



Przygoda z programowaniem dla dzieci i absolutnie początkujących

Warren Sande, Carter Sande



Tytuł oryginału: Hello World!: Computer Programming for Kids and Other Beginners,
2nd Edition

Tłumaczenie: Krzysztof Brauner

ISBN: 978-83-283-3037-5

Original edition copyright © 2014 by Manning Publications Co.

All rights reserved

Polish edition copyright © 2017 by HELION SA.

All rights reserved.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiejkolwiek 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)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:

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

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

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

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

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

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

Spis treści

Wstęp xi

Podziękowania xvii

O książce xix

1 Zaczynamy 1

Instalowanie Pythona 1 ■ Uruchomienie Pythona za pomocą IDLE 2 ■ Poproszę o instrukcje 3 ■ Interakcja z Pythonem 5 ■ Pora na programowanie 7
■ Uruchomienie pierwszego programu 8 ■ Jeśli coś się nie uda 9 ■ Drugi program 12

2 Zapamiętaj: pamięć i zmienne 15

Wejście, przetwarzanie, wyjście 15 ■ Nazwy 17 ■ Z czego może się składać nazwa? 21 ■ Liczby i łańcuchy znakowe 22 ■ Jak „zmienne” są zmienne? 24
Ktoś zupełnie inny 25

3 Prosta matma 28

Cztery podstawowe działania 29 ■ Operatory 31 ■ Kolejność wykonywania działań 31 ■ Dwa inne operatory 33 ■ Liczby ogromne i bardzo małe 35

4 Typy danych 40

Zmiana typów danych 40 ■ Sprawdzanie typu za pomocą funkcji `type()` 44
Błędy wynikające z konwersji typów 44 ■ Zastosowanie konwersji typów 44

5 Dane wejściowe 46

Funkcja `raw_input()` 47 ■ Instrukcja `print` i przecinek 48 ■ Wprowadzanie liczb 50 ■ Pobieranie danych z sieci 52

6 GUI — graficzny interfejs użytkownika 55

Czym jest GUI? 55 ■ Nasz pierwszy GUI 56 ■ Dane wejściowe w przypadku GUI 57
Wybierz swój smak 58 ■ Gra w odgadywanie liczb... powraca 61 ■ Inne elementy GUI 62

7 Decyzje, decyzje 65

Sprawdzanie warunków 65 ■ Wcięcia 67 ■ Czy ja widzę podwójnie? 68
Inne rodzaje porównań 69 ■ Co się stanie, gdy warunek nie zostanie spełniony? 70
Sprawdzanie więcej niż jednego warunku 72 ■ Słowo kluczowe `and` 73 ■ Słowo kluczowe `or` 74 ■ Słowo kluczowe `not` 74

8 Pętla w pętli 78

Pętle licznikowe 79 ■ Korzystanie z pętli licznikowej 81 ■ Na skróty — `range()` 82
Kwestia stylu — nazywanie zmiennych w pętli 84 ■ Zliczanie z określonym krokiem 87
Zliczanie bez użycia liczb 89 ■ Jeśli już o tym mówimy... 89 ■ Opuszczanie pętli — `break` i `continue` 90

9 Tylko dla Ciebie — komentarze 94

Dodawanie komentarzy 94 ■ Komentarze jednowierszowe 95 ■ Komentarze na końcu wiersza 95 ■ Komentarze wielowierszowe 96 ■ Łańcuchy znakowe z potrójnymi cudzysłowami 96 ■ Styl komentarzy 97 ■ Komentarze zamieszczone w tej książce 97 ■ Komentowanie instrukcji 98

10 Zagrajmy 99

Narciarz (Skier) 99

11 Pętle zagnieżdżone i pętle zmienne 104

Pętle zagnieżdżone 104 ■ Pętle zmienne 106 ■ Zmienne pętli zagnieżdżone 107
Jeszcze więcej zmiennych pętli zagnieżdżonych 108 ■ Korzystanie z pętli zagnieżdżonych 110 ■ Obliczanie kalorii 113

12 Gromadzenie danych — listy i słowniki 117

Czym jest lista? 117 ■ Tworzenie listy 118 ■ Dodawanie elementów do listy 118
Co oznacza kropka? 119 ■ Lista może zawierać cokolwiek 120 ■ Pobieranie elementów z listy 120 ■ Wycinanie listy 121 ■ Modyfikowanie elementów 124
Inny sposób na dodawanie elementów do listy 124 ■ Usuwanie elementów z listy 126
Przeszukiwanie listy 127 ■ Listy i pętla 129 ■ Sortowanie list 129 ■ Typy mutowalne i niemutowalne 133 ■ Lista zawierająca inne listy: tabele danych 133
Słowniki 136

13 Funkcje 143

Funkcje jak klocki 143 ■ Wywołanie funkcji 145 ■ Przekazywanie argumentów do funkcji 147 ■ Funkcje z większą liczbą argumentów 149 ■ Funkcje zwracające wartość 151 ■ Zasięg zmiennej 153 ■ Wymuszanie zmiennej globalnej 156 ■ Rada dotycząca nazywania zmiennych 157

14 Obiekty 159

Obiekty w świecie rzeczywistym 160 ■ Obiekty w Pythonie 160 ■ Obiekt = atrybuty + metody 162 ■ Co oznacza kropka? 162 ■ Tworzenie obiektów 162 ■ Przykładowa klasa — HotDog 168 ■ Ukrywanie danych 172 ■ Polimorfizm i dziedziczenie 173 ■ Planowanie 175

15 Moduły 178

Czym jest moduł? 178 ■ Do czego można wykorzystać moduły? 178 ■ Pudełka z klockami 179 ■ Jak się tworzy moduły? 179 ■ Jak korzystać z modułów? 180
Przestrzenie nazw 181 ■ Moduły standardowe 185

16 Grafika 189

Pomocna dłoń — Pygame 189 ■ Okno Pygame 190 ■ Rysowanie w oknie 191
Pojedyncze piksele 200 ■ Obrazki 205 ■ Ruszamy! 207 ■ Animacja 208
Płynniejsza animacja 210 ■ Odbijanie piłki 211 ■ Zawijanie piłki 214

17 Sprajty i wykrywanie zderzeń 217

Sprajty 217 ■ Bęc! Wykrywanie kolizji 223 ■ Zliczanie czasu 228

18 Nowy rodzaj danych wejściowych — zdarzenia 233

Zdarzenia 233 ■ Zdarzenia związane z klawiaturą 235 ■ Zdarzenia związane z myszą 240 ■ Zdarzenia timera 241 ■ Czas na kolejną grę — PyPong 244

19 Dźwięk 256

Ponownie sięgamy po pomoc — mixer 256 ■ Generowanie dźwięków kontra odtwarzanie dźwięków 257 ■ Odtwarzanie dźwięków 257 ■ Sterowanie poziomem głośności 260 ■ Odtwarzanie muzyki z powtarzaniem 262 ■ Dodajemy dźwięk do gry PyPong 263 ■ Jeszcze więcej odłotowych dźwięków 264 ■ Dodajemy muzykę do PyPonga 267

20 Więcej o GUI 272

Korzystanie z PyQt 272 ■ Zapręgamy GUI do pracy 277 ■ Obsługa zdarzeń po raz kolejny 279 ■ Przesuwanie przycisku 280 ■ Bardziej użyteczny GUI 280 ■ TempGUI 281 ■ Co dzisiaj w menu? 287

