozumieć zbiory i słowniki składane (534)

Rozszerzona składnia zbiorów i słowników składanych (534)

Pomiary wydajności implementacji iteratorów (535)

Moduł mytimer (536)

Skrypt mierzący wydajność (536)

Pomiary czasu (537)

Alternatywne moduły mierzące wydajność (539)

Inne sugestie (543)

Pułapki związane z funkcjami (544)

Lokalne nazwy są wykrywane w sposób statyczny (544)

Wartości domyślne i obiekty mutowalne (546)

Funkcje niezwracające wyników (548)

Funkcje zagnieżdżone a zmienne modyfikowane w pętli (548)

Podsumowanie rozdziału (548)

Sprawdź swoją wiedzę - quiz (549)

Sprawdź swoją wiedzę - odpowiedzi (549)

Sprawdź swoją wiedzę - ćwiczenia do części czwartej (550)

Część V: Moduły (553)

21. Moduły - wprowadzenie (555)

Po co używa się modułów? (555)

Architektura programu w Pythonie (556)

Struktura programu (556)

Importowanie i atrybuty (557)

Moduły biblioteki standardowej (558)

Jak działa importowanie (559)

1. Odnalezienie modułu (560)
2. (Ewentualne) Kompilowanie (560)

3. Wykonanie (561)

Ścieżka wyszukiwania modułów (561)

Konfiguracja ścieżki wyszukiwania (563)

Wariacje ścieżki wyszukiwania modułów (564)

Lista sys.path (564)

Wybór pliku modułu (565)

Zaawansowane zagadnienia związane z wyborem modułów (566)

Podsumowanie rozdziału (566)

Sprawdź swoją wiedzę - quiz (567)

Sprawdź swoją wiedzę - odpowiedzi (568)

22. Podstawy tworzenia modułów (569)

Tworzenie modułów (569)

Użycie modułów (570)

Instrukcja import (570)

Instrukcja from (571)

Instrukcja from * (571)

Operacja importowania odbywa się tylko raz (571)

Instrukcje import oraz from są przypisaniami (572)

Modyfikacja zmiennych pomiędzy plikami (573)

Równoważność instrukcji import oraz from (573)

Potencjalne pułapki związane z użyciem instrukcji from (574)

Przestrzenie nazw modułów (575)

Pliki generują przestrzenie nazw (576)

Kwalifikowanie nazw atrybutów (577)

Importowanie a zakresy (578)

Zagnieżdżanie przestrzeni nazw (579)

Przeładowywanie modułów (580)

Podstawy przeładowywania modułów (581)

Przykład przeładowywania z użyciem reload (581)

Podsumowanie rozdziału (582)

Sprawdź swoją wiedzę - quiz (583)

Sprawdź swoją wiedzę - odpowiedzi (584)

23. Pakiety modułów (585)

Podstawy importowania pakietów (585)

Pakiety a ustawienia ścieżki wyszukiwania (586)

Pliki pakietów `__init__.py` (586)

Przykład importowania pakietu (588)

Instrukcja `from` a instrukcja `import` w importowaniu pakietów (589)

Do czego służy importowanie pakietów? (590)

Historia trzech systemów (590)

Względne importowanie pakietów (593)

Zmiany w Pythonie 3.0 (593)

Podstawy importów względnych (594)

Do czego służą importy względne? (595)

Zakres importów względnych (597)

Podsumowanie reguł wyszukiwania modułów (598)

Importy względne w działaniu (598)

Podsumowanie rozdziału (603)

Sprawdź swoją wiedzę - quiz (604)

Sprawdź swoją wiedzę - odpowiedzi (604)

24. Zaawansowane zagadnienia związane z modułami (607)

Ukrywanie danych w modułach (607)

Minimalizacja niebezpieczeństw użycia `from * - _X` oraz `__all__` (608)

Włączanie opcji z przyszłych wersji Pythona (608)

Mieszane tryby użycia - `__name__` oraz `__main__` (609)

Testy jednostkowe z wykorzystaniem `__name__` (610)

Użycie argumentów wiersza poleceń z `__name__` (611)

Modyfikacja ścieżki wyszukiwania modułów (613)

Rozszerzenie `as` dla instrukcji `import` oraz `from` (614)

Moduły są obiektami - metaprogramy (615)

Importowanie modułów za pomocą łańcucha znaków nazwy (617)

Przechodnie przeładowywanie modułów (618)

Projektowanie modułów (621)

Pułapki związane z modułami (622)

