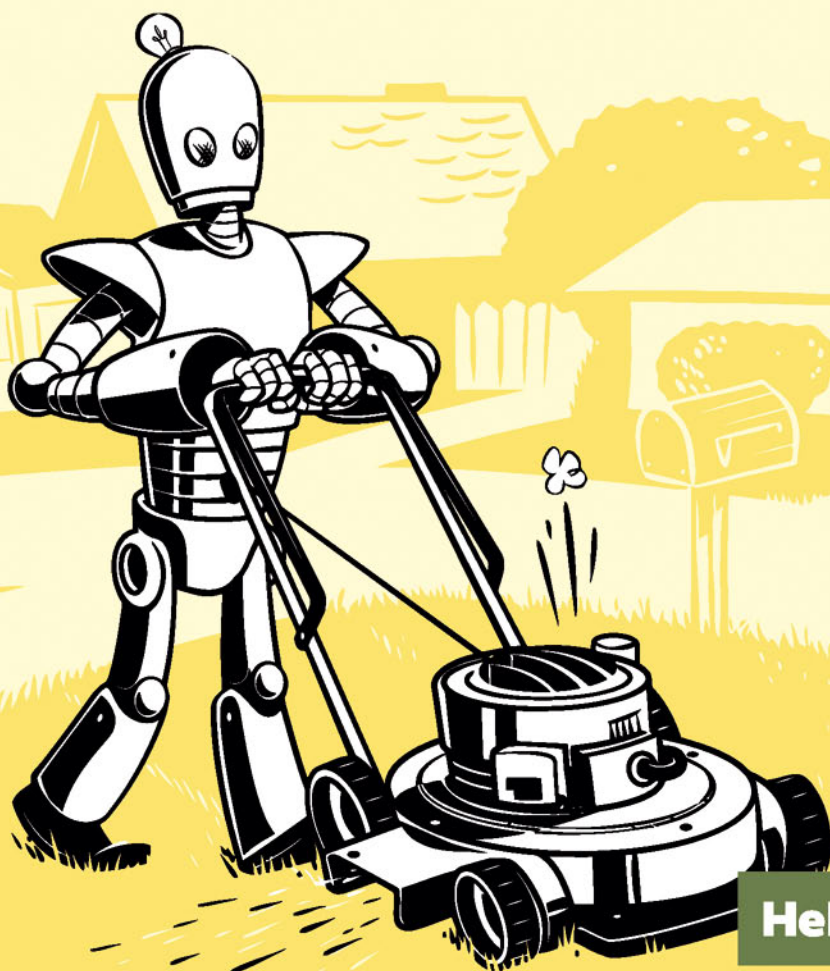


AUTOMATYZACJA NUDNYCH ZADAŃ Z PYTHONEM

NAUKA PROGRAMOWANIA

ALBERT SWEIGART



Helion 

Tytuł oryginału: Automate the Boring Stuff with Python: Practical Programming for Total Beginners

Tłumaczenie: Robert Górczyński

ISBN: 978-83-283-3260-7

Copyright © 2015 by Al Sweigart.

Title of English-language original: Automate the Boring Stuff with Python,
ISBN: 978-1-59327-599-0, published by No Starch Press.

Polish-language edition copyright © 2017 by Helion S.A. 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 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)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
<ftp://ftp.helion.pl/przyklady/autopy.zip>

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
<http://helion.pl/user/opinie/autopy>
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

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

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

Spis treści

| | |
|--|-----------|
| O AUTORZE | 21 |
| O RECENZENCIE TECHNICZNYM | 21 |
| PODZIĘKOWANIA | 23 |
| WPROWADZENIE | 25 |
| Do kogo jest skierowana ta książka? | 26 |
| Konwencje | 27 |
| Czym jest programowanie? | 27 |
| Co to jest Python? | 28 |
| Programiści nie muszą dobrze znać matematyki | 28 |
| Programowanie to aktywność kreatywna | 29 |
| O tej książce | 30 |
| Pobieranie i instalacja Pythona | 31 |
| Uruchomienie środowiska IDLE | 33 |
| Powłoka interaktywna | 33 |
| Jak otrzymać pomoc? | 34 |
| Sprytnie zadawanie pytań dotyczących programowania | 35 |
| Podsumowanie | 36 |

CZĘŚĆ I. PODSTAWY PROGRAMOWANIA W PYTHONIE

37

I.

| | |
|---|-----------|
| PODSTAWY PYTHONA | 39 |
| Wprowadzanie wyrażeń w powłoce interaktywnej | 40 |
| Liczby całkowite, zmiennoprzecinkowe i ciągi tekstowe | 43 |
| Konkatenacja i replikacja ciągu tekstowego | 43 |

| | |
|---|----|
| Przechowywanie wartości w zmiennych | 45 |
| Polecenia przypisania | 45 |
| Nazwy zmiennych | 47 |
| Twój pierwszy program | 48 |
| Analiza programu | 49 |
| Polecenia | 49 |
| Funkcja print() | 50 |
| Funkcja input() | 50 |
| Wyświetlanie imienia użytkownika | 50 |
| Funkcja len() | 51 |
| Funkcje str(), int() i float() | 52 |
| Podsumowanie | 55 |
| Pytania kontrolne | 56 |

2.

KONTROLA PRZEPLÝWU DZIAŁANIA PROGRAMU 59

| | |
|--|----|
| Wartości boolowskie | 60 |
| Operatory porównania | 61 |
| Operatory boolowskie | 63 |
| Binarne operatory boolowskie | 63 |
| Operator not | 64 |
| Łączenie operatorów boolowskich i porównania | 65 |
| Elementy kontroli przepływu działania programu | 65 |
| Warunek | 66 |
| Blok kodu | 66 |
| Wykonywanie programu | 66 |
| Polecenia kontroli przepływu działania programu | 67 |
| Polecenie if | 67 |
| Polecenie else | 68 |
| Polecenie elif | 69 |
| Pętla while | 74 |
| Polecenie break | 78 |
| Polecenie continue | 79 |
| Pętla for i funkcja range() | 83 |
| Import modułów | 87 |
| Polecenie from import | 88 |
| Wcześniejsze zakończenie programu za pomocą sys.exit() | 88 |
| Podsumowanie | 89 |
| Pytania kontrolne | 89 |

3.

FUNKCJE 91

| | |
|--|----|
| Polecenie def wraz z parametrami | 93 |
| Wartość zwrotna funkcji i polecenie return | 93 |
| Wartość None | 95 |

| | |
|---|-----|
| Argumenty w postaci słów kluczowych i funkcja print() | 96 |
| Zasięgi lokalny i globalny | 97 |
| Zmienne lokalne nie mogą być używane w zasięgu globalnym | 98 |
| W zasięgu lokalnym nie można używać zmiennych zdefiniowanych w innych zasięgach lokalnych | 99 |
| Zmienna globalna może być używana w zasięgu lokalnym | 99 |
| Zmienna lokalna i globalna o takiej samej nazwie | 100 |
| Polecenie global | 101 |
| Obsługa wyjątków | 103 |
| Krótki program — odgadnij liczbę | 105 |
| Podsumowanie | 107 |
| Pytania kontrolne | 108 |
| Projekt praktyczny | 108 |
| Problem Collatza | 109 |
| Weryfikacja danych wyjściowych | 109 |

4.

| | |
|--|------------|
| LISTY | 111 |
| Typ danych List | 111 |
| Pobieranie poszczególnych wartości listy za pomocą indeksu | 112 |
| Indeks ujemny | 114 |
| Pobieranie podlisty za pomocą wycinka | 114 |
| Pobieranie długości listy za pomocą polecenia len() | 115 |
| Zmiana wartości na liście za pomocą indeksu | 115 |
| Konkatenacja i replikacja listy | 116 |
| Usunięcie wartości listy za pomocą polecenia del | 116 |
| Praca z listą | 116 |
| Użycie pętli for wraz z listą | 118 |
| Operatory in i not in | 119 |
| Sztuczka pozwalająca na wiele jednoczesnych operacji przypisania | 120 |
| Operatory przypisania i zmiany wartości | 121 |
| Metody | 122 |
| Odszukanie wartości na liście za pomocą metody index() | 122 |
| Dodanie wartości do listy za pomocą metod append() i insert() | 123 |
| Usuwanie wartości z listy za pomocą metody remove() | 124 |
| Sortowanie wartości listy za pomocą metody sort() | 124 |
| Przykładowy program — Magic 8 Ball utworzony za pomocą listy | 126 |
| Typy przypominające listę — ciąg tekstowy i krotka | 127 |
| Modyfikowalne i niemodyfikowalne typy danych | 128 |
| Typ danych krotka | 130 |
| Konwersja typu za pomocą funkcji list() i tuple() | 131 |
| Odwołania | 132 |
| Przekazywanie odwołania | 134 |
| Funkcje copy() i deepcopy() modułu copy | 135 |
| Podsumowanie | 136 |

| | |
|-----------------------------------|-----|
| Pytania kontrolne | 136 |
| Projekty praktyczne | 137 |
| Kod z przecinkami | 137 |
| Obraz na podstawie macierzy | 137 |

5.

SŁOWNIKI I STRUKTURYZACJA DANYCH 139

| | |
|---|-----|
| Typ danych Dictionary | 139 |
| Słownik kontra lista | 140 |
| Metody keys(), values() i items() | 142 |
| Sprawdzenie, czy klucz lub wartość istnieją w słowniku | 143 |
| Metoda get() | 144 |
| Metoda setdefault() | 144 |
| Eleganckie wyświetlanie danych | 146 |
| Użycie struktur danych do modelowania rzeczywistych rozwiązań | 147 |
| Plansza do gry w kółko i krzyżyk | 148 |
| Zagnieżdżone słowniki i listy | 152 |
| Podsumowanie | 154 |
| Pytania kontrolne | 154 |
| Projekty praktyczne | 154 |
| Inwentarz w grze fantasy | 155 |
| Funkcja konwertująca listę na słownik dla inwentarza w grze fantasy | 155 |

6.

OPERACJE NA CIĄGACH TEKSTOWYCH 157

| | |
|---|-----|
| Praca z ciągami tekstowymi | 157 |
| Literały ciągu tekstowego | 158 |
| Indeksowanie i wycinanie ciągów tekstowych | 161 |
| Użycie operatorów in i not in podczas pracy z ciągami tekstowymi | 162 |
| Użyteczne metody ciągu tekstowego | 162 |
| Metody upper(), lower(), isupper() i islower() | 162 |
| Metody typu isX() | 164 |
| Metody startswith() i endswith() | 166 |
| Metody join() i split() | 167 |
| Wyrównywanie tekstu za pomocą metod rjust(), ljust() i center() | 168 |
| Usunięcie białych znaków za pomocą strip(),rstrip() i lstrip() | 170 |
| Kopiowanie i wklejanie ciągów tekstowych za pomocą modułu pyperclip | 171 |
| Projekt — menedżer haseł | 172 |
| Etap 1. Projekt programu i struktur danych | 172 |
| Etap 2. Obsługa argumentów wiersza poleceń | 173 |
| Etap 3. Skopiowanie odpowiedniego hasła | 173 |
| Projekt — dodanie wypunktowania do kodu znaczników Wiki | 175 |
| Etap 1. Kopiowanie i wklejanie ze schowka | 175 |
| Etap 2. Rozdzielenie wierszy tekstu i dodanie gwiazdki | 176 |
| Etap 3. Złączenie zmodyfikowanych wierszy | 177 |

| | |
|---------------------------|-----|
| Podsumowanie | 177 |
| Pytania kontrolne | 178 |
| Projekt praktyczny | 179 |
| Wyświetlenie tabeli | 179 |

CZĘŚĆ II. AUTOMATYZACJA ZADAŃ

181

7.

DOPASOWANIE WZORCA ZA POMOCĄ WYRAZEŃ REGULARNYCH 183

| | |
|--|-----|
| Wyszukiwanie wzorców w tekście bez użycia wyrażeń regularnych | 184 |
| Wyszukiwanie wzorców w tekście z użyciem wyrażeń regularnych | 186 |
| Tworzenie obiektów wyrażeń regularnych | 187 |
| Dopasowanie obiektów wyrażeń regularnych | 187 |
| Przegląd dopasowania za pomocą wyrażenia regularnego | 189 |
| Jeszcze więcej o dopasowaniach wzorca za pomocą wyrażeń regularnych | 189 |
| Grupowanie z użyciem nawiasów | 189 |
| Dopasowanie wielu grup za pomocą potoku | 190 |
| Opcjonalne dopasowanie za pomocą znaku zapytania | 191 |
| Dopasowanie zera wystąpień lub większej liczby wystąpień za pomocą gwiazdki | 192 |
| Dopasowanie jednego wystąpienia lub wielu wystąpień za pomocą plusa | 193 |
| Dopasowanie określonych powtórzeń za pomocą nawiasu klamrowego | 193 |
| Dopasowanie zachłanne i niezachłanne | 194 |
| Metoda findall() | 195 |
| Klasy znaków | 196 |
| Utworzenie własnej klasy znaków | 197 |
| Znaki ^ oraz \$ | 198 |
| Znak wieloznaczny | 199 |
| Dopasowanie wszystkiego za pomocą kropki i gwiazdki | 199 |
| Dopasowanie znaku nowego wiersza za pomocą kropki | 200 |
| Przegląd znaków stosowanych w wyrażeniach regularnych | 200 |
| Dopasowanie bez uwzględnienia wielkości znaków | 201 |
| Zastępowanie ciągu tekstowego za pomocą metody sub() | 202 |
| Zarządzanie skomplikowanymi wyrażeniami regularnymi | 203 |
| Połączenie opcji re.IGNORECASE, re.DOTALL i re.VERBOSE | 203 |
| Projekt — wyodrębnianie numeru telefonu i adresu e-mail | 204 |
| Etap 1. Utworzenie wyrażenia regularnego dopasowującego numer telefonu | 205 |
| Etap 2. Utworzenie wyrażenia regularnego dopasowującego adres e-mail | 206 |
| Etap 3. Wyszukanie wszystkich dopasowań w tekście umieszczonym w schowku | 207 |
| Etap 4. Połączenie dopasowań w celu utworzenia pojedynczego ciągu tekstowego do umieszczenia w schowku | 208 |
| Uruchomienie programu | 208 |
| Pomysły na podobne programy | 209 |
| Podsumowanie | 209 |
| Pytania kontrolne | 210 |

| | |
|--|-----|
| Projekty praktyczne | 211 |
| Wykrywanie silnego hasła | 212 |
| Oparta na wyrażeniu regularnym wersja metody strip() | 212 |

8.

ODCZYT I ZAPIS PLIKÓW **213**

| | |
|---|-----|
| Pliki i ścieżki dostępu do plików | 213 |
| Lewy ukośnik w systemie Windows, prawy ukośnik w systemach macOS i Linux | 214 |
| Bieżący katalog roboczy | 215 |
| Względne kontra bezwzględne ścieżki dostępu | 216 |
| Tworzenie nowych katalogów za pomocą funkcji os.makedirs() | 216 |
| Moduł os.path | 217 |
| Obsługa bezwzględnych i względnych ścieżek dostępu | 217 |
| Ustalenie wielkości pliku i zawartości katalogu | 220 |
| Sprawdzenie poprawności ścieżki dostępu | 221 |
| Proces odczytu i zapisu pliku | 222 |
| Otwieranie pliku za pomocą funkcji open() | 223 |
| Odczyt zawartości pliku | 224 |
| Zapis pliku | 225 |
| Zapis zmiennych za pomocą modułu shelve | 226 |
| Zapis zmiennych za pomocą funkcji pprint.pformat() | 227 |
| Projekt — generowanie losowych plików quizu | 229 |
| Etap 1. Umieszczenie danych quizu w słowniku | 229 |
| Etap 2. Utworzenie pliku quizu i losowe umieszczenie odpowiedzi na pytania | 230 |
| Etap 3. Utworzenie odpowiedzi | 231 |
| Etap 4. Zapis treści w plikach quizu i odpowiedzi | 232 |
| Projekt — schowek przechowujący wiele elementów | 234 |
| Etap 1. Komentarze i konfiguracja pliku binarnego | 235 |
| Etap 2. Zapis zawartości schowka wraz ze słowem kluczowym | 235 |
| Etap 3. Wyświetlenie słów kluczowych i wczytanie treści powiązanej ze słowem kluczowym | 236 |
| Podsumowanie | 237 |
| Pytania kontrolne | 237 |
| Projekty praktyczne | 238 |
| Rozbudowa programu schowka przechowującego wiele elementów | 238 |
| Program Mad Libs | 238 |
| Wyszukiwanie wyrażenia regularnego | 239 |

9.

ORGANIZACJA PLIKÓW **241**

| | |
|--|-----|
| Moduł shutil | 242 |
| Kopiowanie plików i katalogów | 242 |
| Przenoszenie oraz zmiana nazwy plików i katalogów | 243 |
| Trwałe usunięcie plików i katalogów | 245 |
| Bezpieczne usuwanie danych za pomocą modułu send2trash | 246 |

| | |
|---|-----|
| Przejsie przez drzewo katalogu | 246 |
| Kompresja plikow za pomoca modulu zipfile | 248 |
| Odczyt pliku w formacie ZIP | 248 |
| Wyodrębnianie plikow z archiwum ZIP | 249 |
| Utworzenie i dodawanie elementow do archiwum ZIP | 250 |
| Projekt — zmiana plikow z datami w stylu amerykanskim na daty w stylu europejskim | 251 |
| Etap 1. Utworzenie wyrazenia regularnego dla daty w stylu amerykanskim | 251 |
| Etap 2. Identyfikacja w nazwie pliku fragmentow okreslajacych date | 253 |
| Etap 3. Utworzenie nowej nazwy pliku i zmiana nazw plikow | 254 |
| Pomysly na podobne programy | 255 |
| Projekt — utworzenie archiwum ZIP bedacego kopia katalogu | 255 |
| Etap 1. Ustalenie nazwy pliku archiwum ZIP | 256 |
| Etap 2. Utworzenie nowego archiwum ZIP | 257 |
| Etap 3. Przejsie przez drzewo katalogu i dodanie plikow do archiwum ZIP | 258 |
| Pomysly na podobne programy | 259 |
| Podsumowanie | 259 |
| Pytania kontrolne | 260 |
| Projekty praktyczne | 260 |
| Kopiowanie selektywne | 260 |
| Usuniecie niepotrzebnych plikow | 260 |
| Wypelnienie przerw | 260 |

10.

