

W PROSTOCIE TKWI SIŁA



wydanie II

Programowanie w Pythonie

dla
bysstrzaków



Utworzenie
i uruchomienie
pierwszej aplikacji

Rozwiązywanie problemów
i usuwanie błędów

Praca z Anacondą
i używanie
funkcji magicznych

Tytuł oryginału: Beginning Programming with Python For Dummies, 2nd Edition

Tłumaczenie: Agnieszka Górczyńska

ISBN: 978-83-283-5905-5

Original English language edition Copyright © 2018 by John Wiley & Sons, Inc., Hoboken, New Jersey

All rights reserved including the right of reproduction in whole or in part in any form.

This translation published by arrangement with John Wiley & Sons, Inc.

Oryginalne angielskie wydanie © 2018 by John Wiley & Sons, Inc., Hoboken, New Jersey

Wszelkie prawa, włączając prawo do reprodukcji całości lub części w jakiegokolwiek formie, zarezerwowane.

Tłumaczenie opublikowane na mocy porozumienia z John Wiley & Sons, Inc.

Translation copyright © 2019 by Helion S.A.

Wiley, the Wiley Publishing logo, For Dummies, Dla Bystrzaków, the Dummies Man logo, Dummies.com, Making Everything Easier and related trade dress are trademarks or registered trademarks of John Wiley and Sons, Inc. and/or its affiliates in the United States and/or other countries. Used under license. Python is a registered trademark of Python Software Foundation Corporation. All other trademarks are the property of their respective owners.

Wiley, the Wiley Publishing logo, For Dummies, Dla Bystrzaków, the Dummies Man logo, Dummies.com, Making Everything Easier i związana z tym szata graficzna są markami handlowymi John Wiley and Sons, Inc. i/lub firm stowarzyszonych w Stanach Zjednoczonych i/lub innych krajach. Wykorzystywane na podstawie licencji. Python jest zastrzeżonym znakiem towarowym firmy Python Software Foundation Corporation. Wszystkie pozostałe znaki handlowe są własnością ich właścicieli.

Autor oraz HELION SA dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz HELION SA nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://dlabystrzakow.pl/user/opinie/prpyb2>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:

<ftp://ftp.helion.pl/przyklady/prpyb2.zip>

Wydawnictwo HELION

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 231 22 19, 32 230 98 63

e-mail: dlabystrzakow@dlabystrzakow.pl

WWW: <http://dlabystrzakow.pl>

Printed in Poland.

- Kup książkę
- Poleć książkę
- Oceń książkę

- Księgarnia internetowa
- Lubię to! » Nasza społeczność

Spis treści

O autorze	15
Podziękowania od autora	17
Wprowadzenie	19
CZĘŚĆ I: ROZPOCZĘCIE PRACY Z PYTHONEM	25
ROZDZIAŁ 1: Komunikowanie się z komputerem	27
Powody, dla których chciałbyś komunikować się z komputerem	28
Aplikacja to forma komunikacji	29
Zastanów się nad procedurami używanymi codziennie	29
Zapisywanie procedury	30
Traktowanie aplikacji jak każdej innej procedury	31
Komputer traktuje zadania dosłownie	31
Definiowanie aplikacji	32
Komputer używa języka specjalnego	32
Pomoc człowiekowi w rozmowie z komputerem	33
Dlaczego Python jest tak świetny?	34
Powody, dla których warto wybrać Pythona	35
Jak możesz skorzystać na stosowaniu Pythona?	36
Organizacje stosujące Pythona	37
Wyszukiwanie użytecznych aplikacji Pythona	37
Porównanie Pythona z innymi językami programowania	39

ROZDZIAŁ 2:	Instalowanie Pythona	41
	Pobieranie niezbędnej wersji Pythona	41
	Instalowanie Pythona	44
	Praca z systemem Windows	45
	Praca z systemem macOS	47
	Praca z systemem Linux	49
	Uzyskanie dostępu do Pythona w systemie	52
	Używanie systemu Windows	52
	Używanie systemu macOS	55
	Używanie systemu Linux	56
	Sprawdzanie poprawności instalacji	56
 ROZDZIAŁ 3:	 Praca z Pythonem	 59
	Praca w powłoce	60
	Uruchamianie Pythona	60
	Wykorzystanie zalet powłoki	61
	Wykorzystanie zmiennych środowiskowych Pythona	63
	Wydawanie polecenia	65
	Wydawanie poleceń komputerowi	65
	Zakończenie wydawania polecenia	66
	Wyświetlenie wyniku	66
	Korzystanie z pomocy	67
	Tryb pomocy	68
	Prośba o pomoc	69
	Wyjście z trybu pomocy	72
	Pomoc bezpośrednia	72
	Zakończenie pracy z powłoką Pythona	74
 ROZDZIAŁ 4:	 Tworzenie pierwszej aplikacji	 77
	Dlaczego środowisko IDE ma duże znaczenie?	78
	Tworzenie kodu lepszej jakości	78
	Debugowanie	79
	Dlaczego notatnik jest użyteczny?	79
	Pobieranie dystrybucji Anaconda	80
	Pobieranie oprogramowania	80
	Instalowanie dystrybucji Anaconda w systemie Linux	81
	Instalowanie dystrybucji Anaconda w systemie macOS	82
	Instalowanie dystrybucji Anaconda w systemie Windows	83

Pobieranie zbiorów danych i przykładowych fragmentów kodu	87
Używanie Jupyter Notebook	87
Definiowanie repozytorium kodu źródłowego	88
Utworzenie aplikacji	93
Poznajemy komórki	93
Dodawanie komórek dokumentujących	95
Inna treść w komórce	97
Znaczenie wcięć w kodzie	97
Dodawanie komentarzy	99
Poznajemy komentarze	99
Używanie komentarzy jako notatek dla siebie	101
Używanie komentarzy do uniemożliwienia uruchomienia kodu	101
Zakończenie pracy z Jupyter Notebook	102

ROZDZIAŁ 5: **Praca z dystrybucją Anaconda105**

Pobieranie kodu źródłowego	106
Praca z punktami kontrolnymi	107
Definiowanie użycia punktów kontrolnych	108
Zapisywanie punktu kontrolnego	109
Przywracanie punktu kontrolnego	109
Operowanie komórkami notatnika	109
Dodawanie komórek różnych typów	109
Dzielenie i łączenie komórek	110
Przenoszenie komórki	110
Uruchamianie komórki	111
Włączanie i wyłączanie danych wyjściowych	112
Zmiana wyglądu Jupyter Notebook	113
Wyszukiwanie poleceń za pomocą paska poleceń	114
Praca z numerami wierszy	115
Używanie funkcji Cell Toolbar	115
Praca z jądrem	117
Uzyskiwanie pomocy	118
Używanie funkcji magicznych	120
Wyświetlanie uruchomionego procesu	121

