

Mark Lassoﬀ



Programowanie dla początkujących

Poznaj świat programowania!

Tytuł oryginału: Programming for Absolute Beginners

Tłumaczenie: Piotr Rajca

Projekt okładki: Studio Gravite / Olsztyn; Obarek, Pokoński, Pazdrijowski, Zaprucki

Materiały graficzne na okładce zostały wykorzystane za zgodą Shutterstock Images LLC.

ISBN: 978-83-283-1908-0

© 2014 by LearnToProgram.tv, Incorporated.

All rights reserved. No part of this document may be reproduced or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written permission of LearnToProgram.tv, Incorporated.

Polish edition copyright © 2016 by Helion SA.

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 Wydawnictwo HELION 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 Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Wydawnictwo HELION

ul. Kościuszki 1c, 44-100 GLIWICE

tel. 32 231 22 19, 32 230 98 63

e-mail: helion@helion.pl

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

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/propoc>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzje.

Printed in Poland.

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

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

SPIS TREŚCI

| | |
|---|-----------|
| <i>O autorze</i> | 7 |
| <i>O autorze kursu</i> | 7 |
| Rozdział 1. Pisanie pierwszego programu | 9 |
| <i>Czego nauczymy się z tej książki?</i> | 10 |
| <i>Przygotowywanie środowiska do programowania</i> | 11 |
| <i>Pisanie i uruchamianie pierwszego programu</i> | 15 |
| <i>Wyjaśnienie polecenia print</i> | 20 |
| Rozdział 2. Wejście i wyjście | 23 |
| <i>Instrukcje input</i> | 23 |
| <i>Sparametryzowane dane wejściowe</i> | 26 |
| <i>Zapisywanie danych w plikach</i> | 28 |
| Rozdział 3. Zmienne | 31 |
| <i>Deklarowanie i inicjalizacja zmiennych</i> | 31 |
| <i>Operacje arytmetyczne na zmiennych</i> | 34 |
| <i>Stosowanie łańcuchów znaków</i> | 36 |
| Rozdział 4. Instrukcje warunkowe i pętle | 39 |
| <i>Proste instrukcje warunkowe</i> | 39 |
| <i>Co jeszcze?</i> | 41 |
| <i>Złożone wyrażenia warunkowe</i> | 42 |
| <i>Złożone instrukcje warunkowe — instrukcja elif</i> | 44 |
| <i>Pętle while</i> | 45 |
| <i>Nie próbujcie tego w domu!</i> | 47 |
| Rozdział 5. Operacje na danych | 49 |
| <i>Listy</i> | 49 |
| <i>Edycja elementów list</i> | 52 |
| <i>Stosowanie funkcji list</i> | 53 |
| <i>Krotki</i> | 54 |
| <i>Słowniki</i> | 55 |

| | |
|--|-----------|
| Rozdział 6. Podsumowanie wiedzy | 59 |
| <i>Klasy a obiekty</i> | 59 |
| <i>Tworzenie klas w języku Python</i> | 61 |
| <i>Dodawanie metod do klasy</i> | 63 |
| <i>Tworzenie wielu instancji</i> | 64 |

ROZDZIAŁ 5.

OPERACJE NA DANYCH

W poprzednim rozdziale poznałeś struktury sterujące, instrukcje warunkowe oraz pętle — konstrukcje, które mają kluczowe znaczenie we wszystkich językach programowania, pozwalają one bowiem programom podejmować decyzje na podstawie kryteriów określanych przez programistów. Z kolei w tym rozdziale skoncentrujemy się na danych i sposobach ich przechowywania. Przedstawię w nim trzy struktury danych: listy, krotki oraz słowniki. Dotychczas nie poświęcaliśmy zbyt wiele uwagi zagadnieniom związanym z danymi, ale najłatwiej można sobie je wyobrazić jako zorganizowane informacje.

Różne typy informacji są organizowane na różne sposoby. Na przykład, jeśli dane dotyczą planowanych zakupów, to optymalnym sposobem ich organizacji będzie lista. Jeżeli jednak konieczne będzie zapisanie ocen uczniów z różnych przedmiotów w pewnym okresie czasu, to lista już w ogóle się do tego nie nada. Istnieje wiele różnych struktur danych, których używają programiści, a każda z nich jest zoptymalizowana pod kątem innego typu danych.