USUWANIE BLEDOW **261**

| | |
|---|-----|
| Zglaszanie wyjatku | 262 |
| Pobranie stosu wywolow w postaci ciagu tekstowego | 264 |
| Asercje | 265 |
| Uzycie asercji w projekcie symulacji ulicznej sygnalizacji swietlonej | 266 |
| Wykluczenie asercji | 268 |
| Rejestracja danych | 268 |
| Uzycie modulu logging | 268 |
| Nie przeprowadzaj procesu usuwania bledow za pomoca funkcji print() | 270 |
| Poziomy rejestrowania informacji | 271 |
| Wykluczenie rejestrowania informacji | 272 |
| Rejestrowanie informacji w pliku | 273 |
| Debugger srodowiska IDLE | 273 |
| Go | 274 |
| Step | 274 |
| Over | 274 |
| Out | 275 |
| Quit | 275 |
| Debugowanie programu sumujacego liczby | 275 |
| Punkty kontrolne | 277 |
| Podsumowanie | 279 |

| | |
|---|-----|
| Pytania kontrolne | 280 |
| Projekt praktyczny | 281 |
| Debugowanie programu symulującego rzut monetą | 281 |

II.

| | |
|---|------------|
| POBIERANIE DANYCH Z INTERNETU | 283 |
| Projekt — mapIt.py z użyciem modułu webbrowser | 284 |
| Etap 1. Ustalenie adresu URL | 285 |
| Etap 2. Obsługa argumentów wiersza poleceń | 285 |
| Etap 3. Obsługa zawartości schowka i uruchomienie przeglądarki WWW | 286 |
| Pomysły na podobne programy | 287 |
| Pobieranie plików z internetu za pomocą modułu requests | 287 |
| Pobieranie strony internetowej za pomocą funkcji requests.get() | 288 |
| Sprawdzenie pod kątem błędów | 289 |
| Zapis pobranych plików na dysku twardym | 290 |
| HTML | 291 |
| Zasoby pomagające w poznawaniu języka HTML | 291 |
| Krótkie wprowadzenie | 292 |
| Wyświetlenie kodu źródłowego HTML strony internetowej | 293 |
| Wyświetlenie oferowanych przez przeglądarkę WWW narzędzi programistycznych | 293 |
| Użycie narzędzi programistycznych do wyszukiwania elementów HTML | 296 |
| Przetwarzanie kodu HTML za pomocą modułu BeautifulSoup | 297 |
| Utworzenie obiektu BeautifulSoup na podstawie kodu HTML | 297 |
| Wyszukiwanie elementu za pomocą metody select() | 298 |
| Pobieranie danych z atrybutów elementu | 300 |
| Projekt — wyszukiwanie typu „szczęśliwy traf” w Google | 301 |
| Etap 1. Pobranie argumentów wiersza poleceń i żądanie strony wyszukiwarki | 301 |
| Etap 2. Wyszukanie wszystkich wyników | 302 |
| Etap 3. Otworzenie kart przeglądarki WWW dla poszczególnych wyników | 303 |
| Pomysły na podobne programy | 304 |
| Projekt — pobranie wszystkich komiksów z witryny XKCD | 304 |
| Etap 1. Projekt programu | 306 |
| Etap 2. Pobranie strony internetowej | 307 |
| Etap 3. Odszukanie i pobranie obrazu komiksu | 307 |
| Etap 4. Zapis obrazu i odszukanie poprzedniego komiksu | 308 |
| Pomysły na podobne programy | 310 |
| Kontrolowanie przeglądarki WWW za pomocą modułu selenium | 310 |
| Uruchomienie przeglądarki WWW kontrolowanej przez moduł selenium | 310 |
| Wyszukanie elementów na stronie | 311 |
| Kliknięcie na stronie | 314 |
| Wypełnianie i wysyłanie formularzy sieciowych | 314 |
| Symulacja naciśnięcia klawiszy specjalnych | 315 |
| Klikanie przycisków przeglądarki WWW | 316 |
| Więcej informacji na temat modułu selenium | 316 |
| Podsumowanie | 316 |

| | |
|--|-----|
| Pytania kontrolne | 316 |
| Projekty praktyczne | 317 |
| Klient poczty działający w wierszu poleceń | 318 |
| Pobieranie obrazów z witryny internetowej | 318 |
| 2048 | 318 |
| Weryfikacja łącza | 318 |

12.

PRACA Z ARKUSZAMI KALKULACYJNYMI PROGRAMU EXCEL 319

| | |
|---|-----|
| Dokumenty Excela | 320 |
| Instalacja modułu openpyxl | 320 |
| Odczyt dokumentów Excela | 321 |
| Otwieranie istniejącego dokumentu Excela za pomocą openpyxl | 321 |
| Pobranie arkuszy ze skoroszytu | 322 |
| Pobieranie komórek z arkuszy | 322 |
| Konwersja między literami kolumn i liczbami | 324 |
| Pobieranie wierszy i kolumn z arkuszy | 325 |
| Skoroszyty, arkusze i komórki | 327 |
| Projekt — odczyt danych z arkusza kalkulacyjnego | 327 |
| Etap 1. Odczyt danych z arkusza kalkulacyjnego | 328 |
| Etap 2. Wypełnienie struktury danych | 329 |
| Etap 3. Zapis wyników do pliku | 331 |
| Pomysły na podobne programy | 332 |
| Zapis dokumentów Excela | 332 |
| Tworzenie i zapisywanie dokumentów Excela | 332 |
| Tworzenie i usuwanie arkuszy kalkulacyjnych | 333 |
| Zapis wartości w komórkach | 334 |
| Projekt — uaktualnienie skoroszytu | 335 |
| Etap 1. Przygotowanie struktury danych wraz z uaktualnionymi informacjami | 336 |
| Etap 2. Sprawdzenie wszystkich wierszy i skorygowanie nieprawidłowych cen | 337 |
| Pomysły na podobne programy | 338 |
| Ustawienie stylu czcionki komórek | 338 |
| Obiekt Font | 339 |
| Formuły | 341 |
| Dostosowanie wierszy i kolumn do własnych potrzeb | 342 |
| Ustalenie wysokości wiersza i szerokości kolumny | 343 |
| Łączenie i dzielenie komórki | 343 |
| Zablokowane okienka | 344 |
| Wykresy | 345 |
| Podsumowanie | 348 |
| Pytania kontrolne | 348 |
| Projekty praktyczne | 349 |
| Program tworzący tabliczkę mnożenia | 349 |
| Program wstawiający pusty wiersz | 350 |

| | |
|--|-----|
| Program zmieniający położenie komórek arkusza kalkulacyjnego | 350 |
| Przeniesienie zawartości pliku tekstowego do arkusza kalkulacyjnego | 351 |
| Przeniesienie zawartości arkusza kalkulacyjnego do plików tekstowych | 351 |

13.

PRACA Z DOKUMENTAMI PDF I WORDA 353

| | |
|--|-----|
| Dokumenty w formacie PDF | 353 |
| Wyodrębnianie tekstu z dokumentu PDF | 354 |
| Deszyfrowanie dokumentu PDF | 356 |
| Tworzenie dokumentów PDF | 357 |
| Projekt — połączenie wybranych stron z wielu dokumentów PDF | 362 |
| Etap 1. Wyszukanie wszystkich plików w formacie PDF | 363 |
| Etap 2. Otworzenie poszczególnych dokumentów PDF | 364 |
| Etap 3. Dodanie poszczególnych stron | 364 |
| Etap 4. Zapis dokumentu wynikowego | 365 |
| Pomysły na podobne programy | 366 |
| Dokumenty procesora tekstu Microsoft Word | 366 |
| Odczyt dokumentów Worda | 367 |
| Pobranie pełnego tekstu z pliku w formacie .docx | 368 |
| Nadawanie stylu akapitom i obiektom Run | 369 |
| Utworzenie dokumentu Worda z niestandardowymi stylami | 370 |
| Atrybuty obiektu Run | 371 |
| Zapis dokumentów Worda | 372 |
| Dodanie nagłówków | 374 |
| Dodanie znaku podziału wiersza i strony | 375 |
| Dodanie obrazu | 376 |
| Podsumowanie | 376 |
| Pytania kontrolne | 377 |
| Projekty praktyczne | 378 |
| PDF Paranoja | 378 |
| Własne zaproszenia utworzone w dokumencie Worda | 378 |
| Program łamiący hasło dokumentu PDF za pomocą ataku typu brute force | 379 |

14.

PRACA Z PLIKAMI CSV I DANymi JSON 381

| | |
|--|-----|
| Moduł csv | 382 |
| Obiekt Reader | 383 |
| Użycie pętli for do odczytu danych z obiektu Reader | 384 |
| Obiekt Writer | 384 |
| Argumenty w postaci słów kluczowych delimiter i lineterminator | 386 |
| Projekt — usunięcie nagłówka z pliku CSV | 387 |
| Etap 1. Iteracja przez poszczególne pliki CSV | 387 |
| Etap 2. Odczyt zawartości pliku CSV | 388 |
| Etap 3. Zapis pliku CSV bez pierwszego wiersza | 389 |
| Pomysły na podobne programy | 390 |

| | |
|---|-----|
| JSON i API | 390 |
| Moduł json | 392 |
| Odczyt danych JSON za pomocą funkcji loads() | 392 |
| Zapis danych w formacie JSON za pomocą funkcji dumps() | 392 |
| Projekt — pobieranie bieżących danych prognozy pogody | 393 |
| Etap 1. Pobranie z wiersza poleceń informacji o lokalizacji | 393 |
| Etap 2. Pobranie danych w formacie JSON | 394 |
| Etap 3. Wczytanie danych w formacie JSON i wyświetlenie prognozy pogody | 395 |
| Pomysły na podobne programy | 397 |
| Podsumowanie | 397 |
| Pytania kontrolne | 397 |
| Projekty praktyczne | 398 |
| Konwerter danych w formacie Excel do formatu CSV | 398 |

15.

CZAS, HARMONOGRAM ZADAŃ I URUCHAMIANIE PROGRAMÓW 401

| | |
|--|-----|
| Moduł time | 402 |
| Funkcja time.time() | 402 |
| Funkcja time.sleep() | 403 |
| Zaokrąglanie liczb | 404 |
| Projekt — superstoper | 405 |
| Etap 1. Przygotowanie programu do pomiaru czasu | 405 |
| Etap 2. Monitorowanie i wyświetlenie czasu okrążenia | 406 |
| Pomysły na podobne programy | 407 |
| Moduł datetime | 408 |
| Typ danych timedelta | 410 |
| Pauza aż do chwili osiągnięcia określonej daty | 411 |
| Konwersja obiektu datetime na ciąg tekstowy | 412 |
| Konwersja ciągu tekstowego na obiekt datetime | 413 |
| Przegląd funkcji czasu w Pythonie | 414 |
| Wielowątkowość | 415 |
| Przekazanie argumentów funkcji docelowej dla nowego wątku | 417 |
| Kwestie związane ze współbieżnością | 418 |
| Projekt — wielowątkowy program pobierający dane z witryny XKCD | 418 |
| Etap 1. Modyfikacja programu w celu użycia funkcji | 419 |
| Etap 2. Utworzenie i uruchomienie wątków | 420 |
| Etap 3. Zaczekanie na zakończenie działania wszystkich wątków | 421 |
| Uruchamianie innych programów z poziomu Pythona | 421 |
| Przekazanie funkcji Popen() argumentów wiersza poleceń | 424 |
| Harmonogram zadań, launchd i cron | 424 |
| Otwieranie witryn internetowych za pomocą Pythona | 425 |
| Wykonywanie innych skryptów Pythona | 425 |
| Otwieranie plików w ich aplikacjach domyślnych | 426 |

| | |
|---|-----|
| Projekt — prosty program odliczający czas | 427 |
| Etap 1. Odliczanie | 428 |
| Etap 2. Odtworzenie pliku dźwiękowego | 428 |
| Pomysły na podobne programy | 429 |
| Podsumowanie | 430 |
| Pytania kontrolne | 430 |
| Projekty praktyczne | 431 |
| Ładniejszy stoper | 431 |
| Oparty na harmonogramie program pobierający komiksy | 431 |

16.

WYSYLANIE WIADOMOŚCI E-MAIL I TEKSTOWYCH 433

| | |
|--|-----|
| SMTP | 434 |
| Wysyłanie wiadomości e-mail | 434 |
| Nawiązanie połączenia z serwerem SMTP | 435 |
| Wysłanie wiadomości SMTP typu „Witaj” | 436 |
| Włączenie szyfrowania TLS | 437 |
| Logowanie w serwerze SMTP | 437 |
| Wysyłanie wiadomości e-mail | 438 |
| Zamknięcie połączenia z serwerem SMTP | 438 |
| IMAP | 439 |
| Pobieranie i usuwanie wiadomości e-mail za pomocą protokołu IMAP | 439 |
| Nawiązanie połączenia z serwerem IMAP | 440 |
| Logowanie w serwerze IMAP | 441 |
| Wyszukiwanie wiadomości e-mail | 441 |
| Pobieranie wiadomości e-mail i oznaczanie jej jako przeczytanej | 445 |
| Pobieranie adresów e-mail z niezmodyfikowanych wiadomości e-mail | 447 |
| Pobranie treści z niezmodyfikowanej wiadomości e-mail | 448 |
| Usuwanie wiadomości e-mail | 449 |
| Zamknięcie połączenia z serwerem IMAP | 450 |
| Projekt — wysyłanie wiadomości e-mail z przypomnieniami o składkach | 451 |
| Etap 1. Otworzenie pliku Excela | 451 |
| Etap 2. Wyszukanie wszystkich członków klubu, którzy zalegają ze składką | 453 |
| Etap 3. Wysłanie spersonalizowanego przypomnienia | 453 |
| Wysyłanie wiadomości tekstowych za pomocą Twilio | 455 |
| Założenie konta w serwisie Twilio | 456 |
| Wysyłanie wiadomości tekstowych | 456 |
| Projekt — moduł typu „wyślij mi wiadomość SMS” | 459 |
| Podsumowanie | 460 |
| Pytania kontrolne | 461 |
| Projekty praktyczne | 461 |
| Program losowo przypisujący uciążliwe zadania | 461 |
| Przypomnienie o parasolu | 462 |
| Automatyczna rezygnacja z subskrypcji | 462 |
| Kontrola komputera za pomocą wiadomości e-mail | 462 |

17.

| | |
|---|------------|
| PRACA Z OBRAZAMI | 465 |
| Podstawy teorii obrazu cyfrowego | 465 |
| Kolory i wartości RGBA | 466 |
| Współrzędne i krotki pudełek | 467 |
| Praca z obrazami za pomocą modułu pillow | 468 |
| Praca z typem danych Image | 470 |
| Przycinanie obrazu | 471 |
| Kopiowanie i wklejanie obrazów w innych obrazach | 472 |
| Zmiana wielkości obrazu | 474 |
| Rotacja i lustrzane odbicia obrazu | 476 |
| Zmiana poszczególnych pikseli | 478 |
| Projekt — dodanie logo | 480 |
| Etap 1. Otworzenie pliku logo | 481 |
| Etap 2. Iteracja przez wszystkie pliki i otworenie obrazów | 481 |
| Etap 3. Zmiana wielkości obrazu | 482 |
| Etap 4. Dodanie obrazu logo i zapisanie zmian | 483 |
| Pomysły na podobne programy | 485 |
| Rysowanie na obrazach | 486 |
| Rysowanie kształtów | 486 |
| Umieszczanie tekstu na obrazie | 488 |
| Podsumowanie | 490 |
| Pytania kontrolne | 491 |
| Projekty praktyczne | 491 |
| Rozbudowa i poprawa projektów omówionych w rozdziale | 491 |
| Odszukanie na dysku twardym katalogów zawierających zdjęcia | 492 |
| Własne wizytówki | 493 |

18.

| | |
|---|------------|
| KONTROLOWANIE KLAWIATURY I MYSZY ZA POMOCĄ AUTOMATYZACJI GUI | 495 |
| Instalacja modułu pyautogui | 496 |
| Pozostajemy na kursie | 496 |
| Zamknięcie wszystkiego przez wylogowanie się | 497 |
| Pauzy i funkcja bezpiecznej awarii | 497 |
| Kontrola poruszania myszą | 498 |
| Poruszanie kursorem myszy | 499 |
| Pobranie informacji o położeniu kursora myszy | 500 |
| Projekt — gdzie teraz jest kursor myszy? | 500 |
| Etap 1. Import modułu | 501 |
| Etap 2. Konfiguracja kodu zamykającego program oraz pętli działającej w nieskończoność | 501 |
| Etap 3. Pobranie i wyświetlenie bieżących współrzędnych kursora myszy | 502 |

| | |
|---|-----|
| Kontrola działania myszy | 503 |
| Kliknięcie myszą | 504 |
| Przeciąganie myszą | 504 |
| Przewijanie myszą | 506 |
| Praca z ekranem | 508 |
| Wykonanie zrzutu ekranu | 508 |
| Analiza zrzutu ekranu | 509 |
| Projekt — rozbudowa programu mouseNow.py | 509 |
| Rozpoznawanie obrazu | 510 |
| Kontrola klawiatury | 512 |
| Przekazanie ciągu tekstowego z klawiatury | 512 |
| Nazwy klawiszy | 513 |
| Naciskanie i zwalnianie klawiszy | 514 |
| Kombinacja klawiszy | 515 |
| Przegląd funkcji modułu pyautogui | 516 |
| Projekt — automatyczne wypełnianie formularzy | 517 |
| Etap 1. Ustalenie kroków do wykonania | 518 |
| Etap 2. Przygotowanie współrzędnych | 519 |
| Etap 3. Rozpoczęcie wpisywania danych | 521 |
| Etap 4. Obsługa rozwijanych list i przycisków opcji | 522 |
| Etap 5. Wysłanie formularza i oczekiwanie | 523 |
| Podsumowanie | 524 |
| Pytania kontrolne | 525 |
| Projekty praktyczne | 525 |
| Symulowanie zajętości | 525 |
| Bot komunikatora internetowego | 526 |
| Samouczek dotyczący bota grającego w grę | 526 |

DODATKI

527

A

INSTALACJA MODUŁÓW FIRM TRZECICH 529

| | |
|--|-----|
| Narzędzie pip | 529 |
| Instalacja modułów firm trzecich | 530 |

B

URUCHAMIANIE PROGRAMÓW 533

| | |
|--|-----|
| Wiersz shebang | 533 |
| Uruchamianie programów Pythona w Windows | 534 |
| Uruchamianie programów Pythona w systemach macOS i Linux | 535 |
| Uruchamianie programów Pythona z wyłączonymi asercjami | 536 |

C**ODPOWIEDZI NA PYTANIA KONTROLNE 537**

| | |
|-------------------|-----|
| Rozdział 1. | 538 |
| Rozdział 2. | 538 |
| Rozdział 3. | 540 |
| Rozdział 4. | 541 |
| Rozdział 5. | 542 |
| Rozdział 6. | 542 |
| Rozdział 7. | 543 |
| Rozdział 8. | 544 |
| Rozdział 9. | 545 |
| Rozdział 10. | 545 |
| Rozdział 11. | 546 |
| Rozdział 12. | 547 |
| Rozdział 13. | 548 |
| Rozdział 14. | 549 |
| Rozdział 15. | 549 |
| Rozdział 16. | 550 |
| Rozdział 17. | 550 |
| Rozdział 18. | 551 |

SKOROWIDZ 553

11

Pobieranie danych z internetu



W TYCH RZADKICH, KŁOPOTLIWYCH MOMENTACH, KIEDY JESTEM POZBAWIONY DOSTĘPU DO WI-FI, ZDAJĘ SOBIE SPRAWĘ Z TEGO, JAK WIELE CZYNNOŚCI, KTÓRE WYKONUJĘ W MOIM KOMPUTERZE, robię w internecie. Z czystego przyzwyczajenia sprawdzam pocztę elektroniczną, opublikowane przez moich przyjaciół komunikaty w serwisie Twitter lub po prostu szukam odpowiedzi na pytania typu: „Czy Kurtwood Smith zagrał jakieś główne role, zanim w roku 1987 wcielił się w postać Robocopa?¹”.