CZĘŚĆ II: KOMUNIKACJA 125

ROZDZIAŁ 6: **Przechowywanie i modyfikowanie informacji 127**

Przechowywanie informacji	128
Zmienna jako pojemnik	128
Używanie odpowiedniego pojemnika do przechowywania danych	128
Definiowanie podstawowych typów danych w Pythonie	129
Umieszczanie informacji w zmiennej	129
Typy liczbowe	130
Wartość boolowska	134
Ciąg tekstowy	135
Data i godzina	136

ROZDZIAŁ 7: **Zarządzanie informacją 139**

Określanie sposobu postrzegania danych przez Pythona	140
Porównywanie	140
Jak komputer przeprowadza porównania?	141
Praca z operatorami	141
Definiowanie operatorów	142
Kolejność operatorów	149
Tworzenie i używanie funkcji	150
Funkcja jako pakiet kodu	150
Wielokrotne używanie kodu	150
Definiowanie funkcji	151
Uzyskiwanie dostępu do funkcji	153
Przekazywanie informacji do funkcji	153
Zwrot informacji przez funkcję	157
Porównywanie danych wyjściowych funkcji	158
Pobieranie danych wejściowych od użytkownika	159

ROZDZIAŁ 8: **Podejmowanie decyzji 161**

Podejmowanie prostych decyzji za pomocą konstrukcji if	162
Konstrukcja if	162
Używanie konstrukcji if w aplikacji	163
Wybór alternatywy za pomocą konstrukcji if...else	167
Konstrukcja if...else	168
Używanie konstrukcji if...else w aplikacji	168
Używanie konstrukcji if...elif w aplikacji	169
Używanie zagnieżdżonych konstrukcji warunkowych	172
Używanie wielu konstrukcji if lub if...else	172
Łączenie różnych typów konstrukcji warunkowych	174

ROZDZIAŁ 9: Wykonywanie powtarzających się czynności177

Przetwarzanie danych przy użyciu konstrukcji for	178
Polecenie for	179
Tworzenie prostej pętli for	179
Używanie polecenia break w kodzie	180
Używanie polecenia continue w kodzie	182
Używanie polecenia pass w kodzie	183
Używanie polecenia else w kodzie	184
Przetwarzanie danych przy użyciu konstrukcji while	186
Polecenie while	186
Używanie polecenia while w aplikacji	187
Pętle zagnieżdżone	188

ROZDZIAŁ 10: Obsługa błędów191

Dlaczego Python Cię nie rozumie?	192
Źródła błędów	193
Klasyfikacja błędów	194
Rozróżnianie typów błędów	195
Przechwytywanie wyjątków	197
Podstawowa obsługa wyjątków	197
Obsługa wyjątków od bardziej ogólnych do bardziej szczegółowych	208
Zagnieżdżona obsługa błędów	210
Zgłaszanie wyjątków	214
Zgłoszenie wyjątku w sytuacji szczególnej	214
Przekazywanie informacji o błędzie	215
Tworzenie i używanie własnych wyjątków	216
Używanie klauzuli finally	218

CZĘŚĆ III: NAJCZĘŚCIEJ WYKONYWANE ZADANIA221

ROZDZIAŁ 11: Interakcje z pakietami223

Grupowanie kodu	224
Typy pakietów	226
Bufor pakietów	227
Importowanie pakietów	229
Polecenie import	230
Polecenie from...import	232
Wyszukiwanie pakietów na dysku	234

Pobieranie pakietów z innych źródeł	235
Otwieranie powłoki Anacondy	236
Praca z pakietami conda	236
Instalowanie pakietów za pomocą narzędzia pip	241
Wyświetlanie zawartości pakietu	243
Wyświetlanie dokumentacji pakietu	246
Uruchamianie Pydoc	246
Używanie łączy szybkiego dostępu	248
Wpisywanie szukanego wyrażenia	249
Wyświetlanie wyników	250
ROZDZIAŁ 12: Praca z ciągami tekstowymi	253
Warto pamiętać, że ciągi tekstowe są różne	254
Definiowanie znaku przy użyciu liczb	254
Używanie znaków do tworzenia ciągów tekstowych	255
Tworzenie ciągów tekstowych wraz ze znakami specjalnymi	257
Wybór poszczególnych znaków	259
Wycinanie	261
Odszukiwanie wartości w ciągu tekstowym	265
Formatowanie ciągu tekstowego	267
ROZDZIAŁ 13: Zarządzanie listą	271
Organizowanie informacji w aplikacji	272
Porządkowanie danych przy użyciu listy	272
W jaki sposób Python wyświetla listę?	273
Tworzenie listy	274
Dostęp do listy	276
Iteracja przez listę	277
Modyfikowanie listy	278
Przeszukiwanie listy	281
Sortowanie listy	283
Wyświetlanie listy	284
Praca z obiektem Counter	286
ROZDZIAŁ 14: Kolekcje wszystkich typów danych	289
Poznajemy kolekcje	290
Praca z krotką	291
Praca ze słownikiem	294
Tworzenie i używanie słownika	295
Zastępowanie konstrukcji switch słownikiem	298

Tworzenie stosu przy użyciu listy	301
Praca z kolejką	303
Praca z kolejką dwukierunkową	306

ROZDZIAŁ 15: **Tworzenie i używanie klasy**309

Klasa jako metoda pakowania	310
Części klasy	312
Tworzenie definicji klasy	312
Wbudowane atrybuty klasy	313
Praca z metodami	314
Praca z konstruktorami	316
Praca ze zmiennymi	318
Przeciążanie operatorów	322
Tworzenie klasy	324
Definiowanie klasy MyClass	324
Zapisywanie klasy na dysku	325
Używanie klasy w aplikacji	326
Tworzenie nowej klasy poprzez rozszerzenie już istniejącej	327
Tworzenie klasy potomnej	327
Testowanie klasy w aplikacji	329

CZĘŚĆ IV: WYKONYWANIE ZADAŃ ZAAWANSOWANYCH331

ROZDZIAŁ 16: **Przechowywanie danych w pliku**333

W jaki sposób działa trwały magazyn danych?	334
Tworzenie treści dla trwałego magazynu danych	336
Tworzenie pliku	339
Odczytywanie zawartości pliku	343
Uaktualnianie zawartości pliku	345
Usuwanie pliku	349

ROZDZIAŁ 17: **Wysyłanie wiadomości e-mail**351

Co się dzieje, gdy wysyłasz wiadomość e-mail?	352
Wyświetlanie wiadomości e-mail przypomina odczytywanie listu	352
Definiowanie elementów koperty	354
Definiowanie elementów listu	359
Tworzenie wiadomości e-mail	363
Praca z wiadomością w formacie zwykłego tekstu	364
Praca z wiadomością w formacie HTML	365
Wyświetlanie otrzymanej wiadomości e-mail	366

CZĘŚĆ V: DEKALOGI369

ROZDZIAŁ 18: **Dziesięć świetnych zasobów programistycznych371**

Praca z dokumentacją Pythona w internecie	372
Używanie narzędzia LearnPython.org	373
Tworzenie aplikacji internetowych za pomocą Pythona	374
Pobieranie bibliotek dodatkowych	374
Szybsze tworzenie aplikacji za pomocą środowiska IDE	376
Znacznie łatwiejsze sprawdzanie składni	377
Wykorzystanie zalet XML-a	377
Poznanie najczęściej popełnianych błędów w Pythonie przez początkujących programistów	379
Poznanie Unicode	379
Zwiększenie szybkości działania aplikacji	380

