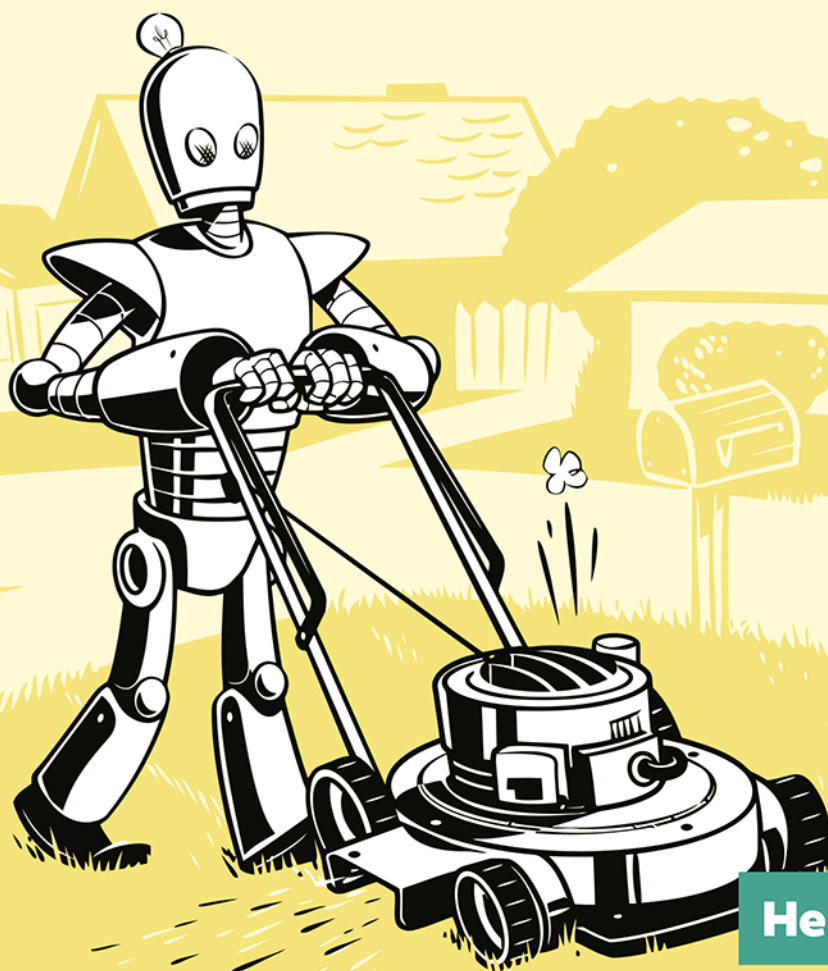


WYDANIE II

AUTOMATYZACJA NUDNYCH ZADAŃ Z PYTHONEM

NAUKA PROGRAMOWANIA

AL SWEIGART



Helion

Tytuł oryginału: Automate the Boring Stuff with Python, 2nd Edition:
Practical Programming for Total Beginners

Tłumaczenie: Robert Górczyński

ISBN: 978-83-289-2386-7

Copyright © 2020 by Al Sweigart. Title of English-language original: *Automate the Boring Stuff with Python, 2nd Edition: Practical Programming for Total Beginners*, ISBN: 978-1-59327-992-9, published by No Starch Press.

Polish-language edition copyright © 2021, 2025 by Helion SA. All rights reserved.

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

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

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

Autor oraz wydawca dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz wydawca nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<https://helion.pl/user/opinie/auto2v>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Helion S.A.

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 230 98 63

e-mail: helion@helion.pl

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

Printed in Poland.

- Kup książkę
- Poleć książkę
- Oceń książkę

- Księgarnia internetowa
- Lubię to! » Nasza społeczność

Spis treści

PODZIĘKOWANIA	19
----------------------	-----------

WPROWADZENIE	21
---------------------	-----------

Do kogo jest skierowana ta książka?	22
Konwencje	23
Czym jest programowanie?	23
Co to jest Python?	24
Programiści nie muszą dobrze znać matematyki	24
Nigdy nie jest za późno na rozpoczęcie nauki programowania	25
Programowanie to aktywność kreatywna	26
O tej książce	26
Pobieranie i instalacja Pythona	29
Pobieranie i instalowanie edytora tekstu Mu	30
Uruchomienie edytora Mu	30
Uruchomienie środowiska IDLE	31
Powłoka interaktywna	31
Instalowanie modułów zewnętrznych	33
Jak otrzymać pomoc?	33
Sprytne zadawanie pytań dotyczących programowania	34
Podsumowanie	35

CZĘŚĆ I. PODSTAWY PROGRAMOWANIA W PYTHONIE

1	
----------	--

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	46
Twój pierwszy program	48
Analiza programu	49
Komentarze	49
Funkcja print()	50
Funkcja input()	50
Wyświetlanie imienia użytkownika	51

Funkcja len()	51
Funkcje str(), int() i float()	52
Podsumowanie	55
Pytania kontrolne	56

2

KONTROLA PRZEPŁYWU DZIAŁANIA PROGRAMU 59

Wartości boolowskie	60
Operatory porównania	61
Operatory boolowskie	63
Binarne operatory boolowskie	63
Operator not	64
Łączenie operatorów boolowskich i porównania	64
Elementy kontroli przepływu działania programu	65
Warunek	65
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
Krótki program — odgadnij liczbę	89
Krótki program — kamień, papier, nożyce	91
Podsumowanie	95
Pytania kontrolne	95

3

FUNKCJE 97

Polecenie def wraz z parametrami	99
Definiowanie, wywoływanie, przekazywanie, argument i parametr	99
Wartość zwrotna funkcji i polecenie return	100
Wartość None	102
Argumenty w postaci słów kluczowych i funkcja print()	102
Stos wywołań	104
Zasięgi lokalny i globalny	106
Zmienne lokalne nie mogą być używane w zasięgu globalnym	107
W zasięgu lokalnym nie można używać zmiennych zdefiniowanych w innych zasięgach lokalnych	108
Zmienna globalna może być używana w zasięgu lokalnym	109
Zmienna lokalna i globalna o takiej samej nazwie	109
Polecenie global	110
Obsługa wyjątków	113
Krótki program — zigzag	115
Podsumowanie	117

Pytania kontrolne	117
Projekt praktyczny	118
Problem Collatza	118
Weryfikacja danych wyjściowych	119

