

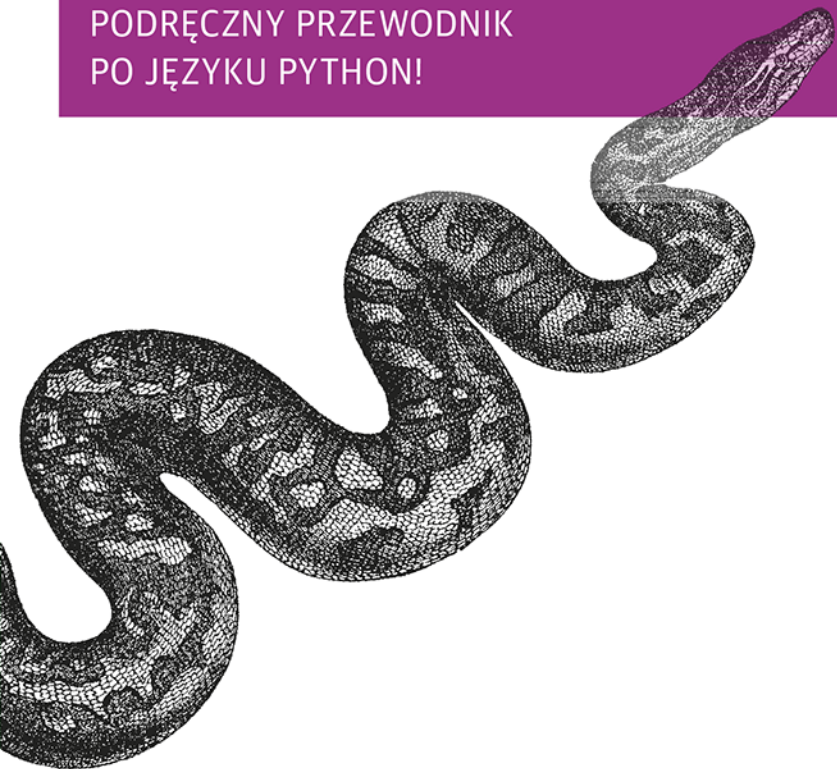
O'REILLY®

Wydanie V

Python

Leksykon kieszonkowy

PODRĘCZNY PRZEWODNIK
PO JĘZYKU PYTHON!



Helion 

Mark Lutz

Tytuł oryginału: Python Pocket Reference, Fifth Edition

Tłumaczenie: Radosław Meryk

ISBN: 978-83-283-6035-8

© 2014, 2019 Helion S.A.

Authorized Polish translation of the English edition Python Pocket Reference, 5th Edition ISBN 9781449357016 © 2014 Mark Lutz.

This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to publish and sell the same.

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 bierze jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Wydawnictwo HELION nie ponosi 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/pylk5v>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

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

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

Spis treści

Wprowadzenie	7
Konwencje	8
Opcje wiersza poleceń Pythona	9
Opcje poleceń Pythona	9
Specyfikacja programu w wierszu polecenia	11
Opcje poleceń Pythona 2.X	12
Zmienne środowiskowe	13
Zmienne operacyjne	13
Zmienne opcji wiersza poleceń	14
Python Launcher dla systemu Windows	15
Dyrektywy plikowe launchera	15
Wiersz poleceń launchera	16
Zmienne środowiskowe launchera	17
Wbudowane typy i operatory	17
Operatory i priorytet ich stosowania	17
Uwagi na temat stosowania operatorów	19
Operacje według kategorii	21
Uwagi na temat działań na sekwencjach	25
Specyficzne typy wbudowane	26
Liczby	26
Ciągi znaków	29
Łańcuchy znaków Unicode	46
Listy	50
Słowniki	56
Krotki	60
Pliki	61
Zbiory	66
Inne typy i konwersje	68

Instrukcje i ich składnia	70
Reguły składniowe	70
Reguły dotyczące nazw	72
Instrukcje	75
Instrukcja przypisania	75
Instrukcja wyrażeniowa	79
Instrukcja print	80
Instrukcja if	82
Instrukcja while	83
Instrukcja for	83
Instrukcja pass	84
Instrukcja break	84
Instrukcja continue	84
Instrukcja del	84
Instrukcja def	85
Instrukcja return	89
Instrukcja yield	89
Instrukcja global	91
Instrukcja nonlocal	91
Instrukcja import	92
Instrukcja from	95
Instrukcja class	97
Instrukcja try	99
Instrukcja raise	102
Instrukcja assert	104
Instrukcja with	104
Instrukcje w Pythonie 2.X	106
Przestrzenie nazw i reguły zasięgu	107
Nazwy kwalifikowane — przestrzenie nazw obiektów	107
Nazwy niekwalifikowane — zasięgi leksykalne	107
Zasięgi zagnieżdżone i domknięcia	109
Programowanie obiektowe	110
Klasy i egzemplarze	111
Atrybuty pseudoprywatne	112
Klasy nowego stylu	113
Formalne reguły dziedziczenia	114

Metody przeciążające operatory	118
Wszystkie typy	119
Kolekcje (sekwencje, mapy)	125
Liczby (operatory dwuargumentowe)	127
Liczby (inne działania)	130
Deskryptory	130
Menedżery kontekstu	131
Metody przeciążające operatory w Pythonie 2.X	132
Funkcje wbudowane	135
Funkcje wbudowane w Pythonie 2.X	157
Wbudowane wyjątki	163
Klasy bazowe (kategorie)	163
Wyjątki szczegółowe	165
Szczegółowe wyjątki OSError	169
Wyjątki kategorii ostrzeżeń	170
Framework ostrzeżeń	171
Wbudowane wyjątki w Pythonie 3.2	172
Wbudowane wyjątki w Pythonie 2.X	173
Wbudowane atrybuty	173
Standardowe moduły biblioteczne	174
Moduł sys	175
Moduł string	183
Funkcje i klasy modułu	183
Stałe	184
Moduł systemowy os	185
Narzędzia administracyjne	186
Stałe wykorzystywane do zapewnienia przenośności	187
Polecenia powłoki	188
Narzędzia do obsługi środowiska	190
Narzędzia do obsługi deskryptorów plików	191
Narzędzia do obsługi nazw ścieżek	194
Zarządzanie procesami	198
Moduł os.path	201
Moduł dopasowywania wzorców re	204
Funkcje modułu	204
Obiekty wyrażeń regularnych	206

Obiekty dopasowania	207
Składnia wzorców	208
Moduły utrwalania obiektów	210
Moduły dbm i shelve	212
Moduł pickle	214
Moduł GUI tkinter i narzędzia	217
Przykład użycia modułu tkinter	217
Podstawowe widgety modułu tkinter	218
Wywołania okien dialogowych	218
Dodatkowe klasy i narzędzia modułu tkinter	220
Porównanie biblioteki Tcl/Tk z modułem tkinter Pythona	220
Moduły i narzędzia do obsługi internetu	222
Inne standardowe moduły biblioteczne	224
Moduł math	225
Moduł time	225
Moduł timeit	227
Moduł datetime	228
Moduł random	228
Moduł json	228
Moduł subprocess	229
Moduł enum	230
Moduł struct	230
Moduły obsługi wątków	231
API baz danych Python SQL	232
Przykład użycia interfejsu API	233
Interfejs modułu	234
Obiekty połączeń	234
Obiekty kursora	235
Obiekty typów i konstruktory	236
Idiomy Pythona i dodatkowe wskazówki	236
Wskazówki dotyczące rdzenia języka	236
Wskazówki dotyczące środowiska	238
Wskazówki dotyczące użytkowania	239
Różne wskazówki	241
Skorowidz	243

Instrukcje

W kolejnych podrozdziałach zostaną opisane wszystkie instrukcje występujące w Pythonie. W każdym podrozdziale zamieszczono format składni instrukcji, a za nim szczegółowe informacje na temat jej wykorzystania. W przypadku instrukcji złożonych każde wystąpienie ciągu *grupa* wewnątrz formatu instrukcji oznacza jedną lub więcej innych instrukcji. Instrukcje te mogą tworzyć blok pod wierszem nagłówka. Dla grupy trzeba zastosować wcięcie pod nagłówkiem, jeśli zawiera inną instrukcję złożoną (*if*, *while* itp.). W przeciwnym razie może się znaleźć w tym samym wierszu co nagłówki instrukcji. Obie poniższe konstrukcje są prawidłowe:

```
if x < 42:
    print(x)
    while x: x = x - 1

if x < 42: print(x)
```

Poniżej zamieszczono szczegółowe informacje wspólne dla wersji Pythona 3.X i 2.X. Szczegóły obowiązujące jedynie w wersji 2.X zestawiono na końcu podrozdziału „Instrukcje w Pythonie 2.X”.