Zacznijmy od prostej listy zakupów.

LISTY



Zapewne każdy kiedyś korzystał z listy zakupów, lecz nie wszyscy mieli okazję zapisywać ją w kodzie programu w języku Python. Poniższy kod pokazuje, jak można to zrobić:

```
groceries = [ "jablka", "ziemniaki", "chleb", "mleko", "masło",  
↳ "pomarańcze", "sałata", "cukier", "płatki" ]
```

W odróżnieniu od zmiennych, które przechowują jedną daną, powyższa lista umożliwiła zapisanie wielu elementów. Bez trudu można sobie wyobrazić mnóstwo sytuacji, w których zastosowanie listy byłoby bardzo wygodne. W powyższym przykładzie wszystkie *elementy* listy są łańcuchami znaków. Równie dobrze można jednak tworzyć listy zawierające liczby zmiennoprzecinkowe, liczby całkowite lub liczby obu tych typów. Poniżej przedstawiłem listę liczb zmiennoprzecinkowych:

Elementy

```
gpas = [ 3.25, 2.26, 1.99, 3.55, 4.0, 3.21, 2.56, 3.06 , 2.72 ]
```

A teraz zaczniesz się robić ciekawiej. Ponieważ w języku Python listy są traktowane jak obiekty, udostępniają kilka wbudowanych poleceń umożliwiających operowanie na nich i na ich zawartości. To ma naprawdę duże znaczenie, gdyż dostępne funkcje pozwalają na wykonywanie takich operacji jak zliczanie liczby elementów bądź ich sortowanie. Oczywiście takie operacje można także wykonywać, używając samodzielnie napisanego kodu, lecz wymagałoby to naprawdę wiele pracy.

UWAGA: Więcej informacji o **obiekciech** znajdziesz w następnym rozdziale, dlatego jeśli to pojęcie nie jest Ci jeszcze znane, nie masz się czym przejmować.

Zanim przejdziemy do kolejnych, bardziej złożonych zagadnień, warto jeszcze poświęcić nieco uwagi sposobom pobierania danych z list. W tym celu wpisz w edytorze program, którego kod zawiera dwie przedstawione wcześniej listy i kilka dodatkowych instrukcji:

```
groceries = [ "jablka", "ziemniaki", "chleb", "mleko", "maslo",  
             ↪ "pomarańcze", "sałata", "cukier", "płatki" ]  
gpas = [ 3.25, 2.26, 1.99, 3.55, 4.0, 3.21, 2.56, 3.06 , 2.72 ]  
print groceries[0]  
print groceries[4]  
print gpas[3]
```

Kiedy już upewnisz się, że poprawnie wpisałeś kod programu, zapisz go i wykonaj za pomocą polecenia python. Jego wyniki będą takie same jak na rysunku 5.1.

```

Terminal
prog4pocz: ~ $ python groceries.py
jablka
maslo
3.55
prog4pocz: ~ $

```

Rysunek 5.1. Program wyświetlający dane pobierane z listy wykonany w serwisie Koding.com

W powyższym przykładzie szczególną uwagę zwróć na instrukcje `print`. Jak widać, każda z nich odwołuje się do jednej ze zdefiniowanych wcześniej list i do konkretnego indeksu w ramach danej listy zapisanego w nawiasach kwadratowych. W języku Python, podobnie jak w większości innych języków programowania, listy są indeksowane, co oznacza, że każdemu ich elementowi jest przyporządkowywana liczba, której można używać, by odwołać się do danego elementu. Pierwszy element listy zawsze ma indeks o wartości 0.