4

LISTY	121
Typ danych List	121
Pobieranie poszczególnych wartości listy za pomocą indeksu	122
Indeks ujemny	124
Pobieranie podlisty za pomocą wycinka	124
Pobieranie długości listy za pomocą polecenia len()	125
Zmiana wartości na liście za pomocą indeksu	125
Konkatenacja i replikacja listy	126
Usunięcie wartości listy za pomocą polecenia del	126
Praca z listą	126
Użycie pętli for wraz z listą	128
Operatory in i not in	129
Sztuczka pozwalająca na wiele jednoczesnych operacji przypisania	130
Używanie funkcji enumerate() z listą	131
Używanie funkcji random.choice() i random.shuffle() z listą	131
Operatory przypisania i zmiany wartości	132
Metody	133
Odszukanie wartości na liście za pomocą metody index()	133
Dodanie wartości do listy za pomocą metod append() i insert()	134
Usuwanie wartości z listy za pomocą metody remove()	135
Sortowanie wartości listy za pomocą metody sort()	136
Odwrócenie kolejności wartości listy za pomocą metody reverse()	137
Przykładowy program — Magic 8 Ball utworzony za pomocą listy	137
Typy danych w postaci sekwencji	139
Modyfikowalne i niemodyfikowalne typy danych	139
Typ danych krotka	142
Konwersja typu za pomocą funkcji list() i tuple()	143
Odwołania	143
Identyfikator i funkcja id()	145
Przekazywanie odwołań	147
Funkcje copy() i deepcopy() modułu copy	147
Krótki program — gra w życie	148
Podsumowanie	153
Pytania kontrolne	154
Projekty praktyczne	154
Kod z przecinkami	155
Rzut monetą	155
Obraz na podstawie macierzy	156

5

SŁOWNIKI I STRUKTURYZACJA DANYCH	157
Typ danych Dictionary	157
Słownik kontra lista	158
Metody keys(), values() i items()	160
Sprawdzenie, czy klucz lub wartość istnieją w słowniku	162
Metoda get()	162
Metoda setdefault()	163

Eleganckie wyświetlanie danych	164
Użycie struktur danych do modelowania rzeczywistych rozwiązań	165
Plansza do gry w kółko i krzyżyk	166
Zagnieżdżone słowniki i listy	171
Podsumowanie	173
Pytania kontrolne	173
Projekty praktyczne	173
Weryfikacja słownika modelującego grę w szachy	174
Inwentarz w grze fantasy	174
Funkcja konwertująca listę na słownik dla inwentarza w grze fantasy	175

6

OPERACJE NA CIĄGACH TEKSTOWYCH

177

Praca z ciągami tekstowymi	177
Literaty ciągu tekstowego	178
Indeksowanie i wycinanie ciągów tekstowych	181
Użycie operatorów in i not in podczas pracy z ciągami tekstowymi	182
Umieszczenie ciągu tekstowego w innym	182
Użyteczne metody ciągu tekstowego	183
Metody upper(), lower(), isupper() i islower()	183
Metody typu isX()	185
Metody startswith() i endswith()	187
Metody join() i split()	188
Podział tekstu za pomocą metody partition()	189
Wyrównywanie tekstu za pomocą metod rjust(), ljust() i center()	190
Usunięcie białych znaków za pomocą strip(), rstrip() i lstrip()	192
Wartości liczbowe znaków pobrane za pomocą funkcji ord() i chr()	193
Kopiowanie i wklejanie ciągów tekstowych za pomocą modułu pyperclip	194
Projekt — schowek dla wielu ciągów tekstowych	195
Etap 1. Projekt programu i struktur danych	195
Etap 2. Obsługa argumentów wiersza poleceń	196
Etap 3. Skopiowanie odpowiedniej wiadomości	196
Projekt — dodanie wypunktowania do kodu znaczników Wiki	197
Etap 1. Kopiowanie i wklejanie ze schowka	198
Etap 2. Rozdzielenie wierszy tekstu i dodanie gwiazdki	199
Etap 3. Złączenie zmodyfikowanych wierszy	200
Krótki program — świńska łacina	200
Podsumowanie	204
Pytania kontrolne	205
Projekt praktyczny	206
Wyświetlenie tabeli	206
Symulator Zombie Dice	206

CZĘŚĆ II. AUTOMATYZACJA ZADAŃ

7

DOPASOWANIE WZORCA ZA POMOCĄ WYRAŻEŃ REGULARNYCH

213

Wyszukiwanie wzorców w tekście bez użycia wyrażeń regularnych	214
Wyszukiwanie wzorców w tekście z użyciem wyrażeń regularnych	216
Tworzenie obiektów wyrażeń regularnych	217
Dopasowanie obiektów wyrażeń regularnych	218
Przegląd dopasowania za pomocą wyrażenia regularnego	218

Jeszcze więcej o dopasowaniach wzorca za pomocą wyrażeń regularnych	219
Grupowanie z użyciem nawiasów	219
Dopasowanie wielu grup za pomocą potoku	221
Opcjonalne dopasowanie za pomocą znaku zapytania	222
Dopasowanie zera wystąpień lub większej liczby wystąpień za pomocą gwiazdki	222
Dopasowanie jednego wystąpienia lub wielu wystąpień za pomocą plusa	223
Dopasowanie określonych powtórzeń za pomocą nawiasu klamrowego	224
Dopasowanie zachłanne i niezachłanne	225
Metoda findall()	225
Klasy znaków	226
Utworzenie własnej klasy znaków	227
Znaki ^ oraz \$	228
Znak wieloznaczny	229
Dopasowanie wszystkiego za pomocą kropki i gwiazdki	229
Dopasowanie znaku nowego wiersza za pomocą kropki	230
Przegląd znaków stosowanych w wyrażeniach regularnych	231
Dopasowanie bez uwzględnienia wielkości znaków	232
Zastępowanie ciągu tekstowego za pomocą metody sub()	232
Zarządzanie skomplikowanymi wyrażeniami regularnymi	233
Połączenie opcji re.IGNORECASE, re.DOTALL i re.VERBOSE	234
Projekt — wyodrębnianie numeru telefonu i adresu e-mail	235
Etap 1. Utworzenie wyrażenia regularnego dopasowującego numer telefonu	236
Etap 2. Utworzenie wyrażenia regularnego dopasowującego adres e-mail	237
Etap 3. Wyszukanie wszystkich dopasowań w tekście umieszczonym w schowku	237
Etap 4. Połączenie dopasowań w celu utworzenia pojedynczego ciągu tekstowego do umieszczenia w schowku	238
Uruchomienie programu	239
Pomysły na podobne programy	239
Podsumowanie	240
Pytania kontrolne	240
Projekty praktyczne	242
Wykrywanie daty	242
Wykrywanie silnego hasła	242
Oparta na wyrażeniu regularnym wersja metody strip()	243

8

WERYFIKACJA DANYCH WEJŚCIOWYCH	245
Moduł PyInputPlus	246
Argumenty w postaci słów kluczowych min, max, greaterThan i lessThan	248
Argument w postaci słowa kluczowego blank	249
Argumenty w postaci słów kluczowych limit, timeout i default	250
Argumenty w postaci słów kluczowych allowRegexes i blockRegexes	251
Przekazanie do inputCustom() niestandardowej funkcji weryfikacji danych wejściowych	252
Projekt — zajęcie kogoś godzinami	253
Projekt — quiz z tabliczki mnożenia	255
Podsumowanie	257
Pytania kontrolne	258
Projekty praktyczne	258
Program przygotowujący kanapki	259
Własna wersja quizu z zakresu tabliczki mnożenia	259

