

C O R Y A L T H O F F

PROGRAMISTA

samouk

Profesjonalny
przewodnik
do samodzielnej
nauki kodowania

Helion 

Tytuł oryginału: The Self-Taught Programmer: The Definitive Guide to Programming Professionally

Tłumaczenie: Piotr Rajca

ISBN: 978-83-283-9427-8

Copyright © 2017 by Cory Althoff

All rights reserved. This book or any portion thereof may not be reproduced or used in any manner whatsoever without the express written permission of the publisher.

Polish edition copyright © 2018, 2022 by Helion S.A.

All rights reserved.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz wydawca 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 wydawca nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Helion S.A.

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 231 22 19, 32 230 98 63

e-mail: helion@helion.pl

WWW: <https://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<https://helion.pl/user/opinie/proprv>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)



Spis treści

Część I. Wprowadzenie do programowania	13
Rozdział 1. Wprowadzenie	15
Struktura książki	16
Zacznijmy od końca	17
Nie jesteś sam	17
Zalety samodzielnej nauki	17
Dlaczego należy programować?	18
Konsekwencja i zainteresowanie	18
Postać książki	19
Technologie stosowane w książce	19
Słownictwo	20
Wyzwanie	20
Rozdział 2. Zaczynamy	21
Czym jest programowanie	21
Czym jest Python	22
Instalowanie Pythona	22
Rozwiązywanie problemów	23
Interaktywna powłoka	23
Zapisywanie programów	24
Uruchamianie programów przykładowych	25
Słownictwo	25
Wyzwanie	26

Rozdział 3.	Wprowadzenie do programowania	27
	Przykłady	28
	Komentarze	28
	Wyświetlanie	29
	Wiersze	30
	Słowa kluczowe	30
	Odstępy	31
	Typy danych	31
	Stałe i zmienne	33
	Składnia	36
	Błędy i wyjątki	36
	Operatory arytmetyczne	37
	Operatory porównania	40
	Operatory logiczne	41
	Instrukcje warunkowe	43
	Instrukcje	47
	Słownictwo	49
	Wyzwania	51
Rozdział 4.	Funkcje	53
	Reprezentacja koncepcji	54
	Funkcje	54
	Definiowanie funkcji	55
	Funkcje wbudowane	57
	Wielokrotne stosowanie funkcji	59
	Parametry wymagane i opcjonalne	60
	Zasięg	61
	Obsługa wyjątków	64
	Łańcuchy dokumentujące	66
	Używanie zmiennych tylko wtedy, gdy to konieczne	67
	Słownictwo	67
	Wyzwania	68
Rozdział 5.	Kontenery	69
	Metody	69
	Listy	70
	Krotki	73
	Słowniki	75
	Kontenery w kontenerach	79
	Słownictwo	81
	Wyzwania	82

Rozdział 6.	Operacje na łańcuchach znaków	83
	Potrójne łańcuchy	84
	Indeksy	84
	Łańcuchy znaków są niezmiennie	85
	Konkatencja	85
	Powielanie łańcuchów znaków	86
	Zmiana wielkości liter	86
	Formatowanie	86
	Dzielenie łańcuchów	87
	Metoda join	88
	Usuwanie odstępów	89
	Zastępowanie	89
	Znajdowanie indeksu	89
	Metoda in	90
	Zabezpieczanie znaków specjalnych	90
	Znak nowego wiersza	91
	Wycinki	91
	Słownictwo	93
	Wyzwania	93
Rozdział 7.	Pętle	95
	Pętle for	95
	Funkcja range	99
	Pętle while	99
	Instrukcja break	100
	Instrukcja continue	101
	Pętle zagnieżdżone	102
	Słownictwo	104
	Wyzwania	104
Rozdział 8.	Moduły	105
	Moduły wbudowane	105
	Importowanie innych modułów	107
	Słownictwo	108
	Wyzwania	108
Rozdział 9.	Pliki	109
	Zapisywanie danych w pliku	109
	Automatyczne zamykanie plików	111
	Odczyt z plików	111
	Pliki CSV	112
	Słownictwo	114
	Wyzwania	114

Rozdział 10.	Łączenie wszystkiego w całość	115
	Wisielec	116
	Wyzwania	119
Rozdział 11.	Praktyka	121
	Do przeczytania	121
	Inne zasoby	121
	Poszukiwanie pomocy	122
Część II. Wprowadzenie do programowania obiektowego		123
Rozdział 12.	Paradygmaty programowania	125
	Stan	125
	Programowanie proceduralne	126
	Paradygmat programowania funkcyjnego	127
	Paradygmat programowania obiektowego	128
	Słownictwo	133
	Wyzwania	134
Rozdział 13.	Cztery filary programowania obiektowego	135
	Hermetyzacja	135
	Abstrahowanie	138
	Polimorfizm	138
	Dziedziczenie	140
	Kompozycja	142
	Słownictwo	143
	Wyzwania	144
Rozdział 14.	Więcej o programowaniu obiektowym	145
	Zmienne klasowe a zmienne instancyjne	145
	Metody magiczne	147
	Is	149
	Słownictwo	150
	Wyzwania	150
Rozdział 15.	Łączenie wszystkiego w całość	151
	Karty	151
	Talia	153
	Klasa gracza	154
	Klasa gry	155
	Wojna	156