21 Formatowanie łańcuchów znakowych 293

Koniec linii 294 ■ Odstępy poziome — tabulatory 296 ■ Wstawianie zmiennych do łańcucha znakowego 298 ■ Formatowanie liczb 299 ■ Inny sposób formatowania 304 ■ Operacje na łańcuchach znakowych 305

22 Odczyt i zapis plików 313

Czym jest plik? 314 ■ Nazwy plików 314 ■ Położenie plików 315 ■ Otwieranie pliku 319 ■ Odczytywanie pliku 320 ■ Pliki tekstowe i pliki binarne 322 ■ Zapisywanie do pliku 323 ■ Zapisywanie bardziej złożonych danych: pickle 327 ■ Czas na kolejną grę — Wisielec (Hangman) 329

23 Zaryzykuj — losowość 337

Czym jest losowość? 338 ■ Rzucanie kością do gry 338 ■ Tworzymy talię kart 343 ■ Szalone ósemki 348

24 Symulacje komputerowe 361

Modelowanie świata rzeczywistego 361 ■ Lądownik księżycowy 362 ■ Kontrolowanie czasu 367 ■ Obiekty reprezentujące czas 368 ■ Zapisywanie czasu do pliku 372 ■ Wirtualny zwierzak 374

25 Narciarz (Skier) — objaśnienie 385

Narciarz 385 ■ Przeszkody 389

26 Bitwa pytonów 399

Bitwa pytonów 399 ■ Tworzenie robota 401 ■ Bardziej rozbudowany robot 403
Układ współrzędnych 404

27 Co dalej? 409

Dla młodszych programistów 409 ■ Python 410 ■ Programowanie gier
i Pygame 410 ■ Programowanie gier w innych językach niż Python 411
„Prosty” znaczy „BASIC” 411 ■ Aplikacje mobilne 411 ■ Rozejrzyj się 411

Dodatek A Zasady nazewnictwa zmiennych 413

Dodatek B Różnice pomiędzy Pythonem 2 i Pythonem 3 415

Dodatek C Klucz odpowiedzi do pytań i zadań 419

Spis listingów 451

Skorowidz 455

Zaczynamy

Instalowanie Pythona

W pierwszej kolejności powinieneś zainstalować Pythona na komputerze, z którego będziesz korzystać.

Nie jest to trudne zadanie, jednak **bardzo Cię zachęcam, abyś skorzystał z programu instalacyjnego *Hello World***; dzięki niemu zainstalujesz prawidłową wersję oprogramowania, którego będziesz używał w trakcie lektury książki. Program instalacyjny znajdziesz na serwerze wydawnictwa Helion¹ (<ftp://ftp.helion.pl/przyklady/hellow.zip>). Wybierz wersję programu odpowiednią dla systemu operacyjnego, z którego korzystasz.

ZA STARYCH, DOBRYCH CZASÓW



Na samym początku rozwoju komputerów osobistych (PC) było znacznie łatwiej. Wiele pierwszych komputerów PC było wyposażonych we wbudowany język programowania o nazwie BASIC i nie trzeba było niczego instalować. Wystarczyło włączyć komputer — na ekranie pojawiał się napis „READY” i można było zacząć pisać programy w BASIC-u. Fajnie, prawda?

Inna sprawa, że napis „READY” to było już wszystko — nie było programów, okienek, menu. Jeśli chciałeś, aby komputer zrobił *cokolwiek*, musiałeś sam napisać program! Nie było edytorów tekstowych, odtwarzaczy multimedialnych, przeglądarek internetowych i innych programów, do których obecnie jesteś przyzwyczajony. Nie było nawet internetu, który mógłbyś przeglądać. Nie było także ładnej grafiki ani dźwięków — poza okazjonalnym „bzyknięciem”, gdy coś źle zrobiłeś.

¹ Pliki do angielskiej wersji książki dostępne są na stronie <http://www.helloworldbook2.com> — przyp. tłum.

Do dyspozycji masz programy instalacyjne dla systemów operacyjnych Windows, Mac OS X i Linux. Wszystkie przykłady zamieszczone w tej książce zostały omówione w oparciu o system Windows, niemniej w systemach Mac OS X i Linux korzystanie z Pythona wygląda bardzo podobnie.

W tej książce będziemy korzystać z Pythona w wersji 2.7.5. Jeśli skorzystałeś z instalatora pobranego ze strony towarzyszącej książce, taką właśnie wersję będziesz miał zainstalowaną na komputerze. W chwili gdy to czytasz, dostępna może być nowsza wersja Pythona. Wszystkie przykłady przedstawione w książce zostały przetestowane w wersji 2.7.5 Pythona. Bardzo możliwe, że będą one także działały na nowszych wersjach 2.x, jednak nie mogę dać na to żadnej gwarancji.

Jeśli na Twoim komputerze jest już zainstalowany Python i nie użyjesz naszego instalatora, upewnij się, że zainstalowanych jest także kilka dodatkowych rzeczy, z których będziemy korzystać. Aby się dowiedzieć, jak to zrobić, zajrzyj do sekcji poświęconej instalacji znajdującej się na stronie internetowej www.helloworldbook2.com. Ponownie jednak zachęcam Cię, abyś skorzystał z instalatora dostępnego na serwerze wydawnictwa Helion (<ftp://ftp.helion.pl/przyklady/hellow.zip>).



Python 2 kontra Python 3

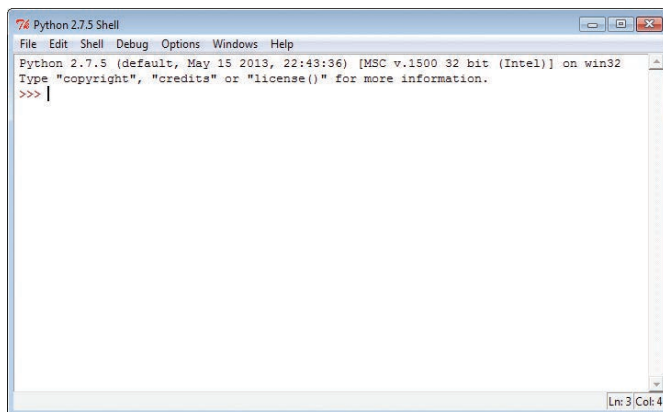
Na kilka lat przed powstaniem niniejszej książki opublikowana została nowa wersja języka Python o nazwie Python 3. Okazało się jednak, że jest to nie tyle zaktualizowana wersja, co zupełnie nowa gałąź rozwoju tego języka. Z tego powodu wielu programistów postanowiło pozostać przy starszej wersji — Python 2. Autorzy Pythona rozwijają i tworzą równoległe zarówno nowe wersje Pythona 2, jak i Pythona 3. W momencie publikacji drugiego wydania tej książki aktualnymi wersjami Pythona były Python 2.7.5 i Python 3.3.0. W tej książce korzystamy z Pythona 2.7.5, ale kod będzie najprawdopodobniej kompatybilny ze wszystkimi kolejnymi wersjami Pythona 2.x.

Więcej szczegółów dotyczących różnic między Pythonem 2 i Pythonem 3 znajdziesz w dodatku B.