W kodzie najwyższego poziomu kolejność instrukcji ma znaczenie (622)

Instrukcja `from` kopiuje nazwy, jednak łączy już nie (623)

Instrukcja `from *` może zaciemnić znaczenie zmiennych (624)

Funkcja `reload` może nie mieć wpływu na obiekty importowane za pomocą `from` (624)

Funkcja `reload` i instrukcja `from` a testowanie interaktywne (625)

Rekurencyjne importowanie za pomocą `from` może nie działać (626)

Podsumowanie rozdziału (627)

Sprawdź swoją wiedzę - quiz (627)

Sprawdź swoją wiedzę - odpowiedzi (628)

Sprawdź swoją wiedzę - ćwiczenia do części piątej (628)

Część VI: Klasy i programowanie zorientowane obiektowo (631)

25. Programowanie zorientowane obiektowo (633)

Po co używa się klas? (634)

Programowanie zorientowane obiektowo z dystansu (635)

Wyszukiwanie dziedziczenia atrybutów (635)

Klasy a instancje (637)

Wywołania metod klasy (638)

Tworzenie drzew klas (638)

Programowanie zorientowane obiektowo oparte jest na ponownym wykorzystaniu kodu (641)

Podsumowanie rozdziału (643)

Sprawdź swoją wiedzę - quiz (644)

Sprawdź swoją wiedzę - odpowiedzi (644)

26. Podstawy tworzenia klas (647)

Klasy generują większą liczbę obiektów instancji (647)

Obiekty klas udostępniają zachowania domyślne (648)

Obiekty instancji są rzeczywistymi elementami (648)

Pierwszy przykład (649)

Klasy dostosowuje się do własnych potrzeb przez dziedziczenie (651)

Drugi przykład (652)

Klasy są atrybutami w modułach (653)

Klasy mogą przechwytywać operatory Pythona (654)

Trzeci przykład (655)

Po co przeciąża się operatory? (657)

Najprostsza klasa Pythona na świecie (658)

Klasy a słowniki (660)

Podsumowanie rozdziału (662)

Sprawdź swoją wiedzę - quiz (662)

Sprawdź swoją wiedzę - odpowiedzi (663)

27. Bardziej realistyczny przykład (665)

Krok 1. - tworzenie instancji (666)

Tworzenie konstruktorów (666)

Testowanie w miarę pracy (667)

Wykorzystywanie kodu na dwa sposoby (668)

Krok 2. - dodawanie metod (669)

Tworzenie kodu metod (671)

Krok 3. - przeciążanie operatorów (673)

Udostępnienie wyświetlania (674)

Krok 4. - dostosowanie zachowania do własnych potrzeb za pomocą klas podrzędnych (675)

Tworzenie klas podrzędnych (675)

Rozszerzanie metod - niewłaściwy sposób (676)

Rozszerzanie metod - właściwy sposób (676)

Polimorfizm w akcji (678)

Dziedziczenie, dostosowanie do własnych potrzeb i rozszerzenie (679)

Programowanie zorientowane obiektowo - idea (680)

Krok 5. - dostosowanie do własnych potrzeb także konstruktorów (680)

Programowanie zorientowane obiektowo jest prostsze, niż się wydaje (682)

Inne sposoby łączenia klas (683)

Krok 6. - wykorzystywanie narzędzi do introspekcji (684)

Specjalne atrybuty klas (686)

Uniwersalne narzędzie do wyświetlania (687)

Atrybuty instancji a atrybuty klas (688)

Rozważania na temat nazw w klasach narzędzi (689)

Ostateczna postać naszych klas (690)

Krok 7. i ostatni - przechowanie obiektów w bazie danych (691)

Obiekty pickle i shelve (691)

Przechowywanie obiektów w bazie danych za pomocą shelve (692)

Interaktywne badanie obiektów shelve (694)

Uaktualnianie obiektów w pliku shelve (695)

Przyszłe kierunki rozwoju (697)

Podsumowanie rozdziału (699)

Sprawdź swoją wiedzę - quiz (699)

Sprawdź swoją wiedzę - odpowiedzi (700)

28. Szczegóły kodu klas (703)

Instrukcja class (703)

Ogólna forma (703)

Przykład (704)

Metody (706)

Przykład metody (707)

Wywoływanie konstruktorów klas nadrzędnych (708)

Inne możliwości wywoływania metod (708)

Dziedziczenie (708)

Tworzenie drzewa atrybutów (709)

Specjalizacja odziedziczonych metod (710)

Techniki interfejsów klas (711)