ROZDZIAŁ 19: **Dziesięć sposobów na zarabianie pieniędzy za pomocą Pythona383**

Praca w dziale zapewnienia jakości	384
Pracownik działu IT w mniejszej organizacji	385
Tworzenie skryptów Pythona dla aplikacji	386
Administrowanie siecią	387
Nauka programowania	387
Pomaganie ludziom w lokalizacji	388
Eksploracja danych	388
Praca z systemami osadzonymi	389
Wykonywanie zadań naukowych	389
Analiza danych w czasie rzeczywistym	390

ROZDZIAŁ 20: **Dziesięć narzędzi usprawniających pracę z Pythonem391**

Śledzenie błędów za pomocą Roundup Issue Tracker	392
Utworzenie środowiska wirtualnego za pomocą VirtualEnv	393
Instalowanie aplikacji za pomocą PyInstaller	395
Przygotowanie dokumentacji programistycznej za pomocą pdoc	396
Opracowanie kodu aplikacji za pomocą Komodo Edit	396
Debugowanie aplikacji za pomocą pydbgr	398
Środowisko interaktywne dzięki użyciu IPython	399
Testowanie aplikacji Pythona za pomocą PyUnit	399
Uporządkowanie kodu za pomocą Isort	400
Kontrola wersji z użyciem Mercuriala	400

ROZDZIAŁ 21:	Dziesięć bibliotek, które powinieneś znać	403
	Przygotowanie bezpiecznego środowiska za pomocą PyCrypto	404
	Praca z bazą danych za pomocą SQLAlchemy	404
	Oglądanie świata za pomocą Map Google	405
	Dodawanie graficznego interfejsu użytkownika za pomocą TkIntera	406
	Dostarczanie eleganckiej prezentacji danych tabelarycznych za pomocą PrettyTable	406
	Usprawnienie dźwięku w aplikacji za pomocą PyAudio	406
	Przeprowadzanie operacji na grafice za pomocą PyQtGraph	408
	Wyszukiwanie informacji za pomocą IRLib	409
	Tworzenie za pomocą JPype środowiska współdziałającego z Javą	409
	Uzyskanie za pomocą Twisted Matrix dostępu do zasobów sieci lokalnej	410
	Używanie dostępu do zasobów internetu za pomocą bibliotek	411
	Skorowidz	413

- » komunikowanie się z komputerem,
- » tworzenie programów pozwalających na komunikację z komputerem,
- » poznanie programów i sposobów ich tworzenia,
- » poznanie powodów, dla których warto używać Pythona.

Rozdział 1

Komunikowanie się z komputerem

Konwersacja z komputerem może brzmieć jak scenariusz filmu science fiction. W końcu załoga statku Enterprise w filmie *Star Trek* regularnie rozmawiała z komputerami. Tak naprawdę komputer często odzywa się do użytkownika. Jednak wraz z powstaniem technologii Siri opracowanej przez Apple'a (<https://www.apple.com/siri/>), Echo opracowanej przez Amazona (<https://www.amazon.com/dp/BooX4WHP5E/>) i innego oprogramowania interaktywnego wspomniana wcześniej konwersacja z komputerem wcale nie jest taka niemożliwa.



Prośnienie komputera o podanie pewnych informacji to jedno, a dostarczanie poleceń to zupełnie coś innego. W rozdziale tym dowiesz się, dlaczego miałbyś chcieć wydawać polecenia komputerowi i jakie korzyści możesz z tego odnieść. Odkryjesz również potrzebę istnienia specjalnego języka do prowadzenia takiej komunikacji i powód, dla którego będziesz używać Pythona. Jednak głównym celem tego rozdziału jest pokazanie, że programowanie to po prostu rodzaj komunikacji, podobny do innych form komunikacji z komputerem.

Powody, dla których chciałbyś komunikować się z komputerem

Rozmowa z maszyną może na początku wydawać się dziwactwem, ale jest koniecznością, ponieważ komputer nie potrafi (jeszcze) czytać w myślach użytkownika. Nawet jeśli komputer potrafiłby czytać w myślach, to wciąż byłaby pewna forma komunikacji z użytkownikiem. Nic nie może się zdarzyć bez wymiany informacji między maszyną i jej użytkownikiem. Operacje takie jak:

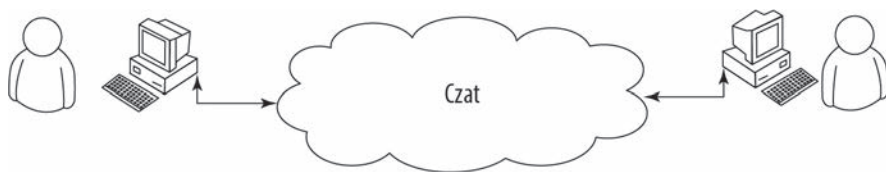
- ▶ odczyt wiadomości e-mail,
- ▶ napisanie tekstu o wakacjach,
- ▶ znalezienie najlepszego prezentu na świecie itd.

są przykładami komunikacji odbywającej się między komputerem i użytkownikiem. Komputer komunikujący się później z innymi maszynami lub użytkownikami, aby wykonać stawiane przed nim zadania, po prostu rozszerza podstawową ideę potrzeby komunikacji do osiągnięciażądanego wyniku.

W większości sytuacji komunikacja zachodzi w sposób praktycznie niewidzialny, o ile tak naprawdę nie zaczniesz się nad nią zastanawiać. Przykładowo, uczestnicząc w czacie internetowym, możesz mieć wrażenie, że rozmawiasz z innym człowiekiem. Jednak tak naprawdę prowadzisz komunikację z komputerem, który z kolei poprzez czat (niezależnie od tego, z czego się ten czat składa) komunikuje się z innym komputerem, a ten z kolei komunikuje się z jego użytkownikiem. Na rysunku 1.1 pokazałem przykład takiej sytuacji.

RYСУNEK 1.1.

Komunikacja z komputerem może być niewidoczna, dopóki nie zaczniesz się nad tym głębiej zastanawiać



Zwróć uwagę na chmurę znajdującą się w środku na rysunku 1.1. Ta chmura może zawierać cokolwiek, ale Ty wiesz, że zawiera przynajmniej inne komputery, w których zostały uruchomione pewne aplikacje. Te komputery pozwalają Tobie i Twoim przyjaciołom komunikować się poprzez czat. Teraz zastanów się nad tym, jak łatwy wydaje się cały ten proces, gdy korzystasz z aplikacji czatu. Nawet pomimo wykonywania wielu zadań w tle wydaje się, że po prostu rozmawiasz z przyjacielem, a proces sam w sobie pozostaje niewidoczny.