Uruchomienie Pythona za pomocą IDLE

Pracę z Pythonem możesz rozpocząć na kilka sposobów. Jednym z nich jest wykorzystanie środowiska IDLE, co za chwilę uczynimy.

Otwórz menu *Start*. W folderze *Python 2.7* znajdziesz pozycję *IDLE (Python GUI)*. Kliknij ją, a otworzy się okno środowiska IDLE. Powinno ono wyglądać podobnie jak na poniższym rysunku.



IDLE jest **powłoką** (ang. *shell*) Pythona. Powłoka umożliwia interakcję z programem za pomocą wprowadzanego tekstu. Powłoka IDLE pozwala więc na interakcję z Pythonem (to z tego powodu na pasku tytułowym jest napisane *Python 2.7.5 Shell*). IDLE jest zarazem graficznym interfejsem użytkownika (ang. *GUI*), dlatego w menu *Start* widoczny jest napis *Python GUI*. IDLE posiada coś jeszcze, ale o tym opowiem za chwilę.

SŁOWNICZEK

GUI oznacza **graficzny interfejs użytkownika** (ang. *graphical user interface*). Jest to program, który składa się z okienek, menu, przycisków, pasków przewijania itp. Programy, które nie posiadają GUI, nazywamy **programami tekstowymi** lub **programami wiersza poleceń**.

Znak `>>>`, który mogłeś zauważyć na poprzednim rysunku, to **znak zachęty** (ang. *prompt*). Program wyświetla znak zachęty, gdy czeka, aż coś napiszesz. Znaki `>>>` oznaczają więc, że Python oczekuje na Twoje instrukcje.

Poproszę o instrukcje

Wydajmy Pythonowi pierwszą instrukcję. Upewnij się, że kursor znajduje się na końcu znaku zachęty `>>>`, i wprowadź:

```
print "Witaj, świecie!"
```

a następnie naciśnij klawisz *Enter* (na niektórych klawiaturach jest on oznaczony jako *Return*). Musisz nacisnąć *Enter* na końcu każdej linii.

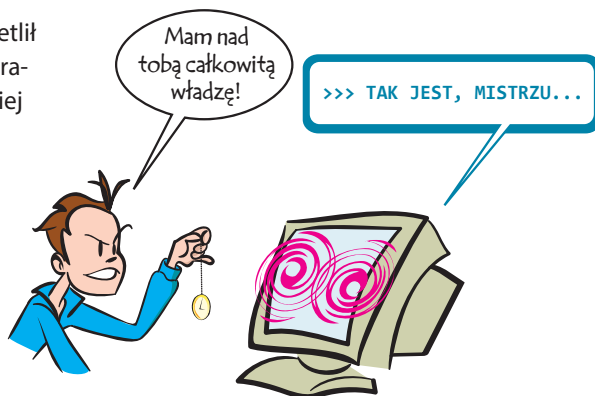
Gdy naciśniesz *Enter*, powinieneś otrzymać następującą odpowiedź:

```
Witaj, świecie!  
>>>
```

Poniższy rysunek przedstawia okno IDLE po wykonaniu instrukcji.

Python wykonał to, co mu kazałeś — wyświetlił komunikat **Witaj, świecie!** (w świecie programowania instrukcja **print**² oznacza najczęściej wyświetlenie czegoś na ekranie komputera, a nie wydrukowanie na drukarce). Ta linia to **instrukcja** w języku Python. Właśnie rozpoczęłeś programowanie! Masz władzę nad komputerem!

Tak przy okazji — stało się już tradycją, że pierwszą rzeczą, jakiej się uczysz podczas



nauki programowania, jest umiejętność sprawienia tego, by komputer wyświetlił napis **Witaj, świecie!** (ang. *Hello World!*). To stąd wziął się tytuł niniejszej książki. Stałeś się częścią tej tradycji — witaj w świecie programowania!

Dobre pytanie! IDLE pomaga Ci w zrozumieniu niektórych rzeczy. Prezentuje różne elementy kodu za pomocą różnych kolorów, abyś mógł je łatwo odróżnić (**kod** to inny sposób na określenie instrukcji, jakie wydajesz komputerowi w danym języku programowania, takim jak np. Python). W dalszej części książki objaśnię, co oznaczają poszczególne kolory.

² W języku angielskim słowo *print* oznacza „wydrukuj” — przyp. tłum.

A jeśli coś nie działa?

Jeśli popełnisz w kodzie jakiś błąd, zapewne zobaczysz na ekranie coś podobnego do tego:

```
>>> pront "Witaj, świecie!"  
SyntaxError: invalid syntax  
>>>
```

Komunikat błędu informuje Cię, że wprowadziłeś polecenie, którego Python nie rozumie. W powyższym przykładzie z powodu literówki zamiast polecenia `print` wprowadziłeś polecenie `pront` i Python nie wie, co z tym poleceniem począć. Jeśli przydarzy Ci się taka sytuacja, spróbuj ponownie wprowadzić instrukcję i upewnij się, że jest ona dokładnie taka sama jak w przykładzie.



Masz rację — dzieje się tak dlatego, że słowo `print` jest słowem kluczowym, a słowo `pront` nim nie jest.

SŁOWNICZEK

Słowo kluczowe (nazywane także **słowem zarezerwowanym**) to specjalne słowo, które jest częścią języka Python.

Interakcja z Pythonem

Przed chwilą miałeś okazję użyć Pythona w trybie interaktywnym. Wpisałeś polecenie (instrukcję), a Python natychmiast ją wykonał.

SŁOWNICZEK

Wykonanie polecenia, instrukcji lub programu to po prostu ich „uruchomienie”.

Spróbujmy zrobić coś innego w trybie interaktywnym. Wpisz takie polecenie:

```
>>> print 5 + 3
```

Powinieneś otrzymać następujący wynik:

```
8  
>>>
```

A więc Python potrafi dodawać! Nie jest to zaskakujące, gdyż komputery są niezwykle biegłe w dziedzinie arytmetyki.

Sprawdźmy jeszcze następującą rzecz:

```
>>> print 5 * 3
15
>>>
```

W przypadku niemal wszystkich komputerów i języków programowania symbol `*` oznacza operację mnożenia. Znak ten nazywamy **gwiazdką**.

Jeśli na lekcjach matematyki zapisujesz „5 razy 3” lub „5 x 3”, będziesz się musiał przyzwyczaić do stosowania w Pythonie znaku `*` (w przypadku większości klawiatur jest on umieszczony nad cyfrą 8).



No tak. A co w takim razie powiesz na to:

```
>>> print 2345 * 6789
15920205
>>>
```



Dobrze. A co w przypadku takiego mnożenia:

```
>>> print 1234567898765432123456789 * 9876543212345678987654321
12193263200731596000609652202408166072245112635269
>>>
```



To prawda — na komputerze możesz wykonywać operacje matematyczne na bardzo dużych liczbach.

Możesz wykonać także inną operację:

```
>>> print "kot" + "pies"
kotpies
>>>
```

Albo wypróbuj takie polecenie:

```
>>> print "Witaj " * 20
Witaj Witaj Witaj Witaj Witaj Witaj Witaj Witaj Witaj Witaj
Witaj Witaj Witaj Witaj Witaj Witaj Witaj Witaj Witaj Witaj
```

Poza tym, że komputery przeprowadzają operacje matematyczne, radzą sobie bardzo dobrze z wielokrotnym wykonywaniem tej samej operacji. W tym przypadku nakazaliśmy Pythonowi wyświetlenie 20 razy napisu `Witaj`.