Indeks

W przypadku przedstawionej w przykładzie listy zakupów indeksy jej elementów będą miały następującą postać:

| INDEKS | ELEMENT LISTY |
|--------|---------------|
| 0 | jablka |
| 1 | ziemniaki |
| 2 | chleb |
| 3 | mleko |
| 4 | maslo |
| 5 | pomarańcze |
| 6 | salata |
| 7 | cukier |
| 8 | plátky |

Odwołanie służące do pobierania danych z list ma następującą, ogólną postać, przedstawioną tutaj w połączeniu z poleceniem `print`:

```

print
nazwaListy[indeks]

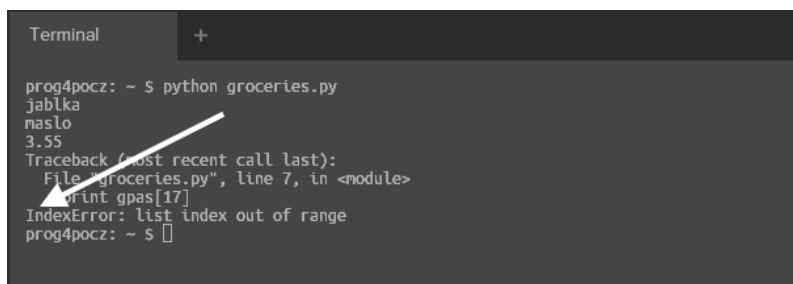
```

Jeśli masz jakieś doświadczenia związane z korzystaniem z innych języków programowania, to zauważyłeś zapewne, że listy są bardzo podobne do tablic. Jeśli to zauważyłeś, to miałeś rację!

W kontekście listy trzeba pamiętać o jeszcze jednym zagadnieniu. Otóż długość list zależy wyłącznie od ich zawartości. Jakakolwiek próba odwołania się do nieistniejącego indeksu spowoduje zakończenie programu i wyświetlenie komunikatu o błędzie. W ramach testu dodaj do poprzedniego programu poniższy wiersz kodu:

```
print gpas[17]
```

Po wprowadzeniu tej zmiany próba wykonania programu spowoduje wyświetlenie wyników przedstawionych na rysunku 5.2.



```
Terminal +
prog4pocz: ~ $ python groceries.py
jablka
maslo
3.55
Traceback (most recent call last):
  File "groceries.py", line 7, in <module>
    print gpas[17]
IndexError: list index out of range
prog4pocz: ~ $
```

Rysunek 5.2. Próba odwołania się do indeksu spoza zakresu listy spowoduje wyświetlenie błędu

EDYCJA ELEMENTÓW LIST

Indeksów można także używać do zmieniania wartości elementów list. Do tego celu używana jest instrukcja przypisania, taka sama jak w przypadku przypisywania wartości do zwyczajnych zmiennych. Poniższa instrukcja zmieni wartość pierwszego elementu listy gpas:

```
gpas[0] = 4.0
```

Zwróć uwagę, że ta instrukcja nie zmienia indeksów pozostałych elementów listy.

STOSOWANIE FUNKCJI LIST

Jak już wspomniałem, listy są obiektami, a to daje możliwość stosowania wielu funkcji skojarzonych z tymi obiektami. Warto zajrzeć do oficjalnej dokumentacji języka, by poznać dostępne funkcje operujące na listach.

Dokumentacja języka Python obejmująca funkcje wykonujące operacje na listach jest dostępna na stronie <https://docs.python.org/2/tutorial/datastructures.html>.

Aby przetestować jedną z tych funkcji, a konkretnie funkcję `append()`, dodaj do poprzedniego programu dwa poniższe wiersze kodu:

`append()`

```
groceries.append("kurczak")
print groceries
```

Jak widać, funkcja `append()` została poprzedzona kropką i dodana do nazwy listy. W powyższym przypadku wywołanie funkcji `append()` doda łańcuch znaków "kurczak" do pierwszego dostępnego elementu listy. Oczywiście jest, że zastosowanie tej funkcji spowodowało powiększenie długości listy.