Część III. Wprowadzenie do narzędzi programistycznych	159
Rozdział 16. Bash	161
Co dalej	162
Znajdowanie Bash	162
Polecenia	163
Ostatnie polecenia	164
Ścieżki względne i bezwzględne	164
Poruszanie się	165
Flagi	166
Pliki ukryte	167
Potoki	168
Zmienne środowiskowe	168
Użytkownicy	169
Dalsza nauka	170
Słownictwo	170
Wyzwania	171
Rozdział 17. Wyrażenia regularne	173
Konfiguracja	173
Proste dopasowania	175
Dopasowywanie początku i końca	176
Dopasowywanie różnych znaków	177
Dopasowywanie cyfr	178
Powtórzenia	179
Dosłowne traktowanie znaków	181
Narzędzia do tworzenia wyrażeń regularnych	182
Słownictwo	182
Wyzwania	182
Rozdział 18. Menedżery pakietów	183
Pakiety	183
Pip	184
Środowiska wirtualne	186
Słownictwo	186
Wyzwania	186
Rozdział 19. Kontrola wersji	187
Repozytoria	188
Rozpoczynanie projektu	189
Wypychanie i wciąganie zmian	190

Przykład wypychania	191
Przykład wciągania	194
Przywracanie wersji	194
diff	195
Dalsze kroki	197
Słownictwo	197
Wyzwania	198
Rozdział 20. Łączenie wszystkiego w całość	199
Kod HTML	200
Pozyskiwanie informacji z witryny Google Wiadomości	201
Słownictwo	204
Wyzwanie	204
Część IV. Wprowadzenie do informatyki	205
Rozdział 21. Struktury danych	207
Struktury danych	207
Stosy	208
Odwracanie łańcucha znaków przy użyciu stosu	210
Kolejki	211
Kolejka po bilety	212
Słownictwo	214
Wyzwania	214
Rozdział 22. Algorytmy	215
FizzBuzz	215
Wyszukiwanie sekwencyjne	216
Palindrom	217
Anagram	218
Zliczanie wystąpień liter	218
Rekurencja	219
Słownictwo	221
Wyzwania	222

Część V. Zdobywanie pracy	223
Rozdział 23. Najlepsze praktyki programistyczne	225
Pisz kod w ostateczności	226
Zasada DRY	226
Prostopadłość	226
Każdy element danych powinien mieć jedną reprezentację	226
Funkcje powinny robić tylko jedną rzecz	227
Jeśli to trwa zbyt długo, zapewne robimy coś źle	227
Wykonuj operacje w optymalny sposób już od samego początku	227
Zachowaj zgodność z konwencjami	228
Używaj dobrego IDE	228
Rejestracja	229
Testowanie	229
Przeglądanie kodu	230
Bezpieczeństwo	230
Słownictwo	231
Rozdział 24. Pierwsza praca w charakterze programisty	233
Określ ścieżkę	233
Zdobywanie początkowego doświadczenia	234
Przygotowania do rozmowy kwalifikacyjnej	235
Rozmowa kwalifikacyjna	235
Jak radzić sobie na rozmowie	236
Rozdział 25. Praca w zespole	237
Opanowanie podstaw	237
Nie pytaj o to, co możesz znaleźć w internecie	238
Modyfikowanie kodu	238
Syndrom oszusta	238
Rozdział 26. Dalsza lektura	239
Klasyka	239
Kursy internetowe	240
Hacker News	240
Rozdział 27. Dalsze kroki	241
Poszukaj mentora	241
Skacz na głęboką wodę	242
Kolejna rada	242
Podziękowania	245
Skorowidz	247

Wprowadzenie

„Większość dobrych programistów programuje nie dlatego, że oczekują zapłaty lub powszechnego uwielbienia, ale dlatego, że jest to świetna zabawa”.