W trybie interaktywnym popracujemy jeszcze trochę później, a teraz...

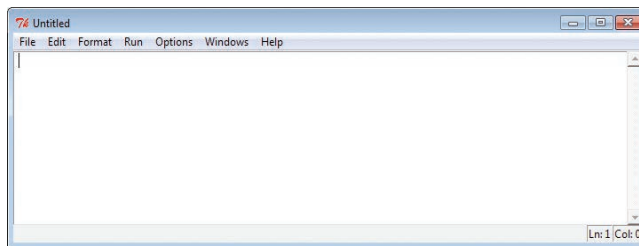
Pora na programowanie

Przykłady, które prezentowałem do tej pory, były pojedynczymi instrukcjami (wydawanymi w trybie interaktywnym). Jest to dobry sposób na to, aby się przekonać, co potrafi Python, ale nie są to tak naprawdę programy. Jak wspomniałem wcześniej, program komputerowy składa się z szeregu instrukcji. Stwórzmy więc nasz pierwszy program w Pythonie.

Zacznijmy od tego, że musisz jakoś wpisać treść programu. Jeśli zrobisz to po prostu w oknie trybu interaktywnego, Python w żaden sposób tego nie „zapamięta”. Musisz skorzystać z edytora tekstowego (takiego jak Notatnik w systemie Windows, TextEdit w systemie Mac OS X lub vi w Linuksie), który potrafi zapisać plik na dysku. Środowisko IDLE jest wyposażone w edytor tekstowy, który sprawdzi się tutaj znacznie lepiej niż Notatnik. Aby otworzyć ten edytor, wybierz z menu IDLE polecenie *File/New Window*.

Gdy mówimy o poleceniach dostępnych w menu (np. *File/New Window*), to pierwsza część (*File* w tym przypadku) oznacza menu główne programu. Znak / wskazuje, że następna pozycja, czyli w tym przypadku *New Window*, znajduje się w menu *File*. Z takiej formy zapisu będziemy korzystać w dalszej części książki.

Pojawi się okno podobne do okna przedstawionego na poniższym rysunku. Na pasku tytułowym znajduje się napis *Untitled*, gdyż nie nadaliśmy jeszcze plikowi nazwy.



Wprowadź teraz w edytorze program z poniższego listingu³.

Listing 1.1. Nasz pierwszy program z prawdziwego zdarzenia

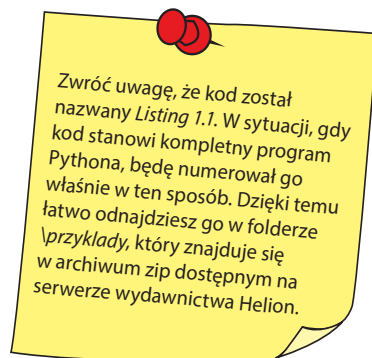
```
print "Uwielbiam pizzę!"
print "pizza " * 20
print "mniam " * 40
print "Najadłem się."
```

Gdy skończysz, zapisz program za pomocą polecenia *File/Save* lub *File/Save As*. Nazwij go *pizza.py*. Program możesz zapisać w dowolnym miejscu na dysku (ważne jest jednak, abyś zapamiętał gdzie — będziesz mógł go później łatwo odnaleźć). Najlepiej jednak będzie, jeśli utworzysz nowy folder, w którym będziesz zapisywał programy w Pythonie. Końcówka nazwy pliku *.py* pełni ważną funkcję — informuje komputer, że nie jest to zwykły plik tekstowy, tylko program napisany w języku Python.

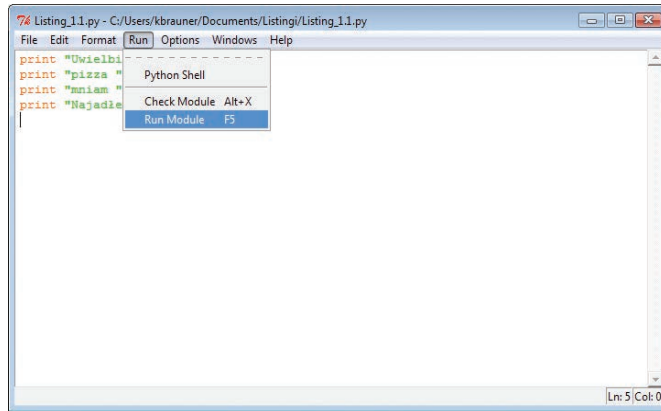
Jak zapewne zauważyłeś, edytor sformatował treść programu, używając różnych kolorów. Niektóre słowa są pomarańczowe, a inne zielone. Dzieje się tak dlatego, że edytor IDLE jest świadomy tego, iż wprowadzasz program Pythona, i wyświetla wszystkie słowa kluczowe w kolorze pomarańczowym, a to, co znajduje się pomiędzy cudzysłowami, wyświetla w kolorze zielonym. Dzięki temu kod Pythona staje się znacznie bardziej czytelny.

Uruchomienie pierwszego programu

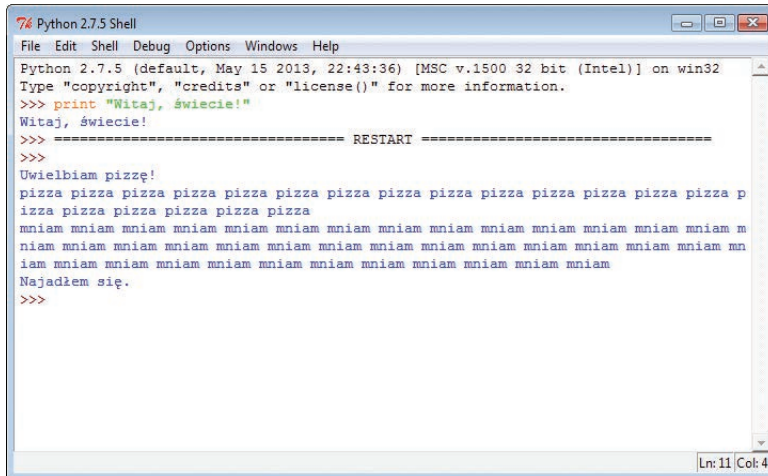
Gdy zapiszesz program na dysku, przejdź do menu *Run* (nadal będąc w edytorze IDLE) i wybierz polecenie *Run Module* (jak na poniższym rysunku). Polecenie to uruchomi Twój program.



³ Aby zapewnić poprawne działania programu, w którym występują polskie znaki diakrytyczne, należy zmienić domyślne ustawienia środowiska IDLE. Z menu *Options* należy wybrać pozycję *Configure IDLE...*, a następnie w oknie dialogowym *IDLE Preferences* należy przejść do zakładki *General* i zaznaczyć opcję *Locale-defined* widoczną w sekcji *Default Source Encoding*. Innym rozwiązaniem jest dodawanie w pierwszej linii każdego nowego programu dodatkowej klauzuli: `# -*- coding: utf-8 -*-` — *przyp. tłum.*



Uaktywni się ponownie okno powłoki Pythona (chodzi o okno, które otworzyło się, gdy po raz pierwszy uruchomiłeś środowisko IDLE) i zobaczysz w nim obraz podobny do tego z poniższego rysunku:



Fragment opatrzonej tekstem **RESTART** informuje, że w tym miejscu uruchomiłeś program. Informacja ta stanie się pomocna, gdy zaczniesz testować swoje programy, uruchamiając je raz za razem.