Aplikacja to forma komunikacji

Komunikacja z komputerem odbywa się za pomocą aplikacji. Jednej aplikacji używasz do odpowiadania na wiadomości e-mail, kolejnej do zakupu produktów, a jeszcze innej do tworzenia prezentacji. *Aplikacja* (czasami określana mianem *apki*) pozwala człowiekowi wyrażać idee w sposób zrozumiały dla komputera i definiuje narzędzia niezbędne do kształtowania danych, które są używane do komunikowania się w określony sposób. Dane potrzebne do przedstawienia treści w prezentacji są inne niż te używane do zakupu prezentu dla matki. Sposób wyświetlania, używania i interpretowania danych jest inny w poszczególnych zadaniach. Dlatego też konieczne jest stosowanie odmiennych aplikacji do interakcji z danymi w sposób, który będzie zrozumiały zarówno dla Ciebie, jak i komputera.

Do dyspozycji masz aplikacje pozwalające na wykonanie niemal każdego ogólnego zadania, jakie obecnie możesz sobie wyobrazić. W istocie prawdopodobnie masz również dostęp do aplikacji, dla których nie znajdujesz nawet zastosowania. Programiści od lat nieustannie tworzą miliony aplikacji różnego typu, więc wcale niełatwe będzie zrozumienie tego, co można zrobić przez utworzenie nowej metody komunikacji z komputerem za pomocą aplikacji. Odpowiedź sprowadza się do zastanowienia się nad danymi i sposobami, w jakie chcesz ich używać. Część danych po prostu nie jest wystarczająco atrakcyjna, aby przyciągnąć uwagę programistów. Ewentualnie możesz potrzebować danych w formacie, który nie jest jeszcze obsługiwany przez żadną aplikację. Dlatego też nie masz sposobu, by poinformować komputer o takich danych, o ile nie utworzysz własnej aplikacji przeznaczonej do ich obsługi.

W kolejnych punktach przedstawię aplikacje z perspektywy pracy z unikatowymi danymi w sposób, który pod wieloma względami można uznać za specjalny. Przykładowo, możesz mieć dostęp do bazy danych klipów wideo i jednocześnie nie mieć żadnych metod dostępu w sposób, który z Twojego punktu widzenia miałby sens. Dane są unikatowe i chcesz mieć do nich dostęp w specjalny sposób. To może oznaczać konieczność utworzenia aplikacji uwzględniającej Twoje potrzeby i charakterystykę danych.

Zastanów się nad procedurami używanymi codziennie

Procedura to po prostu zestaw kroków wykonywanych, aby zrealizować pewne zadanie. Przykładowo, jeśli chcesz przygotować tosty, procedura może przedstawiać się następująco:

1. Wyciągnięcie pieczywa i masła z lodówki.
2. Otwarcie pieczywa i wyciągnięcie dwóch kromek.
3. Zdjęcie pokrywy z tostera.

4. Umieszczenie kromek w otworach tosteru.
5. Przesunięcie dźwigni tosteru w dół i rozpoczęcie ogrzewania pieczywa.
6. Poczekanie na zakończenie ogrzewania pieczywa.
7. Wyciągnięcie kromek z tosteru.
8. Umieszczenie kromek na talerzu.
9. Posmarowanie kromek masłem.

Wprawdzie procedura stosowana przez Ciebie może być inna od przedstawionej tutaj, ale jest mało prawdopodobne, że smarujesz kromki masłem przed ich włożeniem do tosteru. Oczywiście będziesz musiał wyciągnąć pieczywo z opakowania przed włożeniem kromek do tosteru, ponieważ włożenie zapakowanego pieczywa do tosteru może spowodować niechciany efekt. Większość osób tak naprawdę nie zastanawia się nad procedurą przygotowania tostów, a mimo tego ją stosuje.



Komputer nie potrafi wykonywać zadania bez procedury. Komputerowi trzeba podać kroki konieczne do wykonania, kolejność ich wykonywania, a także wskazać wszelkie wyjątki, które będą oznaczały niepowodzenie zadania. Wszystkie te informacje (a nawet więcej) są zamieszczone w aplikacji. Ujmując rzecz najkrócej, można powiedzieć, że aplikacja to po prostu zapisana procedura, za pomocą której wskazujesz komputerowi, co, kiedy i jak ma zrobić. Skoro z procedur korzystasz przez całe życie, tak naprawdę musisz znaleźć sposób na przekazanie tej wiedzy komputerowi, aby miał jak najwięcej informacji o zadaniu przeznaczonym do wykonania.

Zapisywanie procedury

Gdy uczęszczałem do szkoły podstawowej, nauczyciel poprosił uczniów o napisanie wypracowania na temat przygotowania tostów. Gdy oddaliśmy nasze prace, nauczyciel przyniósł na lekcję toster i pieczywo, a następnie każde wypracowanie zostało odczytane i wypróbowane w praktyce. Wprawdzie żadna z procedur przedstawionych w wypracowaniach nie działała zgodnie z oczekiwaniami, ale za to wszystkie dostarczyły mnóstwa śmiechu. W moim wypadku zapomniałem o konieczności wyciągnięcia pieczywa z opakowania, więc nauczyciel miał trudność z włożeniem pieczywa do tosteru. Ta lekcja na dobre utkwiała mi w pamięci. Zapisywanie procedury może być bardzo trudne, ponieważ mimo tego, że dokładnie wiadomo, co ma zostać zrobione, to jednak bardzo często pewne kroki są pomijane. Przyjmuje się założenie, że druga osoba będzie doskonale wiedziała, co należy zrobić.

Wiele zdarzeń w życiu jest związanych z procedurami. Zastanów się nad listą rzeczy do sprawdzenia przez pilota przed rozpoczęciem lotu. Bez dobrej procedury samolot mógłby ulec katastrofie. Przygotowanie doskonałej procedury

zdecydowanie wymaga czasu, ale na pewno to zadanie jest możliwe do wykonania. Być może trzeba będzie podjąć wiele prób, zanim procedura zadziała zgodnie z oczekiwaniami, choć ostatecznie otrzymasz żądany efekt. Jednak zapis procedury to nie wszystko — powinna zostać przetestowana przez kogoś, kto nie był zaangażowany w jej przygotowanie. Podczas pracy z komputerem jest on doskonałym testerem.

Traktowanie aplikacji jak każdej innej procedury

Komputer można porównać do mojego nauczyciela ze szkoły podstawowej z przykładu, który przedstawiłem w poprzednim podrozdziale. Gdy tworzysz aplikację, zapisujesz procedurę definiującą serię kroków, które komputer będzie musiał wykonać, aby zrealizować stawiane przed nim zadanie. Jeżeli pominiesz choćby jeden krok, efekt nie będzie zgodny z oczekiwaniami. Komputer nie będzie wiedział, co miałeś na myśli lub tego, że chciałeś, aby pewne zadanie zostało wykonane automatycznie. Komputer wie jedynie to, że przygotowałeś pewną procedurę, której musi się ściśle trzymać.

Komputer traktuje zadania dosłownie

Człowiek ostatecznie przyzwyczaja się do przygotowanych procedur. Automatycznie nadrabia ewentualne braki w procedurze lub wskazuje czynności, które zostały pominięte. Innymi słowy, człowiek jest w stanie poradzić sobie z problemami powodowanymi przez procedurę.