ODCZYT I ZAPIS PLIKÓW**261**

Pliki i ścieżki dostępu do plików	261
Lewy ukośnik w systemie Windows, prawy ukośnik w systemach macOS i Linux	262
Używanie operatora / do łączenia ścieżek dostępu	264
Bieżący katalog roboczy	266
Katalog domowy	267
Względne kontra bezwzględne ścieżki dostępu	267
Tworzenie nowych katalogów za pomocą funkcji <code>os.makedirs()</code>	267
Obsługa bezwzględnych i względnych ścieżek dostępu	268
Pobieranie fragmentów ścieżki dostępu do pliku	270
Ustalenie wielkości pliku i zawartości katalogu	273
Modyfikowanie listy plików za pomocą wzorców glob	274
Sprawdzenie poprawności ścieżki dostępu	276
Proces odczytu i zapisu pliku	277
Otwieranie pliku za pomocą funkcji <code>open()</code>	278
Odczyt zawartości pliku	279
Zapis pliku	280
Zapis zmiennych za pomocą modułu <code>shelve</code>	281
Zapis zmiennych za pomocą funkcji <code>pprint.pformat()</code>	283
Projekt — generowanie losowych plików quizu	284
Etap 1. Umieszczenie danych quizu w słowniku	285
Etap 2. Utworzenie pliku quizu i losowe umieszczenie odpowiedzi na pytania	286
Etap 3. Utworzenie odpowiedzi	287
Etap 4. Zapis treści w plikach quizu i odpowiedzi	288
Projekt — schowek przechowujący wiele elementów	289
Etap 1. Komentarze i konfiguracja pliku binarnego	290
Etap 2. Zapis zawartości schowka wraz ze słowem kluczowym	291
Etap 3. Wyświetlenie słów kluczowych i wczytanie treści powiązanej ze słowem kluczowym	291
Podsumowanie	292
Pytania kontrolne	293
Projekty praktyczne	293
Rozbudowa programu schowka przechowującego wiele elementów	294
Program <code>Mad Libs</code>	294
Wyszukiwanie wyrażenia regularnego	294

ORGANIZACJA PLIKÓW**295**

Moduł <code>shutil</code>	296
Kopiowanie plików i katalogów	296
Przenoszenie oraz zmiana nazwy plików i katalogów	297
Trwałe usunięcie plików i katalogów	298
Bezpieczne usuwanie danych za pomocą modułu <code>send2trash</code>	299
Przejsięcie przez drzewo katalogu	300
Kompresja plików za pomocą modułu <code>zipfile</code>	302
Odczyt pliku w formacie ZIP	302
Wyodrębnianie plików z archiwum ZIP	303
Utworzenie i dodawanie elementów do archiwum ZIP	304
Projekt — zmiana plików z datami w stylu amerykańskim na daty w stylu europejskim	304
Etap 1. Utworzenie wyrażenia regularnego dla daty w stylu amerykańskim	305
Etap 2. Identyfikacja w nazwie pliku fragmentów określających datę	307
Etap 3. Utworzenie nowej nazwy pliku i zmiana nazw plików	308
Pomysły na podobne programy	309

Projekt — utworzenie archiwum ZIP będącego kopią katalogu	309
Etap 1. Ustalenie nazwy pliku archiwum ZIP	309
Etap 2. Utworzenie nowego archiwum ZIP	310
Etap 3. Przejście przez drzewo katalogu i dodanie plików do archiwum ZIP	311
Pomysły na podobne programy	312
Podsumowanie	312
Pytania kontrolne	313
Projekty praktyczne	313
Kopiowanie selektywne	314
Usunięcie niepotrzebnych plików	314
Wypełnienie przerw	314

11

USUWANIE BŁĘDÓW 315

Zgłaszanie wyjątku	316
Pobranie stosu wywołań w postaci ciągu tekstowego	318
Asercje	319
Użycie asercji w projekcie symulacji ulicznej sygnalizacji świetlnej	321
Rejestracja danych	322
Użycie modułu logging	323
Nie przeprowadzaj procesu usuwania błędów za pomocą funkcji print()	325
Poziomy rejestrowania informacji	325
Wyłączenie rejestrowania informacji	326
Rejestrowanie informacji w pliku	327
Debugger edytora Mu	327
Kontynuuj	328
Krok do wewnątrz	329
Przekrocz	329
Krok na zewnątrz	329
Zatrzymaj	329
Debugowanie programu sumującego liczby	329
Punkty kontrolne	331
Podsumowanie	333
Pytania kontrolne	333
Projekt praktyczny	334
Debugowanie programu symulującego rzut monetą	334

12

POBIERANIE DANYCH Z INTERNETU 337

Projekt — maplt.py z użyciem modułu webbrowser	338
Etap 1. Ustalenie adresu URL	339
Etap 2. Obsługa argumentów wiersza poleceń	339
Etap 3. Obsługa zawartości schowka i uruchomienie przeglądarki WWW	340
Pomysły na podobne programy	341
Pobieranie plików z internetu za pomocą modułu requests	341
Pobieranie strony internetowej za pomocą funkcji requests.get()	342
Sprawdzenie pod kątem błędów	343
Zapis pobranych plików na dysku twardym	344
HTML	345
Zasoby pomagające w poznawaniu języka HTML	346
Krótkie wprowadzenie	346
Wyświetlenie kodu źródłowego HTML strony internetowej	347

Wyświetlenie oferowanych przez przeglądarkę WWW narzędzi programistycznych	348
Użycie narzędzi programistycznych do wyszukiwania elementów HTML	350
Przetwarzanie kodu HTML za pomocą modułu bs4	352
Utworzenie obiektu BeautifulSoup na podstawie kodu HTML	352
Wyszukiwanie elementu za pomocą metody select()	353
Pobieranie danych z atrybutów elementu	355
Projekt — wyświetlenie wyników wyszukiwania	356
Etap 1. Pobranie argumentów wiersza poleceń i żądanie strony wyszukiwarki	357
Etap 2. Wyszukiwanie wszystkich wyników	357
Etap 3. Otworzenie kart przeglądarki WWW dla poszczególnych wyników	358
Pomysły na podobne programy	359
Projekt — pobranie wszystkich komiksów z witryny XKCD	360
Etap 1. Projekt programu	361
Etap 2. Pobranie strony internetowej	362
Etap 3. Odszukanie i pobranie obrazu komiksu	363
Etap 4. Zapis obrazu i odszukanie poprzedniego komiksu	363
Pomysły na podobne programy	365
Kontrolowanie przeglądarki WWW za pomocą modułu selenium	365
Uruchomienie przeglądarki WWW kontrolowanej przez moduł selenium	366
Wyszukiwanie elementów na stronie	368
Kliknięcie na stronie	370
Wypełnianie i wysyłanie formularzy sieciowych	370
Symulacja naciśnięcia klawiszy specjalnych	371
Klikanie przycisków przeglądarki WWW	372
Więcej informacji na temat modułu selenium	372
Podsumowanie	372
Pytania kontrolne	373
Projekty praktyczne	374
Klient poczty działający w wierszu poleceń	374
Pobieranie obrazów z witryny internetowej	374
2048	374
Weryfikacja łącza	375