Instrukcja przypisania

```
cel = wyrażenie
cel1 = cel2 = wyrażenie
cel1, cel2 = wyrażenie1, wyrażenie2
cel1 += wyrażenie

cel1, cel2, ... = obiekt_iterowalny-o-tej-samej-długości
(cel1, cel2, ...) = obiekt_iterowalny-o-tej-samej-długości
[cel1, cel2, ...] = obiekt_iterowalny-o-tej-samej-długości
cel1, *cel2, ... = obiekt_iterowalny-o-odpowiedniej-długości
```

W instrukcjach przypisania cele zawierają *referencje* do obiektów. Instrukcje przypisania powinny mieć jawny format zamieszczony powyżej, w którym:

- *wyrażenia* generują obiekty;
- *cele* mogą być prostymi nazwami (x), kwalifikowanymi atrybutami ($x.attr$) albo indeksami i wycinkami ($x[i]$, $x[i:j:k]$);
- *zmienne* wewnątrz celów nie są deklarowane wcześniej, ale przed użyciem ich w wyrażeniu trzeba nadać im wartość (patrz podrozdział „Wyrażenia atomowe i dynamiczne określanie typów”).

Pierwszy format wymieniony powyżej to *proste* przypisanie. Drugi z formatów — przypisanie *wielocelowe* (ang. *multiple-target*) — służy do przypisania tego samego wyrażenia do każdego z celów. Trzeci format — przypisanie *krotek* — łączy w pary cele z wyrażeniami, od lewej do prawej. Czwarty format — *przypisanie z aktualizacją* — jest skrótem dla połączenia operacji dodawania z przypisaniem (patrz następny podrozdział).

Ostatnie cztery formaty to *przypisanie sekwencji*, które służy do przypisywania komponentów dowolnej sekwencji (bądź innego obiektu iterowalnego) do odpowiadających im celów — od lewej do prawej. Sekwencja lub obiekt iterowalny występujące po prawej stronie mogą być dowolnego typu, ale muszą być tej samej długości, chyba że wśród celów z lewej strony znajdzie się nazwa rozpoczynająca się gwiazdką ($*x$), tak jak w ostatnim formacie. Ten ostatni format jest znany jako *rozszerzone przypisanie sekwencji*. Instrukcja o takim formacie jest dostępna wyłącznie w Pythonie 3.X. Pozwala ona na pobranie dowolnej liczby elementów (patrz podrozdział „Rozszerzone instrukcje przypisania sekwencji (3.X)”).

Przypisania mogą również występować *niejawnie* w innych kontekstach w Pythonie (np. zmienne pętli `for` oraz mechanizm przekazywania argumentów funkcji). Niektóre z formatów instrukcji przypisania można również stosować w innych miejscach (np. przypisania sekwencji w instrukcjach `for`).

Przypisanie z aktualizacją

W Pythonie dostępny jest zbiór dodatkowych formatów instrukcji przypisania. Zestawiono je w tabeli 13. Są to tzw. *przypisania z aktualizacją* — formaty te implikują wyrażenie dwuargumentowe razem z przypisaniem. Na przykład poniższe dwa formaty w przybliżeniu są sobie równoważne:

```
X = X + Y
X += Y
```


Tabela 13. Instrukcje przypisania z aktualizacją

$X += Y$	$X \&= Y$	$X -= Y$	$X = Y$
$X *= Y$	$X \wedge= Y$	$X /= Y$	$X >>= Y$
$X \%= Y$	$X <<= Y$	$X **= Y$	$X //= Y$

Jednak wartość celu X w instrukcji drugiego formatu może być wyznaczona *tylko raz*. Dodatkowo format ten pozwala na zastosowanie działań *w miejscu* dla typów mutowalnych (np. wyrażenie `list1 += list2` automatycznie wywołuje metodę `list1.extend(list2)` zamiast wolniejszej operacji konkatencji implikowanej przez operator `+`). W klasach przypisania w miejscu mogą być przeciążane za pomocą nazw metod rozpoczynających się od `i` (np. `__iadd__()` dla operatora `+=`, `__add__()` dla operatora `+`). W Pythonie w wersji 2.2 wprowadzono nowy format: `X //= Y` (dzielenie całkowite).

Zwykłe instrukcje przypisania sekwencji

W Pythonie 2.X i 3.X dowolną sekwencję (bądź też inny obiekt iterowalny) złożoną z wartości można przypisać do dowolnej sekwencji nazw, pod warunkiem że ich długości są takie same. Podstawowa forma instrukcji przypisania sekwencji działa w większości kontekstów przypisania:

```
>>> a, b, c, d = [1, 2, 3, 4]
>>> a, d
(1, 4)

>>> for (a, b, c) in [[1, 2, 3], [4, 5, 6]]:
...     print(a, b, c)
...
1 2 3
4 5 6
```

Rozszerzone instrukcje przypisania sekwencji (3.X)

W Pythonie 3.X (wyłącznie) instrukcja przypisania sekwencji została rozszerzona, by umożliwić obsługę przypisania sekwencji o dowolnej liczbie elementów — wystarczy poprzedzić gwiazdką jedną ze zmiennych celu przypisania. W takim przypadku długości sekwencji nie muszą być identyczne, natomiast nazwa poprzedzona gwiazdką pobiera niepasujące elementy do nowej listy:

```
>>> a, *b = [1, 2, 3, 4]
>>> a, b
(1, [2, 3, 4])
```

```

>>> a, *b, c = (1, 2, 3, 4)
>>> a, b, c
(1, [2, 3], 4)

>>> *a, b = 'spam'
>>> a, b
(['s', 'p', 'a'], 'm')

>>> for (a, *b) in [[1, 2, 3], [4, 5, 6]]:
...     print(a, b)
...
1 [2, 3]
4 [5, 6]

```

UWAGA

Uogólnienie gwiazdki w Pythonie 3.5 lub w wersjach późniejszych? W Pythonie 3.3 i wcześniejszych wersjach specjalne formy składni `*X` i `**X` mogą występować w trzech miejscach: w *instrukcjach przypisania*, gdzie `*X` pobiera niedopasowane elementy w instrukcjach przypisania sekwencji; w *nagłówkach funkcji*, w których te dwie formy służą do zbierania niepasujących argumentów pozycyjnych i argumentów w postaci słów kluczowych; oraz w *wywołaniach funkcji*, gdzie wymienione dwie formy rozpakowują obiekty iterowalne i słowniki do indywidualnych elementów (argumentów).

W Pythonie 3.4 deweloperzy planowali uogólnić składnię z gwiazdką w taki sposób, by można jej było używać także w *literalach opisujących struktury danych*, w których miałaby ona powodować rozpakowywanie kolekcji do postaci pojedynczych elementów, podobnie jak działa to obecnie w wywołaniach funkcji. Rozpakowująca składnia z gwiazdką będzie mogła być wykorzystywana dla *krotek, list, zbiorów, słowników i obiektów składanych*. Na przykład:

```

[x, *iter]           # rozpakowanie elementów obiektu iterowalnego iter: lista
(x, *iter), {x, *iter} # to samo dla krotki, zbiór
{'x': 1, **dict}     # rozpakowanie elementów słownika: słowniki
[*iter for iter in x] # rozpakowanie elementów obiektu iterowalnego:
                     # obiekty składane

```

Są to dodatkowe miejsca zastosowania składni z gwiazdką oprócz instrukcji przypisania, nagłówków funkcji i wywołań funkcji. Dodatkowo być może zostaną zniesione niektóre obecne ograniczenia w zakresie korzystania z tej składni. Proponowana zmiana została przesunięta na wydanie po wersji 3.4, tuż przed ukazaniem się niniejszej książki, i pozostaje niepewna. W istocie dyskutowano o niej od 2008 roku. Zmiana ta nie będzie rozpatrywana do wydania Pythona 3.5 lub wersji późniejszej i być może nie pojawi się w ogóle. Szczegółowe informacje można znaleźć w dokumencie „What’s New”.