Gdy rozpoczynasz tworzenie aplikacji komputerowej, możesz być sfrustrowany tym, że komputer ściśle wykonuje zleczone mu zadania i dosłownie traktuje polecenia otrzymywane od użytkownika. Przykładowo, jeśli wskażesz komputerowi, że pewna wartość powinna wynosić 5, będzie on szukał dokładnie wartości 5. Wprawdzie człowiek może uznać wartość 4,9 za wystarczająco dobrą, ale komputer nie działa w taki sposób. Z perspektywy komputera to jest wartość 4,9 i nie jest ona równa wartości 5. Ujmując rzecz najkrócej — komputer jest nieelastyczny, nie działa w sposób intuicyjny i jest pozbawiony wyobraźni. Gdy stworzysz procedurę dla komputera, będzie on za każdym razem dokładnie wykonywał to, co mu zlecisz — nigdy nie zmodyfikuje procedury ani też nie uzna, że powinien zrobić coś inaczej.

Definiowanie aplikacji

Jak wcześniej wspomniałem, aplikacja pozwala człowiekowi na wyrażenie pomysłu w sposób zrozumiały dla komputera. Aby osiągnąć ten cel, aplikacja opiera się na jednej procedurze lub większej liczbie procedur wskazujących komputerowi sposób wykonania zadań związanych z przetwarzaniem i wyświetlaniem danych. To, co widzisz na ekranie, jest tekstem pochodzącym z procesora tekstu, ale wyświetlenie tych informacji wymaga procedur pozwalających komputerowi na pobranie danych z dysku, ułożenie ich w zrozumiałej postaci, a następnie przedstawienie użytkownikowi. W kolejnych punktach znacznie dokładniej zdefiniuję specyfikę aplikacji.

Komputer używa języka specjalnego

Język, którym posługuje się człowiek, jest skomplikowany i trudny do zrozumienia. Nawet aplikacje, takie jak Siri i Alexa, mają poważne trudności ze zrozumieniem, co tak naprawdę mówi do nich człowiek. Na przestrzeni lat komputery zyskały umiejętność pobierania danych wyjściowych w postaci języka, którym posługuje się człowiek, a następnie uznawania pewnych słów za polecenia. Jednak mimo wszystko komputer nadal nie jest w stanie na żadnym rozsądnym poziomie rozumieć mowy człowieka. Tę trudność można przedstawić na przykładzie pracy prawnika. Gdy czytasz treść dokumentu przygotowanego przez prawnika, możesz mieć wrażenie, że to jakieś bzdury bez większego znaczenia. Jednak celem takiego zapisu jest przedstawienie idei i koncepcji w sposób uniemożliwiający ich interpretację. Prawnikowi rzadko udaje się osiągnąć sukces pod tym względem, ponieważ mowa człowieka nie jest precyzyjna.

Biorąc pod uwagę to, czego się dowiedziałeś we wcześniejszej części rozdziału, możesz uznać, że komputer nigdy nie będzie bazował na mowie człowieka w celu zrozumienia procedur. Komputer zawsze traktuje polecenia dosłownie, więc używanie języka człowieka do utworzenia aplikacji mogłoby zakończyć się otrzymaniem zupełnie nieoczekiwanych wyników. Dlatego też do komunikowania się z komputerami są używane języki specjalne, nazywane *językami programowania*. Dzięki tym językom specjalnym można tworzyć procedury, które są precyzyjne i całkowicie zrozumiałe zarówno dla człowieka, jak i komputera.



Tak naprawdę komputer nie posługuje się żadnym językiem. Wewnętrznie używa kodu binarnego do zmiany położenia przełączników i przeprowadzania obliczeń matematycznych. Komputer nawet nie rozumie liter i zna tylko cyfry. Specjalna aplikacja zmienia zapisaną w języku programowania procedurę na kod binarny zrozumiały przez komputer. Na potrzeby materiału przedstawionego w książce nie musisz się zajmować niskiego poziomu specyfikacją i sposobem działania komputera na poziomie binarnym. Jednak za interesujący można uznać fakt, że komputer przeprowadza operacje na liczbach, a nie korzysta z żadnego języka.

Pomoc człowiekowi w rozmowie z komputerem

Podczas tworzenia aplikacji trzeba koniecznie pamiętać o jej przeznaczeniu. Aplikacja ma pomóc człowiekowi w komunikowaniu się z komputerem w określony sposób. Każda aplikacja wykorzystuje jakieś dane wejściowe, przechowuje je, przetwarza dane, a następnie wyświetla dane wyjściowe, aby człowiek używający tej aplikacji mógł otrzymać żądany wynik. Niezależnie od rodzaju aplikacji, np. gra lub arkusz kalkulacyjny, podstawowa idea pozostaje taka sama. Komputer działa z danymi, które zostały mu dostarczone przez człowieka, aby otrzymać oczekiwany wynik.

Tworząc aplikację, zapewniasz człowiekowi nową metodę komunikowania się z komputerem. To podejście pozwala innym osobom wyświetlać dane na nowe sposoby. Komunikacja między człowiekiem i komputerem powinna być na tyle łatwa, aby aplikacja tak naprawdę zniknęła z pola widzenia. Zastanów się nad aplikacjami, z których korzystałeś w przeszłości. Najlepsza aplikacja to taka, która pozwala skoncentrować się na używanych danych. Przykładowo, gra zostanie uznana za wciągającą, gdy będziesz koncentrować się na planecie, którą masz za zadanie uratować, lub na pilotowanym statku, a nie na aplikacji pozwalającej na wykonywanie tych zadań.



WSKAZÓWKA

Jednym z najlepszych sposobów na to, by zacząć zastanawiać nad utworzeniem aplikacji, jest podpatrywanie, jak to robią inni. Zapisanie tego, co Ci się podoba, a co nie podoba w innych aplikacjach, to doskonały sposób na odkrycie tego, jak powinny wyglądać aplikacje tworzone przez Ciebie. Oto kilka pytań, na które powinieneś sobie odpowiedzieć na początku pracy nad aplikacją:

- » Co takiego rozprasza mnie w aplikacji?
- » Które funkcje są łatwe w użyciu?
- » Które funkcje są trudne w użyciu?
- » Jak ułatwić aplikacji pracę z moimi danymi?
- » Jak przygotować dane, aby praca z nimi stała się łatwiejsza?
- » Co chcę osiągnąć dzięki mojej aplikacji, czego nie mogę zrobić za pomocą istniejących aplikacji?

Podczas tworzenia aplikacji profesjonalny programista zadaje sobie znacznie więcej pytań. Przedstawiony tutaj zestaw jest dobry na początek, ponieważ pomaga traktować aplikację jako sposób komunikacji człowieka z komputerem. Jeżeli kiedykolwiek byłeś sfrustrowany z powodu używanej aplikacji, doskonale wiesz, co będą czuli inni, jeśli nie zadasz sobie odpowiednich pytań podczas powstawania aplikacji. Komunikacja to najważniejszy element każdej tworzonej aplikacji.