13

PRACA Z ARKUSZAMI KALKULACYJNYMI PROGRAMU EXCEL

377

Dokumenty Excela	378
Instalacja modułu openpyxl	378
Odczyt dokumentów Excela	379
Otwieranie istniejącego dokumentu Excela za pomocą openpyxl	379
Pobranie arkuszy ze skoroszytu	380
Pobieranie komórek z arkuszy	380
Konwersja między literami kolumn i liczbami	382
Pobieranie wierszy i kolumn z arkuszy	383
Skoroszyty, arkusze i komórki	385
Projekt — odczyt danych z arkusza kalkulacyjnego	385
Etap 1. Odczyt danych z arkusza kalkulacyjnego	386
Etap 2. Wypełnienie struktury danych	387
Etap 3. Zapis wyników do pliku	389
Pomysły na podobne programy	390
Zapis dokumentów Excela	390
Tworzenie i zapisywanie dokumentów Excela	391
Tworzenie i usuwanie arkuszy kalkulacyjnych	391
Zapis wartości w komórkach	392

Projekt — uaktualnienie skoroszytu	393
Etap 1. Przygotowanie struktury danych wraz z uaktualnionymi informacjami	394
Etap 2. Sprawdzenie wszystkich wierszy i skorygowanie nieprawidłowych cen	395
Pomysły na podobne programy	396
Ustawienie stylu czcionki komórek	396
Obiekt Font	397
Formuły	398
Dostosowanie wierszy i kolumn do własnych potrzeb	400
Ustalenie wysokości wiersza i szerokości kolumny	400
Łączenie i dzielenie komórki	401
Zablokowane okienka	402
Wykresy	403
Podsumowanie	404
Pytania kontrolne	405
Projekty praktyczne	406
Program tworzący tabliczkę mnożenia	406
Program wstawiający pusty wiersz	407
Program zmieniający położenie komórek arkusza kalkulacyjnego	407
Przeniesienie zawartości pliku tekstowego do arkusza kalkulacyjnego	408
Przeniesienie zawartości arkusza kalkulacyjnego do plików tekstowych	409

14

PRACA Z ARKUSZAMI GOOGLE 411

Instalacja i konfiguracja EZSheets	411
Pobranie danych uwierzytelniających i plików tokenów	412
Unieważnienie pliku danych uwierzytelniających	414
Obiekt skoroszytu	415
Tworzenie, przekazywanie i wyświetlanie skoroszytów	415
Atrybuty skoroszytu	417
Pobieranie i przekazywanie skoroszytów	418
Usuwanie skoroszytu	419
Obiekt arkusza	419
Odczytywanie i zapisywanie danych	420
Tworzenie i usuwanie arkuszy	425
Kopiowanie arkusza	426
Praca z ograniczeniami nakładanymi przez Arkusze Google	427
Podsumowanie	428
Pytania kontrolne	428
Projekty praktyczne	429
Pobieranie danych Formularzy Google	429
Konwertowanie skoroszytów na inne formaty	429
Wyszukiwanie błędów w skoroszytach	429

15

PRACA Z DOKUMENTAMI PDF I WORDA 431

Dokumenty w formacie PDF	431
Wyodrębnianie tekstu z dokumentu PDF	432
Deszyfrowanie dokumentu PDF	434
Tworzenie dokumentów PDF	435
Projekt — połączenie wybranych stron z wielu dokumentów PDF	440
Etap 1. Wyszukanie wszystkich plików w formacie PDF	441
Etap 2. Otworzenie poszczególnych dokumentów PDF	442

Etap 3. Dodanie poszczególnych stron	442
Etap 4. Zapis dokumentu wynikowego	443
Pomysły na podobne programy	444
Dokumenty procesora tekstu Microsoft Word	444
Odczyt dokumentów Worda	445
Pobranie pełnego tekstu z pliku w formacie .docx	446
Nadawanie stylu akapitów i obiektom Run	447
Utworzenie dokumentu Worda z niestandardowymi stylami	449
Atrybuty obiektu Run	449
Zapis dokumentów Worda	451
Dodanie nagłówków	453
Dodanie znaku podziału wiersza i strony	454
Dodanie obrazu	454
Tworzenie dokumentu PDF na podstawie dokumentu Worda	455
Podsumowanie	455
Pytania kontrolne	456
Projekty praktyczne	457
PDF Paranoja	457
Własne zaproszenia utworzone w dokumencie Worda	457
Program łamiący hasło dokumentu PDF za pomocą ataku typu brute force	458

16

PRACA Z PLIKAMI CSV I DANYMI JSON

461

Moduł csv	462
Obiekt reader	463
Użycie pętli for do odczytu danych z obiektu reader	464
Obiekt writer	464
Argumenty w postaci słów kluczowych delimiter i lineterminator	466
Obiekty CSV DictReader i DictWriter	467
Projekt — usunięcie nagłówka z pliku CSV	469
Etap 1. Iteracja przez poszczególne pliki CSV	470
Etap 2. Odczyt zawartości pliku CSV	470
Etap 3. Zapis pliku CSV bez pierwszego wiersza	471
Pomysły na podobne programy	472
JSON i API	473
Moduł json	474
Odczyt danych JSON za pomocą funkcji loads()	474
Zapis danych w formacie JSON za pomocą funkcji dumps()	475
Projekt — pobieranie bieżących danych prognozy pogody	475
Etap 1. Pobranie z wiersza poleceń informacji o lokalizacji	476
Etap 2. Pobranie danych w formacie JSON	477
Etap 3. Wczytanie danych w formacie JSON i wyświetlenie prognozy pogody	478
Pomysły na podobne programy	480
Podsumowanie	480
Pytania kontrolne	481
Projekty praktyczne	481
Konwerter danych w formacie Excel do formatu CSV	481

17

CZAS, HARMONOGRAM ZADAŃ I URUCHAMIANIE PROGRAMÓW

483

Moduł time	483
Funkcja time.time()	484
Funkcja time.sleep()	485

Zaokrąglanie liczb	486
Projekt — superstoper	486
Etap 1. Przygotowanie programu do pomiaru czasu	487
Etap 2. Monitorowanie i wyświetlenie czasu okrążenia	488
Pomysły na podobne programy	489
Moduł datetime	490
Typ danych timedelta	491
Pauza aż do chwili osiągnięcia określonej daty	493
Konwersja obiektu datetime na ciąg tekstowy	493
Konwersja ciągu tekstowego na obiekt datetime	495
Przegląd funkcji czasu w Pythonie	495
Wielowątkowość	496
Przekazanie argumentów funkcji docelowej dla nowego wątku	498
Kwestie związane ze współbieżnością	499
Projekt — wielowątkowy program pobierający dane z witryny XKCD	500
Etap 1. Modyfikacja programu w celu użycia funkcji	500
Etap 2. Utworzenie i uruchomienie wątków	502
Etap 3. Zaczekanie na zakończenie działania wszystkich wątków	502
Uruchamianie innych programów z poziomu Pythona	503
Przekazanie funkcji Popen() argumentów wiersza poleceń	506
Harmonogram zadań, launchd i cron	506
Otwieranie witryn internetowych za pomocą Pythona	507
Wykonywanie innych skryptów Pythona	507
Otwieranie plików w ich aplikacjach domyślnych	508
Projekt — prosty program odliczający czas	509
Etap 1. Odliczanie	509
Etap 2. Odtworzenie pliku dźwiękowego	510
Pomysły na podobne programy	510
Podsumowanie	511
Pytania kontrolne	512
Projekty praktyczne	512
Ładniejszy stoper	512
Oparty na harmonogramie program pobierający komiksy	513
18	
WYSYŁANIE WIADOMOŚCI E-MAIL I TEKSTOWYCH	515
Wysyłanie i odbieranie poczty za pomocą API Gmail	516
Włączenie API Gmail	516
Wysyłanie wiadomości za pomocą konta Gmail	517
Odczytywanie wiadomości za pomocą konta Gmail	518
Wyszukiwanie wiadomości w koncie poczty Gmail	520
Pobieranie załączników z konta Gmail	521
SMTP	521
Wysyłanie wiadomości e-mail	522
Nawiązanie połączenia z serwerem SMTP	522
Wysłanie wiadomości SMTP typu „Witaj”	524
Włączenie szyfrowania TLS	524
Logowanie w serwerze SMTP	525
Wysyłanie wiadomości e-mail	525
Zamknięcie połączenia z serwerem SMTP	526
IMAP	526