Instrukcja wyrażeniowa

wyrażenie

funkcja([wartość, nazwa=wartość, *nazwa, **nazwa...])

obiekt.metoda([wartość, nazwa=wartość, *nazwa, **nazwa...])

Dowolne wyrażenie może być instrukcją (np. występujące samodzielnie w wierszu). Z drugiej strony instrukcje nie mogą występować w żadnym innym kontekście wyrażenia (np. instrukcje przypisania nie zwracają wyników i nie mogą być zagnieżdżane).

Wyrażenia są powszechnie używane do wywoływania funkcji i metod, które nie zwracają wartości, oraz do wyświetlania w trybie interaktywnym. Instrukcje wyrażeniowe są także najpopularniejszym sposobem kodowania dla wyrażeń `yield` oraz wywołań wbudowanej funkcji `print()` z Pythona 3.X (choć w tej książce instrukcje te opisano osobno).

Składnia wywołań

W wywołaniach funkcji i metod aktualne argumenty wywołania są od siebie oddzielone przecinkami i zwykle są dopasowywane według pozycji do argumentów w nagłówkach `def` funkcji. W wywołaniach można opcjonalnie wymienić specyficzne nazwy argumentów, do których będą przekazywane wartości. W tym celu należy skorzystać ze składni argumentu kluczowego `nazwa=wartość`. Argumenty kluczowe są dopasowywane według nazwy zamiast pozycji.

Składnia dowolnych argumentów wywołania

W listach argumentów wywołania funkcji i metod można zastosować specjalną składnię w celu *rozpakowania* kolekcji do dowolnej liczby argumentów. Jeśli `pargs` i `kargs` to odpowiednio sekwencja (lub inny obiekt iterowalny) oraz słownik:

```
f(*pargs, **kargs)
```

to instrukcja o tym formacie wywoła funkcję `f` z argumentami *pozycyjnymi* z obiektu iterowalnego `pargs` oraz z argumentami *kluczowymi* ze słownika `kargs`. Na przykład:

```
>>> def f(a, b, c, d): print(a, b, c, d)
...
>>> f(*[1, 2], **dict(c=3, d=4))
1 2 3 4
```

Składnię tę dodano po to, by była symetryczna ze składnią argumentów nagłówka funkcji, na przykład `def f(*pargs, **kargs)`, która *pobiera*

niedopasowane argumenty. W wywołaniach elementy poprzedzone gwiazdkami są rozpakowywane do pojedynczych argumentów i mogą być łączone z innymi argumentami pozycyjnymi i kluczowymi zgodnie z regułami porządkowania (np. `g(1, 2, foo=3, bar=4, *pargs, **kargs)`).

W Pythonie 2.X podobny efekt można osiągnąć za pomocą wbudowanej funkcji `apply()`, która w Pythonie 3.X została usunięta:

```
apply(f, pargs, kargs)
```

Więcej informacji na temat składni wywołania można znaleźć w podrozdziale „Instrukcja `def`” (włącznie z tabelą 15.).

Instrukcja `print`

W Pythonie 3.X wyświetlanie tekstu do standardowego strumienia wyjściowego przyjęło postać wywołania wbudowanej funkcji. Zwykle jest ona kodowana jako instrukcja wyrażeniowa (w oddzielnym wierszu). Jej sygnatura wywołania jest następująca:

```
print([wartość [, wartość]*]  
      [, sep=łańcuch-znaków] [, end=łańcuch-znaków]  
      [, file=obiekt] [, flush=bool])
```

Każda *wartość* jest wyrażeniem generującym obiekt, dla którego ma być wyświetlony łańcuch znaków `str()`. To wywołanie konfiguruje się za pomocą trzech argumentów, które mogą być wyłącznie argumentami kluczowymi (jeśli argument będzie pominięty lub zostanie przekazana wartość `None`, przyjmie on wartość domyślną):

`sep`

Łańcuch znaków, który ma być umieszczony pomiędzy wartościami (domyślnie spacja: `' '`).

`end`

Łańcuch znaków do umieszczenia na końcu wyświetlanego tekstu (domyślnie znak nowego wiersza: `'\n'`).

`file`

Obiekt postaci pliku, do którego jest zapisywany tekst (domyślnie standardowe wyjście: `sys.stdout`).

`flush`

Wartość `true` lub `false` umożliwiająca włączenie lub wyłączenie wymuszonego opróżnienia strumienia wyjściowego (wprowadzone w Pythonie 3.3 — domyślnie `False`).

Aby wyłączyć separatory w postaci spacji oraz znaki wysuwu wiersza, można przekazać puste lub dostosowane do własnych potrzeb argumenty `sep` i `end`. W celu przekierowania wyjścia można przekazać nazwę pliku za pomocą argumentu `file` (patrz także podrozdział „Pliki”):

```
>>> print(2 ** 32, 'spam')
4294967296 spam

>>> print(2 ** 32, 'spam', sep='')
4294967296spam

>>> print(2 ** 32, 'spam', end=' '); print(1, 2, 3)
4294967296 spam 1 2 3

>>> print(2 ** 32, 'spam', sep='',
...       file=open('out', 'w'))
>>> open('out').read()
'4294967296spam\n'
```

Ponieważ domyślnie funkcja `print` po prostu wywołuje metodę `write()` obiektu, do którego w danym momencie odwołuje się urządzenie `sys.stdout`, to poniższa sekwencja jest równoważna wywołaniu `print(X)`:

```
import sys
sys.stdout.write(str(X) + '\n')
```

Aby przekierować polecenie wyświetlania tekstu do plików bądź obiektów klasy, należy albo przekazać dowolny obiekt z implementacją metody `write()` do argumentu kluczowego `file`, tak jak pokazano wcześniej, albo przypisać urządzenie `sys.stdout` do dowolnego takiego obiektu (patrz także podrozdział „Pliki”):

```
sys.stdout = open('log', 'a') # dowolny obiekt z metodą write()
print('Ostrzeżenie: spam!') # wywołanie jest kierowane do metody write() obiektu
```

Ponieważ do urządzenia wyjściowego `sys.stdout` można przypisać nową wartość, argument kluczowy `file` nie jest konieczny. Często jednak pozwala on na uniknięcie zarówno jawnych wywołań metody `write()`, jak i zapisywania i odtwarzania oryginalnej wartości `sys.stdout` wokół wywołania metody `print`, w przypadku gdy pierwotny strumień danych jest w dalszym ciągu wymagany. Więcej informacji na temat działania instrukcji `print()` w Pythonie 3.X można znaleźć w podrozdziale „Funkcje wbudowane”.