Ponieważ duża część pracy wykonywanej w komputerze wymaga dostępu do internetu, więc byłoby dobrze, gdyby tworzone programy miały dostęp do sieci WWW. *Pobieranie danych z internetu* oznacza umożliwienie programowi pobrania i przetwarzania treści pochodzącej z sieci WWW. Przykładowo Google używa wielu tego rodzaju programów w celu indeksowania stron internetowych dla zbudowanej przez tę firmę wyszukiwarki internetowej. W tym rozdziale poznasz kilka modułów ułatwiających pobieranie danych z internetu w programach Pythona.

¹ Odpowiedź na to pytanie brzmi nie.

- Moduł **webbrowser** jest dostarczany wraz z Pythonem i pozwala na wyświetlenie wskazanej strony w przeglądarce WWW.
- Moduł **requests** pozwala na pobieranie z internetu plików i stron internetowych.
- Moduł **BeautifulSoup** umożliwia przetwarzanie kodu HTML, za pomocą którego są tworzone strony internetowe.
- Moduł **selenium** uruchamia i kontroluje przeglądarkę WWW. Moduł **selenium** ma możliwość wypełniania formularzy oraz symulacji kliknięć myszą w przeglądarce WWW.

Projekt — `mapIt.py` z użyciem modułu `webbrowser`

Zdefiniowana w module `webbrowser` funkcja `open()` może uruchomić przeglądarkę WWW i przejść pod wskazany adres URL. W powłoce interaktywnej wprowadź przedstawione poniżej polecenia.

```
>>> import webbrowser
>>> webbrowser.open('http://inventwithpython.com/')
```

W przeglądarce WWW zostanie utworzona nowa karta, w której będzie wyświetlona witryna znajdująca się pod adresem URL `http://inventwithpython.com/`. To jest jedyne zadanie, jakie może wykonać moduł `webbrowser`. Mimo to funkcja `open()` otwiera przed nami interesujące możliwości. Przykładowo do żmudnych zadań należy kopiowanie do schowka adresu zawierającego nazwę ulicy, numer i miejscowość, aby później wyświetlić podane miejsce w aplikacji sieciowej Mapy Google. Całą operację można zredukować o kilka kroków przez utworzenie prostego skryptu, który na podstawie adresu umieszczonego w schowku automatycznie wyświetli mapę w przeglądarce WWW. W ten sposób musisz jedynie skopiować adres do schowka, a następnie uruchomić skrypt. W przeglądarce WWW otrzymasz wyświetloną mapę.

Poniżej wymienilem w punktach sposób działania programu.

- Pobranie adresu z argumentów wiersza poleceń lub schowka.
- W przeglądarce WWW uruchomienie aplikacji sieciowej Mapy Google i wyświetlenie mapy dla podanego adresu.

Oznacza to, że kod będzie musiał wykonywać przedstawione poniżej zadania.

- Odczyt argumentów wiersza poleceń ze zmiennej `sys.argv`.
- Odczyt zawartości schowka.
- Wywołanie funkcji `webbrowser.open()` w celu otwarcia strony w przeglądarce WWW.

Otwórz nowe okno edytora pliku i zapisz plik pod nazwą *mapIt.py*.

Etap 1. Ustalenie adresu URL

Opierając się na informacjach przedstawionych w dodatku B, skonfiguruj skrypt *mapIt.py* w taki sposób, aby po uruchomieniu go z poziomu wiersza poleceń, na przykład tak:

```
C:\> mapit 870 Valencia St, San Francisco, CA 94110
```

wykorzystał argumenty wiersza poleceń, a nie używał schowka. Jeżeli nie podano żadnych argumentów wiersza poleceń, program powinien automatycznie wykorzystać zawartość schowka.

Przed wszystkim trzeba ustalić, jaki adres URL powinien być użyty dla podanego adresu wskazującego fizyczne miejsce. Po wczytaniu w przeglądarce WWW aplikacji sieciowej Mapy Google (<http://mapy.google.pl/>) i wyszukaniu wymienionego powyżej adresu pasek adresu URL będzie miał postać podobną do <https://www.google.pl/maps/place/870+Valencia+St/@37.7590311,-122.4215096,17z/data=!3m1!4b1!4m2!3m1!1s0x808f7e3dad07a37:0xc86b0b2bb93b73d8>.

Wprawdzie adres został podany w URL, ale jednocześnie zawiera dużą ilość dodatkowego tekstu. W witrynach internetowych w adresie URL często znajdują się dane dodatkowe, które mają pomagać w śledzeniu odwiedzających lub w dostosowaniu witryny do własnych potrzeb. Jeżeli jednak wpiszesz adres URL w postaci <https://www.google.pl/maps/place/870+Valencia+St+San+Francisco+CA/>, wówczas zostaniesz przekierowany na właściwą stronę. Dlatego też nasz program może po prostu otworzyć nową kartę w przeglądarce WWW i przejść pod adres 'https://www.google.pl/maps/place/ciąg_tekstowy_adresu' (gdzie *ciąg_tekstowy_adresu* oznacza adres, który ma zostać pokazany na mapie).

Etap 2. Obsługa argumentów wiersza poleceń

W pliku programu wprowadź przedstawiony poniżej fragment kodu.

```
#!/python3
# mapIt.py — Wyświetla w przeglądarce WWW mapę na podstawie adresu
# podanego w wierszu poleceń lub w schowku.

import webbrowser, sys
if len(sys.argv) > 1:
    # Pobranie adresu z wiersza poleceń.
    address = ' '.join(sys.argv[1:])

# TODO: Pobranie adresu ze schowka.
```

Po wierszu shebang (#!) omawianego programu importujemy moduł `webbrowser`, aby mieć możliwość uruchomienia przeglądarki. Ponadto importujemy moduł `sys`, który pozwoli na odczyt potencjalnych argumentów wiersza poleceń.

Zmienna `sys.argv` przechowuje listę składającą się z nazwy pliku programu oraz argumentów podanych w wierszu poleceń. Jeżeli ta lista będzie zawierała coś więcej niż tylko nazwę pliku skryptu, wówczas wartością zwrótną wywołania `len(sys.argv)` będzie liczba całkowita większa niż 1. To oznacza, że użytkownik podał argumenty w wierszu poleceń.

Argumenty w wierszu poleceń są zwykle rozdzielone spacjami, choć w omawianym przypadku chcemy zinterpretować je wszystkie jako pojedynczy ciąg tekstowy. Ponieważ zmienna `sys.argv` to lista ciągów tekstowych, można ją przekazać metodzie `join()`, której wartością zwrótną jest pojedynczy ciąg tekstowy. Nie chcemy nazwy programu w tym ciągu tekstowym, więc zamiast po prostu `sys.argv` przekazujemy `sys.argv[1:]` i tym samym pozbywamy się pierwszego elementu listy. Ostateczny ciąg tekstowy będzie przechowywany w zmiennej o nazwie `address`.

Jeżeli teraz uruchomisz program za pomocą poniższego polecenia:

```
mapit 870 Valencia St, San Francisco, CA 94110
```

wówczas lista przechowywana w zmiennej `sys.argv` będzie miała następującą postać:

```
['mapIt.py', '870', 'Valencia', 'St, ', 'San', 'Francisco, ', 'CA', '94110']
```

Z kolei wartością zmiennej `address` będzie ciąg tekstowy `'870 Valenci St, San Francisco, CA 94110'`.

Etap 3. Obsługa zawartości schowka i uruchomienie przeglądarki WWW

Wprowadź w kodzie programu przedstawione poniżej zmiany.

```
#!/ python3
# mapIt.py — Wyświetla w przeglądarce WWW mapę na podstawie adresu
# podanego w wierszu poleceń lub w schowku.

import webbrowser, sys, pyperclip
if len(sys.argv) > 1:
    # Pobranie adresu z wiersza poleceń.
    address = ' '.join(sys.argv[1:])
else:
    # Pobranie adresu ze schowka.
    address = pyperclip.paste()

webbrowser.open('https://www.google.pl/maps/place/' + address)
```

Jeżeli w wierszu poleceń nie zostały podane żadne argumenty, program przyjmie założenie, że odpowiedni adres znajduje się w schowku. Zawartość schowka można pobrać za pomocą wywołania `pyperclip.paste()` i przechowywać ją

w zmiennej o nazwie `address`. Na końcu używamy wywołania `webbrowser.open()` do uruchomienia przeglądarki WWW wraz z adresem URL dla aplikacji Mapy Google.

Choć niektóre tworzone programy będą wykonywały ogromne zadania i pozwolą Ci uniknąć wielu godzin pracy, to równie satysfakcjonujące jest korzystanie z programu pozwalającego zaoszczędzić kilka sekund czasu podczas wykonywania często powtarzającego się zadania, takiego jak wyświetlenie danego adresu na mapie. W tabeli 11.1 wymieniłem kroki niezbędne do wykonania, aby wspomniany adres na mapie wyświetlić bez użycia programu `mapIt.py`.

Tabela 11.1. Wyświetlenie adresu na mapie z użyciem `mapIt.py` i bez zastosowania tego programu

| Ręczne wyświetlenie adresu na mapie | Użycie <code>mapIt.py</code> |
|--|---|
| Zaznaczenie adresu. | Zaznaczenie adresu. |
| Skopiowanie adresu. | Skopiowanie adresu. |
| Otworzenie przeglądarki WWW. | Uruchomienie programu <code>mapIt.py</code> . |
| Przejdźcie pod adres URL http://maps.google.pl/ . | |
| Kliknięcie pola tekstowego adresu. | |
| Wklejenie adresu. | |
| Naciśnięcie klawisza <code>Enter</code> . | |

Czy dostrzegasz już, jak program `mapIt.py` pomaga podczas wykonywania omawianego zadania?

Pomysły na podobne programy

Kiedy masz adres URL, moduł `webbrowser` pozwala wyeliminować krok, jakim jest otworenie przeglądarki WWW i przejście pod wskazany adres URL. Poniżej wymieniłem inne programy, które mogą wykorzystać tego rodzaju funkcjonalność.

- Otworzenie w oddzielnych kartach przeglądarki WWW wszystkich łączy znalezionych na stronie.
- Otworzenie przeglądarki WWW i przejście na stronę z prognozą pogody.
- Otworzenie kilku witryn serwisów społecznościowych, z których najczęściej korzystasz.

Pobieranie plików z internetu za pomocą modułu `requests`

Moduł `requests` pozwala na łatwe pobieranie plików z internetu bez konieczności zajmowania się bardziej skomplikowanymi kwestiami, takimi jak błędy sieciowe, problemy z połączeniem i kompresja danych. Moduł `requests` nie jest dostarczany

wraz z Pythonem, więc najpierw trzeba go zainstalować. Z poziomu wiersza poleceń wydaj polecenie `pip install requests`. (W dodatku A znajdziesz więcej informacji na temat instalacji modułów opracowanych przez firmy trzecie).

Moduł `requests` powstał, ponieważ oferowana przez Pythona biblioteka `urllib2` jest zbyt skomplikowana w użyciu. Myślę, że powinieneś wziąć czarny mazak i zamazać ten cały akapit. Zapomnij, że kiedykolwiek wspomniałem o `urllib2`. Jeżeli w programie Pythona trzeba pobrać dane z internetu, po prostu skorzystaj z modułu `requests`.

Teraz przeprowadzimy prosty test w celu sprawdzenia, czy moduł `requests` został prawidłowo zainstalowany. W powłoce interaktywnej wprowadź przedstawione poniżej polecenie.

```
>>> import requests
```

Jeżeli nie zostanie wyświetlony żaden komunikat błędu, to oznacza, że moduł `request` jest zainstalowany poprawnie.

Pobieranie strony internetowej za pomocą funkcji `requests.get()`

Funkcja `requests.get()` pobiera ciąg tekstowy zawierający adres URL przeznaczony do pobrania. Gdy wywołane zostaje `type()` na wartości zwrótej funkcji `requests.get()`, możesz sprawdzić, czy otrzymałeś obiekt `Response`. Obiekt zawiera odpowiedź udzieloną przez serwer WWW na wykonane do niego żądanie. Szczegółowe omówienie obiektu `Response` znajdziesz dalej w tym rozdziale. Natomiast teraz, skoro Twój komputer ma połączenie z internetem, w powłoce interaktywnej wprowadź przedstawione poniżej polecenia.

```
>>> import requests
>>> res = requests.get('http://www.gutenberg.org/files/27062/27062-0.txt') ❶
>>> type(res)
<class 'requests.models.Response'>
>>> res.status_code == requests.codes.ok ❷
True
>>> len(res.text)
167343
>>> print(res.text[:250])
The Project Gutenberg EBook of Romeo i Julia, by William Shakespeare
```

```
This eBook is for the use of anyone anywhere at no cost and with
almost no restrictions whatsoever. You may copy it, give it away or
re-use it under the terms of the Project
```

Podany adres URL prowadzi do tekstowej wersji sztuki *Romeo i Julia* dostępnej w witrynie projektu Gutenberg ❶. Jeżeli chcesz dowiedzieć się, czy żądanie internetowe do wskazanej strony zakończyło się powodzeniem, sprawdź atrybut `status_code` obiektu `Response`. Jeżeli wartością będzie `requests.codes.ok`, wówczas masz pewność, że wszystko przebiegło dobrze ❷. (W protokole HTTP kod

oznaczający zakończenie operacji powodzeniem to 200. Prawdopodobnie znasz także kod 404, który oznacza, że wskazany zasób nie został znaleziony).

Jeżeli żądanie zakończy się powodzeniem, pobrana strona będzie przechowywana w postaci ciągu tekstowego w zmiennej `text` obiektu `Response`. Wymieniona zmienna przechowuje ogromny ciąg tekstowy zawierający tekst całej sztuki. Wywołanie funkcji `len(res.text)` wskazuje, że ciąg zawiera prawie 170000 znaków. Na końcu wywołanie `print(Res.text[:250])` wyświetla jedynie pierwsze 250 znaków tekstu.

Sprawdzenie pod kątem błędów

Jak wcześniej widziałeś, obiekt `Response` ma atrybut `status_code`, którego wartość można sprawdzić. Jeżeli jest nią `requests.codes.ok`, oznacza to, że żądanie zakończyło się powodzeniem. Prościej sposobem sprawdzenia sukcesu jest wywołanie metody `raise_for_status()` obiektu `Response`. Metoda zgłosi wyjątek, jeśli nastąpi błąd podczas pobierania pliku. Gdy pobieranie zakończy się powodzeniem, nic nie zostanie zgłoszone. W powłoce interaktywnej wprowadź przedstawione poniżej polecenia.

```
>>> res = requests.get('http://inventwithpython.com/strona_ktora_nie_istnieje')
>>> res.raise_for_status()
Traceback (most recent call last):
  File "<pyshell#138>", line 1, in <module>
    res.raise_for_status()
  File "C:\Python34\lib\site-packages\requests\models.py", line 773,
    ↪in raise_for_status
    raise HTTPError(http_error_msg, response=self)
requests.exceptions.HTTPError: 404 Client Error: Not Found
```

Metoda `raise_for_status()` to dobry sposób na zagwarantowanie zatrzymania działania programu w przypadku nieprawidłowego pobrania danych. To dobre rozwiązanie, bp program powinien zatrzymać swoje działanie w przypadku wystąpienia pewnego nieoczekiwanego błędu. Jeżeli pobranie nieprawidłowych danych *nie stanowi* problemu dla programu, zawsze możesz opakować wywołanie `raise_for_status()` poleceniami `try` i `except` w celu obsługi tego rodzaju błędu bez powodowania awarii całego programu. Spójrz na poniższy fragment kodu.

```
import requests
res = requests.get('http://inventwithpython.com/strona_ktora_nie_istnieje')
try:
    res.raise_for_status()
except Exception as exc:
    print('Wystąpił następujący problem: %s' % (exc))
```

Powyższe wywołanie metody `raise_for_status()` spowoduje wygenerowanie przez program przedstawionych poniżej danych wyjściowych.

Wystąpił następujący problem: błąd po stronie klienta 404: nie znaleziono zasobu.

Zawsze stosuj wywołania `raise_for_status()` po wywołaniu funkcji `requests.get()`. Powinieneś mieć pewność o prawidłowym pobraniu danych, zanim program będzie kontynuował pracę.

Zapis pobranych plików na dysku twardym

Od tego miejsca, za pomocą metod `open()` i `write()` możesz zapisywać strony internetowe do plików umieszczonych na dysku twardym komputera. Jednak istnieją pewne drobne różnice. Przede wszystkim konieczne jest otworenie pliku w *trybie binarym*, co wymaga przekazania ciągu tekstowego `'wb'` jako drugiego argumentu metody `open()`. Jeśli nawet strona jest w postaci zwykłego tekstu (na przykład pobranego wcześniej pliku ze sztuką *Romeo i Julia*), to i tak konieczne jest zapisanie danych binarnych zamiast tekstowych, aby zachować *kodowanie Unicode* tego tekstu.

KODOWANIE UNICODE

Omówienie kodowania Unicode wykracza poza zakres tematyczny tej książki. Więcej informacji na ten temat znajdziesz w wymienionych poniżej artykułach.

- *Joel on Software: The Absolute Minimum Every Software Developer Absolutely, Positively Must Know About Unicode and Character Sets (No Excuses!)* na stronie <http://www.joelonsoftware.com/articles/Unicode.html>.
- *Pragmatic Unicode* na stronie <http://nedbatchelder.com/text/unipain.html>.

W celu zapisania strony internetowej do pliku można użyć pętli `for` wraz z metodą `iter_content()` obiektu `Response`, tak jak pokazałem poniżej.

```
>>> import requests
>>> res = requests.get('http://www.gutenberg.org/files/27062/27062-0.txt')
>>> res.raise_for_status()
>>> playFile = open('Romeo-i-Julia.txt', 'wb')
>>> for chunk in res.iter_content(100000):
>>>     playFile.write(chunk)

100000
67343
>>> playFile.close()
```

Wartością zwrótną metody `iter_content()` są „fragmenty” treści przetwarzane podczas każdej iteracji pętli. Każdy fragment to po prostu dane typu *bajty*, metoda podaje ilość bajtów znajdujących się w poszczególnych fragmentach.

Sto tysięcy bajtów to rozsądna wielkość, więc przekazujemy wartość 100000 jako argument funkcji `iter_content()`.