— Linus Torvalds

Studiowałem nauki polityczne na Uniwersytecie Clemson. Zwróciłem wtedy uwagę na informatykę i nawet zapisałem się na pierwszym roku na kurs „Wprowadzenie do programowania”, ale szybko z niego zrezygnowałem. Był dla mnie zbyt trudny. Kiedy jednak po zakończeniu studiów zamieszkałem w Dolinie Krzemowej, uznałem, że muszę się nauczyć programowania. Po roku pracowałem jako programista drugiego stopnia w eBay (to lepiej niż programista pierwszego stopnia i gorzej niż starszy programista). Nie chciałem jednak sprawiać wrażenia, że to było łatwe. Było to niezwykle wymagające, ale w przerwach pomiędzy rzucaniem czym popadnie po ścianach miałem sporo frajdy.

Swoją przygodę z nauką programowania rozpocząłem od języka Python, popularnego języka programowania. Jednak celem tej książki nie jest jedynie nauczenie czytelników programowania w konkretnym języku, choć — oczywiście — chodzi także to. Jej celem jest przedstawienie wszystkiego, czego nie można znaleźć w standardowych źródłach informacji. Traktuje ona o rzeczach, których sam nauczyłem się po drodze, zostając programistą. Książka nie jest przeznaczona dla kogoś poszukującego zwyczajnego wprowadzenia do zagadnień programowania, by potem pisać kod w ramach hobby. Napisałem ją specjalnie z myślą o osobach, które pragną zajmować się programowaniem zawodowo. Niezależnie od tego, czy celem jest zostanie programistą, przedsiębiorcą, czy też wykorzystanie nowych umiejętności programistycznych w jeszcze innej profesji, tę książkę napisałem właśnie dla Ciebie.

Nauczenie się nowego języka programowania to tylko część całej batalii. Są także inne umiejętności, które trzeba posiadać, by porozumiewać się w języku informatyków. Nauczę Cię wszystkiego, czego sam nauczyłem się na drodze od zupełnego amatora do profesjonalnego programisty. Napisałem tę książkę, by pokazać osobom marzącym o zostaniu programistami zarys tego, co muszą umieć. Jako programista samouk nie wiedziałem, czego mam się uczyć. Książki zawierające wprowadzenie do programowania wszystkie są takie same. Uczą podstaw programowania w językach Python lub Ruby i pozostawiają nas samym sobie. Osoby, które przeczytały takie książki, często pytały: „Co mam teraz zrobić? Jeszcze nie jestem programistą i nie wiem, czego mam się uczyć w następnej kolejności”. Ta książka jest moją odpowiedzią na to pytanie.

Struktura książki

Wielu zagadnieniom poruszonym w pojedynczym rozdziale tej książki można by poświęcić odrębną pozycję — i w wielu przypadkach tak się właśnie dzieje. Moim celem jednak nie jest opisywanie każdego szczegółu każdego zagadnienia, które może Ci się przydać. Moim celem jest przedstawienie mapy — zarysu wszystkich umiejętności, które będziesz musiał zdobyć, by zawodowo zajmować się programowaniem. Książka została podzielona na pięć części.

Część I. — Wprowadzenie do programowania. Swój pierwszy program napiszesz tak szybko, jak to tylko możliwe. Miejmy nadzieję, że będzie to już dziś.

Część II. — Wprowadzenie do programowania obiektowego. W tej części opisuję różne paradygmaty programowania, koncentrując się na programowaniu obiektowym. Napiszesz w tej części grę, która pokaże ogromne możliwości, jakie daje programowanie. Jestem pewny, że po przeczytaniu tej części książki będziesz zafascynowany programowaniem.

Część III. — Wprowadzenie do narzędzi programistycznych. W tej części książki pokażę różne narzędzia programistyczne, które przeniosą wydajność pracy na wyższy poziom. Na tym etapie jesteś już zapewne zafascynowany programowaniem i chcesz być coraz lepszy. W tym miejscu dowiesz się nieco więcej o systemach operacyjnych, o możliwościach stosowania wyrażeń regularnych w celu poprawy wydajności pracy, o instalowaniu programów napisanych przez innych i zarządzaniu nimi oraz o współpracy z innymi programistami przy wykorzystaniu systemów kontroli wersji.

Część IV. — Wprowadzenie do informatyki. Ta część zawiera krótkie wprowadzenie do informatyki i jest poświęcona dwóm podstawowym zagadnieniom, czyli algorytmom i strukturom danych.

Część V. — Zdobywanie pracy. Ostatnia część książki jest poświęcona najlepszym praktykom programistycznym, zdobywaniu pracy w charakterze programisty, pracy w zespole oraz samodoskonaleniu programistów. Zawiera porady, jak przejść techniczną rozmowę kwalifikacyjną na stanowisko programisty i jak pracować w zespole oraz informacje o tym, jak dalej poszerzać swoje umiejętności.

