

Spis treści

O autorach	9
O recenzencie	10
Przedmowa	11
Rozdział 1. Aktualny stan Pythona	17
Gdzie jesteśmy i dokąd zmierzamy?	18
Co zrobić z kodem w Pythonie 2?	18
Jak być na bieżąco?	20
Dokumenty PEP	21
Aktywne społeczności	22
Inne źródła informacji	25
Podsumowanie	26
Rozdział 2. Nowoczesne środowiska programistyczne Pythona	27
Wymagania techniczne	28
Ekosystem pakietów Pythona	29
Instalowanie pakietów Pythona za pomocą narzędzia pip	29
Izolowanie środowiska uruchomieniowego	31
Izolacja na poziomie aplikacji a izolacja na poziomie systemu	33
Izolacja środowiska na poziomie aplikacji	34
Poetry jako system zarządzania zależnościami	37
Izolacja środowiska na poziomie systemu	41
Konteneryzacja a wirtualizacja	43
Zarządzanie środowiskami wirtualnymi z użyciem Dockera	44
Wirtualne środowiska programistyczne oparte na narzędziu Vagrant	61

Popularne narzędzia do zwiększania produktywności	63
Niestandardowe powłoki Pythona	63
Stosowanie powłoki IPython	65
Stosowanie powłok we własnych skryptach i programach	67
Interaktywne debugery	68
Inne narzędzia do zwiększania produktywności	69
Podsumowanie	71
Rozdział 3. Nowości w Pythonie	72
Wymagania techniczne	73
Niedawne dodatki do języka	73
Operatory scalania i aktualizacji słownika	74
Wyrażenia przypisania	78
Wskazówki dotyczące typów w typach generycznych	81
Parametry czysto pozycyjne	83
Moduł zoneinfo	85
Moduł graphlib	86
Nie tak nowe, ale wciąż błyszczące	90
Funkcja breakpoint()	90
Tryb roboczy	92
Funkcje __getattr__() i __dir__() na poziomie modułu	94
Formatowanie łańcuchów znaków za pomocą obiektów f-string	95
Podkreślenia w literałach liczbowych	96
Moduł secrets	96
Co może się pojawić w przyszłości?	98
Tworzenie sumy typów za pomocą operatora	98
Strukturalne dopasowywanie wzorców	99
Podsumowanie	103
Rozdział 4. Porównanie Pythona z innymi językami	104
Wymagania techniczne	105
Model klas i programowanie obiektowe	105
Dostęp do klas bazowych	106
Wielodziedziczenie i porządek MRO	108
Inicjalizowanie instancji klasy	113
Wzorce dostępu do atrybutów	116
Deskryptory	117
Właściwości	123
Dynamiczny polimorfizm	127
Przeciążanie operatorów	129
Przeciążanie funkcji i metod	135
Klasy danych	138
Programowanie funkcyjne	141
Funkcje lambda	142
Funkcje map(), filter() i reduce()	144
Obiekty i funkcje częściowe	146
Generatory	147
Wyrażenia generatora	148
Dekoratory	149

Wyliczenia	151
Podsumowanie	153
Rozdział 5. Interfejs, wzorce i modułowość	154
Wymagania techniczne	155
Interfejsy	155
Odrobina historii: zope.interface	157
Stosowanie adnotacji funkcji i abstrakcyjnych klas bazowych	164
Tworzenie interfejsów z wykorzystaniem adnotacji określających typ	169
Odwroćcie sterowania i wstrzykiwanie zależności	172
Odwroćcie sterowania w aplikacjach	173
Stosowanie platform do wstrzykiwania zależności	181
Podsumowanie	185
Rozdział 6. Współbieżność	186
Wymagania techniczne	186
Czym jest współbieżność?	187
Wielowątkowość	189
Czym jest wielowątkowość?	189
Obsługa wątków w Pythonie	192
Kiedy należy stosować wielowątkowość?	194
Przykładowa aplikacja wielowątkowa	197
Wieloprocusowość	212
Wbudowany moduł multiprocessing	214
Stosowanie puli procesów	217
Stosowanie modułu multiprocessing.dummy jako interfejsu do obsługi wielowątkowości	219
Programowanie asynchroniczne	220
Kooperatywna wielozadaniowość i asynchroniczne operacje wejścia – wyjścia	221
Słowa kluczowe async i await w Pythonie	222
Praktyczny przykład zastosowania programowania asynchronicznego	225
Dostosowywanie nieasynchronicznego kodu do asynchroniczności za pomocą obiektów future	228
Podsumowanie	231
Rozdział 7. Programowanie sterowane zdarzeniami	233
Wymagania techniczne	234
Czym dokładnie jest programowanie sterowane zdarzeniami?	234
Sterowanie zdarzeniami nie jest tożsame z asynchronicznością	235
Programowanie sterowane zdarzeniami w GUI	236
Komunikacja sterowana zdarzeniami	238
Różne style programowania sterowanego zdarzeniami	240
Styl oparty na wywołaniach zwrotnych	241
Styl oparty na obserwowaniu obiektów	242
Styl oparty na tematach	246
Architektury sterowane zdarzeniami	248
Kolejki zdarzeń i komunikatów	249
Podsumowanie	252