Pobieranie i usuwanie wiadomości e-mail za pomocą protokołu IMAP	527
Nawiązanie połączenia z serwerem IMAP	527
Logowanie w serwerze IMAP	528
Wyszukiwanie wiadomości e-mail	529
Pobieranie wiadomości e-mail i oznaczanie jej jako przeczytanej	533
Pobieranie adresów e-mail z niezmodyfikowanych wiadomości e-mail	534
Pobranie treści z niezmodyfikowanej wiadomości e-mail	535
Usuwanie wiadomości e-mail	536
Zamknięcie połączenia z serwerem IMAP	536
Projekt — wysyłanie wiadomości e-mail z przypomnieniami o składkach	537
Etap 1. Otworzenie pliku Excela	538
Etap 2. Wyszukanie wszystkich członków klubu, którzy zalegają ze składką	539
Etap 3. Wysłanie spersonalizowanego przypomnienia	540
Wysyłanie wiadomości tekstowych za pomocą bramek SMS	541
Wysyłanie wiadomości tekstowych za pomocą Twilio	543
Założenie konta w serwisie Twilio	544
Wysyłanie wiadomości tekstowych	544
Projekt — moduł typu „wyślij mi wiadomość SMS”	546
Podsumowanie	548
Pytania kontrolne	548
Projekty praktyczne	549
Program losowo przypisujący uciążliwe zadania	549
Przypomnienie o parasolu	550
Automatyczna rezygnacja z subskrypcji	550
Kontrola komputera za pomocą wiadomości e-mail	550

19

PRACA Z OBRAZAMI

553

Podstawy teorii obrazu cyfrowego	553
Kolory i wartości RGBA	554
Współrzędne i krotki pudełek	555
Praca z obrazami za pomocą modułu pillow	556
Praca z typem danych Image	558
Przycinanie obrazu	559
Kopiowanie i wklejanie obrazów w innych obrazach	560
Zmiana wielkości obrazu	563
Rotacja i lustrzane odbicie obrazu	564
Zmiana poszczególnych pikseli	566
Projekt — dodanie logo	568
Etap 1. Otworzenie pliku logo	569
Etap 2. Iteracja przez wszystkie pliki i otworzenie obrazów	570
Etap 3. Zmiana wielkości obrazu	571
Etap 4. Dodanie obrazu logo i zapisanie zmian	572
Pomysły na podobne programy	573
Rysowanie na obrazach	574
Rysowanie kształtów	574
Umieszczanie tekstu na obrazie	576
Podsumowanie	578
Pytania kontrolne	579
Projekty praktyczne	579
Rozbudowa i poprawa projektów omówionych w rozdziale	580
Odszukanie na dysku twardym katalogów zawierających zdjęcia	580
Własne wizytówki	581

KONTROLOWANIE KLAWIATURY I MYSZY ZA POMOCĄ AUTOMATYZACJI GUI	583
Instalacja modułu pyautogui	584
Konfiguracja ustawień dostępności w macOS	585
Pozostajemy na kursie	585
Pauzy i funkcja bezpiecznej awarii	585
Zamknięcie wszystkiego przez wylogowanie się	586
Kontrola poruszania myszą	586
Poruszanie kursorem myszy	587
Pobranie informacji o położeniu kursora myszy	588
Kontrola działania myszy	589
Kliknięcie myszą	589
Przeciąganie myszą	590
Przewijanie myszą	592
Planowanie ruchu myszą	592
Praca z ekranem	594
Wykonanie rzutu ekranu	594
Analiza rzutu ekranu	594
Rozpoznawanie obrazu	596
Pobieranie informacji o oknie	598
Pobranie aktywnego okna	598
Inne sposoby na pobieranie okna	599
Przeprowadzanie operacji na oknach	600
Kontrola klawiatury	602
Przekazanie ciągu tekstowego z klawiatury	602
Nazwy klawiszy	603
Naciskanie i zwalnianie klawiszy	604
Kombinacja klawiszy	604
Konfiguracja skryptów automatyzacji GUI	605
Przegląd funkcji modułu pyautogui	607
Projekt — automatyczne wypełnianie formularzy	608
Etap 1. Ustalenie kroków do wykonania	610
Etap 2. Przygotowanie współrzędnych	611
Etap 3. Rozpoczęcie wpisywania danych	612
Etap 4. Obsługa rozwijanych list i przycisków opcji	613
Etap 5. Wystanie formularza i oczekiwanie	614
Wyświetlanie okien dialogowych	615
Podsumowanie	616
Pytania kontrolne	617
Projekty praktyczne	618
Symulowanie zajętości	618
Używanie schowka do odczytywania pola tekstowego	618
Bot komunikatora internetowego	619
Samouczek dotyczący bota grającego w grę	620

A

INSTALACJA MODUŁÓW FIRM TRZECICH	621
Narzędzie pip	621
Instalacja modułów firm trzecich	622
Instalowanie modułów dla edytora Mu	624

B

URUCHAMIANIE PROGRAMÓW

627

Uruchamianie programów z poziomu powłoki	627
Uruchamianie programów Pythona w Windows	629
Uruchamianie programów Pythona w systemie macOS	630
Uruchamianie programów Pythona w systemie Ubuntu Linux	631
Uruchamianie programów Pythona z wyłączonymi asercjami	632

C

ODPOWIEDZI NA PYTANIA KONTROLNE

633

Rozdział 1.	634
Rozdział 2.	634
Rozdział 3.	636
Rozdział 4.	637
Rozdział 5.	638
Rozdział 6.	638
Rozdział 7.	639
Rozdział 8.	640
Rozdział 9.	640
Rozdział 10.	641
Rozdział 11.	641
Rozdział 12.	642
Rozdział 13.	643
Rozdział 14.	644
Rozdział 15.	645
Rozdział 16.	645
Rozdział 17.	646
Rozdział 18.	646
Rozdział 19.	647
Rozdział 20.	648

12

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.

- 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 **bs4** umożliwia przetwarzanie kodu HTML, za pomocą którego są tworzone strony internetowe.