Abstrakcyjne klasy nadrzędne (712)

Abstrakcyjne klasy nadrzędne z Pythona 2.6 oraz 3.0 (713)

Przestrzenie nazw - cała historia (714)

Pojedyncze nazwy - globalne, o ile nie przypisane (715)

Nazwy atrybutów - przestrzenie nazw obiektów (715)

Zen przestrzeni nazw Pythona - przypisania klasyfikują zmienne (715)

Słowniki przestrzeni nazw (718)

Łączy przestrzeni nazw (720)

Raz jeszcze o łańcuchach znaków dokumentacji (722)

Klasy a moduły (723)

Podsumowanie rozdziału (724)

Sprawdź swoją wiedzę - quiz (724)

Sprawdź swoją wiedzę - odpowiedzi (724)

29. Przeciążanie operatorów (727)

Podstawy (727)

Konstruktory i wyrażenia - `__init__` i `__sub__` (728)

Często spotykane metody przeciążania operatorów (728)

Indeksowanie i wycinanie - `__getitem__` i `__setitem__` (730)

Wycinki (730)

Iteracja po indeksie - `__getitem__` (731)

Obiekty iteratorów - `__iter__` i `__next__` (733)

Iteratory zdefiniowane przez użytkownika (734)

Wiele iteracji po jednym obiekcie (735)

Test przynależności - `__contains__`, `__iter__` i `__getitem__` (737)

Metody `__getattr__` oraz `__setattr__` przechwytyją referencje do atrybutów (740)

Inne narzędzia do zarządzania atrybutami (741)

Emulowanie prywatności w atrybutach instancji (741)

Metody `__repr__` oraz `__str__` zwracają reprezentacje łańcuchów znaków (742)

Metoda `__radd__` obsługuje dodawanie prawostronne i modyfikację w miejscu (745)

Dodawanie w miejscu (746)

Metoda `__call__` przechwytyje wywołania (747)

Interfejsy funkcji i kod oparty na wywołaniach zwrotnych (748)

Porównania - `__lt__`, `__gt__` i inne (750)

Metoda `__cmp__` w 2.6 (usunięta w 3.0) (750)

Testy logiczne - `__bool__` i `__len__` (751)

Destrukcja obiektu - `__del__` (752)

Podsumowanie rozdziału (754)

Sprawdź swoją wiedzę - quiz (755)

Sprawdź swoją wiedzę - odpowiedzi (755)

30. Projektowanie z użyciem klas (757)

Python a programowanie zorientowane obiektowo (757)

Przeciążanie za pomocą sygnatur wywołań (lub bez nich) (758)

Programowanie zorientowane obiektowo i dziedziczenie - związek "jest" (759)

Programowanie zorientowane obiektowo i kompozycja - związki typu "ma" (760)

Raz jeszcze procesor strumienia danych (762)

Programowanie zorientowane obiektowo a delegacja - obiekty "opakowujące" (765)

Pseudoprywatne atrybuty klas (767)

Przegląd zniekształcania nazw zmiennych (767)

Po co używa się atrybutów pseudoprywatnych? (768)

Metody są obiektami - z wiązaniem i bez wiązania (770)

Metody niezwiązane w 3.0 (772)

Metody związane i inne obiekty wywoływane (773)

Dziedziczenie wielokrotne - klasy mieszane (775)

Tworzenie klas mieszanych (776)

Klasy są obiektami - uniwersalne fabryki obiektów (785)

Do czego służą fabryki? (787)

Inne zagadnienia związane z projektowaniem (788)

Podsumowanie rozdziału (788)

Sprawdź swoją wiedzę - quiz (789)

Sprawdź swoją wiedzę - odpowiedzi (789)

31. Zaawansowane zagadnienia związane z klasami (791)

Rozszerzanie typów wbudowanych (791)

Rozszerzanie typów za pomocą osadzania (792)

Rozszerzanie typów za pomocą klas podrzędnych (793)

Klasy w nowym stylu (795)

Nowości w klasach w nowym stylu (796)

Zmiany w modelu typów (797)

Zmiany w dziedziczeniu diamentowym (801)

Nowości w klasach w nowym stylu (805)

Sloty (805)

Właściwości klas (809)

Przeciążanie nazw - `__getattr__` i deskrytory (811)

Metaklasy (811)

Metody statyczne oraz metody klasy (811)

Do czego potrzebujemy metod specjalnych? (812)

Metody statyczne w 2.6 i 3.0 (812)