Zastanów się także na sposobem, w jaki pracujesz. Rozpocznij tworzenie procedur dla wykonywanych zadań. Dobrze jest przeanalizować proces krok po kroku i zapisać wszystko to, co przyjdzie Ci na myśl w związku z danym krokiem. Po

zakończeniu pracy poproś kogoś innego o wypróbowanie procedury i sprawdzenie, czy faktycznie działa. Możesz być zaskoczony tym, że nawet gdy włoży się dużo wysiłku, łatwo jest zapomnieć o pewnych krokach.



OSTRZEŻENIE

Powstanie złej aplikacji zwykle ma swoje korzenie w tym, że programista nie wie, do czego ma służyć, co w niej jest specjalnego, do kogo jest adresowana, na czym ma polegać jej działanie itd. Jeżeli zdecydujesz się utworzyć aplikację, upewnij się, że dokładnie wiesz, dlaczego to robisz i co chcesz osiągnąć za jej pomocą. Przygotowanie planu naprawdę pomaga i powoduje, że programowanie przynosi radość. Możesz pracować nad nową aplikacją i jednocześnie obserwować, czy stawiane jej cele zostały osiągnięte. W ten sposób przygotujesz aplikację gotową do użycia i pokazania przyjacielom (będą sądzili, że jesteś niezły, skoro umiesz utworzyć aplikację).

Dlaczego Python jest tak świetny?

Obecnie dostępnych jest wiele języków programowania. Tak naprawdę student może poświęcić cały semestr na poznawanie istniejących języków programowania, a mimo tego nie dowie się o istnieniu wszystkich. (Przeżyłem to osobiście podczas studiów). Być może uważasz, że programiści są zadowoleni z dostępności tak wielu języków programowania i po prostu wybierają jeden, za pomocą którego komunikują się z komputerem. Prawda jest jednak taka, że nieustannie powstają nowe języki programowania.



ZAPAMIĘTAJ

Programiści tworzą nowe języki programowania z ważnych powodów. Każdy z takich języków ma coś specjalnego do zaoferowania — zwykle to, co wykonuje wyjątkowo doskonale. Ponadto informatyka nieustannie ewoluuje, więc języki programowania również muszą to robić, aby nadążać za nowymi technologiami. Jako że tworzenie aplikacji wiąże się z efektywną komunikacją, wielu programistów opanowało różne języki programowania, co pozwala im wybrać język właściwy do wykonania danego zadania. Jeden język może sprawdzać się doskonale podczas pobierania informacji z bazy danych, drugi zaś podczas tworzenia elementów interfejsu użytkownika.

Podobnie jak w wypadku innych języków programowania, także Python radzi sobie doskonale z pewnymi zadaniami i dlatego powinieneś je poznać, zanim rozpoczniesz używać tego języka. Będziesz zaskoczony, jak świetne rozwiązania można stworzyć z zastosowaniem Pythona. Poznanie mocnych i słabych stron języka programowania pomaga w jeszcze lepszym jego wykorzystaniu, a także zapobiega frustracji, która mogłaby się pojawić na skutek wyboru języka niedziałającego najlepiej. W kolejnych sekcjach przedstawię informacje, które pomogą Ci przekonać się do Pythona.

Powody, dla których warto wybrać Pythona

Większość języków programowania powstała w konkretnych celach, które pomagają zdefiniować cechy charakterystyczne danego języka i określić jego możliwości. Naprawdę nie można opracować języka programowania przeznaczonego do wszystkiego, ponieważ programiści mają konkretne cele i potrzeby podczas tworzenia aplikacji. Jeżeli chodzi o Pythona, to podstawowym celem było utworzenie języka programowania pozwalającego programistom zachować maksymalną efektywność i produktywność. Mając to na uwadze, można wymienić kilka powodów, dla których warto wybrać Pythona do utworzenia aplikacji.

- » **Krótszy czas potrzebny na opracowanie aplikacji.** Kod w Pythonie jest zwykle od dwóch do dziesięciu razy krótszy niż porównywalny z nim kod utworzony w językach C/C++ i Java. To oznacza mniejszą ilość czasu poświęconą na tworzenie aplikacji i więcej czasu na jej używanie.
- » **Łatwość w odczycie.** Język programowania jest jak każdy inny język — aby zrozumieć jego działanie, trzeba mieć możliwość jego odczytania. Kod Pythona jest zwykle znacznie łatwiejszy w odczycie niż kod utworzony w innych językach. To oznacza mniejszą ilość czasu potrzebną na interpretację kodu i większą na wprowadzanie w nim zmian.
- » **Krótszy czas nauki.** Twórcy Pythona chcieli opracować język charakteryzujący się mniejszą liczbą dziwnych reguł, aby był łatwiejszy w nauce. W końcu programiści chcą tworzyć aplikacje, a nie poznawać niejasne i trudne języki programowania.



WSKAZÓWKA

Wprawdzie Python zalicza się do popularnych języków programowania, ale nie zawsze jest najpopularniejszy (zależy to od witryny internetowej użytej do ich porównania). Obecnie znajduje się na trzeciej pozycji w rankingu przygotowanym przez organizację TIOBE (<https://www.tiobe.com/tiobe-index/>), monitorującą m.in. dane statystyczne związane z używaniem języków programowania. Z kolei według IEEE Spectrum (<https://spectrum.ieee.org/computing/software/the-2017-top-programming-languages>) Python znajduje się na pierwszym miejscu wśród języków programowania. Według Tech Rapidly (<https://techrapidly.com/top-10-best-programming-languages-learn-2019/>) Python również znajduje się na pierwszym miejscu.

Jeżeli szukasz języka programowania, którego znajomość ułatwi Ci znalezienie pracy, to Python jest doskonałym wyborem — choć Java, C/C++ lub C# może być jeszcze lepszym rozwiązaniem, w zależności od rodzaju pracy, którą chcesz zdobyć. Visual Basic to również dobry wybór, nawet jeśli obecnie nie jest tak popularny jak Python. Wybierz język, który lubisz oraz który spełnia potrzeby związane z tworzeniem aplikacji i pozwala zrealizować cele, jakie zamierzasz osiągnąć. Python został okrzyknięty językiem roku w 2007 oraz 2010 roku, a w lutym 2011 roku zajął czwarte miejsce wśród najpopularniejszych języków. Dlatego jeśli szukasz pracy, to jest naprawdę dobry, ale niekoniecznie najlepszy wybór. Możesz być jednak zaskoczony tym, ilu programistów obecnie używa Pythona,

który stał się jednym z najpopularniejszych języków. Więcej informacji na ten temat znajdziesz w opublikowanym przeze mnie poście na blogu, a znajdziesz go pod adresem <http://blog.johnmuellerbooks.com/2014/07/14/python-as-a-learning-tool/>.

Jak możesz skorzystać na stosowaniu Pythona?