Następnie program wykonuje operacje. Może nie dzieje się tutaj zbyt wiele, ale sprawiłeś, że komputer zrobił to, co mu nakazałeś. Kolejne programy będą bardziej interesujące.

Jeśli coś się nie uda

A co się stanie, gdy popełnisz błąd w programie i z tego powodu on się nie uruchomi? Mogą się pojawić dwa różne rodzaje błędów. Przyjrzyjmy się obu, abyś wiedział, co zrobić, kiedy przydadzą się one Tobie.

Błędy składni

Jeszcze zanim uruchomisz program, środowisko IDLE go zweryfikuje. Jeśli znajdzie w nim jakiś błąd, będzie to najprawdopodobniej **błąd składni** (ang. *syntax error*). Składnia to zbiór reguł rządzących danym językiem programowania. Błąd składni oznacza więc, że wpisałeś coś, co nie jest poprawne w języku Python.

Oto przykład:

```
print "Witaj w świecie Pythona."
print "Mam nadzieję, że podczas nauki programowania będziesz się dobrze bawić."
print Do zobaczenia!"
```

 Tutaj brakuje cudzysłowu

Zapomnieliśmy umieścić cudzysłów między `print` a `Do zobaczenia!`.

Jeśli spróbujesz uruchomić ten program, IDLE wyświetli komunikat: "There's an error in your program: *invalid syntax*". W takiej sytuacji musisz się przyrzeć programowi i znaleźć błąd. Edytor IDLE podświetli na czerwono miejsce, w którym wykrył błąd. Może to nie być dokładna lokalizacja błędu, ale na pewno będzie to wskazanie, że wystąpił on w okolicy zaznaczonego miejsca.

Błędy wykonania

Drugim rodzajem możliwych błędów jest błąd, którego Python (lub IDLE) nie wykryje przed uruchomieniem programu. Błąd ten pojawia się w momencie wykonywania programu, dlatego nazywa się **błędem wykonania** (ang. *runtime error*). Oto przykład takiego błędu:

```
print "Witaj w świecie Pythona"
print "Mam nadzieję, że podczas nauki programowania będziesz się dobrze bawić."
print "Do zobaczenia!" + 5
```

Jeśli zapiszesz ten program, a następnie spróbujesz go uruchomić, to uruchomi się on bez żadnych problemów. Wyświetlą się dwie pierwsze linijki tekstu, a następnie pojawi się komunikat błędu:

```
>>> ===== RESTART =====
```

```
>>>
```

```
Witaj w świecie Pythona
```

```
Mam nadzieję, że podczas nauki programowania będziesz się dobrze bawić.
```

```
Traceback (most recent call last):
```

```
File "C:/Users/kbrauner/Documents/Listingi/error1.py", line 3, in <module>
```

```
    print "Do zobaczenia!" + 5
```

```
TypeError: cannot concatenate 'str' and 'int' objects
```

```
>>>
```

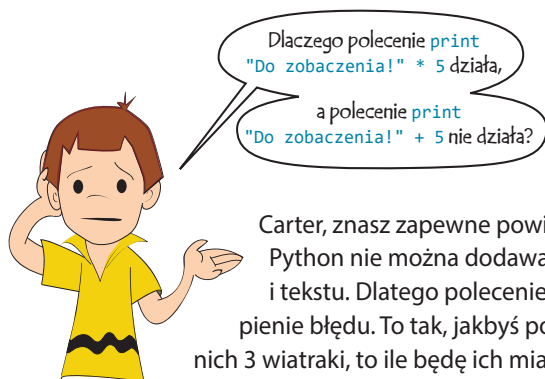
Początek komunikatu błędu

Miejsce, w którym wystąpił błąd

„Zła” linia kodu

Co zdaniem Pythona jest powodem błędu?

Linia rozpoczynająca się od wyrazu `Traceback` wskazuje początek komunikatu błędu. Kolejna linia informuje o tym, gdzie wystąpił błąd — zawiera ona nazwę pliku i numer linii. Następnie wyświetlana jest linia kodu zawierająca błąd. Dzięki temu możesz łatwo dojść do tego, gdzie jest problem. Ostatnia linia komunikatu błędu informuje o tym, co zdaniem Pythona jest powodem błędu.



Carter, znasz zapewne powiedzenie: „Co ma piernik do wiatraka?”. W języku Python nie można dodawać do siebie dwóch różnych rzeczy — np. liczby i tekstu. Dlatego polecenie `print "Do zobaczenia!" + 5` powoduje wystąpienie błędu. To tak, jakbyś powiedział: „Jeśli wezmę 5 pierników i dodam do nich 3 wiatraki, to ile będę ich miał?”. Będziesz miał 8, ale czego? Dodawanie do siebie tych rzeczy nie ma zupełnie sensu. Możesz jednak pomnożyć prawie każdą rzecz daną liczbą razy — otrzymasz w wyniku jeszcze więcej tej samej rzeczy. Jeżeli masz 2 pierniki i pomnożysz je razy 5, otrzymasz 10 pierników! Właśnie z tego powodu instrukcja `print "Do zobaczenia!" * 5` działa prawidłowo.

Myśl jak programista

Nie przejmuj się, jeśli w Twoim programie pojawiają się komunikaty błędów. Dzięki nim łatwiej odnajdziesz błędny kod i go poprawisz. Jeżeli w programie jest błąd, to lepiej, żebyś się o nim dowiedział dzięki komunikatowi błędu. Błędy, które nie powodują pojawienia się komunikatu, są znacznie trudniejsze do zlokalizowania!



Drugi program

Pierwszy program nie robił zbyt wiele — wyświetlał po prostu coś na ekranie. Spróbujmy czegoś nieco bardziej interesującego.

Kod na listingu 1.2 to prosta gra w odgadywanie liczb. Za pomocą polecenia *File/New Window* otwórz nowy plik w edytorze IDLE — podobnie jak to zrobiłeś za pierwszym razem. Przepisz program z listingu 1.2, a następnie zapisz go na dysku. Możesz nadać mu dowolną nazwę, ważne jednak, aby kończyła się ona znakami *.py*. Dobrym wyborem będzie nazwa *NumGuess.py*.

Program składa się z 18 linii kodu i kilku pustych linii, dzięki którym kod będzie bardziej przejrzysty. Przepisywanie go nie powinno Ci zająć wiele czasu. Na razie nie przejmuj się tym, że nie rozumiesz jeszcze, o co w tym programie chodzi — przejdziemy do tego za chwilę.