Alternatywy dla metod statycznych (814)

Używanie metod statycznych i metod klasy (815)

Zliczanie instancji z użyciem metod statycznych (817)

Zliczanie instancji z metodami klasy (818)

Dekoratory i metaklasy - część 1. (820)

Podstawowe informacje o dekoratorach funkcji (820)

Przykład dekoratora (821)

Dekoratory klasy i metaklasy (822)

Dalsza lektura (823)

Pułapki związane z klasami (824)

Modyfikacja atrybutów klasy może mieć efekty uboczne (824)

Modyfikowanie mutowalnych atrybutów klasy również może mieć efekty uboczne (825)

Dziedziczenie wielokrotne - kolejność ma znaczenie (826)

Metody, klasy oraz zakresy zagnieżdżone (827)

Klasy wykorzystujące delegację w 3.0 - `__getattr__` i funkcje wbudowane (829)

Przesadne opakowywanie (829)

Podsumowanie rozdziału (830)

Sprawdź swoją wiedzę - quiz (830)

Sprawdź swoją wiedzę - odpowiedzi (830)

Sprawdź swoją wiedzę - ćwiczenia do części szóstej (831)

Część VII: Wyjątki oraz narzędzia (839)

32. Podstawy wyjątków (841)

Po co używa się wyjątków? (841)

Role wyjątków (842)

Wyjątki w skrócie (843)

Domyślny program obsługi wyjątków (843)

Przechwytywanie wyjątków (844)

Zgłaszanie wyjątków (845)

Wyjątki zdefiniowane przez użytkownika (845)

Działania końcowe (846)

Podsumowanie rozdziału (847)

Sprawdź swoją wiedzę - quiz (849)

Sprawdź swoją wiedzę - odpowiedzi (849)

33. Szczegółowe informacje dotyczące wyjątków (851)

Instrukcja try/except/else (851)

Części instrukcji try (853)

Część try/else (855)

Przykład - zachowanie domyślne (856)

Przykład - przechwytywanie wbudowanych wyjątków (857)

Instrukcja try/finally (857)

Przykład - działania kończące kod z użyciem try/finally (858)

Połączona instrukcja try/except/finally (859)

Składnia połączonej instrukcji try (860)

Łączenie finally oraz except za pomocą zagnieżdżenia (861)

Przykład połączonego try (862)

Instrukcja raise (863)

Przekazywanie wyjątków za pomocą raise (864)

łańcuchy wyjątków w Pythonie 3.0 - raise from (865)

Instrukcja assert (865)

Przykład - wyłapywanie ograniczeń (ale nie błędów!) (866)

Menedżery kontekstu with/as (867)

Podstawowe zastosowanie (867)

Protokół zarządzania kontekstem (868)

Podsumowanie rozdziału (870)

Sprawdź swoją wiedzę - quiz (871)

Sprawdź swoją wiedzę - odpowiedzi (871)

34. Obiekty wyjątków (873)

Wyjątki - powrót do przyszłości (874)

Wyjątki oparte na łańcuchach znaków znikają (874)

Wyjątki oparte na klasach (875)

Tworzenie klas wyjątków (875)

Do czego służą hierarchie wyjątków? (877)

Wbudowane klasy wyjątków (880)

Kategorie wbudowanych wyjątków (881)

Domyślne wyświetlanie oraz stan (882)

Własne sposoby wyświetlania (883)

Własne dane oraz zachowania (884)

Udostępnianie szczegółów wyjątku (884)

Udostępnianie metod wyjątków (885)

Podsumowanie rozdziału (886)

Sprawdź swoją wiedzę - quiz (886)

Sprawdź swoją wiedzę - odpowiedzi (886)

35. Projektowanie z wykorzystaniem wyjątków (889)

Zagnieżdżanie programów obsługi wyjątków (889)

Przykład - zagnieżdżanie przebiegu sterowania (891)

Przykład - zagnieżdżanie składniowe (891)

Zastosowanie wyjątków (893)

Wyjątki nie zawsze są błędami (893)

Funkcje mogą sygnalizować warunki za pomocą raise (893)

Zamykanie plików oraz połączeń z serwerem (894)

Debugowanie z wykorzystaniem zewnętrznych instrukcji try (895)

Testowanie kodu wewnątrz tego samego procesu (895)

Więcej informacji na temat funkcji sys.exc_info (896)

Wskazówki i pułapki dotyczące projektowania wyjątków (897)

Co powinniśmy opakować w try (897)

Jak nie przechwytywać zbyt wiele - unikanie pustych except i wyjątków (898)