Jeśli nie dysponujesz jeszcze żadnymi umiejętnościami w zakresie programowania, podczas lektury wszystkich części książki jak najwięcej samodzielnie ćwicz programowanie. Nie staraj się przeczytać tej książki zbyt szybko. Warto jej używać jak przewodnika i pomiędzy lekturą kolejnych rozdziałów jak najwięcej ćwiczyć.

Zacznijmy od końca

Sposób, w jaki nauczyłem się programować, jest dokładnie odwrotny od standardowego sposobu nauki informatyki i programowania, a struktura tej książki odpowiada mojemu tokowi nauki. W tradycyjnym sposobie nauki na początku wiele czasu poświęca się na poznanie teorii — w rzeczywistości przeznaczają się na to tak wiele czasu, że wiele osób, które skończyły kursy informatyki, wcale nie potrafi pisać programów. Jeff Atwood na swoim blogu, w poście „Why Can't Programmers... Program?”¹ pisze: „Autor, podobnie jak ja, nie może zrozumieć faktu, że 199 na 200 starających się o posadę programisty nie potrafi pisać kodu. Powtarzam: w ogóle nie potrafi napisać żadnego kodu”. To spostrzeżenie skłoniło Atwooda do stworzenia programistycznego wyzwania o nazwie **FizzBuzz** — programistycznego testu używanego na rozmowach kwalifikacyjnych w celu eliminacji kandydatów. Większość kandydatów oblewa ten test i właśnie z tego powodu podczas lektury tej książki tak wiele czasu poświęcimy na naukę umiejętności, które będą niezbędne w praktyce. Jednak nie masz się czego obawiać: nauczysz się także wszystkiego, czego potrzeba, by zdać test FizzBuzz.

Bohater filmu *Szachowe dzieciństwo*, w swojej książce *Sztuka nauki*, opisuje, jak nauczył się grać w szachy w sposób odwrotny do standardowo przyjętego. Zamiast studiować otwarcia, zaczął od nauki gry końcowej (w której na planszy pozostaje jedynie kilka figur). Taka strategia pozwoliła mu lepiej zrozumieć grę i wygrać wiele zawodów. Na podobnej zasadzie uważam, że lepsze efekty daje nauczenie się najpierw programowania i poznawanie teorii później, kiedy już będziemy chcieli wiedzieć, jak wszystko działa. Właśnie z tego względu teoretyczne zagadnienia informatyki wprowadzam dopiero w czwartej części książki, a w dodatku przedstawiam je w jak najkrótszej postaci. Choć teoria jest ważna, jednak jej znajomość stanie się jeszcze cenniejsza, kiedy zostanie poparta praktycznymi doświadczeniami programistycznymi.

Nie jesteś sam

Nauka programowania poza szkołami czy uczelniami staje się coraz popularniejsza. W 2015 roku serwis Stack Overflow (internetowa społeczność programistów) przeprowadził ankietę, z której wynikało, że ponad 48% respondentów nie ma żadnego wykształcenia związanego z programowaniem lub informatyką².

Zalety samodzielnej nauki

Kiedy zostałem zatrudniony w eBay, znalazłem się w zespole programistów mających tytuły magistrów informatyki zdobyte na uniwersytetach Stanforda, Cal, Duke, były w nim nawet dwie osoby po

¹ Dlaczego programiści nie mogą... programować? — przyp. tłum.

² <http://www.infoworld.com/article/2908474/application-development/stack-overflowsurvey-finds-nearly-half-have-no-degree-in-computer-science.html>

doktoratach. Dla mnie, osoby w wieku 25 lat, było krępujące, że moi 21-letni koledzy wiedzieli o programowaniu i informatyce 10 razy więcej niż ja.

Niezależnie od tego, jak onieśmialające byłoby pracowanie z osobami z tytułami licencjatów, magistrów czy też nawet doktorów informatyki, nigdy nie zapominaj, że będziesz dysponował tym, czym dysponowałem ja — „zaletą samodzielnej nauki”. Nie sięgnąłeś po tę książkę, bo kazał Ci to zrobić nauczyciel czy wykładowca, ale dlatego, że pragniesz się czegoś nauczyć, a pragnienie poznania czegoś jest największą zaletą i przewagą, jaką można dysponować. Poza tym nie wolno zapominać, że wiele z osób, które w świecie komputerów odniosło największe sukcesy, było właśnie samoukami. Steve Wozniak, współzałożyciel firmy Apple, jest programistą samoukiem. Podobnie jak Margaret Hamilton, która otrzymała Prezydencki Medal Wolności za swoją pracę w NASA nad misją Apollo, David Karp — założyciel Tumblr, Jack Dorsey — założyciel Twittera oraz Kevin Systrom — założyciel Instagrama.

Dlaczego należy programować?