Listing 1.2. Gra w odgadywanie liczb⁴

```
import random
sekret = random.randint(1, 99)  ← Wybranie losowej liczby
propozycja = 0
proba = 0
print "AHOJ! Jam jest pirat Roberts Straszliwy i mam dla Ciebie zagadkę!"
print "Jest nią tajemna liczba od 1 do 99. Na odgadnięcie jej masz 6 prób."
while propozycja != sekret and proba < 6:
    propozycja = input("Jaka to liczba?")  ← Pobranie propozycji gracza
    if propozycja < sekret:
        print "Za mała, psuabracie!"
    elif propozycja > sekret:
        print "Za duża, szczerze lądowy!"

    proba = proba + 1  ← Zwiększenie liczby prób

if propozycja == sekret:
    print "Stop! Udało Ci się! Odgadłeś moją tajemną liczbę!"
else:
    print "Wykorzystałeś wszystkie próby! Powodzenia następnym razem, koleżko!"
    print "Tajemna liczba to", sekret
```

Gracz ma do wykorzystania tylko 6 prób

Wyświetlenie komunikatu na końcu rozgrywki

Kiedy będziesz przepisywać kod, zwróć uwagę na wcięcia linii występujące po instrukcji *while* oraz na dodatkowe wcięcia po instrukcjach *if* i *elif*. Zwróć także uwagę na dwukropki na końcu niektórych linii. Jeśli umieścisz dwukropek w odpowiednim miejscu, edytor automatycznie zrobi wcięcie w kolejnym wierszu.

⁴ Język Python (i wiele innych języków programowania) nie pozwala na stosowanie w nazwach zmiennych polskich znaków diakrytycznych, więc tutaj i w dalszej części książki zmienne będą ich pozbawione — *przyp. tłum.*

- Uruchomiłeś edytor tekstowy IDLE i wprowadziłeś swój pierwszy program.
- Uruchomiłeś swój pierwszy program.
- Dowiedziałeś się, czym są komunikaty błędów.
- Uruchomiłeś swój drugi program — grę w odgadywanie liczb.

Sprawdź swoją wiedzę

1. W jaki sposób można uruchomić środowisko IDLE?
2. Do czego służy instrukcja `print`?
3. Jak wygląda w Pythonie symbol mnożenia?
4. Jaki napis wyświetla IDLE w momencie uruchamiania programu?
5. Jak można inaczej określić uruchomienie programu?

Zadania praktyczne

1. Korzystając z trybu interaktywnego, oblicz za pomocą Pythona, ile minut ma tydzień.
2. Napisz prosty program, który wyświetli trzy linie tekstu: Twoje nazwisko, Twoją datę urodzenia i Twój ulubiony kolor. Wynik powinien wyglądać podobnie do poniższego:

```
Nazywam się Warren Sande.  
Urodziłem się 1 stycznia 1970.  
Mój ulubiony kolor to niebieski.
```

Zapisz program, a następnie go uruchom. Jeśli program nie działa zgodnie z oczekiwaniami lub jeśli wyświetla się komunikat błędu, spróbuj znaleźć błąd i popraw program.

Skorowidz

A

AI, *Patrz*: sztuczna inteligencja
animacja, 207, 208, 209, 210, 218, 377
 klatka, *Patrz*: klatka
 szybkość, 228, 229
aplikacja mobilna, 411
argument, 86, 147, 149, 150, 151
Artificial Intelligence, *Patrz*: sztuczna inteligencja
atrapa, *Patrz*: funkcja szkieletowa
atrybut, 160, 161, 162

B

bajt, 195, 314
BASIC, 411
biblioteka
 Cocoa Touch, 411
 PyMunk, 410
 standardowa, 185
bit, xii, 195, 314
blitowanie, 207, 219, 236
blok kodu, 67, 79, 144
 try-except, 378
błąd, 5, 9, 11
 konwersji typów, 44
 obsługa, 378
 przechwytywanie, 378
 składni, 10

wykonania, 10
zaokrąglenia, 42

Boo, 411

C

Chipmunk Physics, 410
cwd, *Patrz*: katalog bieżący
cyfra binarna, xii
czas, 368, 369, 370
 mierzenie, 370
 rzeczywisty, 367
zapisywanie do pliku, 372
znacznik, 368

D

dane
 struktura, 134
 zapisywanie, 327, 328, 329
typ, *Patrz*: typ
ukrywanie, *Patrz*: hermetyzacja
walidacja, 354
wejściowe, 15, 46, 47, 50, 58, 313, *Patrz też*:
 wejście
 z sieci, 52
wyjściowe, 15, 313, *Patrz też*: wyjście
data, 368, 369
dekrementacja, 25, 35

dokumentacja, 94, 95
 drzewo decyzyjne, 111
 dziedziczenie, 173, 174
 dźwięk, 256
 generowanie, 257
 odtworzenie, 257
 w tle, 261
 z powtarzaniem, 262
 poziom głośności, 260

E

EasyGUI, 56, 62
 edytor tekstowy, 7, 8

F

folder, 315, *Patrz też:* katalog
 format
 .ui, 275
 MP3, 257
 Ogg Vorbis, 257
 wave, 257, 323
 WMA, 257
 funkcja, 143
 abs, 387
 argument, *Patrz:* argument
 choice, 339
 choicebox, 59
 clock.get_fps, 229, 230
 clock.tick, 229, 230, 235, 368
 close, 321, 325
 definiowanie, 143, 144
 dump, 328
 enterbox, 60
 float, 40, 44, 50, 321, 371
 importowanie, 186
 input, 47, 51, 416
 int, 40, 43, 51, 300, 321
 integerbox, 61
 load, 329
 msgbox, 56, 57
 open, 319, 320, 324
 parametr, *Patrz:* parametr
 print, 415
 pygame.image.load, 205, 206
 random.randint, 187, 338
 range, 82, 85, 129, 417

krok, 87
 raw_input, 46, 47, 49, 50, 51, 416
 spritecollide, 225
 str, 40, 44
 szkieletowa, 175, 176
 time.delay, 226, 228, 367
 time.sleep, 185, 186, 367
 type, 44
 wbudowana, 40
 wynik, 151, 152
 wywoływanie, 144, 145

G

grafika, 189, 192
 animacja, *Patrz:* animacja
 wymazywanie, 209, 280
 GUI, 3, 55, 272, 375
 tworzenie, 56

H

hermetyzacja, 172, 173

I

IDLE, 3, 4, 5
 indeks, 36, 120, 121
 inkrementacja, 25, 35
 instrukcja, xii
 del, 126
 elif, 70
 else, 71, 185
 for, 185
 if, 66, 185
 print, 4, 48, 49, 294, 327, 415
 return, 152
 wstrzymywanie, 98
 iteracja, 80, 84, 89
 iterator, 83

J

Java, 411
 język programowania, xii
 BASIC, 411
 Boo, 411
 Java, 411
 Lua, 411
 obiektowy, 159

Objective-C, 411
 Python, *Patrz:* Python
 Scratch, 409
 Squeak Etoys, 409

K

katalog, 315, *Patrz też:* folder
 bieżący, 317
 klasa, 162, 218, 278
 group, 224
 instancja, 162, 218, 278
 nazwa, 170
 pochodna, 174, 175
 pygame.time.Clock, 228, 229
 timedelta, 370
 klatka, 228
 kod binarny, xii
 kolekcja, 138
 kolor
 nazwa, 194
 podstawowy, 194
 RGB, 194, 195
 kombinacja, 110
 komentarz, 94, 95
 jednowierszowy, 95
 na końcu wiersza, 95
 styl, 97
 wielowierszowy, 96
 komponent, *Patrz:* widżet
 kontrolka, *Patrz:* widżet
 krotka, 133