Plik *Romeo-i-Julia.txt* teraz już istnieje w bieżącym katalogu roboczym. Zwróć uwagę, że nazwa pliku na stronie internetowej to *27062-0.txt*, natomiast zapisany na dysku twardym ma zupełnie inną nazwę. Obsługą pobierania zawartości stron internetowych zajmuje się moduł `requests`. Strona po pobraniu stanowi po prostu dane w programie. Jeśli nawet utracisz połączenie z internetem po pobraniu strony internetowej, wszystkie dane strony nadal pozostaną w komputerze.

Wartością zwrótną metody `write()` jest liczba bajtów zapisanych w pliku. W przedstawionym powyżej przykładzie pierwszy fragment składał się ze 100000 bajtów, natomiast pozostała część wymagała jedynie 67343 bajtów.

Poniżej przedstawiam pełny proces pobierania i zapisu pliku.

1. Wywołanie funkcji `requests.get()` w celu pobrania pliku.
2. Wywołanie funkcji `open()` wraz z argumentem `'wb'` w celu utworzenia nowego pliku w trybie binarnym.
3. Iteracja przez obiekt `Response` za pomocą metody `iter_content()`.
4. Wywołanie `write()` w trakcie każdej iteracji, aby umieścić treść w pliku.
5. Wywołanie funkcji `close()` w celu zamknięcia pliku.

To już wszystko, co dotyczy modułu `requests`! Pętla `for` i metoda `iter_content()` mogą wydawać się skomplikowane w porównaniu z opartym na funkcjach `open()`, `write()` i `close()` rozwiązaniem, którego używamy podczas pracy z plikami tekstowymi. Jednak przedstawione podejście gwarantuje, że moduł `requests` nie będzie zużywał zbyt dużej ilości pamięci nawet podczas pobierania ogromnych plików. Więcej informacji na temat innych funkcji modułu `requests` znajdziesz na stronie <http://requests.readthedocs.io/en/master/>.

HTML

Zanim będziesz mógł zająć się poważniej stronami internetowymi, najpierw powinieneś poznać podstawy języka HTML. Zobaczysz również, jak uzyskać dostęp do oferowanych przez przeglądarkę WWW użytecznych narzędzi programistycznych, z pomocą których pobieranie danych z internetu stanie się jeszcze łatwiejsze.

Zasoby pomagające w poznawaniu języka HTML

Hipertekstowy język znaczników (ang. *hypertext markup language*, czyli **HTML**) to format, w którym są zapisywane strony internetowe. W tym rozdziale przyjąłem założenie, że znasz HTML przynajmniej w zakresie podstawowym. Jeżeli mimo wszystko potrzebujesz pewnego wprowadzenia, sugeruję zapoznanie się z materiałem przedstawionym na jednej z wymienionych poniżej stron internetowych.

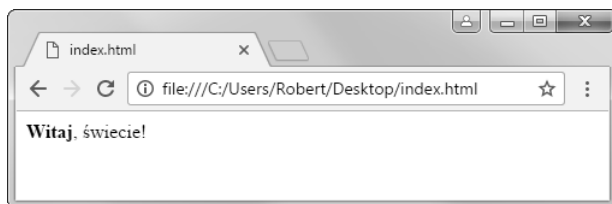
- <http://htmldog.com/guides/html/beginner/>
- <https://www.codecademy.com/learn/web>
- <https://developer.mozilla.org/en-US/docs/Learn/HTML>

Krótkie wprowadzenie

Kiedy minęło już sporo czasu od chwili, gdy miałeś okazję analizować jakikolwiek kod HTML, tutaj znajdziesz krótkie omówienie podstaw. Plik w formacie HTML jest plikiem zwykłego tekstu wraz z rozszerzeniem *.html*. Tekst w tego rodzaju pliku jest ujęty w tak zwane *znaczniki*, którymi są po prostu słowa umieszczone w nawiasach ostrych. Znaczniki wskazują przeglądarce WWW sposób formatowania strony internetowej. Znaczniki otwierający i zamykający mogą oblewać pewien tekst i tym samym tworzą tak zwany *element*. Z kolei *tekst* (lub *wewnętrzny HTML*) to zawartość umieszczona między znacznikami otwierającym i zamykającym. Przykładowo poniższy fragment kodu HTML powoduje wyświetlenie w przeglądarce WWW komunikatu *Witaj, świecie!*, przy czym słowo *Witaj* będzie pogrubione.

```
<strong>Witaj</strong>, świecie!
```

Po wygenerowaniu przez przeglądarkę WWW tekst ten będzie wyglądał tak, jak pokazałem na rysunku 11.1.



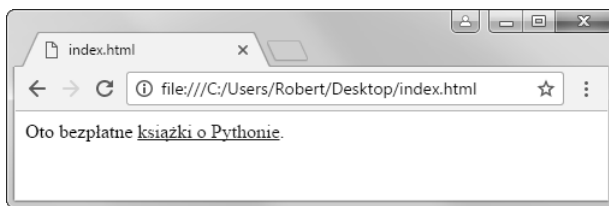
Rysunek 11.1. Komunikat „Witaj, świecie!” wyświetlony przez przeglądarkę WWW

Znacznik otwierający `` informuje, że obejmowany przez niego tekst ma zostać pogrubiony. Z kolei znacznik zamykający `` wskazuje przeglądarce WWW, gdzie kończy się pogrubiony tekst.

W języku HTML mamy wiele różnych znaczników. Część z nich obsługuje dodatkowe właściwości w postaci *atrybutów* definiowanych wewnątrz nawiasu ostrego. Przykładowo znacznik `<a>` zawiera tekst, który powinien być wyświetlony w postaci łącza. Adres URL dla tego łącza jest określany przez atrybut `href`. Poniżej przedstawiłem przykład.

```
Oto bezpłatne <a href="http://inventwithpython.com">książki o Pythonie</a>.
```

Po wygenerowaniu przez przeglądarkę WWW tekst ten będzie wyglądał tak, jak pokazałem na rysunku 11.2.



Rysunek 11.2. Łącze wygenerowane przez przeglądarkę WWW

Niektóre elementy mają atrybut `id` używany do unikatowej identyfikacji elementu na stronie. Bardzo często będziesz nakazywać programowi wyszukiwanie elementów na podstawie ich atrybutów `id`. Dlatego też ustalenie tego atrybutu za pomocą wbudowanych w przeglądarkę WWW narzędzi programistycznych jest dość często spotykanym zadaniem podczas tworzenia programów pobierających dane z internetu.

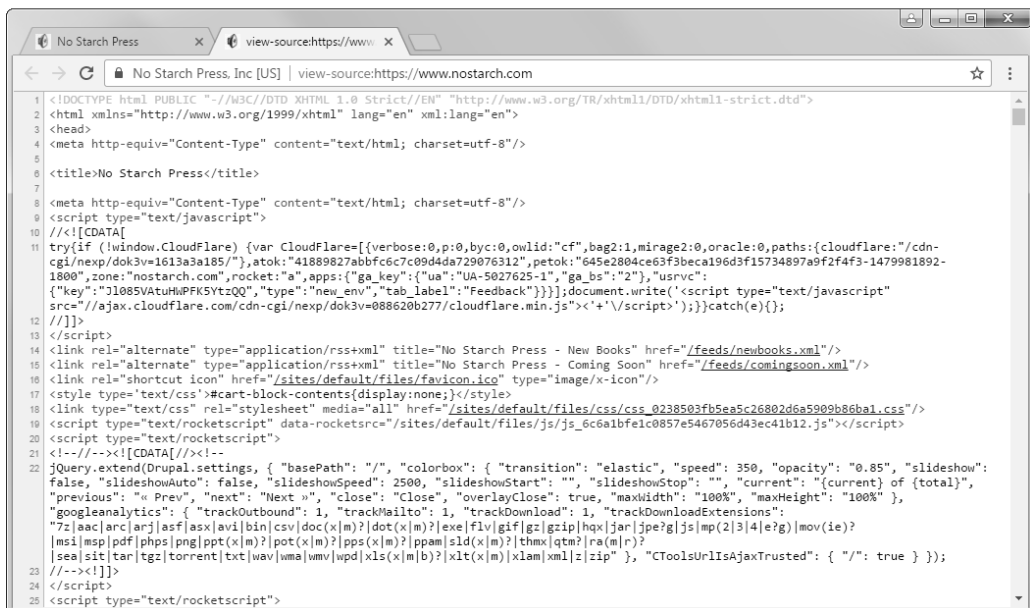
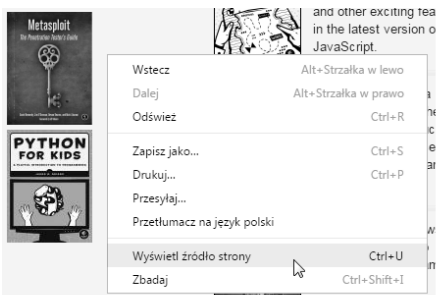
Wyświetlenie kodu źródłowego HTML strony internetowej

Nierzadko będzie występowała konieczność przeanalizowania kodu źródłowego HTML stron internetowych, z którymi współpracują tworzone przez Ciebie programy. W tym celu kliknij prawym przyciskiem myszy dowolną stronę internetową w przeglądarce WWW, a następnie wybierz opcję *Wyświetl źródło* lub *Wyświetl źródło strony*, aby faktycznie zobaczyć kod HTML tej strony (patrz rysunek 11.3). To jest tekst w rzeczywistości otrzymywany przez przeglądarkę WWW. Trzeba w tym miejscu dodać, że przeglądarka WWW „wie”, w jaki sposób wyświetlić, inaczej *wygenerować*, stronę internetową na podstawie kodu HTML.

Gorąco zachęcam do wyświetlenia kodu źródłowego HTML jednej z Twoich ulubionych stron internetowych. Nie przejmuj się, jeśli nie zrozumiesz w pełni tego, co zobaczysz, patrząc na kod źródłowy. Nie musisz być mistrzem w tworzeniu kodu HTML, aby pisać proste programy pobierające dane z internetu — przecież nie zamierzasz samodzielnie budować witryn internetowych. Musisz jedynie wiedzieć, jak pobrać dane z istniejącej witryny.

Wyświetlenie oferowanych przez przeglądarkę WWW narzędzi programistycznych

Kod źródłowy strony możesz wyświetlić również za pomocą narzędzi programistycznych wbudowanych w przeglądarkę WWW. Przykładowo w przeglądarkach Chrome i Internet Explorer dla Windows narzędzia programistyczne są zainstalowane. Wystarczy nacisnąć klawisz `F12`, aby zostały wyświetlone na ekranie (patrz rysunek 11.4). Ponowne naciśnięcie klawisza `F12` zamyka te narzędzia. W przeglądarce Chrome narzędzia programistyczne można wyświetlić za pomocą opcji menu *Więcej narzędzi/Narzędzia dla programistów*. Z kolei w systemie macOS trzeba nacisnąć klawisze `Command+Option+I`, aby wyświetlić narzędzia programistyczne wbudowane w przeglądarkę Chrome.



Rysunek 11.3. Wyświetlenie kodu źródłowego strony internetowej

Jeżeli używasz przeglądarki Firefox, narzędzia programistyczne zostaną wyświetlone po naciśnięciu klawiszy *Ctrl+Shift+C* (platformy Windows i Linux) lub *Command+Option+C* (platforma macOS). Układ tych narzędzi jest niemal identyczny z układem narzędzi wbudowanych w przeglądarkę Chrome.

W przeglądarce Safari przejdź do karty *Zaawansowane* okna preferencji, a następnie zaznacz pole wyboru *Pokazuj menu Programowanie na pasku menu*. Po włączeniu wymienionego menu narzędzia programistyczne będziesz mógł wyświetlić po naciśnięciu klawiszy *Command+Option+I*.

Po włączeniu lub zainstalowaniu narzędzi programistycznych w przeglądarce WWW możesz kliknąć dowolny element strony internetowej, a następnie z menu kontekstowego wybrać opcję *Skontroluj element*. W ten sposób przejdiesz do fragmentu kodu HTML odpowiedzialnego za wygenerowanie klikniętego elementu strony. Ta możliwość okaże się użyteczna, gdy rozpoczniemy przetwarzanie kodu HTML w programach pobierających dane z internetu.



Rysunek 11.4. Okno narzędzi programistycznych w przeglądarce Chrome

NIE UŻYWAJ WYRAŻEŃ REGULARNYCH DO PRZETWARZANIA KODU HTML

Odszukanie określonego fragmentu kodu HTML w ciągu tekstowym wydaje się idealnym zadaniem dla wyrażeń regularnych. Jednak odradzam takie podejście. Istnieje znacznie więcej różnych sposobów, na jakie może być formatowany kod HTML, który nadal będzie uznany za prawidłowy. Próba dopasowania tych wszystkich możliwych odmian za pomocą wyrażenia regularnego może być żmudna i podatna na błędy. Moduł opracowany specjalnie do przetwarzania kodu HTML nosi nazwę BeautifulSoup i otrzymane za jego pomocą wyniki będą prawdopodobnie miały znacznie mniej błędów.

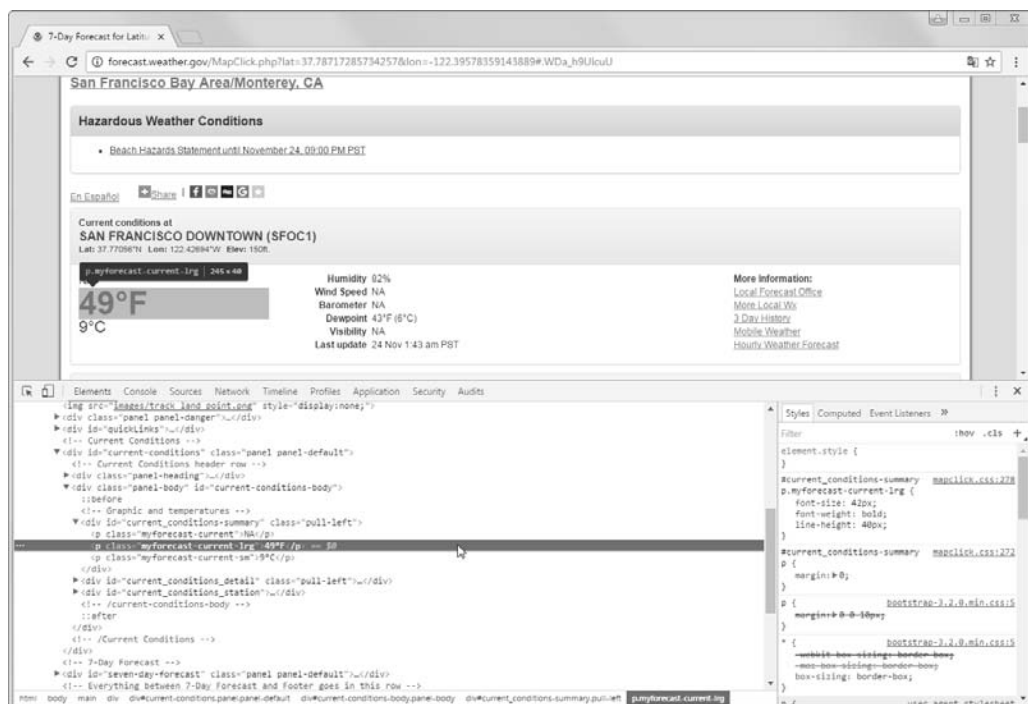
Więcej dokładniejszych informacji na temat tego, dlaczego nie powinniśmy przetwarzać kodu HTML za pomocą wyrażeń regularnych znajdziesz na stronie <http://stackoverflow.com/questions/1732348/regex-match-open-tags-except-xhtml-self-contained-tags/1732454#1732454>.

Użycie narzędzi programistycznych do wyszukiwania elementów HTML

Kiedy Twój program pobierze stronę internetową za pomocą modułu requests, treść tej strony otrzymasz w postaci jednego dużego ciągu tekstowego. Teraz musisz ustalić, które fragmenty kodu HTML odpowiadają interesującym Cię informacjom na pobranej stronie internetowej.

Tutaj z pomocą mogą przyjść wbudowane w przeglądarkę WWW narzędzia programistyczne. Przyjmujemy założenie, że chcesz utworzyć program pobierający z witryny <http://www.weather.gov/> dane dotyczące prognozy pogody. Przed napisaniem jakiegokolwiek kodu musisz przeprowadzić małe badania. Jeżeli odwiedź wymienioną witrynę i podasz kod pocztowy 94105, przejdziesz na stronę zawierającą prognozę pogody dla wymienionego regionu.

Co zrobić w sytuacji, gdy jesteś zainteresowany pobraniem informacji o temperaturze dla regionu o podanym kodzie pocztowym? Kliknij prawym przyciskiem myszy tę informację na stronie, a następnie z wyświetlonego menu kontekstowego wybierz opcję *Zbadaj*. Na ekranie pojawi się okno narzędzi programistycznych, które będzie wyświetlało kod HTML odpowiedzialny za wygenerowanie klikniętego fragmentu strony. Na rysunku 11.5 pokazałem narzędzia programistyczne wraz z kodem HTML generującym fragment strony wyświetlający temperaturę.



Rysunek 11.5. Użycie narzędzi programistycznych do analizy elementu wyświetlającego temperaturę

W oknie narzędzi programistycznych możesz zobaczyć, że kod HTML odpowiedzialny za wyświetlenie temperatury na tej stronie internetowej to `<p class="myforecast-current -lrg">49°F</p>`. To jest dokładnie to, czego szukamy! Wydaje się, że informacje o temperaturze są umieszczone w elemencie `<p>` o klasie `myforecast-current-lrg`. Skoro już wiesz, czego szukasz, moduł Beautiful Soup pomoże Ci w odszukaniu właściwego ciągu tekstowego.

Przetwarzanie kodu HTML za pomocą modułu Beautiful Soup

BeautifulSoup to moduł przeznaczony do wyodrębniania informacji ze strony HTML (sprawdza się w tym znacznie lepiej niż wyrażenia regularne). W kodzie Pythona nazwą tego modułu jest `bs4` (skrót od Beautiful Soup, version 4). W celu zainstalowania modułu należy z poziomu wiersza poleceń wydać polecenie `pip install beautifulsoup4`. (W dodatku A znajdziesz więcej informacji na temat instalacji modułów opracowanych przez firmy trzecie). Wprawdzie `beautifulsoup4` to nazwa używana podczas instalacji, ale w kodzie Pythona stosowana jest nazwa `bs4`, dlatego też, aby zaimportować moduł, w kodzie należy wydać polecenie `import bs4`.

W przykładach omawianych w tym rozdziale za pomocą modułu BeautifulSoup będziemy *przetwarzać* (to znaczy analizować i identyfikować) fragmenty pliku HTML znajdującego się na dysku twardym. Otwórz nowe okno edytora pliku i wprowadź poniższy fragment kodu. Następnie zapisz plik pod nazwą `example.html`. Ewentualnie możesz pobrać ten plik ze strony <ftp://ftp.helion.pl/przyklady/autopy.zip>.

```
<!-- To jest przykładowy plik example.html. -->

<html><head><title>Tytuł mojej witryny internetowej</title></head>
<body>
<p>Pobierz książkę o języku <strong>Python</strong> z <a href="http://
inventwithpython.com">mojej witryny internetowej</a>.</p>
<p class="slogan">Poznaj Pythona w łatwy sposób!</p>
<p>Autor <span id="author">Al Sweigart</span></p>
</body></html>
```

Jak możesz zobaczyć, nawet prosty plik HTML zawiera wiele różnych znaczników i atrybutów. W skomplikowanych witrynach internetowych kod bardzo szybko może stać się zagmatwany. Na szczęście moduł BeautifulSoup niezwykle ułatwia pracę z kodem HTML.