Do utworzenie każdej aplikacji możesz wykorzystać dowolny język programowania. Jeżeli wybierzesz niewłaściwy, proces będzie powolny, podatny na błędy i pewnie znienawidzisz go — jednak zadanie będzie mogło zostać wykonane. Oczywiście większość z nas woli unikać bolesnych doświadczeń, warto więc dowiedzieć się, do tworzenia jakich aplikacji Python jest najczęściej wybierany. Oto lista najczęstszych zastosowań Pythona (oczywiście programiści używają go także w innych celach):

- ▶▶ **Tworzenie przykładowych aplikacji.** Programista bardzo często musi utworzyć tzw. *prototyp*, czyli przykładową aplikację, zanim będzie mógł otrzymać zasoby pozwalające na opracowanie rzeczywistej aplikacji. Ponieważ Python stawia na produktywność, można wykorzystać go do szybkiego tworzenia prototypów aplikacji.
- ▶▶ **Aplikacje skryptowe oparte na przeglądarce WWW.** Wprawdzie JavaScript jest prawdopodobnie najpopularniejszym językiem programowania używanym do tworzenia aplikacji działających w przeglądarce WWW, ale Python znajduje się na drugim miejscu w tej kategorii. Python oferuje funkcjonalność niedostępną dla JavaScriptu (więcej informacji na ten temat znajdziesz w poście opublikowany na stronie <https://blog.glyphobet.net/essay/2557/>), a jego wysoka efektywność pozwala na szybsze (co obecnie jest ogromną zaletą) tworzenie aplikacji działających w przeglądarce WWW.
- ▶▶ **Projektowanie aplikacji matematycznych, naukowych i inżynierskich.** Za interesujące należy uznać to, że Python zapewnia dostęp do naprawdę świetnych bibliotek ułatwiających tworzenie aplikacji matematycznych, naukowych i inżynierskich. Dwie najpopularniejsze biblioteki tego typu to NumPy (<http://www.numpy.org/>) i SciPy (<https://www.scipy.org/>). Dzięki nim można znacznie skrócić czas potrzebny na opracowanie specjalizowanego kodu odpowiedzialnego za obsługę zadań najczęściej wykonywanych przez matematyków, naukowców i inżynierów.
- ▶▶ **Praca z XML-em.** Język XML (ang. *eXtensible Markup Language*) jest obecnie podstawą dla większości magazynów danych w internecie i aplikacji. W przeciwieństwie do większości języków programowania, w których XML jest tylko pewnym dodatkiem, w Pythonie jego obsługa jest jedną z najważniejszych cech języka. Jeżeli musisz pracować z usługą sieciową, czyli najważniejszą metodą wymiany informacji w internecie (a także każdej aplikacji intensywnie używającej danych XML), wówczas Python jest doskonałym wyborem.

- ▶▶ **Współpraca z bazami danych.** W biznesie bazy danych są powszechnie wykorzystywane. Python nie jest językiem przeznaczonym do wykonywania zapytań, jak SQL (ang. *Structured Query Language*) lub LINQ (ang. *Language INtegrated Query*), ale doskonale radzi sobie w zakresie obsługi baz danych. Nawiązywanie połączenia z bazą danych i przetwarzanie przechowywanych w niej informacji jest w Pythonie bardzo łatwe.
- ▶▶ **Opracowywanie interfejsu użytkownika.** Python nie jest podobny do języków takich jak C#, oferujących wbudowane narzędzia graficzne pozwalające na tworzenie interfejsu użytkownika przez przeciąganie i upuszczanie kontrolerek. Jednak oferuje dość obszerną liczbę frameworków graficznego interfejsu użytkownika (ang. *Graphical User Interface*, GUI) — czyli rozszerzeń ułatwiających pracę z grafiką (więcej informacji na ten temat znajdziesz na stronie <https://wiki.python.org/moin/GuiProgramming>). Część tych frameworków jest dostarczana wraz z narzędziami graficznymi, które ułatwiają proces przygotowania interfejsu użytkownika. Rzecz w tym, że Python nie musi stosować tylko jednej metody przygotowania interfejsu użytkownika — możesz wybrać tę, która najlepiej pasuje do Twoich potrzeb.

Organizacje stosujące Pythona

Python jest naprawdę dobry w wykonywaniu zadań, do których został przeznaczony. Dlatego też wiele ogromnych organizacji wykorzystuje Pythona przynajmniej do pewnych zadań związanych z tworzeniem aplikacji. Powinieneś wybrać język programowania obsługiwany przez te organizacje, ponieważ wkładają one pieniądze na jego usprawnienie. W tabeli 1.1 wymieniłem organizacje najczęściej wykorzystujące Pythona do różnych zadań.



WSKAZÓWKA

W tabeli wymieniłem zaledwie ułamek organizacji dość intensywnie wykorzystujących Pythona. Znacznie pełniejszą ich listę znajdziesz na stronie <https://www.python.org/about/success/>. Liczba historii przedstawiających zakończone sukcesem wdrożenia Pythona stała się tak duża, że nawet lista na podanej stronie prawdopodobnie nie jest pełna. Osoby odpowiedzialne za jej obsługę powinny utworzyć kategorie, aby lepiej ułożyć elementy tej listy.

Wyszukiwanie użytecznych aplikacji Pythona

W komputerze możesz mieć aplikację utworzoną w Pythonie i nawet o tym nie wiedzieć. Obecnie Python jest wykorzystywany w wielu różnych aplikacjach dostępnych na rynku, począwszy od narzędzi działających w powłoce, aż po w pełni wyposażone pakiety CAD/CAM. Część tych aplikacji działa w urządzeniach mobilnych, inne zaś w ramach ogromnych usług używanych przez korporacje.

Praktycznie wszystko można zrobić za pomocą Pythona i dlatego warto zobaczyć, do jakich celów jest wykorzystywany przez innych. Wprawdzie w internecie znajduje się wiele list aplikacji utworzonych w Pythonie, ale prawdopodobnie najlepsza lista jest dostępna na stronie <https://wiki.python.org/moin/Applications>.

TABELA 1.1. Największe organizacje używające Pythona

Organizacja	Adres URL	Rodzaj aplikacji
Alice Educational Software — Carnegie Mellon University	https://www.alice.org/	aplikacje edukacyjne
Fermilab	https://www.fnal.gov/	aplikacje naukowe
Go.com	http://go.com/	aplikacje oparte na przeglądarce WWW
Google	https://www.google.com/	silnik wyszukiwarki internetowej
Industrial Light & Magic	https://www.ilm.com/	wszystkie potrzeby związane z programowaniem
Lawrence Livermore National Library	https://www.llnl.gov/	aplikacje naukowe
NASA	https://www.nasa.gov/	aplikacje naukowe
Giełda w Nowym Jorku	https://www.nyse.com/index	aplikacje oparte na przeglądarce WWW
Redhat	https://www.redhat.com/en	narzędzia instalacji systemu Linux
Yahoo!	https://www.yahoo.com/	część poczty elektronicznej Yahoo!
YouTube	http://www.youtube.com/	silnik graficzny
Zope — Digital Creations	https://www.zope.org/	aplikacja publikacji danych

Jako programista Pythona na pewno chcesz poznać narzędzia, które mogą ułatwić Ci pracę. Tak zwane *narzędzia programistyczne* oferują pewien poziom automatyzacji podczas tworzenia procedur wskazujących komputerowi zadania przeznaczone do wykonania. Dysponując narzędziami programistycznymi, można zmniejszyć ilość pracy koniecznej do przygotowania działającej aplikacji. Programiści uwielbiają dzielić się listami swoich ulubionych narzędzi programistycznych, a doskonałą listę takich narzędzi, podzieloną na kategorie, znajdziesz na stronie <https://www.python.org/about/apps/>.