L

liczba
 całkowita, 29, 41, 43, 416
 wprowadzanie, 61
 wyświetlanie, 300
 formatowanie, 299, 300, 304
 losowa, 187, 337
 notacja
 naukowa, 35, 36, 37, 38, 302
 wykładnicza, 36, 37
 rzeczywista, 29
 zmiennoprzecinkowa, 29, 30, 41, 416
 wprowadzanie, 61
 wyświetlanie, 301

licznik, 343
 linia, 202
 szerokość, 198
 lista, 58, 81, 117, 120, 138, 151
 element, 118
 dodawanie, 118, 124
 modyfikowanie, 124
 pobieranie, 120
 usuwanie, 126, 127
 wstawianie, 125
 kopia, 122, 130, 131
 niemutowalna, 133
 przeszukiwanie, 127
 separator, 306
 sortowanie, 125, 129, 132
 w odwrotnej kolejności, 130
 tworzenie, 118
 wycinanie, 121, 123
 wycinek, 122
 wydłużanie, 125
 zawierająca znaki, 86
 losowość, 337, *Patrz też:* prawdopodobieństwo
 Lua, 411

Ł

łańcuch
 formatujący, 298
 znakowy, 22, 86, 96
 dodawanie, *Patrz:* łańcuch znakowy
 konkatenacja
 dzielenie, 305
 formatowanie, 286, 293
 konkatenacja, 23, 295, 307
 usuwanie części, 310
 wielkość liter, 311
 wprowadzanie, 47
 wyszukiwanie, 308, 309
 zamiana na liczbę zmiennoprzecinkową, 43

M

menu, 287, 375
 dodawanie pozycji, 288
 klawisz Alt, 290
 tworzenie, 288
 metoda, 160, 161, 162
 __init__, 165, 166, 235, 401

metoda

- __str__, 166
- append, 118, 119, 124, 125, 126
- combine, 370
- decode, 371
- draw.rect, 198, 199, 364
- endswith, 309
- extend, 125
- format, 304, 305
- index, 128
- insert, 125, 126
- isfile, 373
- join, 307
- keys, 139
- kill, 394
- lower, 311
- łańcuchowa, 305
- move, 221, 280
- now, 370, 371
- pop, 127
- pygame.display.flip, 191, 192
- pygame.draw.circle, 191
- pygame.draw.lines, 202
- pygame.draw.rect, 197
- pygame.mixer.music.set_volume, 260
- readlines, 320, 321, 322
- remove, 126
- reverse, 130
- screen.blit, 207
- seek, 322
- setGeometry, 280
- sort, 129
- sorted, 132, 139
- specjalna, 166
- split, 306, 373
- startswith, 308
- strip, 310
- Surface.set_at, 204
- upper, 311
- write, 326, 327

moduł, 143, 178, 179

- datetime, 368
- importowanie, 184, 186
- os, 373
- pickle, 327, 373
- pygame.font.Font, 249, 364

- pygame.mixer, 256
- pygame.mixer.music, 258, 260, 261, 267
- PyQt, 272, 281, 288, 329, 330, 377
- random, 187, 338
- sprite, 217, 218
- standardowy, *Patrz*: biblioteka standardowa
- time, 185, 228
- tworzenie, 179, 180

N

- nazwa, 16, 17, 18, 19, 25, 26
- zasady, 21, 26, 413
- zmiennej licznikowej, 84

O

- obiekt, 119, 143, 159, 162
 - atrybut, *Patrz*: atrybut
 - date, 369, 370
 - datetime, 368, 369, 370, 373
 - ekran, 192
 - font, 249
 - inicjalizacja, 164, 258
 - instancja, 163, 167
 - pygame.mixer, 258
 - pygame.mixer.Sound, 258
 - rect, 196, 197
 - atrybuty, 198
 - reprezentujący plik, 319
 - time, 369, 370
 - timedelta, 370, 371, 372
 - tworzenie, 162

Objective-C, 411

okno

- dialogowe, 57, 58, 383
- IDLE, *Patrz*: IDLE
- liczbowe, 61
- wielkość, 59, 60, 220, 221
- wprowadzania, 60
- wyboru, 59, 62

Open Graphics Language, *Patrz*: OpenGL

OpenGL, 410

operand, 31

operator, 31

- dekrementacji, 35, 75
- inkrementacji, 35, 75
- matematyczny, 75

modulo, 34, 75, 300
 porównania, 69, 70, 75
 potęgowania, 33, 38, 75
 przypisania, 31, 75
 relacyjny, 70
 selekcji, 162
 oprogramowanie, xii
 otwarte, xiv

P

pamięć, 16
 struktura, 315, 316
 zarządzanie, 153, 155
 parametr, 150, *Patrz też:* argument
 pasek
 narzędzi, 375, 376
 postępu, 375, 376
 tytułowy, 375
 permutacja, 110
 pętla
 for, *Patrz:* pętla licznikowa
 licznikowa, 78, 79, 81, 89
 nieskończona, 80
 obsługi zdarzeń, 190, 191, 234
 warunkowa, 78, 89
 kończenie, 90, 91
 while, 78, 190, *Patrz też:* pętla warunkowa
 zagnieżdżona, 104, 105, 112
 podwójnie, 108
 zmienna, 107
 PhoneGap, 411
 piksel, 193, 200
 rysowanie, 201, 204
 pixel-perfect collision detection, *Patrz:* wykrywanie
 zderzeń precyzyjne
 plik, 314
 binarny, 322
 dźwiękowy, 257
 format, *Patrz:* format
 nazwa, 314, 319
 otwieranie, 319, 320, 322
 w trybie do odczytu, 319, 320, 324
 w trybie do zapisu, 319, 324, 325, 328
 w trybie dołączania, 319, 324, 325
 położenie, 314, 315

 ścieżka, 316
 tekstowy, 322
 zamykanie, 321
 zapisywanie, 323, 325, 326, 327
 podłańcuch, 310
 polimorfizm, 173
 powierzchnia ekranu, 193
 kopia, 192, 194
 podwójne buforowanie, 194
 współrzędne, 195
 powłoka, 3
 powtarzanie klawiszy, 239
 prawdopodobieństwo, 337, 338, 339
 prędkość, 363
 procedura, *Patrz:* zdarzenie procedura obsługi
 program, 7
 debugowanie, 98, 110
 dokumentacja, *Patrz:* dokumentacja
 sterowany zdarzeniami, 233, 235
 tekstowy, 3, 55
 wiersza poleceń, 3, 55
 zatrzymywanie, 80
 projektowanie
 wstępujące, 351
 zstępujące, 351
 prompt, *Patrz:* znak zachęty
 przestrzeń nazw, 181, 250
 globalna, 183
 importowanie, 184
 lokalna, 183, 186
 przycisk, 274
 przesuwanie, 280
 Push Button, 274, 282
 przypisanie, 17, 25
 operator, *Patrz:* operator przypisania
 przyspieszenie, 363
 Pygame, 100, 189, 217, 256, 363, 377, 410
 PyOpenGL, 410
 pytanie, 68
 Python, xiii, xiv
 dokumentacja, 410
 instalowanie, 1
 system pomocy, 63
 wersja, 2, 30, 47, 47, 69, 73, 83, 415, 416, 417
 konwersja programu, 418