Utworzenie obiektu BeautifulSoup na podstawie kodu HTML

Funkcja `bs4.BeautifulSoup()` musi być wywołana wraz z ciągiem tekstowym zawierającym kod HTML przeznaczony do przetworzenia. Wartością zwrótną tej funkcji jest obiekt BeautifulSoup. W powłoce interaktywnej komputera, który ma połączenie z internetem, wprowadź przedstawione poniżej polecenia.

```
>>> import requests, bs4
>>> res = requests.get('http://nostarch.com')
>>> res.raise_for_status()
>>> noStarchSoup = bs4.BeautifulSoup(res.text)
>>> type(noStarchSoup)
<class 'bs4.BeautifulSoup'>
```

W powyższym fragmencie kodu użyliśmy wywołania `requests.get()` w celu pobrania strony głównej z witryny No Starch Press, a następnie przekazaliśmy funkcji `bs4.BeautifulSoup()` atrybut `text` otrzymanej odpowiedzi. Zwrócony obiekt `BeautifulSoup` jest przechowywany w zmiennej o nazwie `noStarchSoup`.

Istnieje również możliwość wczytania kodu HTML z pliku znajdującego się na dysku twardym komputera. To wymaga przekazania obiektu `File` do wywołania `bs4.BeautifulSoup()`. W powłoce interaktywnej wprowadź przedstawione poniżej polecenia (upewnij się, że bieżący katalog roboczy zawiera plik o nazwie `example.html`).

```
>>> exampleFile = open('example.html')
>>> exampleSoup = bs4.BeautifulSoup(exampleFile)
>>> type(exampleSoup)
<class 'bs4.BeautifulSoup'>
```

Po otrzymaniu obiektu `BeautifulSoup` można wykorzystać jego metody w celu wyszukiwania konkretnych fragmentów dokumentu HTML.

Wyszukiwanie elementu za pomocą metody `select()`

Element strony internetowej można pobrać z obiektu `BeautifulSoup` za pomocą wywołania metody `select()` i przekazania jej ciągu tekstowego w postaci *selektora* CSS szukanego elementu. Pod pewnymi względami selektory przypominają wyrażenia regularne — określają szukany wzorzec, w tym przypadku na stronach HTML, a nie w ogólnych ciągach tekstowych.

Pełne omówienie składni selektorów CSS wykracza poza zakres tematyczny tej książki². Poniżej przedstawiłem tylko kilka krótkich informacji o selektorach. W tabeli 11.2 wymieniłem najczęściej stosowane wzorce selektorów CSS.

Różne wzorce selektorów można ze sobą łączyć i tym samym zdefiniować znacznie bardziej skomplikowane dopasowania. Przykładowo `soup.select('p #author')` spowoduje dopasowanie dowolnego elementu, który ma atrybut `id` o wartości `author`, o ile znajduje się wewnątrz elementu `<p>`.

Metoda `select()` zwróci listę obiektów `Tag`, za pomocą których moduł `BeautifulSoup` przedstawia element HTML. Lista będzie zawierała po jednym obiekcie

² Dobre wprowadzenie do tematu selektorów znajdziesz na stronach <https://www.w3.org/TR/CSS2/selector.html>, <http://ryanstutorials.net/css-tutorial/css-selectors.php> i <http://htmldog.com/guides/css/beginner/selectors/>.

Tabela 11.2. Przykłady selektorów CSS

| Selektor przekazany metodzie select() | Dopasowuje... |
|--|---|
| <code>soup.select('div')</code> | Wszystkie elementy o nazwie <code><div></code> . |
| <code>soup.select('#author')</code> | Element, którego atrybut <code>id</code> ma wartość <code>author</code> . |
| <code>soup.select('.notice')</code> | Wszystkie elementy używające atrybutu CSS <code>class</code> o wartości <code>notice</code> . |
| <code>soup.select('div span')</code> | Wszystkie elementy o nazwie <code></code> , które zostały umieszczone w elemencie o nazwie <code><div></code> . |
| <code>soup.select('div > span')</code> | Wszystkie elementy o nazwie <code></code> , które są <i>bezpośrednio</i> w elemencie o nazwie <code><div></code> i między nimi nie istnieje żaden inny element. |
| <code>soup.select('input[name]')</code> | Wszystkie elementy o nazwie <code><input></code> , które mają atrybut <code>name</code> o dowolnej wartości. |
| <code>soup.select('input[type="button"]')</code> | Wszystkie elementy o nazwie <code><input></code> , które mają atrybut <code>type</code> o wartości <code>button</code> . |

Tag dla każdego dopasowania w obiekcie `BeautifulSoup`. Wartości obiektów `Tag` mogą być przekazywane funkcji `str()` w celu wyświetlenia przedstawianych przez nie znaczników HTML. Wartości obiektów `Tag` mogą mieć również atrybut `attrs` pokazujący podane w postaci słownika wszystkie atrybuty HTML danego znacznika. Używając przygotowanego wcześniej pliku o nazwie `example.html`, w powłoce interaktywnej wprowadź przedstawione poniżej polecenia.

```
>>> import bs4
>>> exampleFile = open('example.html')
>>> exampleSoup = bs4.BeautifulSoup(exampleFile.read())
>>> elems = exampleSoup.select('#author')
>>> type(elems)
<class 'list'>
>>> len(elems)
1
>>> type(elems[0])
<class 'bs4.element.Tag'>
>>> elems[0].getText()
'Al Sweigart'
>>> str(elems[0])
'<span id="author">Al Sweigart</span>'
>>> elems[0].attrs
{'id': 'author'}
```

Powyższy fragment kodu wyciągnie z naszego przykładowego pliku HTML element o atrybucie `id="author"`. Wywołania `select('#author')` używamy do zwrócenia listy wszystkich elementów wraz z atrybutem `id="author"`. Listę obiektów `Tag` przechowujemy w zmiennej o nazwie `elems`. Wywołanie `len(elems)` pokazuje, że mamy tylko jeden obiekt `Tag` na liście, czyli było tylko jedno dopasowanie. Wywołanie `getText()` w elemencie zwraca jego tekst lub wewnętrzny kod HTML tego elementu. Tekstem elementu jest treść znajdująca się między znacznikami otwierającym i zamykającym. W omawianym przykładzie to będzie `'Al Sweigart'`.

W wyniku przekazania elementu do funkcji `str()` zwracany jest ciąg tekstowy zawierający znaczniki otwierający i zamykający oraz tekst elementu. Na końcu `attrs` daje słownik wraz z atrybutem elementu, tutaj `'id'`, oraz jego wartości, czyli `'author'`.

Z obiektu `BeautifulSoup` można wyodrębnić również elementy `<p>`. W powłoce interaktywnej wprowadź przedstawione poniżej polecenia.

```
>>> pElems = exampleSoup.select('p')
>>> str(pElems[0])
'<p>Pobierz książkę o języku <strong>Python</strong> z <a href="http://
inventwithpython.com">mojej witryny internetowej</a>.</p>'
>>> pElems[0].getText()
'Pobierz książkę o języku Python z mojej witryny internetowej.'
>>> str(pElems[1])
'<p class="slogan">Poznaj Pythona w łatwy sposób!</p>'
>>> pElems[1].getText()
'Poznaj Pythona w łatwy sposób!'
>>> str(pElems[2])
'<p>Autor <span id="author">Al Sweigart</span></p>'
>>> pElems[2].getText()
'Author Al Sweigart'
```

Tym razem wywołanie `select()` daje listę trzech dopasowań, które są przechowywane w zmiennej `pElems`. Za pomocą funkcji `str()` użytej wraz z `pElems[0]`, `pElems[1]` i `pElems[2]` możemy wyświetlić każdy element jako ciąg tekstowy. Natomiast wywołanie `getText()` w poszczególnych elementach wyświetla znajdujący się w nich tekst.

Pobieranie danych z atrybutów elementu

Metoda `get()` w obiektach `Tag` znacznie ułatwia uzyskanie dostępu do wartości atrybutów w danym elemencie. Wymienionej metodzie przekazujemy ciąg tekstowy nazwy atrybutu, która z kolei zwraca wartość tego atrybutu. Używając przygotowanego wcześniej pliku o nazwie `example.html`, w powłoce interaktywnej wprowadź przedstawione poniżej polecenia.

```
>>> import bs4
>>> soup = bs4.BeautifulSoup(open('example.html'))
>>> spanElem = soup.select('span')[0]
>>> str(spanElem)
'<span id="author">Al Sweigart</span>'
>>> spanElem.get('id')
'author'
>>> spanElem.get('some_nonexistent_addr') == None
True
>>> spanElem.attrs
{'id': 'author'}
```

W powyższym fragmencie kodu metoda `select()` wyszukała wszystkie elementy ``, a następnie pierwszy dopasowany umieściła w zmiennej `spanElem`. Przekazanie funkcji `get()` nazwy atrybutu `'id'` powoduje zwrot jego wartości, czyli `'author'`.

Projekt — wyszukiwanie typu „szczęśliwy traf” w Google

Kiedy szukam informacji na dany temat za pomocą wyszukiwarki internetowej Google, nie przeglądam po kolei otrzymanych wyników wyszukiwania. Klikając łącza prawym przyciskiem myszy (lub z naciśniętym klawiszem *Ctrl*), otwieram kilka pierwszych wyników w nowych kartach, aby później się z nimi zapoznać. Z wyszukiwarki internetowej Google korzystam na tyle często, że podejście typu: otworzenie przeglądarki WWW, wpisanie szukanego tematu i kolejne otwieranie wyników w nowych kartach — stało się żmudne. Byłoby znacznie lepiej, gdybym mógł wpisać w wierszu poleceń szukany temat, a komputer automatycznie uruchomiłby przeglądarkę i kilka pierwszych wyników otworzył w nowych kartach. Dlatego też teraz zajmiemy się przygotowaniem skryptu, który wykona tego rodzaju zadanie.

Poniżej wymieniałem w punktach sposób działania programu.

- Pobranie słów kluczowych wyszukiwania z argumentów wiersza poleceń.
- Pobranie strony wyników wyszukiwania.
- Otworzenie nowej karty dla każdego znalezionej wyniku.

Oznacza to, że kod będzie musiał wykonywać przedstawione poniżej zadania.

- Odczyt argumentów wiersza poleceń z listy `sys.argv`.
- Pobranie za pomocą modułu `requests` strony wyników wyszukiwania.
- Wyszukanie łącz dla każdego wyniku wyszukiwania.
- Wywołanie funkcji `webbrowser.open()` w celu przejścia do przeglądarki WWW.

Otwórz nowe okno edytora pliku i zapisz plik pod nazwą *lucky.py*.

Etap 1. Pobranie argumentów wiersza poleceń i żądanie strony wyszukiwarki

Zanim w ogóle przejdziemy dalej, najpierw musisz ustalić adres URL strony wyświetlającej wyniki wyszukiwania. Patrząc na pasek adresu przeglądarki WWW po przeprowadzeniu jakiegokolwiek wyszukiwania w Google, możesz ustalić, że interesujący nas adres URL ma postać `http://www.google.pl/search?q=SZUKANE_WYRAŻENIE`. Moduł `requests` może pobrać tę stronę, a następnie możemy wykorzystać moduł `BeautifulSoup` do wyodrębnienia z otrzymanego dokumentu HTML

łączy wyników wyszukiwania. Na koniec użyjemy modułu `webbrowser` do otwarcia wspomnianych łączy na kartach przeglądarki WWW.

W pliku wprowadź przedstawiony poniżej kod.

```
#!/python3
# lucky.py — Otwiera kilka wyników wyszukiwania Google.

import requests, sys, webbrowser, bs4

print('Wyszukiwanie...') # Komunikat wyświetlany podczas pobierania strony Google.
res = requests.get('http://google.pl/search?q=' + ' '.join(sys.argv[1:]))
res.raise_for_status()

# TODO: Pobranie łączy z kilkoma pierwszymi wynikami wyszukiwania.

# TODO: Otworzenie karty przeglądarki WWW dla każdego wyniku wyszukiwania.
```

Użytkownik dostarczy szukane wyrażenia za pomocą argumentów wiersza poleceń podczas uruchamiania programu. Argumenty będą przechowywane na liście `sys.argv` w postaci ciągów tekstowych.

Etap 2. Wyszukanie wszystkich wyników

Teraz będziemy musieli użyć modułu `BeautifulSoup` do wyodrębnienia pierwszych kilku wyników wyszukiwania z pobranego dokumentu HTML. W tym miejscu rodzi się pytanie, jak wybrać odpowiedni selektor do tego zadania? Nie można po prostu wyszukać wszystkich znaczników `<a>`, ponieważ na pobranej stronie znajduje się wiele łączy, które nas nie interesują. Dlatego też konieczne jest przeanalizowanie strony wyników wyszukiwania za pomocą wbudowanych w przeglądarkę WWW narzędzi programistycznych i w ten sposób podjęcie próby ustalenia selektora, który pozwoli na wyodrębnienie jedynie interesujących nas łączy.

Po przeprowadzeniu wyszukiwania w Google wyrażenia *Beautiful Soup* przejdź do narzędzi programistycznych i zajrzyj do dowolnie wybranego elementu łącza na stronie. Tego rodzaju elementy wyglądają na niezwykle skomplikowane i mają postać podobną do przedstawionej `Beautiful Soup: We called him Tortoise because he taught us.`.

Nie ma absolutnie żadnego znaczenia, że element wygląda na bardzo skomplikowany. Musimy po prostu odnaleźć wzorzec, który mają wszystkie łącza przedstawiające wyniki wyszukiwania. Jednak ten element `<a>` nie ma niczego, co łatwo odróżniałoby go od nieinteresujących nas elementów `<a>` na stronie.

Wprowadź w kodzie przedstawione poniżej zmiany.

```
#!/python3
# lucky.py — Otwiera kilka wyników wyszukiwania Google.

import requests, sys, webbrowser, bs4

--cięcie--

# Pobranie łączy z kilkoma pierwszymi wynikami wyszukiwania.
soup = bs4.BeautifulSoup(res.text)

# Otworzenie karty przeglądarki WWW dla każdego wyniku wyszukiwania.
linkElems = soup.select('.r a')
```

Gdy rozejrzysz się nieco w okolicach analizowanego wcześniej elementu <a>, zauważysz element, taki jak <h3 class="r">. Po przejrzaniu pozostałej części kodu źródłowego HTML można dojść do wniosku, że klasa r jest używana jedynie dla łączy zawierających wyniki wyszukiwania. Nie musisz wiedzieć, czym jest klasa CSS o nazwie r lub na czym polega jej działanie. Po prostu użyjemy jej do odnalezienia interesujących nas elementów <a>. Możemy utworzyć obiekt BeautifulSoup na podstawie kodu HTML pobranej strony, a następnie wykorzystać selektor '.r a' do odszukania wszystkich elementów <a> znajdujących się wewnątrz elementów o przypisanej klasie CSS r.

Etap 3. Otworzenie kart przeglądarki WWW dla poszczególnych wyników

Na koniec nakazujemy programowi otworenie kart przeglądarki WWW dla poszczególnych wyników. Na końcu programu wprowadź poniższy fragment kodu.

```
#!/python3
# lucky.py — Otwiera kilka wyników wyszukiwania Google.

import requests, sys, webbrowser, bs4

--cięcie--

# Otworzenie karty przeglądarki WWW dla każdego wyniku wyszukiwania.
linkElems = soup.select('.r a')
numOpen = min(5, len(linkElems))
for i in range(numOpen):
    webbrowser.open('http://google.pl' + linkElems[i].get('href'))
```

Domyślnie w nowych kartach otwieramy jedynie pięć pierwszych wyników wyszukiwania, używając modułu webbrowser. Jednak użytkownik mógł szukać czegoś, co pojawiło się w mniejszej liczbie wyników wyszukiwania. Wywołanie `soup.select()` zwraca listę wszystkich elementów dopasowanych za pomocą

selektora `'r a'`. Dlatego też liczba kart przeznaczonych do otworzenia wynosi 5 lub odpowiada wielkości liczby (pod uwagę brana jest mniejsza wartość).

Wbudowana w Pythonie funkcja `min()` zwraca argument o mniejszej wartości z przekazanych liczb całkowitych lub zmiennoprzecinkowych. (Istnieje również wbudowana funkcja `max()`, która zwraca większy z przekazanych argumentów). Funkcję `min()` można wykorzystać do ustalenia, czy na liście znajduje się mniej niż pięć łączy. Liczbę łączy przeznaczonych do otworzenia w nowych kartach przechowujemy w zmiennej `numOpen`. Następnie za pomocą pętli `for` przeprowadzamy iterację przez te łączy, wywołując `range(numOpen)`.

W trakcie każdej iteracji pętli wywołanie funkcji `webbrowser.open()` otwiera nową kartę w przeglądarce WWW. Zwróć uwagę, że wartość atrybutu `href` w zwracanych elementach `<a>` nie zawiera początkowego fragmentu `http://` ↪ `google.pl`, więc trzeba przeprowadzić konkatenację podanego fragmentu i ciągu tekstowego będącego wartością atrybutu `href`.

Teraz możemy natychmiast otworzyć pięć pierwszych wyników wyszukiwania w Google, na przykład wyrażenia *Python programming tutorials*, przez wywołanie w wierszu poleceń `lucky python programming tutorials!` (W dodatku A znajdziesz informacje o tym, jak można łatwo uruchamiać programy Pythona w różnych systemach operacyjnych).

Pomysły na podobne programy

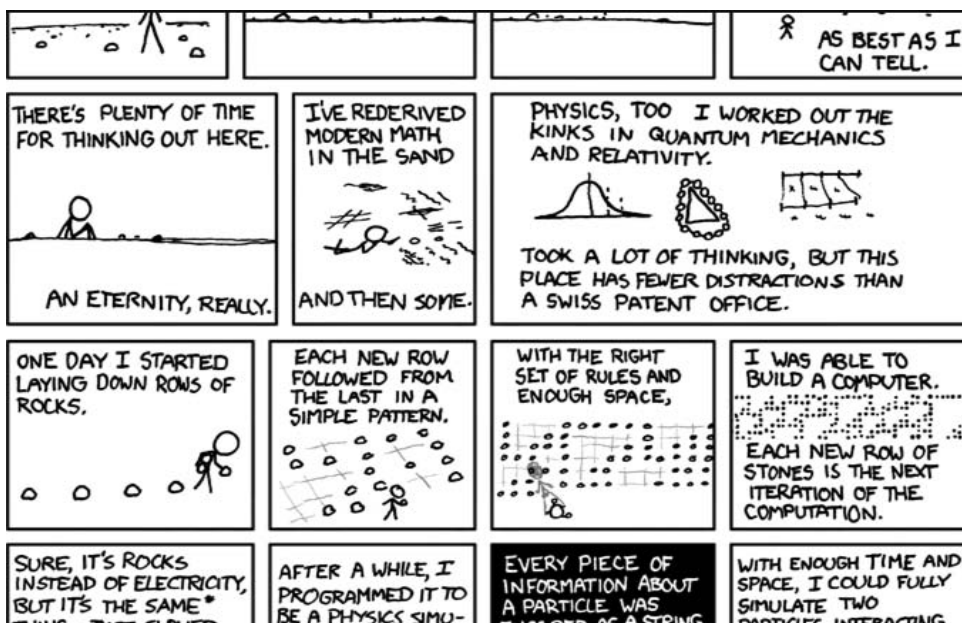
Zaletą kart w przeglądarce WWW jest możliwość łatwego otwierania łączy w nowych kartach, z którymi będzie można zapoznać się później. Program automatycznie otwierający jednocześnie kilka łączy może być wygodnym skrótem dla niektórych zadań. Oto one.