Jak nie przechwytywać zbyt mało - korzystanie z kategorii opartych na klasach (900)

Podsumowanie jądra języka Python (901)

Zbiór narzędzi Pythona (901)

Narzędzia programistyczne przeznaczone do większych projektów (902)

Podsumowanie rozdziału (906)

Sprawdź swoją wiedzę - quiz (906)

Sprawdź swoją wiedzę - odpowiedzi (906)

Sprawdź swoją wiedzę - ćwiczenia do części siódmej (907)

Część VIII: Zagadnienia zaawansowane (909)

36. łańcuchy znaków Unicode oraz łańcuchy bajtowe (911)

Zmiany w łańcuchach znaków w Pythonie 3.0 (912)

Podstawy łańcuchów znaków (913)

Kodowanie znaków (913)

Typy łańcuchów znaków Pythona (915)

Pliki binarne i tekstowe (916)

łańcuchy znaków Pythona 3.0 w akcji (918)

Literały i podstawowe właściwości (918)

Konwersje (919)

Kod łańcuchów znaków Unicode (920)

Kod tekstu z zakresu ASCII (921)

Kod tekstu spoza zakresu ASCII (921)

Kodowanie i dekodowanie tekstu spoza zakresu ASCII (922)

Inne techniki kodowania łańcuchów Unicode (923)

Konwersja kodowania (925)

łańcuchy znaków Unicode w Pythonie 2.6 (925)

Deklaracje typu kodowania znaków pliku źródłowego (928)

Wykorzystywanie obiektów bytes z Pythona 3.0 (929)

Wywołania metod (929)

Operacje na sekwencjach (930)

Inne sposoby tworzenia obiektów bytes (931)

Mieszanie typów łańcuchów znaków (931)

Wykorzystywanie obiektów bytearray z Pythona 3.0 (i 2.6) (932)

Wykorzystywanie plików tekstowych i binarnych (935)

Podstawy plików tekstowych (935)

Tryby tekstowy i binarny w Pythonie 3.0 (936)

Brak dopasowania typu i zawartości (938)

Wykorzystywanie plików Unicode (939)

Odczyt i zapis Unicode w Pythonie 3.0 (939)

Obsługa BOM w Pythonie 3.0 (941)

Pliki Unicode w Pythonie 2.6 (943)

Inne zmiany narzędzi łańcuchów znaków w Pythonie 3.0 (944)

Moduł dopasowywania wzorców re (944)

Moduł danych binarnych struct (945)

Moduł serializacji obiektów pickle (947)

Narzędzia do analizy składniowej XML (948)

Podsumowanie rozdziału (951)

Sprawdź swoją wiedzę - quiz (952)

Sprawdź swoją wiedzę - odpowiedzi (952)

37. Zarządzane atrybuty (955)

Po co zarządza się atrybutami? (955)

Wstawianie kodu wykonywanego w momencie dostępu do atrybutów (956)

Właściwości (957)

Podstawy (957)

Pierwszy przykład (958)

Obliczanie atrybutów (959)

Zapisywanie właściwości w kodzie za pomocą dekoratorów (960)

Deskryptory (961)

Podstawy (962)

Pierwszy przykład (964)

Obliczone atrybuty (966)

Wykorzystywanie informacji o stanie w deskryptorach (967)

Powiązania pomiędzy właściwościami a deskryptorami (968)

Metody `__getattr__` oraz `__getattribute__` (970)

Podstawy (971)

Pierwszy przykład (973)

Obliczanie atrybutów (974)

Porównanie metod `__getattr__` oraz `__getattribute__` (975)

Porównanie technik zarządzania atrybutami (976)

Przechwytywanie atrybutów wbudowanych operacji (979)

Powrót do menedżerów opartych na delegacji (983)

Przykład - sprawdzanie poprawności atrybutów (986)

Wykorzystywanie właściwości do sprawdzania poprawności (986)

Wykorzystywanie deskryptorów do sprawdzania poprawności (988)

Wykorzystywanie metody `__getattr__` do sprawdzania poprawności (990)

Wykorzystywanie metody `__getattribute__` do sprawdzania poprawności (991)

Podsumowanie rozdziału (992)

Sprawdź swoją wiedzę - quiz (992)

Sprawdź swoją wiedzę - odpowiedzi (993)

38. Dekoratory (995)

Czym jest dekorator? (995)

Zarządzanie wywołaniami oraz instancjami (996)

Zarządzanie funkcjami oraz klasami (996)