Instrukcja `print` w Pythonie 2.X

W Pythonie 2.X wyświetlanie jest instrukcją, a nie funkcją wbudowaną. Jest to instrukcja o następującej postaci:

```
print [wartość [, wartość]* [,]]
print >> plik [, wartość [, wartość]* [,]]
```

Instrukcja `print` z Pythona 2.X wyświetla drukowalne reprezentacje wartości w strumieniu `stdout` (używając bieżącego ustawienia `sys.stdout`) oraz dodaje spacje pomiędzy wartościami. Dodanie przecinka na końcu powoduje wyłączenie znaku wysuwu wiersza, który jest standardowo dodawany na końcu listy. Jest to równoważne użyciu klauzuli `end=' '` w Pythonie 3.X:

```
>>> print 2 ** 32, 'spam'
4294967296 spam

>>> print 2 ** 32, 'spam',; print 1, 2, 3
4294967296 spam 1 2 3
```

Instrukcja `print` z Pythona 2.X pozwala także na podanie nazwy pliku wyjściowego (lub podobnego mu obiektu), który będzie pełnił funkcję celu dla wyświetlanego tekstu zamiast strumienia `sys.stdout`.

```
fileobj = open('log', 'a')
print >> fileobj, "Ostrzeżenie: spam!"
```

Jeśli obiekt pliku ma wartość `None`, wykorzystywany jest strumień `sys.stdout`. Składnia Pythona 2.X `>>` jest równoważna użyciu argumentu kluczowego `file=F` z Pythona 3.X. W instrukcji `print` Pythona 2.X nie istnieje odpowiednik argumentu `sep=S`, chociaż wiersze mogą być sformatowane i wyświetlone jako pojedynczy element.

W instrukcji `print` Pythona 2.X można używać nawiasów okrągłych. W przypadku wielu elementów tworzą one krotki. Aby skorzystać z funkcji wyświetlania Pythona 3.X w Pythonie 2.X, należy uruchomić poniższe instrukcje (w interaktywnej sesji lub na początku skryptu) — z tego mechanizmu można skorzystać zarówno w *Pythonie 2.X* (w celu zachowania zgodności w przód z wersją 3.X), jak i w *Pythonie 3.X* (w celu zachowania wstecznej zgodności z Pythonem 2.X):

```
from __future__ import print_function
```

Instrukcja `if`

```
if warunek:
    grupa
[elif warunek:
    grupa]*
[else:
    grupa]
```

Instrukcja `if` pozwala na wybór jednego lub kilku działań (bloków instrukcji) i uruchamia grupę instrukcji powiązaną z pierwszym warun-

kiem `if` lub `elif`, który jest prawdziwy, albo wykonuje grupę `else`, jeśli wszystkie warunki `if` (`elif`) mają wartość `false`. Człony `elif` i `else` są opcjonalne.

Instrukcja `while`

```
while warunek:  
    grupa  
[else:  
    grupa]
```

Pętla `while` to instrukcja pętli ogólnego przeznaczenia, która wykonuje pierwszą grupę instrukcji, gdy warunek na początku instrukcji jest prawdziwy. Instrukcja uruchamia grupę `else`, w przypadku gdy nastąpi zakończenie działania pętli bez wykonania instrukcji `break`.

Instrukcja `for`

```
for cel in obiekt_iterowalny:  
    grupa  
[else:  
    grupa]
```

Pętla `for` realizuje iterację po sekwencji (bądź innym obiekcie iterowalnym). Przypisuje elementy obiektu iterowalnego do zmiennej `cel` i w każdej iteracji wykonuje pierwszą grupę instrukcji. Instrukcja `for` uruchamia opcjonalną grupę `else`, w przypadku gdy nastąpi zakończenie działania pętli bez wykonania instrukcji `break`. Zmienną `cel` instrukcji `for` może być dowolny obiekt, który może się znaleźć po lewej stronie instrukcji przypisania (np. `for (x, y) in lista_krotek`).

Od Pythona w wersji 2.2 instrukcja `for` najpierw próbuje uzyskać obiekt *iteratora* `I` za pomocą metody `iter(obiekt_iterowalny)`, a następnie wielokrotnie wywołuje metodę `I.__next__()` obiektu — do czasu wystąpienia wyjątku `StopIteration` (w Pythonie 2.X metoda `I.__next__()` ma nazwę `I.next()`). Jeśli nie można uzyskać żadnego obiektu iteratora (np. nie zdefiniowano metody `__iter__`), to instrukcja `for` działa poprzez wielokrotne indeksowanie obiektu `obiekt_iterowalny` z użyciem coraz większych wartości przesunięcia, aż zostanie zgłoszony wyjątek `IndexError`.

Iteracja w Pythonie odbywa się w wielu kontekstach, włącznie z instrukcjami pętli `for`, obiektami składanymi i instrukcjami `map()`. Więcej informacji na temat mechanizmów używanych przez pętlę `for` oraz w pozostałych kontekstach iteracji można znaleźć w podrozdziale „Protokół iteracji”.

Instrukcja pass

pass

Instrukcja-wypełniacz, która nie wykonuje żadnych działań. Używa się jej wtedy, gdy jest to syntaktycznie konieczne (np. w odniesieniu do namiastek funkcji). W Pythonie 3.X podobny efekt można uzyskać za pomocą wielokropka (...).

Instrukcja break

break

Powoduje natychmiastowe zakończenie najbliższej instrukcji pętli `while` lub `for` z pominięciem powiązanych z nimi grup `else` (o ile takie istnieją). Wskazówka: w celu wyjścia z wielopoziomowych pętli można skorzystać z instrukcji `raise` i `try`.

Instrukcja continue

continue

Powoduje natychmiastowe przejście na początek najbliższej instrukcji pętli `while` lub `for` i wznowienie wykonywania grupy instrukcji w pętli.

Instrukcja del

```
del nazwa
del nazwa[i]
del nazwa[i:j:k]
del nazwa.atrybut
```

Instrukcja `del` usuwa nazwy, elementy, wycinki i atrybuty, a także powiązania. W pierwszej postaci *nazwa* jest dosłowną nazwą zmiennej. W ostatnich trzech formach instrukcji *nazwa* może być dowolnym wyrażeniem (z użyciem nawiasów, jeśli są potrzebne do określenia priorytetu). Na przykład: `del a.b()[1].c.d`.

Instrukcja ta służy przede wszystkim do struktur danych, a nie zarządzania pamięcią. Usuwa również referencje do obiektów wywoływanych wcześniej. Może spowodować ich usunięcie przez mechanizm odśmiecania, w przypadku gdy nie będzie do nich odwołania w innych miejscach. Proces odśmiecania zachodzi jednak automatycznie i nie musi być wymuszany za pomocą instrukcji `del`.

A

ASCII, *Patrz:* typ tekstowy ASCII
asercja, 104

B

backport, *Patrz:* poprawka
bajt, 22, 23
baza danych relacyjna, 232
biblioteka
 Tk, 217, 220
 tkinter, 217

błąd
 interpretera, 168
 sekwencji, 164
 składni, 167, 171
 systemowy, 164, 169
bufor, 158

C

cel
 operatora %, 33, 34
 zagnieżdżanie, 35
 zastępowanie, 33, 35, 36
ciąg
 pusty, 30
 tekstowy, 39, 46
 znaków, 22, 23, 30, 138, 159
 '`\t\n\r\v\f`', 185
 '`01234567`', 185
 '`0123456789`', 184
 '`0123456789abcdefABCDEF`', 185
 '`abcdefghijklmnopqrstuvwxyz`',
 184

'`ABCDEFGHIJKLMNQRST
UVWXYZ`', 184
formatowanie, 32, 33, 34
internowanie, 178
przestankowych, 185

D

debugger, 182
decimal, *Patrz:* liczba dziesiętna
dekodowanie Unicode, 153
dekorator, 88, 98, 110, 150
 @staticmethod, 152
depth first left right, *Patrz:* DFLR
deskryptor, 131
 klas, 192
 plików, 192
 pliku, 191, 193, 200
destruktor, 120
DFLR, 114
diament, 113, 115
diamond pattern, *Patrz:* diament
dictionary comprehension, *Patrz:*
 słownik składany
domknięcie, 109, 110
dowiązanie symboliczne, 202
dziedziczenie, 111, 114, 163, 192
 nowego stylu, 115, 116, 117
 wielokrotne, 113