- Otworzenie wszystkich stron produktu po przeprowadzeniu wyszukiwania w witrynie, takiej jak Amazon.
- Otworzenie wszystkich łączy prowadzących do opinii o danym produkcie.
- Otworzenie łączy wyników do zdjęcia po przeprowadzeniu wyszukiwania zdjęcia w witrynach, takich jak Flickr lub Imgur.

Projekt — pobranie wszystkich komiksów z witryny XKCD

Blogi i inne regularnie uaktualniane witryny internetowe zwykle mają stronę główną wraz z aktualnym postem oraz przycisk typu *Poprzedni* pozwalający na przejście do poprzedniego postu. Wcześniejszy post również ma przycisk *Poprzedni* i tak dalej. W ten sposób powstaje ścieżka od najnowszego do najstarszego postu opublikowanego w danej witrynie internetowej. Jeżeli chcesz zapoznać się z treścią strony, gdy nie masz połączenia z internetem, musisz ręcznie przechodzić pomiędzy poszczególnymi stronami i zapisywać każdą z nich. Jednak to niezwykle nużące zadanie i dlatego warto opracować program, który będzie to robił automatycznie.

Pokazana na rysunku 11.6 XKCD to publikująca komiksy popularna witryna internetowa doskonale wpisująca się w przedstawiony powyżej schemat. Na stronie głównej witryny XKCD pod adresem <http://xkcd.com/> znajduje się przycisk *Prev* pozwalający na przejście do wcześniejszych komiksów. Ręczne pobranie każdego komiksu zajmie wielk, ale za pomocą odpowiedniego skryptu będzie można zrobić to w ciągu zaledwie kilku minut.



Rysunek 11.6. XKCD to witryna internetowa publikująca komiksy o różnej tematyce, np. przykład romantyzm, sarkazm, matematyka, język i tak dalej

Poniżej wymienilem w punktach sposób działania programu.

- Wczytanie strony głównej XKCD.
- Zapisanie obrazu zawierającego komiks opublikowany na danej stronie.
- Przejście na stronę wskazywaną przez łącze *Previous Comic*.
- Powtarzanie operacji, dopóki nie zostanie pobrany najstarszy komiks.

Oznacza to, że kod będzie musiał wykonywać przedstawione poniżej zadania.

- Pobieranie stron za pomocą modułu `requests`.
- Odszukanie za pomocą modułu `BeautifulSoup` adresu URL obrazu komiksu.
- Pobranie i zapisanie na dysku twardym obrazu komiksu za pomocą metody `iter_content()`.
- Odszukanie adresu URL łącza *Previous Comic* i powtórzenie całej procedury od początku.

Otwórz nowe okno edytora pliku i zapisz plik pod nazwą `downloadXkcd.py`.

Etap 1. Projekt programu

Jeżeli po wyświetleniu strony komiksu otworzysz wbudowane w przeglądarce narzędzia dla programistów i przeanalizujesz elementy na stronie, będziesz mógł poczynić następujące ustalenia.

- Adres URL pliku obrazu komiksu jest zdefiniowany w atrybucie href elementu ``.
- Element `` znajduje się wewnątrz elementu `<div id="comic">`.
- Przycisk *Prev* ma atrybut HTML `rel` o wartości `prev`.
- Przycisk *Prev* najstarszego komiksu ma adres URL w postaci `http://xkcd.com/#`, co wskazuje na brak wcześniejszych stron.

Wprowadź w pliku przedstawiony poniżej fragment kodu.

```
#!/ python3
# downloadXkcd.py — Pobiera wszystkie komiksy opublikowane w witrynie XKCD.

import requests, os, bs4

url = 'http://xkcd.com'          # Początkowy adres URL.
os.makedirs('xkcd', exist_ok=True) # Komiksy są przechowywane w katalogu ./xkcd.
while not url.endswith('#'):
    # TODO: Pobranie strony.

    # TODO: Ustalenie adresu URL pliku obrazu komiksu.

    # TODO: Pobranie obrazu.

    # TODO: Zapis obrazu w katalogu ./xkcd.

    # TODO: Pobranie adresu URL w przycisku Prev.

print('Gotowe!')
```

W powyższym fragmencie kodu mamy zmienną o nazwie `url` rozpoczynającą się od ciągu tekstowego `'http://xkcd.com'`. Wartość tej zmiennej jest nieustannie uaktualniana (w pętli `while`) adresem URL łącza *Prev* znajdującego się na bieżącej stronie. W trakcie każdej iteracji pętli jest pobierany plik obrazu komiksu znajdujący się pod adresem wskazywanym przez zmienną `url`. Gdy adres URL będzie się kończył znakiem `#`, nastąpi opuszczenie pętli `while`.

Plik obrazu zostanie pobrany do podkatalogu o nazwie `xkcd` umieszczonego w bieżącym katalogu roboczym. Wywołanie `os.makedirs()` gwarantuje istnienie wymienionego katalogu. Argument w postaci `exist_ok=True` uniemożliwia funkcji zgłoszenie wyjątku, jeśli ten katalog już istnieje. Pozostała część kodu to po prostu komentarze przedstawiające resztę programu.

Etap 2. Pobranie strony internetowej

Przejdziemy do implementacji kodu odpowiedzialnego za pobranie strony internetowej. Wprowadź w kodzie przedstawione poniżej zmiany.

```
#!/ python3
# downloadXkcd.py — Pobiera wszystkie komiksy opublikowane w witrynie XKCD.

import requests, os, bs4

url = 'http://xkcd.com' # Początkowy adres URL.
os.makedirs('xkcd', exist_ok=True) # Komiksy są przechowywane w katalogu ./xkcd.
while not url.endswith('#'):
    # Pobranie strony.
    print('Pobieranie strony %s...' % url)
    res = requests.get(url)
    res.raise_for_status()

    soup = bs4.BeautifulSoup(res.text)

    # TODO: Ustalenie adresu URL pliku obrazu komiksu.

    # TODO: Pobranie obrazu.

    # TODO: Zapis obrazu w katalogu ./xkcd.

    # TODO: Pobranie adresu URL w przycisku Prev.

print('Gotowe!')
```

Najpierw wyświetlana jest wartość zmiennej `url`, aby użytkownik wiedział, z jakiego adresu URL program pobiera dane. Następnie za pomocą funkcji `requests.get()` modułu `requests` faktycznie pobieramy stronę. Jak zwykle, kolejnym wywołaniem po pobraniu danych jest metoda `raise_for_status()` obiektu `Response` w celu zgłoszenia wyjątku i zakończenia działania programu, jeśli wystąpił jakikolwiek problem podczas pobierania danych. Gdy wszystko przebiegło bez zakłóceń, na podstawie tekstu pobranej strony tworzymy obiekt `BeautifulSoup`.

Etap 3. Odszukanie i pobranie obrazu komiksu

W kodzie programu wprowadź przedstawione poniżej zmiany.

```
#!/ python3
# downloadXkcd.py — Pobiera wszystkie komiksy opublikowane w witrynie XKCD.

import requests, os, bs4

--cięcie--

# Ustalenie adresu URL pliku obrazu komiksu.
comicElem = soup.select('#comic img')
if comicElem == []:
```

```

        print('Nie udało się odnaleźć pliku obrazu komiksu.')
    else:
        comicUrl = 'http:' + comicElem[0].get('src')
        # Pobranie obrazu.
        print('Pobieranie obrazu %s...' % (comicUrl))
        res = requests.get(comicUrl)
        res.raise_for_status()

        # TODO: Zapis obrazu w katalogu ./xkcd.

        # TODO: Pobranie adresu URL w przycisku Prev.

print('Gotowe!')
```

Po przeprowadzonej za pomocą narzędzi dla programistów analizie strony głównej XKCD wiesz, że element `` pliku obrazu komiksu znajduje się wewnątrz elementu `<div>` zawierającego atrybut `id` o wartości `comic`. Dlatego też selektor `'#comic img'` pozwoli na wyodrębnienie właściwego elementu `` z obiektu `BeautifulSoup`.

Kilka stron XKCD ma treść specjalną, która nie jest po prostu plikiem obrazu. Nie stanowi to żadnego problemu, po prostu je pominiemy. Jeżeli selektor nie dopasuje żadnych elementów, wówczas wartością zwrótną `soup.select('#comic img')` będzie pusta lista. W takim przypadku program może po prostu wyświetlić komunikat błędu i przejść dalej bez pobierania obrazu.

W przeciwnym razie selektor zwróci listę, na której znajduje się jeden element ``. Teraz wystarczy wartość jego atrybutu `src` przekazać do wywołania `requests.get()`, aby tym samym pobrać plik obrazu komiksu.

Etap 4. Zapis obrazu i odszukanie poprzedniego komiksu

W kodzie programu wprowadź przedstawione poniżej zmiany.

```

#!/ python3
# downloadXkcd.py — Pobiera wszystkie komiksy opublikowane w witrynie XKCD.

import requests, os, bs4

--ciąćcie--

    # Zapis obrazu w katalogu ./xkcd.
    imageFile = open(os.path.join('xkcd', os.path.basename(comicUrl)), 'wb')
    for chunk in res.iter_content(100000):
        imageFile.write(chunk)
    imageFile.close()

    # Pobranie adresu URL w przycisku Prev.
    prevLink = soup.select('a[rel="prev"]')[0]
    url = 'http://xkcd.com' + prevLink.get('href')

print('Gotowe!')
```

Na tym etapie plik obrazu komiksu jest przechowywany w zmiennej `res`. Trzeba więc zapisać obraz do pliku na dysku twardym.

Wywołaniu funkcji `open()` należy przekazać nazwę pliku. Zmienna `comicUrl` będzie miała wartość, taką jak `'http://imgs.xkcd.com/comics/heartbleed_explanation.png'`. Jak możesz zauważyć, przypomina ona ścieżkę dostępu do pliku. Wartość tej zmiennej można przekazać wywołaniu `os.path.basename()`, aby w wyniku otrzymać ostatnią część adresu URL, czyli `'heartbleed_explanation.png'`. Następnie tę część możemy wykorzystać w charakterze nazwy pliku dla obrazu zapisywanego na dysku twardym. Za pomocą wywołania `os.path.join()` łączymy tę nazwę wraz z nazwą katalogu `xkcd`, ponieważ dzięki wymienionej funkcji program użyje lewych ukośników (`\`) w systemie Windows oraz prawych ukośników (`/`) w systemach macOS i Linux. Gdy mamy określoną nazwę pliku, można już wywołać funkcję `open()` wraz z atrybutem `'wb'`, aby otworzyć nowy plik w trybie binarnym.

Powinieneś już pamiętać, bo pisałem o tym na początku tego rozdziału, że w celu zapisania plików pobieranych za pomocą modułu `requests` konieczne jest przeprowadzenie iteracji przez wartość zwrótną metody `iter_content()`. Kod w pętli `for` zapisuje dane w pliku obrazu we fragmentach o maksymalnej wielkości 100000 bajtów, a następnie zamyka plik. W tym momencie plik obrazu znajduje się już na dysku twardym Twojego komputera.

Dalej selektor `'a[rel="prev"]'` identyfikuje element `<a>` wraz z atrybutem `rel` o wartości `prev`. Wartość atrybutu `href` tego elementu zawiera adres URL wcześniejszego komiksu, który zapisujemy w zmiennej `url`. Następnie pętla `while` ponownie rozpoczyna proces pobierania danych dla wskazanego komiksu.

Dane wyjściowe wygenerowane przez ten program będą przedstawiały się podobnie do pokazanych poniżej.

```
Pobieranie strony http://xkcd.com...
Pobieranie obrazu http://imgs.xkcd.com/comics/phone_alarm.png...
Pobieranie strony http://xkcd.com/1358/...
Pobieranie obrazu http://imgs.xkcd.com/comics/nro.png...
Pobieranie strony http://xkcd.com/1357/...
Pobieranie obrazu http://imgs.xkcd.com/comics/free_speech.png...
Pobieranie strony http://xkcd.com/1356/...
Pobieranie obrazu http://imgs.xkcd.com/comics/orbital_mechanics.png...
Pobieranie strony http://xkcd.com/1355/...
Pobieranie obrazu http://imgs.xkcd.com/comics/airplane_message.png...
Pobieranie strony http://xkcd.com/1354/...
Pobieranie obrazu http://imgs.xkcd.com/comics/heartbleed_explanation.png...
--cięcie--
```

Ten projekt jest dobrym przykładem programu, który może automatycznie podążać za linkami w celu pobrania ogromnej ilości danych z internetu. Więcej informacji na temat pozostałych funkcji modułu `BeautifulSoup` znajdziesz w dokumentacji dostępnej na stronie <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>.

Pomysły na podobne programy

Pobieranie stron i podążanie za łączami to podstawowe zadania wielu programów pobierających dane z internetu. Podobne programy mogą wykonywać przedstawione poniżej operacje.

- Utworzenie kopii zapasowej całej witryny internetowej przez podążanie za jej wszystkimi łączami.
- Skopiowanie wszystkich postów opublikowanych na forum internetowym.
- Powielenie katalogu produktów na wyprzedaży w sklepie internetowym.

Moduły `requests` i `BeautifulSoup` sprawdzają się doskonale, o ile jesteś w stanie ustalić adres URL, który należy przekazać wywołaniu `requests.get()`. Jednak czasami określenie takiego adresu nie jest łatwe. Ewentualnie witryna internetowa, po której ma się poruszać Twój program, wymaga najpierw zalogowania użytkownika. Moduł o nazwie `selenium` daje Twoim programom potężne możliwości w zakresie wykonywania tak skomplikowanych zadań.

Kontrolowanie przeglądarki WWW za pomocą modułu selenium

Moduł `selenium` pozwala Pythonowi na bezpośrednie, programowe kontrolowanie przeglądarki WWW przez klikanie łącz i wypełnianie formularzy sieciowych. To wszystko odbywa się w prawie taki sam sposób, w jaki ze strony internetowej korzysta człowiek. Moduł `selenium` pozwala na współdziałanie ze stronami internetowymi w znacznie bardziej zaawansowany sposób niż oferowany przez moduły `requests` i `BeautifulSoup`. Ponieważ uruchamia przeglądarkę WWW, więc działa nieco wolniej. Ponadto trudno nie zauważyć jego działania w tle, jeśli trzeba na przykład pobrać pewne pliki z internetu.

Więcej informacji oraz dokładne omówienie instalacji modułów opracowanych przez firmy trzecie znajdziesz w dodatku A.

Uruchomienie przeglądarki WWW kontrolowanej przez moduł selenium

W omawianym tutaj przykładzie wykorzystamy przeglądarkę WWW o nazwie Firefox. Tę właśnie przeglądarkę będziemy kontrolować. Jeżeli jeszcze nie zainstalowałeś przeglądarki Firefox, możesz ją pobrać z witryny <http://getfirefox.com/>.

Import modułu `selenium` przebiega nieco inaczej niż modułów wcześniejszych. Zamiast polecenia `import selenium` trzeba wydać polecenie `from selenium import webdriver`. (Dokładne wyjaśnienie powodu, dla którego moduł `selenium` został skonfigurowany w taki właśnie sposób wykracza poza zakres tematyczny książki).

Teraz będzie już można uruchomić przeglądarkę Firefox kontrolowaną przez moduł selenium.³ W powłoce interaktywnej wprowadź przedstawione poniżej polecenia.

```
>>> from selenium import webdriver
>>> browser = webdriver.Firefox()
>>> type(browser)
<class 'selenium.webdriver.firefox.webdriver.WebDriver'>
>>> browser.get('http://inventwithpython.com')
```

Zauważysz, że po wywołaniu `webdriver.Firefox()` następuje uruchomienie wskazanej przeglądarki WWW (tutaj to Firefox). Wywołanie `type()` z wartością udostępnioną przez `webdriver.Firefox()` ujawnia typ danych `WebDriver`. Natomiast wywołanie `browser.get('http://inventwithpython.com')` powoduje przekierowanie przeglądarki WWW do witryny `http://inventwithpython.com/`. W tym momencie okno przeglądarki Firefox powinno wyglądać tak, jak pokazałem na rysunku 11.7.

Wyszukanie elementów na stronie

Obiekty typu `WebDriver` mają całkiem sporą ilość metod przeznaczonych do wyszukiwania elementów na stronie. Zostały podzielone na metody typu `find_element_*` i `find_elements_*`. Metody typu `find_element_*` zwracają pojedynczy obiekt `WebElement` przedstawiający pierwszy element na stronie, który został dopasowany do zapytania. Z kolei metody typu `find_elements_*` zwracają listę obiektów `WebElement_*` dla *każdego* dopasowanego elementu na stronie.

W tabeli 11.3 wymieniłem kilka przykładów metod typu `find_element_*` i `find_elements_*` wywoływanych w obiekcie `WebDriver`, który jest przechowywany w zmiennej o nazwie `browser`.

Poza metodami `*_by_tag_name()`, argumenty wszystkich pozostałych metod uwzględniają wielkość znaków. Jeżeli na stronie nie istnieją elementy możliwe do dopasowania przez metodę, moduł `selenium` zwróci wyjątek `NoSuchElement`. Jeśli nie chcesz, aby ten wyjątek doprowadził do awarii programu, w kodzie powinieneś zastosować polecenia `try` i `except`.

Więcej informacji o obiekcie `WebElement` można zdobyć przez odczyt atrybutów lub wywołanie metod wymienionych w tabeli 11.4.

³ Dla używanej przeglądarki WWW (tutaj Firefox) niezbędny jest jeszcze sterownik. W omawianym przykładzie skorzystamy ze sterownika Geckodriver, który możesz pobrać ze strony <https://github.com/mozilla/geckodriver/releases>. Po rozpakowaniu archiwum otrzymasz plik `geckodriver.exe`. Ponieważ musi znajdować się w katalogu podanym w zmiennej systemowej `path`, więc najlepiej skopiuj go w katalogu, w którym zainstalowałeś Pythona — *przyp. tłum.*



Rysunek 11.7. Po wywołaniach `webdriver.Firefox()` i `get()` w środowisku IDLE na ekranie pojawia się okno wskazanej przeglądarki WWW (tutaj to Firefox)

Otwórz nowe okno edytora pliku i wpisz w nim poniższy fragment kodu.

```

from selenium import webdriver
browser = webdriver.Firefox()
browser.get('http://inventwithpython.com')
try:
    elem = browser.find_element_by_class_name('bookcover')
    print('Znaleziono element <%s> wraz z taką nazwą klasy!' % (elem.tag_name))
except:
    print('Nie udało się znaleźć elementu wraz z podaną nazwą klasy.')
  