Rozdział 8. Elementy metaprogramowania	253
Wymagania techniczne	254
Czym jest metaprogramowanie?	254
Stosowanie dekoratorów do modyfikowania działania funkcji przed jej użyciem	255
Następny krok: dekoratory klas	256
Przechwytywanie procesu tworzenia instancji klasy	260
Metaklasy	263
Ogólna składnia	264
Stosowanie metaklas	267
Pułapki związane z metaklasami	270
Stosowanie metody <code>__init_subclass__()</code> jako alternatywy dla metaklas	271
Generowanie kodu	273
Funkcje <code>exec</code> , <code>eval</code> i <code>compile</code>	273
Drzewa składni abstrakcyjnej	274
Haczyki importu	276
Ważne przykłady generowania kodu w Pythonie	276
Podsumowanie	279
Rozdział 9. Łączenie Pythona z kodem w C i C++	280
Wymagania techniczne	281
C i C++ jako podstawa rozszerzalności w Pythonie	282
Kompilowanie i wczytywanie w Pythonie rozszerzeń napisanych w C	283
Kiedy należy używać rozszerzeń?	285
Zwiększanie wydajności kluczowych fragmentów kodu	285
Integrowanie istniejącego kodu napisanego w różnych językach	286
Integrowanie zewnętrznych bibliotek dynamicznych	287
Tworzenie wydajnych niestandardowych typów danych	287
Pisanie rozszerzeń	288
Rozszerzenia w czystym C	289
Pisanie rozszerzeń za pomocą Cythona	304
Wady korzystania z rozszerzeń	310
Dodatkowa złożoność	310
Trudniejsze debugowanie	311
Komunikacja z bibliotekami dynamicznymi bez używania rozszerzeń	312
Moduł <code>cypes</code>	312
CFFI	318
Podsumowanie	320
Rozdział 10. Automatyzacja testów i kontroli jakości	321
Wymagania techniczne	322
Zasady programowania sterowanego testami	322
Pisanie testów z użyciem platformy <code>pytest</code>	325
Parametryzacja testów	331
Konfiguracje testów w platformie <code>pytest</code>	334
Stosowanie „fałszywych” obiektów	342
Atrapy i moduł <code>unittest.mock</code>	345

Automatyzacja kontroli jakości	349
Pokrycie kodu testami	349
Narzędzia do poprawiania stylu i lintery	353
Statyczna analiza typów	356
Testowanie mutacyjne	358
Przydatne narzędzia związane z testami	363
Generowanie realistycznych danych	363
Generowanie dat i czasu	365
Podsumowanie	366
Rozdział 11. Tworzenie pakietów i udostępnianie kodu w Pythonie	367
Wymagania techniczne	368
Tworzenie pakietów bibliotek i ich udostępnianie	368
Budowa pakietu Pythona	369
Rodzaje dystrybucji pakietów	377
Rejestrowanie i publikowanie pakietów	381
Wersjonowanie pakietów i zarządzanie zależnościami	383
Instalowanie własnych pakietów	387
Pakiety przestrzeni nazw	388
Skrypty i punkty wejścia w pakietach	390
Tworzenie pakietów aplikacji i usług do użytku w internecie	394
Manifest Twelve-Factor App	394
Korzystanie z Dockera	396
Zarządzanie zmiennymi środowiskowymi	398
Rola zmiennych środowiskowych w platformach do tworzenia aplikacji	402
Tworzenie samodzielnych aplikacji wykonywalnych	406
Kiedy samodzielne aplikacje wykonywalne są przydatne?	407
Popularne narzędzia	407
Bezpieczeństwo kodu Pythona w pakietach wykonywalnych	414
Podsumowanie	415
Rozdział 12. Monitorowanie pracy i wydajności aplikacji	417
Wymagania techniczne	418
Rejestrowanie błędów i logów	418
Podstawy rejestrowania logów w Pythonie	419
Zalecane praktyki z obszaru rejestrowania logów	430
Rozproszone rejestrowanie logów	433
Rejestrowanie błędów w celu ich późniejszej analizy	435
Instrumentacja kodu z wykorzystaniem niestandardowych wskaźników	438
Stosowanie aplikacji Prometheus	440
Śledzenie rozproszone aplikacji	448
Śledzenie rozproszone za pomocą Jaegera	451
Podsumowanie	456

Rozdział 13. Optymalizacja kodu	457
Wymagania techniczne	458
Częste przyczyny niskiej wydajności	458
Złożoność kodu	459
Nadmierne wykorzystanie zasobów i ich wyciekanie	462
Nadmierna liczba operacji wejścia – wyjścia i operacji blokujących	463
Profilowanie kodu	464
Profilowanie procesora	465
Profilowanie wykorzystania pamięci	472
Zmniejszanie złożoności przez wybór odpowiednich struktur danych	480
Przeszukiwanie listy	481
Stosowanie zbiorów	482
Stosowanie modułu collections	482
Architektoniczne kompromisy	487
Stosowanie heurystyk i algorytmów aproksymacyjnych	487
Stosowanie kolejek zadań i przetwarzania odroczonego	489
Stosowanie probabilistycznych struktur danych	491
Zapisywanie wyników w pamięci podręcznej	492
Podsumowanie	500
Skorowidz	501