E

egzemplarz
 atrybut, 117
 tworzenie, 119
element widok, 57

F

- fraction, *Patrz:* ułamek
- funkcja, *Patrz też:* instrukcja, moduł
 - `__import__`, 141
 - abs, 135
 - adnotacja, 86
 - all, 135
 - anonimowa, 18
 - any, 135
 - apply, 157
 - argument, 85
 - domyślny, 87
 - kluczowy, 86
 - ascii, 136, 157
 - basestring, 158
 - bin, 136
 - bool, 121, 136
 - buffer, 158
 - bytearray, 136
 - bytes, 121, 136
 - callable, 137
 - chr, 137
 - classmethod, 89, 98, 137
 - cmp, 158
 - coerce, 158
 - compile, 106, 137, 204, 206
 - complex, 138
 - deklaracja opisująca, *Patrz:*
 - dekorator
 - delattr, 138
 - dict, 57, 138
 - dir, 125, 138
 - divmod, 138
 - enumerate, 138
 - eval, 106, 139, 150, 167
 - exec, 106, 139, 157, 167
 - execfile, 106, 158
 - execv, 198, 199
 - execve, 198
 - execvp, 198
 - fabryka, 88
 - file, 159
 - filter, 54, 139
 - float, 140
 - format, 121, 140
 - frozenset, 66, 140
 - generators, 18, 89, 90
 - getattr, 140
 - getswitchinterval, 177, 178
 - globals, 140
 - hasattr, 140
 - hash, 140
 - help, 72, 141
 - hex, 134, 141
 - id, 141
 - imp.reload, 93, 160
 - importlib.import_module, 93
 - input, 141, 159, 160, 165
 - int, 142
 - intern, 159
 - isinstance, 142
 - issubclass, 142
 - iter, 54, 142
 - len, 125, 143
 - list, 143
 - locals, 143
 - lokalna, 85
 - long, 159
 - map, 53, 54, 143
 - max, 144
 - memoryview, 144, 157
 - metoda klasy, 75, 98
 - min, 144
 - namiaszka, 84
 - next, 54, 55, 126, 142, 144, 167
 - normpath, 201
 - object, 145
 - oct, 145
 - okalająca, 108, 109
 - open, 61, 145, 149, 159, 162, 192
 - operator.index, 130
 - ord, 149, 160
 - os.popen, 189
 - os._exit, 199
 - os.abort, 198
 - os.access, 197
 - os.altsep, 187
 - os.chdir, 191, 194
 - os.chmod, 194
 - os.chown, 194
 - os.close, 192
 - os.defpath, 188
 - os.devnull, 188

os.dup, 192
os.environ, 190
os.execl, 198
os.execlp, 198
os.execv, 199
os.execve, 198
os.execvp, 199
os.execvpe, 199
os.extsep, 187
os.fdopen, 192
os.fork, 199
os.fstat, 192
os.ftruncate, 192
os.getcwd, 191, 194
os.getenv, 190
os.geteuid, 199
os.getpid, 199
os.getppid, 199
os.getuid, 199
os.isatty, 193
os.kill, 200
os.linesep, 188
os.link, 194
os.listdir, 194
os.lseek, 193
os.lstat, 195
os.makedirs, 195
os.mkdir, 195
os.mkfifo, 195, 200
os.nice, 200
os.open, 193, 194
os.path.abspath, 201
os.path.basename, 201
os.path.commonprefix, 201
os.path.dirname, 202
os.path.exists, 202
os.path.expanduser, 202
os.path.expandvars, 202
os.path.getatime, 202
os.path.getmtime, 202
os.path.getsize, 202
os.path.isabs, 202
os.path.isdir, 202
os.path.isfile, 202
os.path.islink, 202
os.path.ismount, 202
os.path.join, 203
os.path.normcase, 203
os.path.normpath, 203
os.path.realpath, 203
os.path.samefile, 203
os.path.sameopenfile, 203
os.path.samestat, 203
os.path.split, 203
os.path.splitdrive, 203
os.pathsep, 188
os.pipe, 193, 200
os.plock, 200
os.putenv, 190
os.read, 193
os.readlink, 196
os.remove, 196
os.removedirs, 196
os.rename, 196
os.renames, 196
os.rmdir, 196
os.spawnv, 200
os.spawnve, 200
os.startfile, 189
os.stat, 196
os.strerror, 191
os.symlink, 196
os.system, 188
os.times, 191
os.umask, 191
os.uname, 191
os.unlink, 196
os.utime, 197
os.wait, 200
os.waitpid, 201
os.walk, 197
os.write, 193
pow, 149
print, 149, 150, 157
property, 89, 113, 150
range, 150, 161
raw_input, 160, 165
re.compile, 106, 137, 204, 206
reduce, 160
reload, 93, 160
repr, 120, 136, 150, 157
reversed, 151
round, 151

funkcja, *Patrz też*: instrukcja, moduł
równoległa, 231
set, 66, 151
setattr, 151
setcheckinterval, 232
setswitchinterval, 232
slice, 151
sorted, 152
staticmethod, 89, 98, 152
str, 121, 150, 152, 153, 161
string.capitalize, 184
string.Formatter, 184
string.maketrans, 184
string.Template, 184
sum, 153
super, 8, 115, 153, 154
sys.__stderr__, 182
sys.__stdin__, 182
sys.__stdout__, 182
sys._getframe, 177
sys.argv, 175
sys.builtin_module_names, 175
sys.byteorder, 175
sys.copyright, 176
sys.displayhook, 176
sys.dont_write_bytecode, 176
sys.exc_info, 176
sys.excepthook, 176
sys.exec_prefix, 176
sys.executable, 177
sys.exit, 177
sys.flags, 177
sys.float_info, 177
sys.getcheckinterval, 177
sys.getdefaultencoding, 177
sys.getfilesystemencoding, 177
sys.getrecursionlimit, 178
sys.getrefcount, 178
sys.getsizeof, 178
sys.getswitchinterval, 177, 178
sys.getwindowsversion, 178
sys.hexversion, 178
sys.implementation, 178
sys.int_info, 178
sys.intern, 178
sys.last_traceback, 179
sys.last_type, 179
sys.last_value, 179
sys.maxsize, 179
sys.maxunicode, 179
sys.modules, 179
sys.path, 179
sys.platform, 180
sys.prefix, 180
sys.ps1, 180
sys.ps2, 180
sys.setcheckinterval, 180
sys.setdefaultencoding, 181
sys.setprofile, 181
sys.setrecursionlimit, 181
sys.setswitchinterval, 181
sys.settrace, 182
sys.stderr, 182
sys.stdin, 182
sys.stdout, 182
sys.sys.version, 183
sys.thread_info, 183
sys.tracebacklimit, 183
sys.version_info, 183
sys.winver, 183
ślądu systemu, 182
tuple, 155
tworzenie, 85
type, 68, 113, 155
unichr, 160
unicode, 161
vars, 156
warnings.warn, 171
wbudowana, 108
nazwa, 74
współbieżna, 231
xrange, 161
zagnieżdżona, 91, 109
zip, 156

G

garbage collector, *Patrz*: odśmianie
geometry manager, *Patrz*: menedżer
geometrii
gniazdo, 113
GUI, 15, 217