```

W powyższym fragmencie kodu uruchamiamy przeglądarkę WWW o nazwie Firefox i przekierowujemy ją do podanego adresu URL. Na tej stronie próbujemy wyszukać elementy o nazwie klasy 'bookcover'. Jeżeli tego rodzaju element istnieje, wyświetlamy jego nazwę za pomocą atrybutu `tag_name`. Natomiast w przypadku nieznalezienia elementu wyświetlony będzie zupełnie inny komunikat.

Tabela 11.3. Oferowane przez moduł selenium metody obiektu WebDriver przeznaczone do wyszukiwania elementów

| Nazwa metody | Zwrócony obiekt lub lista obiektów WebDriver |
|---|--|
| <code>browser.find_element_by_class_name(nazwa)</code> <code>browser.find_elements_by_class_name(nazwa)</code> | Elementy używające klasy CSS o podanej <i>nazwie</i> . |
| <code>browser.find_element_by_css_selector(selektor)</code> <code>browser.find_elements_by_css_selector(selektor)</code> | Elementy dopasowane przez <i>selektor</i> CSS. |
| <code>browser.find_element_by_id(id)</code> <code>browser.find_elements_by_id(id)</code> | Elementy o dopasowanej wartości atrybutu <i>id</i> . |
| <code>browser.find_element_by_link_text(tekst)</code> <code>browser.find_elements_by_link_text(tekst)</code> | Elementy <code><a></code> , które zawierają całkowicie dopasowany podany <i>tekst</i> . |
| <code>browser.find_element_by_partial_link_text(tekst)</code> <code>browser.find_elements_by_partial_link_text(tekst)</code> | Elementy <code><a></code> , które zawierają podany <i>tekst</i> . |
| <code>browser.find_element_by_name(nazwa)</code> <code>browser.find_elements_by_name(nazwa)</code> | Elementy o dopasowanej wartości atrybutu <i>nazwa</i> . |
| <code>browser.find_element_by_tag_name(nazwa)</code> <code>browser.find_elements_by_tag_name(nazwa)</code> | Elementy o dopasowanej <i>nazwie</i> znacznika (wielkość liter nie ma znaczenia, element <code><a></code> będzie dopasowany zarówno przez <code>'a'</code> , jak i <code>'A'</code>). |

Tabela 11.4. Atrybuty i metody obiektu WebElement

| Atrybut lub metoda | Opis |
|-----------------------------------|--|
| <code>tag_name</code> | Nazwa znacznika, na przykład <code>'a'</code> dla elementu <code><a></code> . |
| <code>get_attribute(nazwa)</code> | Wartość atrybutu o podanej <i>nazwie</i> w elemencie. |
| <code>text</code> | Tekst wewnątrz elementu, na przykład <code>'witaj'</code> w elemencie <code>witaj</code> . |
| <code>clear()</code> | W przypadku elementów pola lub obszaru tekstowego ta metoda powoduje usunięcie tekstu wyświetlanego przez element. |
| <code>is_displayed()</code> | Zwraca wartość <code>True</code> , jeśli element jest widoczny, w przeciwnym razie wartością zwrótną jest <code>False</code> . |
| <code>is_enabled()</code> | W przypadku elementów danych wejściowych zwraca wartość <code>True</code> , jeśli element jest włączony. W przeciwnym razie wartością zwrótną jest <code>False</code> . |
| <code>is_selected()</code> | W przypadku elementów, takich jak pole wyboru lub pole opcji, zwraca wartość <code>True</code> , jeśli element jest włączony. W przeciwnym razie wartością zwrótną jest <code>False</code> . |
| <code>location</code> | Słownik wraz z kluczami <code>'x'</code> i <code>'y'</code> wskazującymi położenie elementu na stronie. |

Program wygeneruje następujące dane wyjściowe.

```
Znaleziono element <img> wraz z taką nazwą klasy!
```

Udało nam się znaleźć element o nazwie klasy `'bookcover'` oraz nazwie znacznika `'img'`.

Kliknięcie na stronie

Obiekty `WebElement` zwracane przez metody typu `element_*` i `find_elements_*` zawierają metodę `click()` symulującą kliknięcie myszą tego elementu. Metoda ta może być używana w celu podążania za łączem, dokonywania wyboru za pomocą przycisków opcji, kliknięcia przycisku wysyłającego formularz lub wywołania każdej innej akcji, która następuje po kliknięciu elementu myszą. W powłoce interaktywnej wprowadź przedstawione poniżej polecenia.

```
>>> from selenium import webdriver
>>> browser = webdriver.Firefox()
>>> browser.get('http://inventwithpython.com')
>>> linkElem = browser.find_element_by_link_text('Read It Online')
>>> type(linkElem)
<class 'selenium.webdriver.remote.webelement.WebElement'>
>>> linkElem.click() # Podążanie za łączem zatytułowanym "Read It Online".
```

W powyższym fragmencie kodu przeglądarka Firefox przechodzi do witryny `http://inventwithpython.com/`, pobiera obiekt `WebElement` dla elementu `<a>` zawierającego tekst `Read It Online`, a następnie symuluje kliknięcie tego elementu `<a>`. To przypomina kliknięcie łącza przez człowieka, a przeglądarka WWW podąży za tym łączem.

Wypełnianie i wysyłanie formularzy sieciowych

Symulowanie naciśnięcia klawiszy w polu tekstowym na stronie internetowej sprowadza się do odszukania elementu `<input>` lub `<textarea>` dla danego pola tekstowego, a następnie do wywołania metody `send_keys()`. W powłoce interaktywnej wprowadź przedstawione poniżej polecenia.

```
>>> from selenium import webdriver
>>> browser = webdriver.Firefox()
>>> browser.get('http://mail.yahoo.com')
>>> emailElem = browser.find_element_by_id('login-username')
>>> emailElem.send_keys('nieprawdziwy_adres_e-mail')
>>> passwordElem = browser.find_element_by_id('login-passwd')
>>> passwordElem.send_keys('12345')
>>> passwordElem.submit()
```

O ile Yahoo! nie zmieni identyfikatora pól tekstowych przeznaczonych do podania nazwy użytkownika i hasła, przedstawiony powyżej fragment kodu spowoduje wypełnienie tych pól podanym tekstem. (Za pomocą wbudowanych w przeglądarkę WWW narzędzi programistycznych zawsze możesz sprawdzić wspomniane identyfikatory). Wywołanie metody `submit()` w dowolnym elemencie ma dokładnie taki sam efekt jak kliknięcie przycisku `Wyslij` formularza zawierającego ten element. (Równie dobrze można użyć wywołania `emailElem.submit()`, a omawiany kod nadal będzie wykonywał to samo zadanie).

Symulacja naciśnięcia klawiszy specjalnych

Moduł selenium zapewnia obsługę klawiszy, których nie można zasymulować przez użycie pewnych wartości w postaci ciągu tekstowego. Działanie tego rodzaju klawiszy przypomina sekwencje sterujące. Wartości symulujące naciśnięcia tych klawiszy są przechowywane w module `selenium.webdriver.common.keys`. Ponieważ nazwa modułu jest bardzo długa, warto umieszczać na początku programu polecenie `from selenium.webdriver.common.keys import Keys`. Dzięki temu możesz później używać polecenia `Keys` wszędzie tam, gdzie normalnie musiałbyś pisać `selenium.webdriver.common.keys`. W tabeli 11.5 wymienilem najczęściej używane zmienne `Keys`.

Tabela 11.5. Powszechnie używane zmienne zdefiniowane w module `selenium.webdriver.common.keys`

| Atrybuty | Opis |
|--|---|
| <code>Keys.DOWN</code> , <code>Keys.UP</code> , <code>Keys.LEFT</code> , <code>Keys.RIGHT</code> | Klawisze kursora na klawiaturze. |
| <code>Keys.ENTER</code> , <code>Keys.RETURN</code> | Klawisze <i>Enter</i> i <i>Return</i> . |
| <code>Keys.HOME</code> , <code>Keys.END</code> , <code>Keys.PAGE_DOWN</code> , <code>Keys.PAGE_UP</code> | Klawisze <i>Home</i> , <i>End</i> , <i>Pagedown</i> i <i>Pageup</i> . |
| <code>Keys.ESCAPE</code> , <code>Keys.BACK_SPACE</code> , <code>Keys.DELETE</code> | Klawisze <i>Esc</i> , <i>Backspace</i> i <i>Delete</i> . |
| <code>Keys.F1</code> , <code>Keys.F2</code> , ..., <code>Keys.F12</code> | Klawisze od <i>F1</i> do <i>F12</i> znajdujące się na górze klawiatury. |
| <code>Keys.TAB</code> | Klawisz <i>Tab</i> . |

Jeśli na przykład kursor aktualnie nie znajduje się w polu tekstowym, wówczas naciskając klawisze *Home* i *End*, można poruszać się między — odpowiednio — początkiem i końcem strony. W powłocie interaktywnej wprowadź przedstawione poniżej polecenia i zwróć uwagę, jak za pomocą wywołań `send_keys()` można przewijać zawartość strony.

```
>>> from selenium import webdriver
>>> from selenium.webdriver.common.keys import Keys
>>> browser = webdriver.Firefox()
>>> browser.get('http://nostarch.com')
>>> htmlElem = browser.find_element_by_tag_name('html')
>>> htmlElem.send_keys(Keys.END)      # Przewinięcie na koniec strony.
>>> htmlElem.send_keys(Keys.HOME)    # Przewinięcie na początek strony.
```

Znacznik `<html>` jest znacznikiem bazowym w plikach HTML. Cała zawartość dokumentu HTML jest umieszczona między znacznikami `<html>` i `</html>`. Wywołanie `browser.find_element_by_tag_name('html')` to dobry początek na symulowanie naciśnięć klawiszy na stronie. Takie podejście będzie użyteczne, jeśli na przykład nowa zawartość strony jest wczytywana po jej przewinięciu do końca.

Klikanie przycisków przeglądarki WWW

Moduł selenium może symulować także klikanie różnych przycisków przeglądarki WWW. Do tego celu służą wymienione poniżej metody.

- `browser.back()`. Kliknięcie przycisku *Wstecz*.
- `browser.forward()`. Kliknięcie przycisku *Do przodu*.
- `browser.refresh()`. Kliknięcie przycisku *Odśwież*.
- `browser.quit()`. Kliknięcie przycisku *Zamknij okno*.

Więcej informacji na temat modułu selenium

Możliwości oferowane przez moduł selenium są znacznie większe niż przedstawione w rozdziale. Moduł pozwala na modyfikację plików cookie przeglądarki WWW, wykonywanie rzutów stron internetowych, a także na uruchamianie własnego kodu JavaScript. Jeżeli chcesz dowiedzieć się więcej na temat tych możliwości, zajrzyj do oficjalnej dokumentacji modułu selenium, którą znajdziesz na stronie <http://selenium-python.readthedocs.io/>.

Podsumowanie

Większość nudnych zajęć nie ogranicza się wyłącznie do związanych z plikami znajdującymi się na dysku twardym komputera lokalnego. Możliwość programowego pobierania stron internetowych pozwala na rozbudowę programów, które w ten sposób mogą się łączyć z internetem. Moduł `requests` znacznie ułatwia pobieranie danych. Mając nawet jedynie podstawową wiedzę o koncepcjach stosowanych w HTML i selektorach CSS, można wykorzystać moduł `BeautifulSoup` do przetwarzania pobieranych stron internetowych.

Jednak w celu przeprowadzenia pełnej automatyzacji zadań związanych z siecią WWW będziesz musiał przejąć bezpośrednią kontrolę nad przeglądarką WWW, na co pozwala moduł `selenium`. Moduł ten umożliwi automatyczne logowanie się w witrynach internetowych oraz wypełnianie formularzy sieciowych. Ponieważ przeglądarka WWW to obecnie narzędzie, za pomocą którego najczęściej wysyłamy i pobieramy informacje przez internet, warto mieć moduł `selenium` w arsenale dostępnych narzędzi.

Pytania kontrolne

1. Pokróćce omów różnice między modułami `webbrowser`, `requests`, `BeautifulSoup` i `selenium`.
2. Jaki typ obiektów jest zwracany przez funkcję `requests.get()`? Jak można uzyskać dostęp do pobranej treści jako ciągu tekstowego?

3. Jaka metoda modułu `requests` pozwala na sprawdzenie, czy dane zostały pobrane prawidłowo?
4. Jak można otrzymać kody stanu HTTP dla odpowiedzi udzielonej na żądania wysyłane za pomocą modułu `requests`?
5. Jak można zapisać w pliku odpowiedź udzieloną na żądanie wysłane za pomocą modułu `requests`?
6. Jaki jest skrót klawiszowy pozwalający na otworenie wbudowanych w przeglądarkę WWW narzędzi dla programistów?
7. Jak można wyświetlić (za pomocą narzędzi dla programistów) kod HTML określonego elementu na stronie internetowej?
8. Jak przedstawia się ciąg tekstowy selektora CSS dopasowującego element o atrybucie `id` o wartości `main`?
9. Jak przedstawia się ciąg tekstowy selektora CSS dopasowującego elementy zawierające klasę CSS o nazwie `highlight`?
10. Jak przedstawia się ciąg tekstowy selektora CSS dopasowującego wszystkie elementy `<div>` znajdujące się wewnątrz innego elementu `<div>`?
11. Jak przedstawia się ciąg tekstowy selektora CSS dopasowującego element `<button>` wraz z atrybutem `value` o wartości `favorite`?
12. Przyjmujemy założenie, że masz obiekt `Tag` modułu `BeautifulSoup` przechowywany w zmiennej `spam` dla elementu `<div>Witaj, świecie!</div>`. Jak możesz pobrać ciąg tekstowy `'Witaj, świecie!'` z tego obiektu `Tag`?
13. Jak będziesz przechowywać wszystkie atrybuty obiektu `Tag` modułu `BeautifulSoup` w zmiennej o nazwie `linkElem`?
14. Wykonanie polecenia `import selenium` nie działa. Jak prawidłowo zaimportujesz moduł `selenium`?
15. Jaka jest różnica między metodami typu `find_element_*` i `find_elements_*`?
16. Jakie metody ma obiekt `WebElement` modułu `selenium` przeznaczony do symulowania kliknięć myszą i naciśnięć klawiszy na klawiaturze?
17. Masz możliwość wykonania wywołania `send_keys(Keys.ENTER)` w przycisku wysyłającym formularz sieciowy w obiekcie `WebElement`. Jaki jest jeszcze łatwiejszy sposób na wysłanie formularza sieciowego za pomocą modułu `selenium`?
18. Jak za pomocą modułu `selenium` można symulować kliknięcie przycisków przeglądarki WWW, takich jak `wstecz`, `do przodu` i `odśwież`.

Projekty praktyczne

W celu zdobycia doświadczenia utwórz programy wykonujące omówione poniżej zadania.

Klient poczty działający w wierszu poleceń

Utwórz program, który będzie pobierał argumenty wiersza poleceń w postaci adresu e-mail i ciągu tekstowego, a następnie za pomocą modułu `selenium` zaloguje się do Twojego konta e-mail i wyświetli wiadomość. Adresat wiadomości i jej treść są podawane jako argumenty programu. (Dla tego programu rozsądne może być utworzenie oddzielnego konta poczty elektronicznej).

Dobrze byłoby dodać funkcję powiadamiania. Możesz utworzyć również podobny program przeznaczony do wysyłania komunikatów z serwisów Facebook lub Twitter.

Pobieranie obrazów z witryny internetowej

Utwórz program, który będzie korzystał z serwisów przeznaczonych do dzielenia się zdjęciami, na przykład takich jak Flickr i Imgur. Zadaniem programu ma być wyszukiwanie pewnej kategorii zdjęć, a następnie pobranie wszystkich zwróconych w wyniku wyszukiwania. Możesz też utworzyć program, który będzie współdziałał z dowolnym serwisem dzielenia się zdjęciami oferującym funkcję wyszukiwania.

2048

2048 to prosta gra, w której zadaniem gracza jest łączenie pól przez ich przesuwanie w górę, w dół, w lewo i w prawo za pomocą klawiszy kursora. W tej grze można uzyskać bardzo wysoki wynik, nieustannie przesuwając pola zgodnie ze wzorcem w górę, w prawo, w dół, w lewo. Utwórz program otwierający stronę z grą 2048 dostępną pod adresem <https://gabrielecirulli.github.io/2048/>, a następnie przekazujący naciśnięcia klawiszy w górę, w prawo, w dół i w lewo, aby faktycznie symulować grę.

Weryfikacja łącza

Utwórz program, który po otrzymaniu adresu URL strony internetowej spróbuje pobrać wszystkie strony internetowe, do których prowadzą łącza z podanej. Program powinien oznaczać wszystkie strony generujące kod stanu HTTP 404 (czyli „nie znaleziono”) i wyświetlać je na liście nieprawidłowych łączy.