Po dodaniu powyższego kodu do wcześniejszego programu i jego wykonaniu program wyświetli wyniki przedstawione na rysunku 5.3.



```
Terminal
prog4pocz: ~ $ python groceries.py
jablka
naslo
3:55
['jablka', 'ziemiaki', 'chleb', 'mleko', 'naslo', 'pomarancze', 'salata', 'cukier', 'platk', 'kurczak']
prog4pocz: ~ $
```

Rysunek 5.3. Warto zauważyć, że w zawartości listy wyświetlonej za pomocą polecenia `print` znalazł się także łańcuch znaków "kurczak"

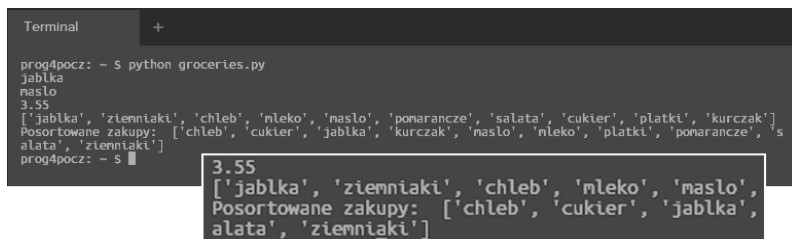
Warto przetestować jeszcze jedną z funkcji operujących na listach. Funkcja `sort()` robi dokładnie to, co sugeruje jej nazwa — sortuje elementy listy.

Można jej używać analogicznie jak funkcji `append()`. Poniżej przedstawiłem kolejne dwa wiersze kodu, które należy dodać do przykładowego programu.

`sort()`

```
groceries.sort()
print "Posortowane zakupy: ", groceries
```

Uruchom teraz ten program za pomocą polecenia `python`, a wygeneruje on wyniki przedstawione na rysunku 5.4.



```
Terminal
prog4pocz: ~ $ python groceries.py
jablka
maslo
3.55
['jablka', 'ziemniaki', 'chleb', 'mleko', 'maslo', 'pomarancze', 'salata', 'cukier', 'platk', 'kurczak']
Posortowane zakupy: ['chleb', 'cukier', 'jablka', 'kurczak', 'maslo', 'mleko', 'platk', 'pomarancze', 's
alata', 'ziemniaki']
prog4pocz: ~ $
```

Rysunek 5.4. Po wywołaniu funkcji `sort()` lista `groceries` zostanie przeindeksowana. W elemencie o indeksie 0 zostanie zapisany łańcuch "chleb", w elemencie o indeksie 1 — łańcuch "cukier", i tak dalej

KROTKI

Krotki są bardzo podobne do list, choć różnią się od nich pod jednym bardzo ważnym względem: nie są dynamiczne. Po zdefiniowaniu krotki jej elementy nie mogą być zmieniane. Zacznijmy od napisania nowego programu, demonstrującego tworzenie i stosowanie krotek:

```
widths = (600, 800, 1024, 1280, 1366, 1920)
print widths
print widths[2]
widths[0] = 500
```

Po uruchomieniu program ten wygeneruje wyniki przedstawione na rysunku 5.5.

```

Terminal
+
prog4pocz: ~ $ python widths.py
(600, 800, 1024, 1280, 1366, 1920)
1024
Traceback (most recent call last):
  File "widths.py", line 4, in <module>
    widths[0] = 500
TypeError: 'tuple' object does not support item assignment
prog4pocz: ~ $

```

Rysunek 5.5. Warto zwrócić uwagę, że można się odwoływać zarówno do całej krotki, jak i do jej konkretnego elementu, jednak próba przypisania wartości wybranemu elementowi krotki spowodowała zgłoszenie błędu. Krotki nie są dynamiczne, dlatego po zainicjowaniu ich wartości nie można zmieniać

Krotki służą do przechowywania danych, które nie zmieniają się w trakcie działania programu. Na przykład w programie przedstawionym powyżej jest bardzo mało prawdopodobne, by zdefiniowane w krotce szerokości ekranów uległy zmianie.

SŁOWNIKI

Słowniki są kolejną strukturą danych dostępną w języku Python. Idealnie się one nadają do przechowywania par danych, określanych także czasami jako **parę klucz – wartość**. W przypadku danych tego typu, zamiast indeksów liczbowych, każda informacja jest indeksowana przy użyciu klucza, którego postać określa programista. Przykładami danych, które z powodzeniem można zapisywać za pomocą słowników, są nazwy województw i ich stolic albo nazwy miast i ich kody pocztowe.