¹ Odpowiedź na to pytanie brzmi nie.

- 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('https://inventwithpython.com/')
```

W przeglądarce WWW zostanie utworzona nowa karta, w której będzie wyświetlona witryna znajdująca się pod adresem URL `https://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 wymienię w punktach sposób działania programu.

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

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

1. Odczyt argumentów wiersza poleceń ze zmiennej `sys.argv`.
2. Odczyt zawartości schowka.
3. 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 wykonać 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 (<https://maps.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!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* → *wy_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 12.1 wymienilem kroki niezbędne do wykonania, aby wspomniany adres na mapie wyświetlić bez użycia programu *mapIt.py*.

Tabela 12.1. Wyświetlenie adresu na mapie z użyciem mapIt.py i bez zastosowania tego programu

Ręczne wyświetlenie adresu na mapie	Użycie mapIt.py
1. Zaznaczenie adresu.	1. Zaznaczenie adresu.
2. Skopiowanie adresu.	2. Skopiowanie adresu.
3. Otworzenie przeglądarki WWW.	3. Uruchomienie programu <i>mapIt.py</i> .
4. Przejście pod adres URL <i>https://mapy.google.pl/</i> .	
5. Kliknięcie pola tekstowego adresu.	
6. Wklejenie adresu.	
7. Naciśnięcie klawisza <i>Enter</i> .	

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

Pomysły na podobne programy

Gdy masz adres URL, moduł webbrowser pozwala wyeliminować krok, jakim jest otworenie przeglądarki WWW i przejście pod wskazany adres URL. Poniżej wymienilem 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 --user 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
❶ >>> res = requests.get('https://www.gutenberg.org/files/27062/27062-0.txt')
>>> type(res)
<class 'requests.models.Response'>
❷ >>> res.status_code == requests.codes.ok
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). Pełną listę kodów stanu HTTP i ich znaczenie znajdziesz na stronie https://pl.wikipedia.org/wiki/Kod_odpowiedzi_HTTP.

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.

Jeżeli wykonanie żądania zakończy się niepowodzeniem i zostanie wyświetlony komunikat błędu, taki jak `Failed to establish a new connection` lub `Max retries exceeded`, wówczas sprawdź połączenie z internetem. Nawiązywanie połączenia z serwerem może być dość skomplikowanym procesem i dlatego nie zamierzam w tym miejscu przedstawiać pełnej listy potencjalnych problemów. Przyczyny najczęściej występujących błędów możesz łatwo ustalić po wpisaniu w ulubionej wyszukiwarce internetowej otrzymanego komunikatu błędu.

Sprawdzenie pod kątem błędów

Jak wcześniej widziałeś, obiekt `Response` ma atrybut `status_code`, którego wartość można sprawdzić. Jeżeli jest nią `requests.codes.ok` (czyli zmienna przechowująca wartość w postaci liczby całkowitej 200), oznacza to, że żądanie zakończyło się powodzeniem. Proszym 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('https://inventwithpython.com/strona_ktora_nie_istnieje')
>>> res.raise_for_status()
Traceback (most recent call last):
  File <stdin>, line 1, in <module>

  File "C:\Users\A1\AppData\Local\Programs\Python\Python37\lib\site-packages\
↳requests\models
.py", line 940, in raise_for_status
    raise HTTPError(http_error_msg, response=self)
requests.exceptions.HTTPError: 404 Client Error: Not Found for url:
https://inventwithpython.com/strona_ktora_nie_istnieje.html
```

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('https://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 https://inventwithpython.com/strona_ktora_nie_istnieje.

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 binarnym*, 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 <https://www.joelonsoftware.com/articles/Unicode.html>.
- *Pragmatic Unicode* na stronie <https://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('https://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 <https://requests.readthedocs.io/en/master/>.

HTML

Zanim będziesz mógł zająć się poważniej stronami internetowymi, najpierw powinienś 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.

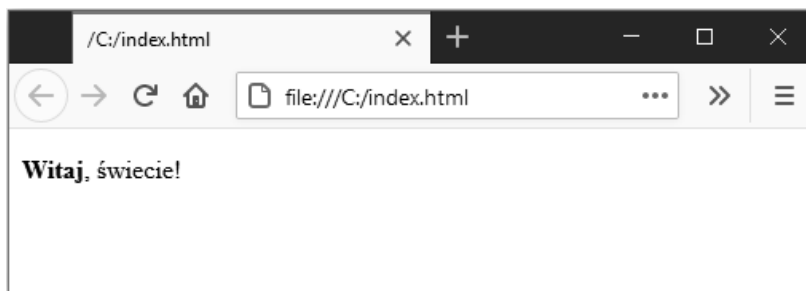
- <https://htmldog.com/guides/html/beginner/>
- <https://www.codecademy.com/learn/learn-html>
- <https://developer.mozilla.org/en-US/docs/Learn/HTML>

Krótkie wprowadzenie

Gdy 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 12.1.



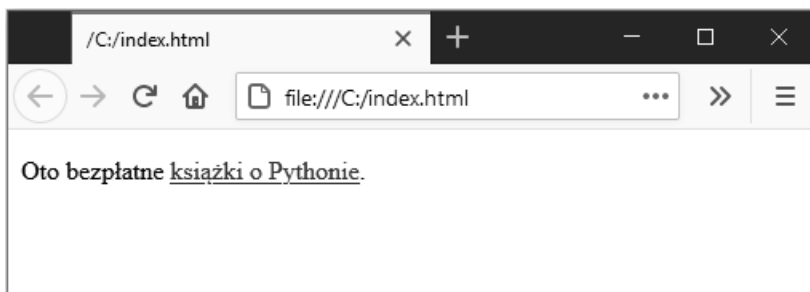
Rysunek 12.1. Komunikat „Witaj, świecie!” wyświetlony przez przeglądarkę WWW

Znacznik otwierający `<strong>` informuje, że obejmowany przez niego tekst ma zostać pogrubiony. Z kolei znacznik zamykający `</strong>` 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="https://inventwithpython.com">książki o Pythonie</a>`.

Po wygenerowaniu przez przeglądarkę WWW tekst ten będzie wyglądał tak, jak pokazałem na rysunku 12.2.



Rysunek 12.2. 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 (w systemie macOS kliknij z naciśniętym klawiszem `Ctrl`) dowolną stronę internetową w przeglądarce WWW, a następnie wybierz opcję *Wyświetl źródło* lub *Pokaż źródło strony*, aby faktycznie zobaczyć kod HTML tej strony (patrz rysunek 12.3). 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 12.4). 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.

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.