Programowanie może pomóc w karierze niezależnie do profesji. Uczenie się programowania wzmacnia. Uwielbiam wymyślanie nowych pomysłów i zawsze chodzi mi po głowie jakiś nowy projekt, który chciałbym rozpocząć. Kiedy nauczyłem się programowania, mogłem sięgnąć i realizować swoje pomysły bez poszukiwania kogoś, kto mógłby to zrobić dla mnie.

Programowanie sprawi, że można być lepszym we wszystkim, co się robi. Nie ma wielu zagadnień, które nie skorzystałyby na umiejętności dokładnego przeanalizowania i określenia rozwiązania problemu. Ostatnio musiałem wykonać bardzo żmudne zadanie polegające na poszukiwaniu lokum na witrynie Craigslist. Potrafiłem jednak napisać program, który robił to za mnie i wysłał mi wyniki e-mailem. Nauka programowania może więc uwolnić od konieczności wykonywania powtarzających się czynności.

Jeśli ktoś chce stać się programistą, warto wiedzieć, że zapotrzebowanie na nich jest coraz większe i nie ma dostatecznie wielu odpowiednio wykwalifikowanych kandydatów, by zaspokoić potrzeby. Szacuje się, że do 2020 roku ponad milion wakatów na programistów pozostanie nieobsadzonych³. Jeśli nawet zostanie zawodowym programistą nie jest Twoim celem, umiejętność programowania jest także ceniona w takich branżach jak nauka oraz finanse.

Konsekwencja i zainteresowanie

By poprawiać swoje umiejętności programistyczne, należy programować codziennie. Jedyną rzeczą, jaka ewentualnie mogłaby powstrzymać Cię przed codziennym ćwiczeniem, jest brak zainteresowania programowaniem. Poznaj zatem dwie metody, by do tego nie dopuścić.

Jeśli nie masz żadnego doświadczenia programistycznego, a rozpoczęcie tej podróży napawa Cię obawą, jestem pewny, że i tak sobie poradzisz. Istnieje wiele nieprawdziwych opinii dotyczących

³ <http://www.wsj.com/articles/computer-programming-is-a-trade-lets-act-like-it-1407109947?mod=e2fb>

programowania, takich jak ta, że wymaga ono doskonałej znajomości matematyki. Otóż nie wymaga. Wcale nie trzeba być świetnym z matematyki, by się uczyć programowania, choć jej znajomość może pomóc. A zatem bardzo wiele zagadnień opisywanych w tej książce będzie łatwiejszych do opanowania, niż można przypuszczać.

Kiedy zaczynałem, używałem listy zadań do wykonania, która zapewniała, że programowałem każdego dnia i zachowywałem koncentrację.

Jeśli potrzebujesz dodatkowej pomocy, Tim Ferriss, specjalista do spraw efektywności, zaleca stosowanie następującej techniki pozwalającej na zachowanie wysokiego poziomu motywacji: daj przyjacielowi lub jakiemuś członkowi rodziny pewną sumę pieniędzy wraz z instrukcją, by Ci ją oddał, kiedy zrealizujesz cele w zadanych ramach czasowych, bądź też, by przekazał jakiejś nielubianej organizacji, jeśli Ci się to nie uda.

Postać książki

Każdy kolejny rozdział tej książki bazuje na poprzednich. Starałem się unikać w niej wielokrotnego wyjaśniania tych samych pojęć, więc warto starać się je zapamiętywać. Ważne terminy w miejscu ich pierwszego wystąpienia są zapisywane pogrubioną czcionką. Na końcu każdego rozdziału znajduje się słowniczek, w którym wszystkie te ważne terminy są definiowane. Na końcu każdego rozdziału znajdują się także zadania, których celem jest poprawienie umiejętności programistycznych, tam także znajdują się odnośniki do rozwiązań tych zadań.

Technologie stosowane w książce

Książka uczy pewnych technologii, by zapewnić Ci możliwość zdobycia jak największej liczby praktycznych umiejętności programistycznych. Staram się jednak zachować neutralność w doborze technologii, koncentrując się nie na nich, lecz raczej na ogólnych koncepcjach.

W niektórych przypadkach musiałem jednak wybrać jedną spośród wielu dostępnych technologii. W rozdziale 19, „Kontrola wersji” (jeśli nie wiesz, czym ona jest, wyjaśnię o co chodzi nieco później), przedstawiam podstawowe informacje o stosowaniu systemu Git — popularnego systemu kontroli wersji. Wybrałem Git, gdyż uważam że stanowi przemysłowy standard kontroli wersji. W większości przykładów programistycznych używam języka Python, gdyż jest bardzo popularnym językiem do nauki programowania, a jego kod jest łatwy do czytania i analizy nawet dla osób, które nigdy wcześniej go nie używały. Oprócz tego osoby znające Pythona są bardzo poszukiwane niemal na każdym rynku pracy.