Klucz

Wartość

Przeanalizujmy następujący program demonstrujący zastosowanie słowników:

```

states = { 'dolnoslaskie': 'Wroclaw',
           'lubelskie': 'Lublin',
           'malopolskie': 'Krakow',
           'pomorskie': 'Gdansk',
           'slaskie': 'Katowice',
           'wielkopolskie': 'Poznan' }

print states['slaskie']

```

Po uruchomieniu powyższego programu w wierszu poleceń wyświetli on łańcuch znaków `slaskie`. Najważniejsze jednak w tym przykładzie jest to, że kluczem skojarzonym z wartością `"Katowice"` jest `"slaskie"`. Zwrócić uwagę na polecenie `print` oraz sposób, w jaki odwołuje się ono do wartości zapisanej w słowniku (przedstawiony na rysunku 5.6).



Rysunek 5.6. Sposób odwoływania się do wartości przechowywanych w słowniku

W języku Python słowniki, podobnie jak listy, są obiektami, z którymi są skojarzone funkcje. Przyjrzymy się zatem nieco dokładniej jednej z takich funkcji.

Zmodyfikuj w poniższy sposób kod poprzedniego programu:

```
states = { 'dolnoslaskie': 'Wroclaw',
           'lubelskie': 'Lublin',
           'malopolskie': 'Krakow',
           'pomorskie': 'Gdansk',
           'slaskie': 'Katowice',
           'wielkopolskie': 'Poznan' }

print states
print "Wartosci:", states.values()
print "Klucze:", states.keys()
```

Kiedy wprowadzisz te zmiany i uruchomisz program z poziomu wiersza poleceń, wyświetli on wyniki przedstawione na rysunku 5.7.

The screenshot shows a terminal window with the following output:
Terminal +
prog4pocz: ~/Documents/kody/R5 \$ python caps.py
{'malopolskie': 'Krakow', 'wielkopolskie': 'Poznan', 'lubelskie': 'Lublin', 'slaskie': 'Katowice', 'pomorskie': 'Gdansk', 'dolnoslaskie': 'Wroclaw'}
Wartosci: ['Krakow', 'Poznan', 'Lublin', 'Katowice', 'Gdansk', 'Wroclaw']
Klucze: ['malopolskie', 'wielkopolskie', 'lubelskie', 'slaskie', 'pomorskie', 'dolnoslaskie']
prog4pocz: ~ \$

Below the terminal output, there is a highlighted box containing the following text:
Wartosci: ['Krakow', 'Poznan', 'Lublin',
Klucze: ['malopolskie', 'wielkopolskie',

Rysunek 5.7. Wyniki przedstawiające zastosowanie dwóch funkcji słowników: `values()` i `keys()`

Jak widać, funkcja `values()` zwraca wszystkie wartości zapisane w słowniku, natomiast funkcja `keys()` — wszystkie klucze.

Ten krótki rozdział może stanowić jedynie ogólny wstęp do zagadnień korzystania z danych w pisanych programach. Niemal każdy program, w mniejszym lub większym stopniu, operuje na danych. Dlatego efektywne posługiwanie się danymi jest bardzo ważnym elementem zestawu narzędzi każdego programisty.

SKOROWIDZ

B

biblioteka, 27

D

dane, 49
konwersja, 25
typ, *Patrz:* typ
wejściowe, 23
zapisywanie w pliku, 28

E

edytor tekstów, 15
nano, 15
vi, *Patrz:* vi

H

Hosmer Jonathan, 14

I

instrukcja
elif, 44, 45
if, 40, 45
if else, 41, 45
input, 23, 25
raw_input, 25
warunkowa, 39, 42, 44
warunek, 40
interpreter, 15
iteracja, 45

J

język Python, *Patrz:* Python

K

klasa, 59
instancja, 59
tworzenie, 61, 64
metoda, 60
dodawanie, 63
tworzenie, 61
właściwość, 60
kod formatujący, 15
Koding.com, 13
komentarz, 32
krotka, 54