H

hash, *Patrz:* skrót

I

indeksowanie, 19, 25

instrukcja, 71, 75, *Patrz też:* funkcja

assert, 104, 165

blok, 71

break, 84, 100

class, 74, 97, 98, 111

continue, 84, 100

def, 85, 87, 106, 111

del, 84, 138

exec, 106, 141

for, 53, 54, 83, 84, 90, 152

from, 95, 96, 97, 141, 165

global, 91

if, 82

import, 92, 93, 95, 141, 165

pakiet, 93

nonlocal, 91, 106, 109

pass, 84

print, 80, 81, 82, 106

przypisania, 75, 98, 107, 111

krotek, 76

sekwencji, 76, 77, 78

wielocelowego, 76

raise, 84, 100, 102, 103, 106

return, 89, 91, 100

składnia, 70

try, 84, 99, 100, 101, 103, 104, 106,
109, 164, 168

klauzula, 100, 101

while, 83, 84

with, 104, 105, 106, 131

wyraźniowa, 79

yield, 20, 89, 90

interfejs

API, 232, 233

Berkeley DB, 211

dopasowywania wyrażeń
regularnych, 204

menedżera geometrii, 218

modułu, 234

obiektów zwracanych, 148

PyDoc, 72

usług systemu operacyjnego, 185
internet, 222

interpreter, 175, 178

interaktywny, 74

ścieżka dostępu, 177

wątek, 181

wersja, 183

inwersja, *Patrz:* operacja bitowa NOT

iteracja, 55, 138

kontekst, 54

protokół, 54, 55, 89, 126

widoku, 57

iterator, 54, 83, 89

K

klasa, 69, 97, 98, 110

atrybut, 98, 111, 117

prywatny, 112

bazowa, *Patrz:* klasa nadrzędna

egzemplarz, 102, 111

Exception, 103

gniazdo, *Patrz:* gniazdo

klasyczna, 113, 114

metoda, 111

nadrzędna, 97

nazwa, 74

nowego stylu, 103, 111, 113, 114,
116, 117, 150

Pickler, 216

String, 39

Unpickler, 216

wyjątków, *Patrz:* wyjątek klasa

klasa-deskryptor, 130

klasa-właściciel, 130

klauzula

as, 92, 96, 101, 104

else, 100

except, 100, 101, 102, 109

finally, 100, 102, 104, 168

from, 90, 102

klucz, 21, 23, 57, 126, 144, 212

krotki, 57

mapowania, 166

rejestr, 183

skrót, 121

klucz
słownika, 178
widok, 57
wyszukiwania, 178
klucz-wartość, 57
kod
bajtowy, 92
blok
wejściowy, 104
zagnieżdżony, 104
zamykający, 104
wielokrotne wykorzystanie, 98
źródłowy, 92
wcięcia, 166
kolejka FIFO, 195
kolekcja, 125
długość, 143
komentarz, 72
komunikat, 104
konstruktor, 120
krotka, 19, 22, 23, 33, 60, 61, 134, 155
klucz, 57
kodowanie, 33
przypisanie, 76
zagnieżdżona, 33
kursor, 235

L

launcher, 15, 17
liczba, 26
całkowita, 142, 159
dziesiętna, 26, 28
ułamkowa, *Patrz:* ułamek
zespolona, 26, 27, 138
zmiennoprzecinkowa, 27, 140,
142, 151
lista, 22, 23, 50, 156, 228
literał, 50
obiekt
wstawianie, 52
wyszukiwanie, 51
pozycja, 50
składana, 52, 53, 54, 55, 56, 110
z zagnieżdżonymi pętlami for, 53
sortowanie, 51
wyrażeń, 52

literał, 18, 26
b'ccc', 49
bytes, 29
'ccc', 46
listy, *Patrz:* lista literał
łańcuchowy, 31
słownikowy, *Patrz:* słownik literał
tworzenie, 30
u'ccc', 47, 49
znakowy, 31

Ł

łańcuch
wyjątków, *Patrz:* wyjątek łańcuch
znaków, *Patrz:* ciąg znaków

M

mapa, 110, 125
indeksowanie, 126
menedżer
geometrii, 218
kontekstu, 104, 131
protokół, 105
zagnieżdżony, 105
metaklasa, 99, 156
Method Resolution Order, *Patrz:* MRO
metoda, 21
__abs__, 24, 130
__add__, 119, 127
__and__, 128
__bool__, 121, 125, 132, 133
__bytes__, 121, 132
__call__, 99, 122, 137, 221
__cmp__, 22, 123, 124, 133, 158
__coerce__, 134
__complex__, 25, 130
__contains__, 22, 125
__del__, 120, 234, 235
__delattr__, 123
__delete__, 113, 131
__delitem__, 23, 24, 41, 127, 133
__delslice__, 23, 134
__dict__, 123
__dir__, 125, 132
__div__, 127, 134

__divmod__, 128
 __enter__, 105, 131
 __eq__, 121, 123, 124, 133
 __exit__, 105, 106, 132
 __float__, 25, 130
 __floordiv__, 128
 __format__, 121
 __ge__, 123, 124, 133
 __get__, 113, 116, 117, 131
 __getattr__, 112, 113, 114, 116, 118, 122, 123, 125, 132
 __getattribute__, 113, 116
 __getitem__, 22, 23, 54, 118, 119, 125, 126, 127, 133, 151
 __getslice__, 23, 133
 __gt__, 123
 __hash__, 57, 121, 140
 __hex__, 130, 133, 134, 141
 __iadd__, 41, 77, 129
 __iand__, 129
 __ifloordiv__, 129
 __ilshift__, 129
 __imod__, 129
 __import__, 93
 __imul__, 41, 129
 __index__, 130, 133, 134, 136, 141, 145
 __init__, 90, 93, 94, 99, 103, 119, 120, 156
 __instancecheck__, 125, 132
 __int__, 24, 130
 __invert__, 24, 130
 __ior__, 129
 __ipow__, 129
 __irshift__, 129
 __isub__, 129
 __iter__, 22, 23, 54, 55, 83, 90, 125, 126, 143
 __itruediv__, 129, 134
 __ixor__, 129
 __le__, 123
 __len__, 23, 24, 121, 125, 126, 127, 132, 133, 151
 __long__, 24, 134
 __lshift__, 128
 __lt__, 22, 123, 124, 133
 __metaclass__, 99, 135
 __mod__, 128
 __mul__, 127
 __ne__, 123, 124
 __neg__, 24, 130
 __new__, 99, 119, 120, 156
 __next__, 54, 90, 126, 132, 142, 144
 __nonzero__, 121, 125, 132, 133
 __oct__, 130, 133, 134, 145
 __or__, 128
 __pos__, 24, 130
 __pow__, 128
 __radd__, 128, 129
 __rand__, 128
 __rcmp__, 133
 __rdivmod__, 128
 __repr__, 120, 121
 __reversed__, 127, 151
 __rfloordiv__, 128
 __rlshift__, 128
 __rmod__, 41, 128
 __rmul__, 128
 __ror__, 128
 __round__, 130, 132
 __rpow__, 128
 __rrshift__, 128
 __rshift__, 128
 __rsub__, 128
 __rtruediv__, 128, 134
 __rxor__, 128
 __set__, 113, 117, 131
 __setattr__, 112, 114, 117, 118, 122, 123
 __setitem__, 23, 41, 126, 133
 __setslice__, 23, 134
 __slots__, 113, 124, 125, 132
 __str__, 120, 121, 135, 153
 __sub__, 127
 __subclasscheck__, 125, 132
 __truediv__, 127, 132
 __unicode__, 135, 161
 __X__, 98
 __xor__, 128
 bytes.translate, 184
 close, 90, 165
 D.clear, 58
 D.copy, 58
 D.get, 59

- metoda
 - D.has_key, 59
 - D.items, 58
 - D.iteritems, 59
 - D.keys, 58
 - D.pop, 59
 - D.popitem, 59
 - D.setdefault, 59
 - D.update, 59
 - D.values, 58
 - D.viewitems, 59
- delim.join, 46
- destruktor, 120
- dict.fromkeys, 59
- działania dwuargumentowego,
 - 128, 129
- działania na zbiorach, 67
- file, 61
 - file.close, 64
 - file.closed, 65
 - file.fileno, 64
 - file.flush, 64
 - file.isatty, 64
 - file.mode, 65
 - file.name, 65
 - file.seek, 64
 - file.tell, 64
 - file.truncate, 64
- for line in infile, 63
- formatująca, 34, 38
 - składnia, 36
- generatora, 90
- has_key, 56, 57
- I.__class__, 113
- I.__next__, 54, 55, 83, 142, 143, 167
- I.next, 54, 55, 143
- infile, 62
 - infile.read, 62
 - infile.readline, 63
 - infile.readlines, 63
- io.BytesIO, 62, 148
- io.StringIO, 62, 148
- isinstance, 125
- issubclass, 125
- items, 56, 57
- iter, 83
- iteratora, 56
- iteritems, 56
- iterkeys, 56
- intervalvalues, 56
- keys, 56, 57
- klasy, 137
 - abstrakcyjnej, 166
 - String, 39
- konstruktor, 120
- L.append, 51
- L.clear, 52
- L.copy, 52
- L.count, 52
- L.extend, 51
- L.index, 51
- L.insert, 52
- L.pop, 52
- L.remove, 52
- L.reverse, 51
- L.sort, 51, 52, 57
- obiektgniazda.makefile, 62
- obiekту string, 183
- open, 61, 62, 63, 66, 212
- outfile, 63
 - outfile.write, 63
 - outfile.writelines, 64
- prawostronna, 128
- przeciążająca operator, 98
- przeciążania klas, 118
- przeciążania operatorów, 68, 118, 132
- przypisania z aktualizacją, 129
- reversed, 51
- s.capitalize, 184
- S.capitalize, 44
- S.casefold, 44
- S.center, 45
- S.count, 43
- S.endswith, 43
- S.expandtabs, 44
- S.find, 42
- S.format, 44
- S.index, 42
- s.join, 184
- S.join, 43
- S.ljust, 44
- S.lower, 44
- S.lstrip, 44
- S.replace, 43