W rozdziale wspominałem już o dwóch takich narzędziach — bibliotekach naukowych NumPy i SciPy. W pozostałej części książki poznasz jeszcze wiele innych narzędzi. Przygotuj własną listę ulubionych narzędzi programistycznych Pythona.

Porównanie Pythona z innymi językami programowania

Porównywanie języków programowania jest pod pewnymi względami niebezpieczne, ponieważ wybór języka to przede wszystkim kwestia własnych upodobań, co zostało dowiedzione naukowo. Zanim więc zostaną zaatakowane przez zwolenników innych języków programowania, chciałbym podkreślić, że używam nie tylko Pythona. Nie istnieje coś takiego jak najlepszy język programowania, a co najwyżej — najlepszy język sprawdzający się w konkretnej aplikacji. Mając to na względzie, w kolejnych sekcjach przedstawiłem ogólne porównanie Pythona z innymi językami. (Więcej informacji na ten temat znajdziesz na stronie <https://wiki.python.org/moin/LanguageComparisons>).

C#

Wiele osób uważa, że Microsoft po prostu skopiował Javę, aby utworzyć C#. Dlatego też C# ma takie same zalety (i wady) jak Java. Podstawowym celem utworzenia C# było opracowanie lepszego rodzaju języka C/C++ — przynajmniej łatwiejszego w poznawaniu i użyciu. W tej sekcji chcę się skoncentrować na porównaniu C# i Pythona. Można więc stwierdzić, że w porównaniu z C# Python charakteryzuje się następującymi zaletami:

- » znacznie łatwiejszy do poznania,
- » znacznie zwięźlejszy kod,
- » w pełni dostępny jako *open source*,
- » lepsza obsługa na różnych platformach sprzętowych,
- » łatwy w użyciu w wielu środowiskach programistycznych,
- » łatwy do rozbudowy za pomocą Javy i C/C++,
- » rozbudowana obsługa funkcjonalności wymaganej przez naukowców i inżynierów.

Java

Przez wiele lat programiści czekali na język pozwalający na jednokrotne utworzenie aplikacji, a następnie na uruchamianie jej gdziekolwiek. Java została zaprojektowana do sprawnego działania na dowolnej platformie. Swoje funkcjonowanie opiera na pewnych sztuczkach, które odkryjesz w dalszej części książki. W tym momencie wystarczy wiedzieć, że ten język odniósł tak duży sukces w zakresie uruchamiania aplikacji na różnych platformach, że inne języki muszą emulować Javę (z różnym wynikiem). Pomimo tego Python ma pewne ważne zalety w porównaniu z Javą:

- » znacznie łatwiejszy do poznania,
- » znacznie zwiększony kod,
- » rozbudowane zmienne (rodzaj pojemników w pamięci komputera) przeznaczone do przechowywania różnych danych na podstawie wymagań aplikacji po jej uruchomieniu (dynamiczne stosowanie typów),
- » krótszy czas tworzenia aplikacji.

Perl

Perl początkowo był akronimem od *Practical Extraction and Report Language*. Obecnie ten język programowania jest nazywany Perl i przy tym pozostajmy. Jednak mimo tego Perl nadal ukazuje swoje korzenie pod tym względem, że doskonale radzi sobie z pobieraniem informacji z bazy danych i przedstawianiem ich w postaci raportu. Oczywiście możliwości Perla są znacznie większe i można go wykorzystać do utworzenia praktycznie każdej aplikacji. (Osobiście użyłem Perla do utworzenia usługi sieciowej). W porównaniu z Perlem Python charakteryzuje się następującymi zaletami:

- » znacznie łatwiejszy do poznania,
- » znacznie łatwiejszy w odczycie kodu,
- » rozbudowany mechanizm ochrony danych,
- » lepsza integracja z Javą,
- » mniejsza zależność od poszczególnych platform.

R

Naukowcy często mają trudny orzech do zgryzienia, gdy stają przed wyborem — R czy Python — ponieważ oba wymienione języki programowania zapewniają możliwości w zakresie analizy statystycznej i dostarczają narzędzia graficzne używane przez naukowców do poznawania wzorców w danych. Oba języki są dostępne również jako *open source* i obsługiwane na wielu różnych platformach. Jednak R jest nieco bardziej specjalizowany niż Python i bardziej ukierunkowany na rynek akademicki. W efekcie, w porównaniu z R, Python charakteryzuje się następującymi zaletami:

- » kładzie nacisk na produktywność i czytelność kodu,
- » został zaprojektowany do użycia w przedsiębiorstwach,
- » zapewnia łatwiejsze debugowanie,
- » stosuje spójne techniki tworzenia kodu,
- » jest elastyczniejszy,
- » jest znacznie łatwiejszy do poznania.

PROGRAM PARTNERSKI

— GRUPY HELION —



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA
Helion 

Język Python — idealny na start!

Oferujący potężne możliwości i dynamiczny Python jest używany do tworzenia wielu różnych aplikacji. Został opracowany jako prawdziwie niezależny od platformy — dzięki temu jest doskonałym narzędziem dla początkujących programistów, zwłaszcza tych, którzy chcą szybko poznać nowy język. Zamieszczone w książce polecenia pozwalają w dość krótkim czasie krok po kroku opanować podstawy Pythona.

W książce:

- Pobieranie i instalowanie Pythona
- Używanie powłoki
- Jupyter Notebook i jego zastosowanie
- Używanie różnych typów danych
- Praca z pakietami

John Paul Mueller jest wolnym strzelcem i redaktorem technicznym. Napisał ponad 100 książek i ponad 600 artykułów o różnorodnej tematyce — od sieci po sztuczną inteligencję, od zarządzania bazami danych po inne obszary programowania. Jest konsultantem, przygotowuje różnego rodzaju egzaminy certyfikacyjne. Ma własną witrynę internetową pod adresem: <http://johnmuellerbooks.com>.

dla
bystrzaków

Zamówienia telefoniczne:



0 801 339900



0 601 339900

septem
septem.pl

Sprawdź najnowsze promocje:
• <http://dlabystrzakow.pl/promocje>
Książki najchętniej czytane:
• <http://dlabystrzakow.pl/bestsellery>
Zamów informacje o nowościach:
• <http://dlabystrzakow.pl/novosci>

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: radys@dlabystrzakow.pl
<http://dlabystrzakow.pl>

Helion

Cena 59,00 zł

ISBN 978-83-283-5905-5



9 788328 359055