L

liczba
całkowita, 20, 25, 32
zmiennoprzecinkowa, 20, 25, 32,
33
lista, 49
dodawanie danych, 53
edycja danych, 52
indeks, 51, 52
liczba elementów, 50
pobieranie danych, 50, 51
sortowanie, 50, 53

Ł

łańcuch znaków, 18, 25, 32, 36
długość, 37

N

nano, 15
plik, 18
polecenie, 21

O

obiekt, 59
operacja
arytmetyczna, 34, 35
na łańcuchach znaków, 36, 37
operator
!=, 42
==, 42
and, 43
porównania, 42
przypisania, 32

P

para klucz – wartość, 55
pętla, 39, 45
nieskończona, 47
while, 45, 46
polecenie
append, 53
file.write, 29
import, 27
print, 18, 20, 51
sort, 53
procesor tekstów, 15
program
Terminal, *Patrz:* Terminal
uruchamianie, 18
Python, 11, 13, 14
bloki kodu, 40
interpreter, *Patrz:* interpreter
rozszerzenie, 18
tryb interaktywny, 35

S

słownik, 55, 56
słowo kluczowe class, 61

T

Terminal, 11
typ, 31, 32

V

vi, 15
plik zapisywanie, 18
polecenie, 21
tryb
poleceń, 15, 17
wstawiania, 16

W

wartość logiczna, 40
wiersz poleceń, 11, 15, 17, 23, 24, 26
argument, 26, 27
wyrażenie, 20
arytmetyczne, 34

Z

zmienna, 24
deklarowanie, 31
dynamiczna, 33
inicjalizacja, 31
znak
!=, 42
#, 32
%, 34
==, 42

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ 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 WYDAWNICZA

 **Helion SA**

Jeżeli chcesz się nauczyć programować, musisz poznać podstawowe zasady i typowe konstrukcje — pętle, instrukcje warunkowe i zmienne, które są wspólne dla wszystkich powszechnie używanych języków programowania. Jeżeli zamierzasz ugruntować fundamenty, by uczyć się konkretnych języków, sięgnij po tę książkę.

Dzięki niej napiszesz i uruchomisz swój pierwszy program.

Następnie nauczysz się pobierać dane od użytkownika, zapisywać wyniki w plikach oraz zrozumiesz, jak działają zmienne. Kolejny krok to nauka korzystania z pętli i instrukcji warunkowych — w ten sposób będziesz w stanie zaimplementować wiele nawet bardzo zaawansowanych programów. Mark Lassoфф wszystkie koncepcje prezentuje na przykładzie języka Python, którego składnia jest zbliżona do innych języków, takich jak JavaScript, Java czy C. Siegnij po tę książkę i wkrocz w świat programowania!

Dzięki tej książce:

- Poznasz podstawy programowania
- Nauczysz się korzystać z podstawowych konstrukcji — pętli, instrukcji warunkowych
- Wykorzystasz zmienne
- Pobierzesz dane od użytkownika oraz zapiszesz wyniki do pliku
- Zgłębisz podstawy języka Python

Wkrocz w świat programowania!

Mark Lassoфф — urodzony programista. Swoją przygodę z programowaniem rozpoczął na komputerze Commodore 64, tworząc gry w języku BASIC. Ma bogate doświadczenie w pisaniu oprogramowania, stron oraz aplikacji internetowych. Jego pasją jest uczenie programowania innych — dziesięć lat działalności w tym obszarze sprawia, że dziś jest uznawany za jednego z najlepszych nauczycieli.

sięgnij po **WIĘCEJ**



KOD KORZYŚCI

Helion

37818 numer katalogowy

księgarnia Internetowa



<http://helion.pl>

zamówienia telefoniczne



0 801 339900



0 601 339900

Sprawdź najnowsze promocje:

● <http://helion.pl/promocje>

Książki najchętniej czytane:

● <http://helion.pl/bestsellery>

Zamów informacje o nowościach:

● <http://helion.pl/nowosci>

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

ISBN 978-83-283-1908-0



9 788328 319080

Informatyka w najlepszym wydaniu

cena: 19,90 zł