Wykorzystywanie i definiowanie dekoratorów (997)

Do czego służą dekoratory? (997)

Podstawy (998)

Dekoratory funkcji (998)

Dekoratory klas (1002)

Zagnieżdżanie dekoratorów (1004)

Argumenty dekoratorów (1006)

Dekoratory zarządzają także funkcjami oraz klasami (1006)

Kod dekoratorów funkcji (1007)

Śledzenie wywołań (1007)

Możliwości w zakresie zachowania informacji o stanie (1009)

Uwagi na temat klas I - dekorowanie metod klas (1012)

Mierzenie czasu wywołania (1017)

Dodawanie argumentów dekoratora (1019)

Kod dekoratorów klas (1021)

Klasy singletona (1021)

Śledzenie interfejsów obiektów (1023)

Uwagi na temat klas II - zachowanie większej liczby instancji (1027)

Dekoratory a funkcje zarządzające (1028)

Do czego służą dekoratory? (raz jeszcze) (1029)

Bezpośrednie zarządzanie funkcjami oraz klasami (1031)

Przykład - atrybuty "prywatne" i "publiczne" (1033)

Implementacja atrybutów prywatnych (1033)

Szczegóły implementacji I (1035)

Uogólnienie kodu pod kątem deklaracji atrybutów jako publicznych (1036)

Szczegóły implementacji II (1038)

Znane problemy (1039)

W Pythonie nie chodzi o kontrolę (1043)

Przykład: Sprawdzanie poprawności argumentów funkcji (1044)

Cel (1044)

Prosty dekorator sprawdzający przedziały dla argumentów pozycyjnych (1045)

Uogólnienie kodu pod kątem słów kluczowych i wartości domyślnych (1047)

Szczegóły implementacji (1050)

Znane problemy (1052)

Argumenty dekoratora a adnotacje funkcji (1053)

Inne zastosowania - sprawdzanie typów (skoro nalegamy!) (1054)

Podsumowanie rozdziału (1055)

Sprawdź swoją wiedzę - quiz (1056)

Sprawdź swoją wiedzę - odpowiedzi (1056)

39. Metaklasy (1061)

Tworzyć metaklasy czy tego nie robić? (1061)

Zwiększające się poziomy magii (1062)

Wady funkcji pomocniczych (1064)

Metaklasy a dekoratory klas - runda 1. (1066)

Model metaklasy (1068)

Klasy są instancjami obiektu type (1068)

Metaklasy są klasami podrzędnymi klasy type (1070)

Protokół instrukcji class (1071)

Deklarowanie metaklas (1071)

Tworzenie metaklas (1073)

Prosta metaklasa (1073)

Dostosowywanie tworzenia do własnych potrzeb oraz inicjalizacja (1074)

Pozostałe sposoby tworzenia metaklas (1074)

Instancje a dziedziczenie (1077)

Przykład - dodawanie metod do klas (1078)

Ręczne rozszerzanie (1078)

Rozszerzanie oparte na metaklasie (1080)

Metaklasy a dekoratory klas - runda 2. (1081)

Przykład - zastosowanie dekoratorów do metod (1084)

Ręczne śledzenie za pomocą dekoracji (1084)

Śledzenie metaklas oraz dekoratorów (1085)

Zastosowanie dowolnego dekoratora do metod (1086)

Metaklasy a dekoratory klas - runda 3. (1088)

Podsumowanie rozdziału (1091)

Sprawdź swoją wiedzę - quiz (1092)

Sprawdź swoją wiedzę - odpowiedzi (1092)

Dodatki (1093)

Dodatek A: Instalacja i konfiguracja (1095)

Instalowanie interpretera Pythona (1095)

Czy Python jest już zainstalowany? (1095)

Skąd pobrać Pythona (1096)

Instalacja Pythona (1097)

Konfiguracja Pythona (1098)

Zmienne środowiskowe Pythona (1098)

Jak ustawić opcje konfiguracyjne? (1100)

Opcje wiersza poleceń Pythona (1103)

Uzyskanie pomocy (1104)

Dodatek B: Rozwiązania ćwiczeń podsumowujących poszczególne części książki (1105)

Część I Wprowadzenie (1105)

Część II Typy i operacje (1107)

Część III Instrukcja i składnia (1112)

Część IV Funkcje (1114)

Część V Moduły (1121)

Część VI Klasy i programowanie zorientowane obiektowo (1125)

Część VII Wyjątki oraz narzędzia (1132)

Skorowidz (1139)