Skorowidz

A

- adres URL, 285
- akapit, 369
- analiza
 - programu, 49
 - zrzutu ekranu, 509
- API, application programming interface, 391
- archiwum ZIP, 249, 256–258
- argumenty, 96
 - funkcji range(), 85
 - wiersza poleceń, 301, 424
- arkusz kalkulacyjny, 319, 327
- asercje, 265, 536
- atrybuty obiektu
 - Run, 371
 - WebElement, 313
- automatyzacja
 - GUI, 495
 - zadań, 181, 527

B

- białe znaki, 170
- bieżący katalog roboczy, 215
- binarne operatory boolowskie, 63
- błąd typu SyntaxError, 42, 43
- błędy, 41, 103, 261

C

- ciąg tekstowy, 43, 127, 157
 - indeksowanie, 161
 - konwersja obiektu datetime, 412
 - kopiowanie, 171

- literały, 158
- metody, 162
- niezmodyfikowany, 159
- potrójne apostrofy, 159
- stos wywołań, 264
- usuwanie białych znaków, 170
- wklejanie, 171
- wycinanie, 161
- wyrównywanie, 168
- wyszukiwanie wzorców, 184, 186
- zastępowanie, 202
- cron, 424
- CSV, comma-separated values, 381
 - usunięcie nagłówka, 387
- cudzysłów, 158
- czas, 402

D

- dane JSON, 381
- data, 408
 - w stylu amerykańskim, 251
- debugger, 273
- debugowanie, 275, 281
- demon launchd, 424
- deszyfrowanie dokumentu PDF, 356
- dokumenty
 - Excela, 320
 - PDF, 353
 - Worda, 353, 366
- dopasowanie, 201
 - jednego wystąpienia, 193
 - niezachłanne, 194
 - obiektów wyrażeń regularnych, 187

- dopasowanie
 - określonych powtórzeń, 193
 - opcjonalne, 191
 - wielu grup, 190
 - wszystkiego, 199
 - zachłanne, 194
 - zera wystąpień, 192
 - znaku nowego wiersza, 200
- dostęp do powłoki interaktywnej, 40
- dostosowanie wierszy i kolumn, 342
- drzewo katalogu, 246, 258
- dzielenie komórki, 343

E

- ekran, 508
- e-mail, 434, 438
- Excel, 319
 - dostosowanie wierszy i kolumn, 342
 - formuły, 341
 - odczyt dokumentów, 321
 - pobieranie wierszy i kolumn, 325
 - wykresy, 345
 - zablokowane okienka, 344
 - zapis dokumentów, 332

F

- format
 - .docx, 368
 - CVS, 381
 - JSON, 381, 394
 - PDF, 353
 - ZIP, 248
- formularze, 314, 517, 523
- formuły, 341
- funkcja, 91
 - add_heading(), 374
 - copy(), 135
 - deepcopy(), 135
 - dumps(), 392
 - float(), 52
 - input(), 50
 - int(), 52
 - len(), 51, 115
 - list(), 131
 - loads(), 392
 - open(), 223

- os.makedirs(), 216
- Popen(), 424
- pprint.pformat(), 227
- print(), 50, 270
- print(a), 96
- range(), 83, 85
- requests.get(), 288
- str(), 52
- sys.exit(), 88
- time.sleep(), 403
- time.time(), 402
- tuple(), 131

funkcje

- czasu, 414
- modułu pyautogui, 516

G

- gra w kółko i krzyżyk, 148
- grupowanie z użyciem nawiasów, 189

H

- harmonogram zadań, 424
- HTML, hypertext markup language, 291
- HTTP, hypertext transfer protocol, 434

I

- IMAP, internet message access protocol, 439
- import modułów, 87
- indeks, 112
 - ujemny, 114
- informacje
 - o lokalizacji, 393
 - o położeniu kursora, 500
- inkrementacja, 121
- instalacja modułu, 529
 - openpyxl, 320
 - pyautogui, 496
- interfejs programowania aplikacji, API, 391
- iteracja, 387

J

- język HTML, 291
- JSON, JavaScript Object Notation, 381, 390

K

- katalog roboczy, 215
- klasy znaków, 196
- klawiatura, 495, 512
- klawisze specjalne, 315
- klikanie
 - na stronie, 314
 - przycisków przeglądarki, 316
- klucz, 143
- klucze wyszukiwania serwera IMAP, 444
- kodowanie Unicode, 290
- kolejność operacji, 40
- kolory, 466
 - CMYK, 467
 - RGB, 467
- kolumna, 343
- kombinacja klawiszy, 515
- komentarz wielowierszowy, 160
- komórka, 327
- kompresja plików, 248
- konkatenacja, 43
 - listy, 116
- kontrola
 - działania myszy, 498, 503
 - klawiatury, 512
 - przeglądarki, 310
 - przepływu działania programu, 59, 65
- konwersja
 - ciągu tekstowego, 413
 - liter na liczby, 324
 - obiektu datetime, 412
- kopiowanie
 - hasła, 173
 - obrazów, 472
 - plików i katalogów, 242
 - stron, 357
- krotka, 127, 130, 467
- kształty, 486

L

- launchd, 424
- liczby
 - całkowite, 43
 - zmiennoprzecinkowe, 43
- listy, 111
 - dodawanie wartości, 123
 - konkatenacja, 116

- operatory, 119
- pobieranie długości, 115
- pobieranie podlisty, 114
- pobieranie wartości, 112
- replikacja, 116
- sortowanie wartości, 124
- usuwanie wartości, 116, 124
- użycie pętli for, 118
- wyszukiwanie wartości, 122
- zmiana wartości, 115
- literały ciągu tekstowego, 158
- logo, 480
- logowanie w serwerze
 - IMAP, 441
 - SMTP, 437
- lokalizacja, 393
- lustrzane odbicia obrazu, 476

Ł

- łączenie
 - komórki, 343
 - operatorów, 42
 - operatorów boolowskich, 65

M

- mechanizm cron, 424
- menedżer hasel, 172
- metoda, 122
 - add_picture(), 376
 - append(), 123
 - center(), 168
 - endswith(), 166
 - findall(), 195
 - get(), 144, 300
 - index(), 122
 - insert(), 123
 - islower(), 162
 - isupper(), 162
 - items(), 142
 - join(), 167
 - keys(), 142
 - ljust(), 168
 - lower(), 162
 - remove(), 124
 - rjust(), 168
 - select(), 298
 - setdefault(), 144

- metoda
 - split(), 167
 - startswith(), 166
 - strftime(), 412
 - strip(), 212
 - sub(), 202
 - time.sleep(), 411
 - upper(), 162
 - values(), 142
- metody
 - ciągu tekstowego, 162
 - listy, 122
 - obiektu WebDriver, 313
 - obiektu WebElement, 313
 - słownika, 142
 - typu isX(), 164
- modelowanie rzeczywistych rozwiązań, 147
- moduł BeautifulSoup, 284, 297
- moduł, 87
 - copy, 135
 - csv, 382
 - datetime, 408
 - json, 392
 - logging, 268
 - openpyxl, 320, 321
 - os.path, 217
 - pillow, 466, 468
 - pyautogui, 496, 516
 - PyPDF2, 354
 - requests, 284, 287
 - selenium, 284, 310, 315, 316
 - send2trash, 246
 - shelve, 226
 - shutil, 242
 - time, 402
 - webbrowser, 284
 - zipfile, 248
- moduły firm trzecich, 529
- monitorowanie, 406
- mysz, 495
 - kontrola działania, 503

N

- naciskanie klawiszy, 514
- nakładanie stron, 359
- narzędzia programistyczne przeglądarki, 293
- narzędzie pip, 529

- nawias klamrowy, 193
- nazwa
 - pliku, 254
 - zmiennej, 47, 100
- nazwy klawiszy, 513

O

- obiekt
 - BeautifulSoup, 297
 - datetime, 412, 413
 - Font, 339
 - Reader, 383
 - Run, 369, 371
 - WebDriver, 313
 - Writer, 384
- obiekty wyrażeń regularnych, 187
- obraz, 465
 - kolory, 466
 - kopiowanie, 472
 - lustrzane odbicia, 476
 - moduł pillow, 468
 - prycinanie, 471
 - rotacja, 476
 - rozpoznawanie, 510
 - umieszczanie tekstu, 488
 - wklejanie, 472
 - zmiana pikseli, 478
 - zmiana wielkości, 474, 482
- obsługa
 - argumentów wiersza poleceń, 173, 285
 - przycisków opcji, 522
 - rozwijanych list, 522
 - wyjątków, 103
 - zawartości schowka, 286
- odczyt
 - danych JSON, 392
 - dokumentów Excela, 321
 - pliku, 213, 222, 224
 - pliku CSV, 388
- odliczanie czasu, 428
- odpowiedniki tekstowe i liczbowe, 55
- odtworzenie pliku dźwiękowego, 428
- odwołania, 132
- opcja
 - re.DOTALL, 203
 - re.IGNORECASE, 203
 - re.VERBOSE, 203

operacje przypisania, 120

operator

in, 119, 162

not, 64

not in, 119, 162

przypisania, 121

operatory boolowskie, 63

binarne, 63

matematyczne, 41

porównania, 61, 65

organizacja plików, 241

otwieranie

dokumentu Excela, 321

kart przeglądarki, 303

plików, 426

witryn internetowych, 425

P

PDF, Portable Document Format, 353

deszyfrowanie dokumentu, 356

łączenie stron, 362

szyfrowanie dokumentu, 361

tworzenie dokumentu, 357

wyodrębnianie tekstu, 354

pętla

for, 83, 118, 384

while, 74, 75, 85

pierwszy program, 48

pliki

archiwum ZIP, 249, 250

CSV, 381, 387

dźwiękowe, 428

kompresja, 248

kopiowanie, 242

odczyt, 222, 224

otwieranie, 223

ścieżka dostępu, 213

trwale usunięcie, 245

ustalanie wielkości, 220

zapis, 222, 225

zapis na dysku, 290

zgłaszanie wyjątku, 262

zmiana nazw, 243, 254

pobieranie

adresów e-mail, 447

arkuszy kalkulacyjnych, 322

danych z internetu, 283

danych z witryny, 418

długości listy, 115

komórek, 322

obrazu komiksu, 307

plików z internetu, 287

strony internetowej, 288, 307

wiadomości e-mail, 439, 445

wierszy i kolumn, 325

podlisty, 114

polecenia

kontroli przepływu działania programu, 67

przypisania, 45

polecenie

break, 78

continue, 79

def, 93

del, 116

elif, 69

else, 68

except, 109

from import, 88

global, 101

idle3, 40

if, 67

return, 93

try, 109

połączenie z serwerem

IMAP, 440, 450

SMTP, 435, 438

pomiar czasu, 405

poruszanie kursorem myszy, 499

potok, 190

potrójne apostrofy, 159

powłoka interaktywna, 40

poziomy rejestrowania informacji, 271

problem Collatza, 109

program

Mad Libs, 238

Magic 8 Ball, 126

mouseNow.py, 509

projekt

automatyczne wypełnianie formularzy, 517

dodanie logo, 480

dodanie wypunktowania, 175

generowanie losowych plików quizu, 229

kursor myszy, 500

łączenie stron dokumentów PDF, 362

mapIt.py, 284

menedżer haseł, 172

odczyt danych z arkusza kalkulacyjnego, 327

projekt

- pobieranie bieżących danych, 393
- pobieranie komiksów, 304
- program odliczający czas, 427
- program wielowątkowy, 418
- rozbudowa programu mouseNow.py, 509
- schowek przechowujący wiele elementów, 234
- superstoper, 405
- symulacja ulicznej sygnalizacji świetlnej, 266
- uaktualnienie skoroszytu, 335
- usunięcie nagłówka z pliku CSV, 387
- utworzenie archiwum ZIP, 255
- wyodrębnianie adresu e-mail, 204
- wyodrębnianie numeru telefonu, 204
- wysyłanie wiadomości e-mail, 451
- wysyłanie wiadomości tekstowej, 459
- wyszukiwanie w Google, 301
- zmiana daty pliku, 251

protokół

- IMAP, 439
- SMTP, 434

przeciąganie myszą, 504

- przekazywanie odwołania, 134
- przenoszenie plików, 243
- przetwarzanie kodu HTML, 297
- przewijanie myszą, 506
- przezroczyste piksele, 475
- prycinanie obrazu, 471
- przypisanie, 45, 120
- punkty kontrolne, 277

R

rejestrowanie

- danych, 268
- informacji, 272
- informacji w pliku, 273

replikacja

- ciągu tekstowego, 43
- listy, 116

rotacja

- obrazu, 476
- stron, 359

rozpoznawanie obrazu, 510

rysowanie

- kształtów, 486
- na obrazach, 486

S

selektory CSS, 299

serwer

- IMAP, 439
- SMTP, 435, 438

serwis Twilio, 456

skoroszyt, 327

słowa kluczowe, 96

słowniki, 139

słowo kluczowe

- delimiter, 386
- lineterminator, 386

SMTP, simple mail transfer protocol, 434

sortowanie, 124

stos wywołań, 264

struktura danych, 329, 336

strukturyzacja danych, 139

styl czcionki komórek, 338

szyfrowanie

- dokumentu PDF, 361
- TLS, 437

Ś

ścieżka dostępu, 213

bezwzględna, 216

sprawdzenie poprawności, 221

względna, 216

środowisko IDLE, 273

T

tworzenie

archiwum ZIP, 255, 257

arkuszy kalkulacyjnych, 333

dokumentu PDF, 357

dokumentu Worda, 370

katalogów, 216

obiektów wyrażeń regularnych, 187

obiektu BeautifulSoup, 297

wątków, 420

typ danych, 43

Dictionary, 139

Image, 470

krotka, 130

List, 111

timedelta, 410

- typy danych
 - modyfikowalne, 128
 - niemodyfikowane, 128

U

- uaktualnienie skoroszytu, 335
- układ współrzędnych, 467
- Unicode, 290
- UNIX, 427
- uruchamianie programów, 421, 533
 - przeglądarki WWW, 286
 - w systemach macOS i Linux, 535
 - w Windows, 534
 - z wyłączonymi asercjami, 536
 - wątków, 420
- usługa
 - bramki SMS, 456
 - Twilio, 456
- ustalenie adresu URL, 285
- usuwanie
 - arkuszy kalkulacyjnych, 333
 - białych znaków, 170
 - błędów, 261, 270
 - danych, 246
 - nagłówka, 387
 - plików i katalogów, 245
 - wiadomości e-mail, 439, 449
- użycie
 - asercji, 266
 - modułu logging, 268
 - pętli for, 118

W

- wartości
 - boolowskie, 60
 - RGBA, 466
- wartość
 - None, 95
 - zwrotna funkcji, 93
- wątek, 417, 420
- wczytanie danych w formacie JSON, 395
- weryfikacja danych wyjściowych, 109
- wiadomości e-mail, 433, 434, 438
 - niezmodyfikowane, 447, 448
 - przeczytane, 445
 - usuwanie, 449
 - wysyłanie, 451

- wielowątkowość, 415
- wiersz, 343
 - shebang, 533
- witryna XKCD, 304
- wklejanie obrazów, 472
- własna klasa znaków, 197
- Word, 353
 - dodanie nagłówków, 374
 - dodanie obrazu, 376
 - nadawanie stylu akapitom, 369
 - niestandardowe style, 370
 - odeczyt dokumentów, 367
 - zapis dokumentów, 372
 - znak podziału wiersza, 375
- współbieżność, 418
- współrzędne kursora myszy, 502
- wycinanie ciągów tekstowych, 161
- wycinek, 114
- wyjątek, 103, 262
- wykonywanie
 - programu, 66
 - skryptów, 425
- wykresy, 345
- wyłączenie
 - asercji, 268
 - rejestrowania informacji, 272
- wyodrębnianie
 - plików, 249
 - tekstu, 354
- wypełnienie struktury danych, 329
- wypunktowanie, 175
- wyrażenia regularne, 183
 - dopasowanie, 189, 201
 - dopasowanie niezachłanne, 194
 - dopasowanie zachłanne, 194
 - klasy znaków, 196
 - opcje zaawansowane, 203
 - przegląd znaków, 200
 - własne klasy znaków, 197
 - wyszukiwanie wzorców, 184, 186
- wyrównywanie tekstu, 168
- wysyłanie
 - formularzy, 168, 314
 - wiadomości e-mail, 433, 438, 451
 - wiadomości SMTP, 436
 - wiadomości tekstowych, 433–456
- wyszukiwanie
 - elementów IITML, 296, 313
 - elementów na stronie, 311

wyszukiwanie
wiadomości e-mail, 441
wyników, 302
wzorców, 184, 186

wyświetlanie
danych, 146
czasu, 406
kodu HTML, 293
tabeli, 179

Z

zablokowane okienka, 344
zagnieżdżone
listy, 152
słowniki, 152
zakończenie działania wątków, 421
zaokrąglanie liczb, 404
zapis
danych w formacie JSON, 392
do pliku, 331
dokumentów Excela, 332
dokumentów Worda, 372
obrazu, 308
pliku, 213, 222, 225, 290
pliku CSV, 389
wartości w komórkach, 334
zmiennych, 226, 227
zasięg
globalny, 97
lokalny, 97, 99

zastępowanie ciągu tekstowego, 202

zmiana
daty pliku, 251
nazwy plików, 243, 254
poszczególnych pikseli, 478
wielkości obrazu, 474, 482

zmiennie, 45
globalne, 97, 99
lokalne, 97, 98

znak
\$, 198
^, 198
gwiazdki, 192, 199
kropki, 199, 200
lewego ukośnika, 214
plusa, 193
podziału strony, 375
podziału wiersza, 375
prawego ukośnika, 214
wieloznaczny, 199
zapytania, 191
znaki sterujące, 158
zrzut ekranu, 508
zwalnianie klawiszy, 514

Ż

żądanie strony wyszukiwarki, 301

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION

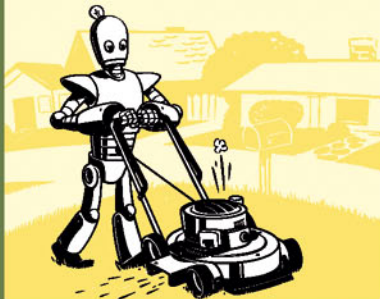


- 1. ZAREJESTRUJ SIĘ**
- 2. PREZENTUJ KSIĄŻKI**
- 3. ZBIERAJ PROWIZJĘ**

Zmień swoją stronę WWW
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

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



NIECH TWÓJ KOMPUTER CIĘ WYRĘCZY!

Komputer jest niezwykle wszechstronnym narzędziem, które może posłużyć do błyskawicznego wykonywania przeróżnych czynności. Dotyczy to również zadań, które zwykle zajmują mnóstwo czasu, a przy tym są męczące i nudne. Zamiast przez wiele godzin porównywać czy wprowadzać dane, lepiej dobrze zaprogramować komputer, który wykona takie zadania w ciągu kilku sekund. Aby to osiągnąć, wcale nie trzeba być profesjonalnym programistą!

Jeśli nie masz pojęcia o programowaniu, aie chcesz wykorzystać komputer do automatyzacji prostych, powtarzalnych operacji, trzymasz w ręku właściwą książkę. Podstawy programowania przedstawiono tu w sposób klarowny i zrozumiały, poszczególne działania wyjaśniono krok po kroku, a propozycje praktycznych projektów pozwalają na gruntowne przyswojenie materiału i poszerzenie wiedzy. Do nauki wykorzystano język Python, ponieważ jest on stosunkowo prosty, popularny, a przy tym wciąż rozwijany. Szybko przekonasz się, że czasu i wysiłku można zaoszczędzić dzięki automatyzacji zadań za pomocą krótkich i łatwych programów!

W książce znajdziesz między innymi:

- podstawy programowania w Pythonie
- opis kontroli przepływu działania programu
- informacje o pracy na danych tekstowych i plikach
- wiadomości o pracy z plikami CSV i danymi JSON
- harmonogramy zadań i opis kontroli czasu uruchamiania programu
- możliwości kontrolowania klawiatury i myszy za pomocą automatyzacji GUI

Albert Sweigart ----- programista, projektant oprogramowania i nauczyciel kodowania. Autor licznych książek o Pythonie dla początkujących. Urodził się w Houston w stanie Teksas, a obecnie mieszka w San Francisco. Prowadzi blog dostępny pod adresem: <http://coffeeghost.net>.

Helion

księgarnia internetowa



<http://helion.pl>

zamówienia telefoniczne



0 801 339900



0 601 339900

Informatyka w najlepszym wydaniu

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/novosci>

sięgnij po WIĘCEJ



KOD KORZYŚCI

ISBN 978-83-283-3260-7



9 788328 332607

cena: 89,00 zł