Oczywiście, do wykonywania przykładów prezentowanych w tej książce konieczny będzie komputer. Każdy komputer ma **system operacyjny** — program pełniący rolę pośrednika pomiędzy komponentami sprzętowymi komputera a człowiekiem. To, co widzimy, patrząc na ekran komputera, jest nazywane **graficznym interfejsem użytkownika** (w skrócie GUI od angielskich słów: *graphical user interface*) i stanowi jeden z elementów systemu operacyjnego.

Istnieją trzy najpopularniejsze systemy operacyjne używane na komputerach biurkowych i laptopach; są to **Windows**, **Unix** oraz **Linux**. Windows jest systemem stworzonym przez firmę Microsoft. Unix z kolei powstał już w latach 70. ubiegłego wieku. Obecny system operacyjny używany przez firmę Apple bazuje właśnie na Uniksie. Od tej pory będę używał nazwy Unix także wtedy, gdy będę miał na myśli system operacyjny stosowany na komputerach Apple. Z kolei Linux — system rozpowszechniany jako **oprogramowanie otwarte** (typu *open source*) — jest używany na ogromnej większości istniejących obecnie **serwerów**. Serwer to komputer lub program komputerowy wykonujący jakieś zadanie, na przykład udostępniający witrynę WWW. Oprogramowanie otwarte (nazywane także oprogramowaniem typu *open source*) to program, który nie jest własnością żadnej firmy ani osoby i który może być rozpowszechniany i modyfikowany bez żadnych ograniczeń. Linux oraz Unix są **systemami uniksowymi**, co oznacza, że są do siebie bardzo podobne. W tej książce zakładam, że używasz komputera działającego pod kontrolą systemów Windows, Unix lub Ubuntu (popularnej wersji systemu operacyjnego Linux).

Słownictwo

FizzBuzz. Test programistyczny stosowany do odsiewania kandydatów.

Graficzny interfejs użytkownika (GUI). Ta część systemu operacyjnego, którą użytkownik może oglądać na ekranie komputera.

Linux. System operacyjny typu *open source* używany na większości serwerów na świecie.

Oprogramowanie typu *open source*. Oprogramowanie, które nie jest własnością żadnej firmy ani konkretnej osoby; nad jego rozwojem pracuje grupa ochotników.

Serwer. Komputer lub program komputerowy wykonujący konkretne zadanie, takie jak udostępnianie witryny WWW.

System operacyjny. Program pełniący rolę pośrednika pomiędzy fizycznymi komponentami komputera a człowiekiem.

Uniksowy system operacyjny. Unix lub Linux.

Unix. System operacyjny stworzony w latach 70. ubiegłego wieku. Bazuje na nim obecny system używany na komputerach firmy Apple.

Windows. System operacyjny firmy Microsoft.

Wyzwanie

1. Przygotuj codzienną listę rzeczy do wykonania, obejmującą ćwiczenie programowania.

Rozwiązania można znaleźć w przykładach dołączonych do książki, które możesz pobrać z serwera FTP wydawnictwa Helion <ftp://ftp.helion.pl/przyklady/proprs.zip>.

Skorowidz

A

abstrahowanie, 138, 143
administrator, 170
algorytmy, 215

- iteracyjne, 221
- rekurencyjne, 219, 221
- wyszukiwanie, 216, 221

anagram, 218, 221

B

Bash, 161, 170
baza kodu, 197
bezpieczeństwo, 230
błąd, 36

- składniowy, 50

D

debugger, 231
definiowanie funkcji, 55
dekrementacja, 34, 50
dodawanie do poczekalni, 197
dopasowywanie

- cyfr, 178
- początku i końca, 176
- różnych znaków, 177

DRY, Don't Repeat Yourself, 226, 231
drzewo, 164
dziedziczenie, 140, 143

E

efekt uboczny, 133
epoka, 214

F

FIFO, First In First Out, 211
FizzBuzz, 20, 215
flagi, 166
framework internetowy, 186
funkcja range, 99
funkcje, 53, 67, 227

- definiowanie, 55
- parametry opcjonalne, 60
- parametry wymagane, 60
- wbudowane, 57, 68
- wielokrotne stosowanie, 59
- wywoływanie, 67

G

Google Wiadomości, 201
gra

- w wojnę, 151, 156
- Wisielec, 116

graficzny interfejs użytkownika, GUI, 20

H

Hacker News, 240
 hermetyzacja, 135, 143
 HTML, 200, 204

I

IDE, 228
 IDLE, 23, 228
 importowanie modułów, 107
 indeks, 84

- końcowy, 93
- początkowy, 93
- ujemny, 93

 inkrementacja, 34, 50
 instalowanie, 22
 instancja klasy, 129, 133
 instrukcja, 47, 50