- S.rfind, 42
- S.rindex, 42
- S.rjust, 45
- S.rstrip, 44
- s.split, 184
- S.split, 43
- S.splitlines, 43
- S.startswith, 43
- S.strip, 44
- S.swapcase, 44
- S.title, 45
- S.translate, 45
- S.upper, 44
- S.zfill, 45
- send, 90
- separatora, 46
- statyczna, 152
- str.format, 38, 41, 121, 140, 184
- StringIO.stringIO, 148
- sys.exit, 168
- T.count, 61
- T.index, 61
- throw, 90
- values, 56, 57
- viewitems, 57
- viewkeys, 57
- viewvalues, 57
- write, 81
- X.__iter__, 54
- X.__round__, 151
- xreadlines, 66
- moduł, 94, *Patrz też:* funkcja
 - __builtin__, 135, 157
 - anydbm, 211
 - atrybut prywatny, 112
 - biblioteczny, 174, 175
 - builtins, 135
 - datetime, 228
 - dbm, 61, 210, 211, 212, 213
 - docelowy, 92
 - dopasowywania wzorców tekstowych re, 41
 - enum, 230
 - glob, 186
 - imp, 160
 - importowanie, 92, 95, 96
 - interfejs, 234

- internetowy, 222
- json, 228
- kwalifikacja, 92
- math, 225
- multiprocessing, 186
- nazwa, 75
- obiekt, *Patrz:* obiekt modułu
- obsługi wątków, 231
- os, 62, 185
- os.path, 185, 201
- pickle, 61, 210, 211, 214, 215
- queue, 186, 232
- re, 204
- shelve, 61, 211, 212
- signal, 186
- socket, 186
- string, 183, 184
- struct, 230
- subprocess, 186, 229
- sys, 159, 175, 186
- tempfile, 186, 194
- threading, 186, 232
- time, 225
- timeit, 227
- tkinter, 217, 218, 219, 220
- utrwalania obiektów, 210
- weakref, 167
- MRO, 8, 114, 115, 154

N

- nazwa
 - kwalifikowana, 107
 - niekwalifikowana, 107, 108
 - zasięg, 108

O

- obiekt, 145
 - bufora, *Patrz:* bufor
 - dopasowania, 207
 - egzemplarza, 98
 - file, 192
 - generatora, 55, 89
 - integer, 130
 - iteratora, 142

obiekt
iterowalny, 54, 66, 67, 83, 89, 143,
151, 156, 160
klasy, 97, 111
kursora, *Patrz*: kursor
modułu, 92
NotImplemented, 124, 127
pliku, 61
ramki, 177
referencja, 75
rozmiar, 178
serializacja, 214
shelve, 212
składany, 109
str, 29, 39
string, 39, 183
utrwalanie, 210
widoku, 57
widoku pamięci, 144
wyrażeń regularnych, 206
object-oriented programming,
Patrz: programowanie obiektowe
odśmiecianie, 18, 120
odzworowanie, 21, 23, 24
OOP, *Patrz*: programowanie obiektowe
operacja
bitowa
AND, 18
NOT, 19
OR, 18
XOR, 18
katalogowa, 170
logiczna, 21, 22
AND, 18
negacja, 18
OR, 18
matematyczna, 225
plikowa, 170
porównania, 22
wejściowa, 146
wycinania, 126
wyjściowa, 147
zmiennoprzecinkowa, 165, 177
operator, 19, 127
%, 33, 34
dwuargumentowy, 128
porównania, 20, 123

równości, 18
trójargumentowy, 18
wyraźeniowy, 18, 19
ostrzeżenie, 170, 171
blokowanie, 171

P

pakiet, 93, 94, 96
pamięć
globalna, 231
wyczerpywanie się, 166
parsowanie, 230
plik, 61
.pth, 92
.pyc, 93
__init__.py, 93, 94
buforowanie, 66
dbm, 213
deskryptor, 148, 191, 193, 200
JSON, 228
koniec, 165
modyfikacja, 202
otwieranie, 145, 162, 212
tryb, 146
py.exe, 15
pyw.exe, 15
rozmiar, 202
shelve, 213
ścieżka, 194, 195, 196, 197, 201
tryb otwarcia, 65
tworzenie, 212
wejściowy, 62
wykonywalny, 199
zamykanie, 192
polecenie
help, 40
powłoki, 185, 188
python -m pydoc -b, 72
poprawka, 57
print, 121
proces
potomny, 199, 200, 201
tworzenie, 198, 199
uprzejmość, 200

- programowanie
 - funkcyjne, 110
 - obiektywne, 110
- protokół
 - iteracji, *Patrz:* iteracja protokół
 - menedżera kontekstu, *Patrz:* menedżer kontekstu protokół
 - sekwencji, *Patrz:* sekwencja protokół
- przepełnienie, 167
- przerwanie, 166
- przeźrzenie nazw, 91, 94, 107, 111, 138, 160
 - obiekty, 107
- punkt montowania, 202
- Python
 - idiomy, 236
 - implementacja, 7
 - launcher, *Patrz:* launcher
 - rdzeń języka, 236
 - środowisko, 238
 - użytkowanie, 239, 240
 - wersja, 15, 178, 183

S

- sekwencja, 21, 22, 23, 25, 125
 - działanie, 32
 - indeksowanie, 126
 - modyfikowalna, *Patrz:* sekwencja mutowalna
 - mutowalna, 23, 29, 46, 50
 - niemutowalna, 29, 46, 60
 - protokół, 126
 - przypisanie, 77
- serializacja, 214
- set comprehension, *Patrz:* zbiór składany
- skrót, 121
- slicing, *Patrz:* rozcinanie
- slot, 124, *Patrz:* gniazdo
- słownik, 23, 24, 56, 138, 140, 156, 213, 228
 - literał, 57
 - modułów, 179
 - poprawka, *Patrz:* poprawka porównywanie, 57
 - składany, 56, 57

- słowo zarezerwowane, 73, 74, 118
- SQL, 232
- SQLite, 233
- stała
 - ascii_letters, 184
 - ascii_lowercase, 184
 - ascii_uppercase, 184
 - digits, 184
 - hexdigits, 185
 - octdigits, 185
 - printable, 185
 - punctuation, 185
 - tekstowa, 31
 - whitespace, 185
- stos, 168, 177, 178, 181
- strumień, 229
 - standardowy, 182
- symbol zachęty, 180
- synonim, 92, 96
- system, 180
 - operacyjny, 185
 - plików, 212