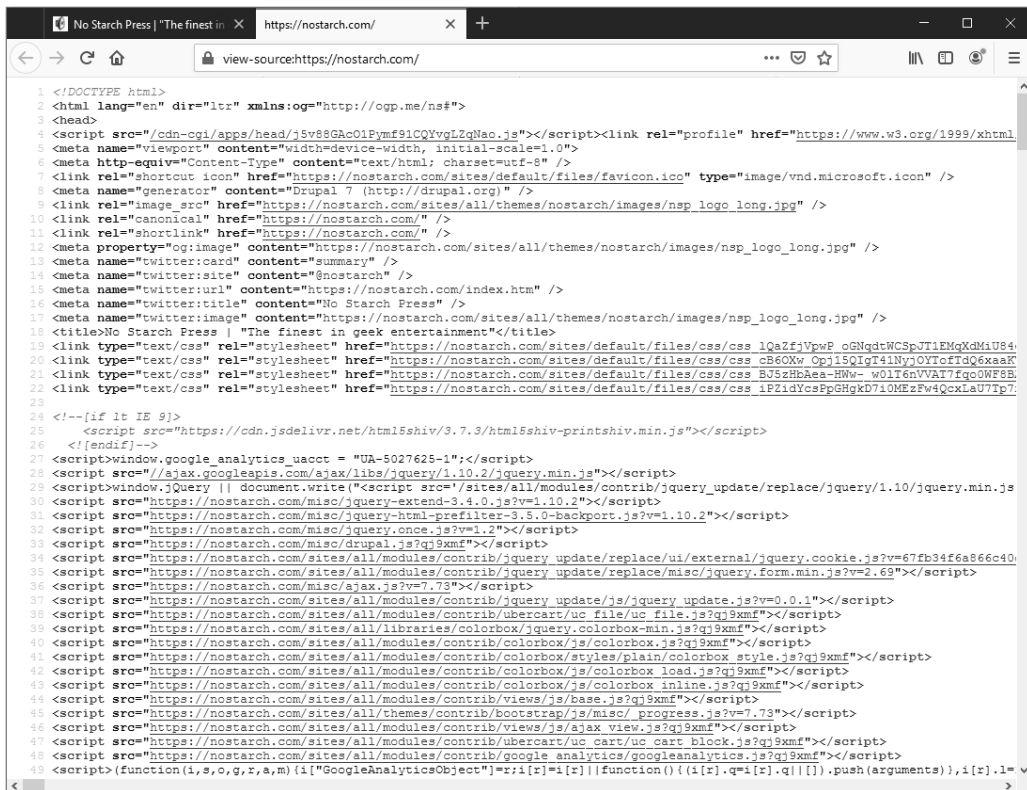
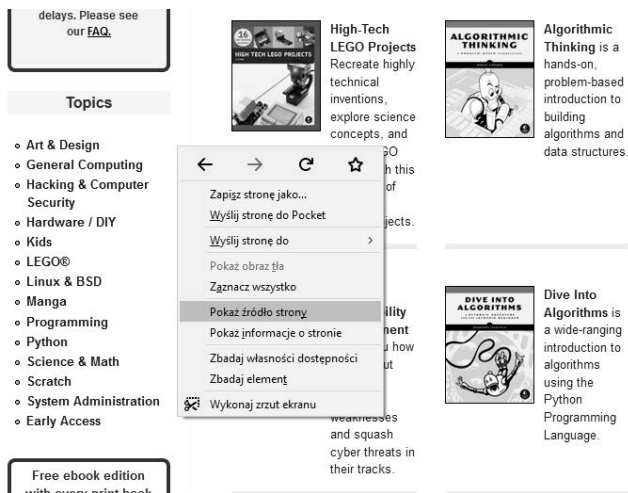
W przeglądarce Safari przejdź do karty *Zaawansowane* okna preferencji, a następnie zaznacz pole wyboru *Pokazuj menu Programowanie na pasku menu*. Po włączeniu wymienionego menu narzędzia programistyczne będziesz mógł wyświetlić po naciśnięciu klawiszy *Command+Option+I*.

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.

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 *bs4* i otrzymane za jego pomocą wyniki będą prawdopodobnie miały znacznie mniej błędów.

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



Rysunek 12.3. Wyświetlenie kodu źródłowego strony internetowej



Rysunek 12.4. Okno narzędzi programistycznych w przeglądarce Chrome

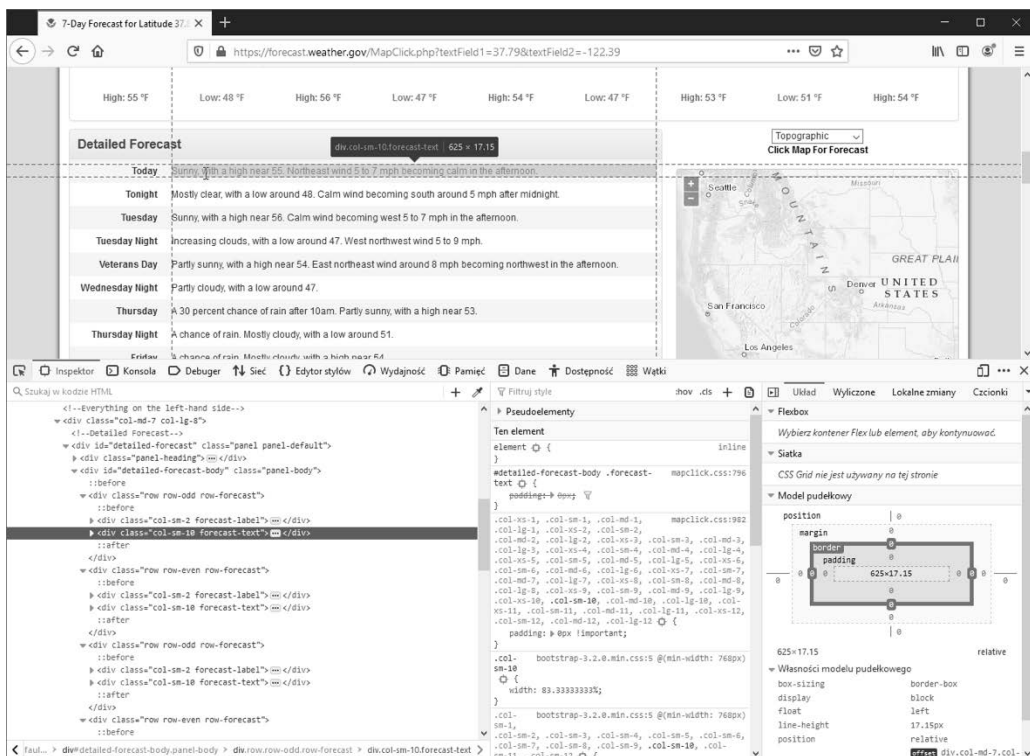
Użycie narzędzi programistycznych do wyszukiwania elementów HTML

Gdy 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 <https://www.weather.gov/> dane dotyczące prognozy pogody. Przed napisaniem jakiegokolwiek kodu musisz przeprowadzić małe badania. Jeżeli odwiedzisz 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 (w systemie macOS kliknij z naciśniętym klawiszem Ctrl) 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 12.5 pokazałem narzędzia programistyczne wraz z kodem HTML generującym fragment strony wyświetlający temperaturę. Należy pamiętać, że w przypadku zmiany układu strony witryny internetowej <https://www.weather.gov/> konieczne będzie powtórzenie tego procesu i przeanalizowanie nowych elementów.