- break, 100, 104
- continue, 101, 104
- elif, 45, 50
- else, 44
- if, 44, 50
- if-else, 50
- with, 114

 instrukcje

- proste, 50
- warunkowe, 43, 50
- złożone, 50

 interaktywna powłoka, 23
 interfejs wiersza poleceń, 161, 170
 iterowanie, 104

J

jajko wielkanocne, 182
 język Python, 22
 języki programowania

- niskiego poziomu, 25
- wysokiego poziomu, 25

K

karty do gry, 151
 katalog roboczy, 170
 klasy, 133

- bazowe, 143
- pochodne, 143

 klauzula, 47, 50
 klient, 143
 klucz, 81
 kod, 25

- HTML, 200
- produkcyjny, 225, 231

 kolejka, queue, 211
 kolejność operacji, 50
 komentarze, 28, 49
 kompozycja, 142, 143
 konkatencja, 85
 kontener, 69

- krotka, 73
- lista, 70
- modyfikowalny, 81
- niezmienny, 81
- słownik, 75
- w kontenerze, 79

 kontrola wersji, 187
 konwencja, 67
 krotki, 73
 kursy internetowe, 240

L

liczba

- całkowita, 49
- zmiennoprzecinkowa, 49

 LIFO, Last In First Out, 208
 listy, 70, 81

Ł

łańcuch dokumentujący, 66, 68
łańcuchy znaków, 49
 dodawanie znaków, 88
 dzielenie, 87
 formatowanie, 86
 indeksy, 84
 konkatencja, 85
 odwracanie, 210
 potrójne, 84
 powielanie, 86
 usuwanie odstępów, 89
 wyszukiwanie, 90
 zabezpieczanie, 90
 zastępowanie, 89
 zmiana wielkości liter, 86
 znajdowanie indeksu, 89

M

menedżer pakietów, 183, 186
 apt-get, 186
 pip, 186
mentor, 241
metadane, 186
metoda, 69, 81, 129, 133
 in, 90
 index, 89
 join, 88
 replace, 89
 split, 87
 strip, 89
 upper, 86
metody
 magiczne, 130, 133, 147
 prywatne, 137, 150
moduł wbudowany, 105, 108
modyfikowanie kodu, 238

N

nagłówek, 51
najlepsze praktyki programistyczne, 225
narzędzia programistyczne, 159
numer rewizji, 197

O

obiekty, 31, 49
 iterowalne, 81
 pliku, 114
 typu bool, 49
 typu NoneType, 49
obsługa wyjątków, 64, 68
odczyt, 114
odkładanie, 214
odstęp, 31
operand, 50
operatory
 przypisania, 33
 arytmetyczne, 37, 50
 logiczne, 41, 50
 porównania, 40, 50
oprogramowanie typu open source, 20

P

pakiety, 183, 186
palindrom, 217, 221
para klucz-wartość, 75, 82
paradygmat programowania, 125, 133
 funkcyjnego, 127
 obiektowego, 128
parametry, 54
 opcjonalne, 60, 68
 wymagane, 60, 68
parsowanie, 204
pętla, 104
 for, 95, 104
 while, 99, 104

pętle

- nieskończone, 104
- wewnętrzne, 104
- zagnieżdżone, 102
- zewnętrzne, 104

pliki

- automatyczne zamykanie, 111
- CSV, 112, 114
- odczytywanie danych, 111
- ukryte, 167
- zapisywanie danych, 109

polecenie, 163

- git diff, 195
- grep, 175
- pip, 184

polimorfizm, 138, 143

pomoc, 122

potoki, 168, 170

powierzchnia ataku, 232

powtórzenia, 179

praca

- programisty, 233
- w zespole, 237

produkcja, 231

program

- IDLE, 23
- pip, 184

programista, 233

programowanie, 21, 25

- funkcyjne, 127, 133
- obiektywne, 123, 128, 133, 135
 - abstrahowanie, 138
 - dziedziczenie, 140
 - hermetyzacja, 135
 - kompozycja, 142
 - polimorfizm, 138
- proceduralne, 126, 133