Q

Qt Designer, 273, 281
 formularz, 273
 klasa, 277
 tytuł, 375
 przycisk, *Patrz*: przycisk
 widżet, *Patrz*: widżet

R

rachunek prawdopodobieństwa, 341, *Patrz też*:
 prawdopodobieństwo
 rect collision detection, *Patrz*: wykrywanie zderzeń
 z wykorzystaniem prostokątów otaczających
 referencja do bieżącej instancji, 167
 renderowanie, 249
 runtime error, *Patrz*: błąd wykonania

S

Scratch, 409
 serwer proxy, 52
 shell, *Patrz*: powłoka
 skrót klawiaturowy, 287, 290
 w systemie Mac OS, 291
 słownik, 117, 136, 137, 138
 element, 140
 klucz, 137, 139
 sortowanie, 139
 wartość, 137, 139
 słowo
 kluczowe, 5, 40
 and, 73, 128
 break, 90, 91
 class, 278
 continue, 90, 91
 def, 143
 from, 184
 global, 156, 157
 if, 66
 import, 180
 in, 127, 140, 310
 not, 74, 128
 or, 74, 128
 pass, 176
 while, 90
 zarezerwowane, 5

software, *Patrz*: oprogramowanie
 sprajt, 217, 218, 363
 grupowanie, 224
 image, 218
 rect, 218, 221, 226
 zderzenie, *Patrz*: wykrywanie zderzeń
 Squeak Etoys, 409
 stała, 106
 struktura danych, *Patrz*: dane struktura
 stwierdzenie, 68
 symbol formatujący, 299, 300, 301, 302, 303
 symulacja komputerowa, 361, 374
 łądowania, 362, 363
 syntax error, *Patrz*: błąd składni
 system
 Android, 411
 binarny, 42, 314
 szewron, 327
 sztuczna inteligencja, 399
 szybkość, 363

T

tabela, 133, 134
 tablica
 asocjacyjna, 139
 mieszająca, 139
 tabulator, 296
 tekstu formatowanie, 113
 test
 logiczny, 70
 warunkowy, 70
 timer, 377
 timestamp, *Patrz*: znacznik czasu
 tryb interaktywny, 5, 7, 20
 typ, 40, 314
 boolowski, 127
 float, 40, 44
 int, 40, 43
 konwersja, 40, 44
 błąd, 44
 list, 122
 mutowalny, 133, 139
 niemutowalny, 133, 139
 sprawdzanie, 44
 str, 40, 44

znak

%i, 298, 299
 %i, 300
 %s, 298, 299
 &, 290
 *, *Patrz*: znak gwiazdka
 **, *Patrz*: znak gwiazdka podwójna
 „, *Patrz*: znak przecinek
 ., *Patrz*: znak kropka
 /, *Patrz*: znak ukośnik
 //, *Patrz*: znak ukośnik podwójny
 :, *Patrz*: znak dwukropek
 [], *Patrz*: znak nawias kwadratowy
 {}, *Patrz*: znak nawias klamrowy
 \, *Patrz*: znak ukośnika wstecznego
 \n, 296
 \t, 296, *Patrz też*: znak tabulacji
 \t, 296
 ^, 33
 +=, *Patrz*: inkrementacja, operator inkrementacji
 <, 69, 75
 <>, 69, 75
 =, *Patrz*: znak równości
 -=, *Patrz*: dekrementacja, operator dekrementacji
 ==, *Patrz*: operator porównania, znak równości
 podwójny
 >, 69, 75
 >>, 327
 >>>, *Patrz*: znak zachęty

cudzysłów, 8, 10, 17, 18, 22
 potrójny, 23, 96, 98
 diakrytyczny, 8
 dwukropek, 12, 67, 123, 131, 144
 gwiazdka, 3, 6, 75, 184, 185, 186, 187
 podwójna, 33, 75
 hash, *Patrz*: znak krzyżyk, komentarz
 kontynuacji, *Patrz*: znak ukośnika wstecznego
 końca linii, 53, 294, 296, 322
 kropka, 162
 krzyżyk, 95, *Patrz też*: komentarz
 łańcuch, *Patrz*: łańcuch znakowy
 myślnik, 29, 75
 nawias klamrowy, 137
 nawias kwadratowy, 81
 niedrukowalny, 307
 podkreślenia, 413
 procent, 34, 75, 298
 wyświetlanie, 303
 przecinek, 48, 49, 150, 295
 przekierowania, 327
 równości, 3, 17, 75
 podwójny, 69, *Patrz też*: operator porównania
 specjalny, 296
 tabulacji, 113, 296
 ukośnik, 29, 75, 317
 podwójny, 30
 wsteczny, 114, 243, 296, 297, 317
 zachęty, 3, 17

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

Witaj w niezwykłym świecie programowania!

Nauka programowania to świetny pomysł, ale musimy Cię ostrzec: programowanie wciąga! Jeśli nauczysz się porozumiewać z komputerem w jego języku, szybko zorientujesz się, że napisanie własnej gry, skryptu do złożonych obliczeń czy funkcjonalnej aplikacji do codziennego użytku daje wielką satysfakcję i sprawia mnóstwo radości. Co więcej, programowania można się nauczyć niemal w każdym wieku. Wystarczy odrobina cierpliwości i konsekwencji!

Trzymasz w ręce chyba najlepszy podręcznik do nauki programowania dla osób absolutnie początkujących — nawet takich, które komputera używają wyłącznie do przeglądania stron i obsługi poczty. W sposób szczególny ten przewodnik nadaje się dla dzieci, ale skorzystają z niego również dorośli, którzy chcą poznać podstawy programowania. Jasno i klarownie przedstawiono tu wszystkie niezbędne informacje, a liczne (bardzo zabawne) przykłady pozwalają na głębsze zrozumienie prezentowanych treści. Nauka odbywa się w Pythonie. Jest to język łatwy w nauce, a przy tym bardzo popularny i wciąż rozwijany. Co więcej, przyswojenie sobie Pythona pozwala na szybką naukę innych języków!

W tym podręczniku znajdziesz informacje m.in. o tym:

- jak przygotować się do pracy: zainstalować Pythona i napisać pierwsze linie kodu
- czym są i do czego służą zmienne, typy danych i operatory
- czym jest GUI i jak go napisać
- jakie są rodzaje pętli, co to są funkcje i na czym polega obsługa zdarzeń
- jak przygotować symulacje komputerowe

Warren Sander — jest inżynierem systemów elektronicznych. Na co dzień korzysta z Pythona zarówno do nauczania programowania, jak i do pisania w pełni profesjonalnych skryptów.

Carter Sander — jest uczniem o wielkiej pasji do technologii. Troszczy się o szkolną sieć informatyczną, chętnie pomaga kolegom w rozwiązywaniu problemów z komputerami, a ponadto uwielbia jeździć na rowerze i grać w stare gry wideo.

Helion

księgarnia internetowa

<http://helion.pl>

zamówienia telefoniczne



0 801 339900



0 601 339900

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

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

Informatyka w najlepszym wydaniu



ISBN 978-83-283-3037-5



9 788328 330375

cena: 59,00 zł