Ś

- ścieżka, 201
 - nazwa, 194, 195, 196, 197

T

- tabela translacji, 184
- tablica bajtowa, 22, 23
- target, *Patrz:* cel
- test
 - diagnostyczny, 104
 - is*, 45
 - zawartości, 45
- tryb
 - otwarcia plików, *Patrz:* plik tryb otwarcia
 - tekstowy Unicode, 63
- typ, 113
 - Boolean, 21, 69
 - Ellipsis, 68
 - konwersja, 69
 - liczbowy, 24, 25, 26, 27
 - logiczny, 68

typ

- łańcuchowy, 29, 32
 - bytearray, 41, 46, 48, 49
 - bytes, 41, 46, 48
 - str, 29, 32, 41, 46
- model dynamiczny, 17
- None, 68
- NotImplemented, 68
- numeryczny, 21
- obiektowy, 17
- odpowiadający jednostkom programu, 69
- set, 28
- tekstowy
 - ASCII, 48, 136, 149
 - Unicode, 41, 46, 47, 48, 49, 135, 149, 153, 160, 161, 179
- wbudowany, 127
- wyliczeniowy, 230

U

- ułamek, 26, 28
- Unicode, *Patrz:* typ tekstowy Unicode
- uprawnienia dostępu, 170

W

- wartość widok, 57
- wątek, 183, 231
- widget, 217, 218, 219, 220
- widok
 - elementu, 57
 - klucza, *Patrz:* klucz widok
 - wartości, 57
- wiersz poleceń, 175, 177, 186, 229
 - flaga -O, 104
 - format, 9, 12
 - opcja, 9, 12, 14
 - specyfikacja programu, 11
- wycinanie, 126
- wycinek, 151, 166
 - prosty, 25
 - przypisanie, 26
 - rozszerzony, 25

wyjątek, 103

- ArithmeticError, 164
- AssertionError, 104, 165
- AttributeError, 165
- BaseException, 163
- BlockingIOError, 169
- BrokenPipeError, 169
- BufferError, 164
- BytesWarning, 171
- ChildProcessError, 169
- ConnectionAbortedError, 169, 170
- ConnectionError, 169
- ConnectionRefusedError, 169, 170
- ConnectionResetError, 170
- DeprecationWarning, 171
- EnvironmentError, 172
- EOFError, 165
- Exception, 164, 173
- FileExistsError, 170
- FileNotFoundError, 170
- FloatingPointError, 164, 165
- FutureWarning, 171
- GeneratorExit, 90, 164, 165
- ImportError, 165
- ImportWarning, 171
- IndentationError, 166
- IndexError, 54, 166
- InterruptedError, 170
- IOError, 145, 162, 172
- IsADirectoryError, 170
- KeyboardInterrupt, 164, 166
- KeyError, 166
- klasa, 103
 - bazowa, 163
- LookupError, 164
- łańcuch, 102
- MemoryError, 166
- NameError, 165, 166
- nazwa, 74
- nieobsłużony, 183
- nieprzechwycony, 179
- NotImplemented, 166
- NotImplementedError, 166
- OSError, 164, 172, 173
- OverflowError, 164, 167
- PendingDeprecationWarning, 171

- PermissionError, 170
- ProcessLookupError, 170
- przechwytywanie, 100
- ReferenceError, 167
- ResourceWarning, 171
- RuntimeError, 165, 167
- RuntimeWarning, 171
- StopIteration, 54, 91, 126, 142, 145, 167
- SyntaxError, 165, 167
- SyntaxWarning, 171
- SystemError, 168
- SystemExit, 164, 168
- TabError, 168
- TimeoutError, 170
- TypeError, 168
- UnboundLocalError, 168
- UnicodeDecodeError, 169
- UnicodeEncodeError, 169
- UnicodeError, 168
- UnicodeTranslateError, 169
- UnicodeWarning, 171
- UserWarning, 171
- ValueError, 165, 169
- VMSError, 173
- Warning, 170
- wbudowany, 163
- WindowsError, 173
- ZeroDivisionError, 164, 169
- zgłaszanie, 100, 102
- wyrażenie
 - formatujące łańcuch znaków, 33, 34, 38
 - generatorowe, 54, 55
 - in, 57
 - jako instrukcja, 79
 - lambda, 86, 87, 110
 - listowe, *Patrz:* lista składana
 - regularne, 204, 206, 207, 208, 211
 - składnia, 209, 210
 - rozcinające, 20
 - yield, 89
- wywołanie zwrotne, 182
- wzorzec
 - diamentu, 113, 115
 - wyrażeń regularnych, 171, 204

Z

- zakres, 166
- zasięg
 - funkcji wbudowanych, 108
 - globalny, 108
 - leksykalny, 107
 - lokalny, 108
 - nazw niekwalifikowanych, 108
 - zagnieżdżony statycznie, 109
- zbiór, 66
 - działania, 67
 - składany, 56
 - zamrożony, 140
- zmienna, 17, 72
 - globalna, 91
 - lokalna, 107, 143
 - nazwa, 17, 72, 73, 74
 - operacyjna, 13
 - pętli wyrażeń generatorowych, 55
 - PYTHONCASEOK, 14
 - PYTHONDEBUG, 15
 - PYTHONDONTWRITEBYTECODE, 15
 - PYTHONFAULTHANDLER, 14
 - PYTHONHASHSEED, 14
 - PYTHONHOME, 14
 - PYTHONINSPECT, 15
 - PYTHONIOENCODING, 14
 - PYTHONNOUSERSITE, 15
 - PYTHONOPTIMIZE, 15
 - PYTHONPATH, 13, 92, 179
 - PYTHONSTARTUP, 14
 - PYTHONUNBUFFERED, 15
 - PYTHONVERBOSE, 15
 - PYTHONWARNINGS, 15
 - sys.path, 92
 - środowiska powłoki, 198
 - środowiskowa, 13
- znak, 29
 - ",, 71
 - #, 72
 - \$, 73
 - %, 33, 34, 36
 - %=, 129
 - &=, 129
 - *, 33, 78, 86

znak

******, 78

****=**, 129

***=**, 129

„, 36

..., 68

/, 201

//=, 129

/=, 129

;, 72

?, 73

@, 88

[.], 52

^=, 129

_, 74

__, 74, 118

{}, 56, *Patrz:* znak nawias klamrowy

|=, 129

~, 202

+=, 129

<<=, 129

-=, 129

>>=, 129

apostrof, 30

cudzysłów, 30

desygatora, 30

nawias klamrowy, 34

oddzielający

komponenty ścieżki

wyszukiwania, 188

nazwę pliku od rozszerzenia,


187

przestankowy, 185

spacja biała, 42, 72

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 

Najlepsze rozwiązania typowych problemów!

Język Python błyskawicznie zdobył uznanie programistów na całym świecie. Sprawdza się doskonale w pisaniu skryptów oraz narzędzi, w dużym projekcie także nie zawiedzie oczekiwani. Python korzysta z automatycznego zarządzania pamięcią oraz umożliwia obiektowe i funkcyjne podejście do tworzonego programu. Liczna społeczność skupiona wokół tego języka na bieżąco wymienia się wiedzą na temat jego praktycznych zastosowań. Dzięki temu uzyskasz odpowiedzi na trapiące Cię pytania.

Przyda Ci się także sprawdzone źródło informacji. Ten przewodnik należy do serii *Leksykon kieszonkowy* i charakteryzuje się niezwykle zwięzłym, przejrzystym układem najważniejszych treści oraz poręczną formą. Znajdziesz tu szczegółowe informacje na temat typów wbudowanych, wyjątków, programowania obiektowego oraz przetwarzania nazw i reguł zasięgu. To wydanie zostało zaktualizowane o mnóstwo nowości, takich jak wykorzystanie Python Launcher w systemie Windows czy formalne reguły dziedziczenia. To doskonałe źródło wiedzy na temat języka Python!

Mark Lutz

– znany i ceniony na całym świecie instruktor programowania w Pythonie.

Dzięki tej książce:

- poznasz podstawy Pythona
- zapoznasz się z zasadami programowania w tym języku
- poznasz typy wbudowane
- wykorzystasz standardowe moduły

Helion 

 helion.pl

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

Sprawdź nasze szkolenia!

SZKOLENIA



AKADEMIA IT & BUSINESS

WWW.SZKOLENIA.HELION.PL

KOD KORZYŚCI
Sięgnij po więcej! ▶



ISBN 978-83-283-6035-8