prostokątność, 226, 231

przeglądanie kodu, 230, 232

przesłanianie metod, 141, 143

przypadek bazowy, 221

przywracanie wersji, 194

pseudokod, 50

Python, 22

R

rejestracja, 229, 231

rekurencja, 219, 221

repozytorium, 188, 197

centralne, 197

lokalne, 197

rewizja, 192, 197

rozmowa kwalifikacyjna, 235

rozpoczynanie projektu, 189

S

separator, 114

serwer, 20

składnia, 36, 50

słowniki, 75, 81

słowo kluczowe, 30, 49

is, 149

stałe, 33, 49

stan, 133

globalny, 125, 133

stos, stack, 208

struktura

danych, 197, 207

kolejka, 211

stos, 208

katalogów, 164

sterująca, 50

syndrom oszusta, 238

system

kontroli wersji

Git, 187

SVN, 187

operacyjny, 20

Ś

ścieżki

pliku, 114

bezwzględne, 164, 170

względne, 164, 170

środowisko wirtualne, 186

T

talia kart, 153
terminal, 163
test FizzBuzz, 215
testowanie, 229, 231
tworzenie
 instancji klasy, 130, 133
 wycinka, 91, 93
typ danych, 31, 49
 Bool, 49
 Float, 49
 Int, 49
 NoneType, 49
 Str, 49
typy danych
 operator przypisania, 50

U

uprawnienia, 170
uruchamianie programów, 25
użytkownicy, 169
używanie zmiennych, 67

W

wartość, 82
wciąganie, 190, 194, 197
Web scraper, 204
wiersz poleceń, 161, 170
wiersze, 30
wycinki, 91
wyjątki, 36, 50, 64
wypychanie, 190, 197
wyrażenia regularne, 173
 dopasowywanie cyfr, 178
 dopasowywanie początku i końca, 176
 dopasowywanie różnych znaków, 177
 narzędzia, 182
 niezachłanne, 182
 powtórzenia, 179

 proste dopasowania, 175
 zachłanne, 182
 znaki zwyczajne, 181
wyrażenie, 50
wyszukiwanie sekwencyjne, 216, 221
wyświetlanie, 29
wywoływanie, 67

Z

zabezpieczanie, 93
zależności, 186
zapis, 114
zapisywanie programów, 24
zasada DRY, 226
zasięg, 68
 globalny, 61, 68
 lokalny, 61, 68
zatwierdzanie, 197
 plików, 192
zdejnowanie, 214
zdobywanie doświadczenia, 234
zestaw, 51
zintegrowane środowisko programistyczne,
 IDE, 228
zliczanie wystąpień liter, 218
zmiana wielkości liter, 86
zmiennie, 33, 50
 globalne, 61, 68
 indeksowe, 104
 instancyjne, 129, 133, 145, 150
 klasowe, 145, 150
 prywatne, 137, 150
 publiczne, 137, 150
 środowiskowe, 168, 170
znacznik HTML, 204
znajdowanie indeksu, 89
znak, 32, 49
 nowego wiersza, 91
znaki specjalne, 90

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 

JEŚLI CHCESZ NAUCZYĆ SIĘ PROGRAMOWANIA, ZACZNIJ OD RAZU!

Nie wystarczy znajomość jednego języka programowania, aby zostać programistą. W rzeczywistości trzeba opanować dość szeroki zakres pojęć i paradygmatów, a także zrozumieć zagadnienia związane z algorytmami. Trzeba być na bieżąco z nowymi technologiami i narzędziami. Należy również poznać i zacząć stosować dobre praktyki programistyczne oraz przyswoić sobie zasady pracy w zespole. Przede wszystkim jednak priorytetem jest sama praktyka, ponieważ wielu programistów wciąż ma problem z pisaniem poprawnego kodu.

Jeśli chcesz być profesjonalistą i postanowiłeś nauczyć się wszystkiego, co jest do tego potrzebne, to wzięłeś do ręki właściwą książkę. Zawiera ona znacznie więcej informacji niż proste wprowadzenie do konkretnego języka programowania. Opisano tu najpotrzebniejsze technologie, elementy kodu i zasady ich stosowania, sporo miejsca poświęcono obiektowości. Zapoznasz się również z najważniejszymi narzędziami programistycznymi i nauczysz się dobrych praktyk programistycznych. To wszystko czyni z tej książki świetną mapę umiejętności, dzięki którym szybko i sprawnie staniesz się prawdziwym profesjonalistą!

W tej książce między innymi:

- wprowadzenie do **programowania struktury** danych i algorytmy
- **pakiety i kontrola** wersji
- **testowanie i bezpieczeństwo** tworzonych aplikacji
- **zasady pracy** zespołu programistów

C O R Y A L T H O F F

jest programistą samoukiem i autorem książek. Pracował dla eBaya i kilku innych firm w Dolinie Krzemowej. Biegłe posługuje się Pythonem, Javą, JavaScriptem i kilkoma innymi językami programowania.

	
	helion.pl
	HELION SA ul. Kościuszki 1c 44-100 Gliwice tel.: 32 230 98 63 helion@helion.pl

ISBN 978-83-283-9427-8

9 788328 394278
Cena: 59,00 zł