Rysunek 12.5. 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 `<div class="col-sm-10 forecast-text">Sunny, with a high near 55. Northeast wind 5 to 7 mph becoming calm in the afternoon.</div>`. To jest dokładnie to, czego szukamy! Wydaje się, że informacje o temperaturze są umieszczone w elemencie `<div>` o klasie CSS `forecast-text`. Prawym przyciskiem myszy kliknij ten element w narzędziach programistycznych przeglądarki WWW, a następnie z menu kontekstowego wybierz opcję *Copy/CSS Selector*. Spowoduje to skopiowanie do schowka ciągu tekstowego w postaci podobnej do `'div.row-odd:nth-child(1) > div:nth-child(2)'`. Tego ciągu tekstowego będzie można później użyć z metodą `select()` modułu BeautifulSoup lub `find_element_by_css_selector()` modułu Selenium, co dokładnie wyjaśnię dalej w tym rozdziale. Skoro już wiesz, czego szukasz, moduł BeautifulSoup pomoże Ci w odszukaniu właściwego ciągu tekstowego.

Przetwarzanie kodu HTML za pomocą modułu bs4

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 --user beautifulsoup4`. (W dodatku A znajdziesz więcej informacji na temat instalacji modułów opracowanych przez firmy trzecie). Wprowadź `beautifulsoup4` do 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 nową kartę edytora pliku w Mu i wprowadź poniższy fragment kodu. Następnie zapisz plik pod nazwą *example.html*. Ewentualnie możesz pobrać ten plik pod adresem <https://ftp.helion.pl/przyklady/autop2.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="https://
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('https://nostarch.com')
>>> res.raise_for_status()
```

```
>>> noStarchSoup = bs4.BeautifulSoup(res.text, 'html.parser')
>>> 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()`. Drugi argument wskazuje modułowi `BeautifulSoup` komponent, który ma zostać wykorzystany do analizy danych HTML.

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, 'html.parser')
>>> type(exampleSoup)
<class 'bs4.BeautifulSoup'>
```

Użyty tutaj komponent `'html.parser'` jest dostarczany razem z Pythonem. Po zainstalowaniu modułu zewnętrznego `lxml` będziesz miał możliwość skorzystania z szybszego komponentu `'lxml'`. Zapoznaj się z zamieszczonymi w dodatku A informacjami na temat zainstalowania tego modułu za pomocą polecenia `pip install --user lxml`. Jeżeli zapomnisz o drugim argumencie wywołania `BeautifulSoup()`, nastąpi wygenerowanie komunikatu ostrzeżenia `UserWarning: No parser was explicitly specified`.

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 książki (na stronie <https://automatetheboringstuff.com/list-of-css-selector-tutorials.html> znajdziesz listę zasobów zawierających więcej informacji dotyczących składni selektorów CSS). Poniżej przedstawiłem tylko kilka krótkich informacji o selektorach. W tabeli 12.2 wymienilem najczęściej stosowane wzorce selektorów CSS.

Tabela 12.2. Przykłady selektorów CSS

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

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>`. Zamiast samemu tworzyć selektor, można prawym przyciskiem myszy kliknąć element w przeglądarce WWW, a następnie z menu kontekstowego wybrać opcję *Zbadaj element*. Po wyświetleniu narzędzi programistycznych przeglądarki WWW należy prawym przyciskiem myszy kliknąć kod HTML elementu, a następnie z menu kontekstowego wybrać opcję *Copy/CSS Selector* w celu skopiowania do schowka ciągu tekstowego selektora. Później będzie można wkleić go w tworzonym kodzie źródłowym.

Metoda `select()` zwróci listę obiektów `Tag`, za pomocą których moduł `BeautifulSoup` przedstawia element HTML. Lista będzie zawierała po jednym obiekcie `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(), 'html.parser')
>>> elems = exampleSoup.select('#author')
>>> type(elems) # elems to lista obiektów Tag.
<class 'list'>
>>> len(elems)
1
>>> type(elems[0])
<class 'bs4.element.Tag'>
>>> str(elems[0]) # Obiekt Tag to ciąg tekstowy.
'<span id="author">Al Sweigart</span>'
>>> elems[0].getText()
```

```
'Al Sweigart'  
>>> 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="https://  
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()  
'Autor 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'), 'html.parser')
>>> 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 `<span>`, a następnie pierwszy dopasowany umieściła w zmiennej `spanElem`. Przekazanie funkcji `get()` nazwy atrybutu `'id'` powoduje zwrot jego wartości, czyli `'author'`.

Projekt — wyświetlenie wyników wyszukiwania

Gdy szukam informacji na dany temat za pomocą wyszukiwarki internetowej Google, nie przeglądam po kolei otrzymanych wyników wyszukiwania. Klikając łączą 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 i wyświetli wyniki wyszukiwania w repozytorium Python Package Index pod adresem <https://pypi.org/>. Tego rodzaju program można dostosować do wielu innych witryn internetowych, choć Google i DuckDuckGo często wykorzystują technologie utrudniające pobieranie danych ze stron wyników wyszukiwania wygenerowanych przez wymienione wyszukiwarki internetowej.

Poniżej wymienię w punktach sposób działania programu.

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

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

1. Odczyt argumentów wiersza poleceń z listy `sys.argv`.
2. Pobranie za pomocą modułu `requests` strony wyników wyszukiwania.

3. Wyszukanie łączy dla każdego wyniku wyszukiwania.
4. Wywołanie funkcji `webbrowser.open()` w celu przejścia do przeglądarki WWW.

Otwórz nowe okno edytora pliku i zapisz plik pod nazwą *searchpypi.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 repozytorium Python Package Index, możesz ustalić, że interesujący nas adres URL ma postać *https://pypi.org/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 otworzenia wspomnianych łączy na kartach przeglądarki WWW.

W pliku wprowadź przedstawiony poniżej kod.

```
#!/ python3
# searchpypi.py — Otwiera kilka wyników wyszukiwania w Python Package Index.

import requests, sys, webbrowser, bs4

print('Wyszukiwanie...')      # Komunikat wyświetlany podczas pobierania strony wyników
                             # wyszukiwania.
res = requests.get('https://pypi.org/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 wyrażenia *Beautiful Soup* przejdź do narzędzi programistycznych i zajrzyj do dowolnie wybranego elementu łączy na stronie. Tego rodzaju elementy wyglądają na niezwykle skomplikowane i mają postać podobną do przedstawionej `<a class="package-snippet" href="HYPERLINK" view-source:https://pypi.org/project/xml-parser/" />/project/xml-parser/">`.

Nie ma absolutnie żadnego znaczenia, że element wygląda na bardzo skomplikowany. Musimy po prostu odnaleźć wzorzec, który mają wszystkie łączy przedstawiające wyniki wyszukiwania.

Wprowadź w kodzie przedstawione poniżej zmiany.

```
#!/ python3
# searchpypi.py — Otwiera kilka wyników wyszukiwania w Python Package Index.
import requests, sys, webbrowser, bs4

--cięcie--

# Pobranie łączy z kilkoma pierwszymi wynikami wyszukiwania.
soup = bs4.BeautifulSoup(res.text, 'html.parser')
# Otworzenie karty przeglądarki WWW dla każdego wyniku wyszukiwania.
linkElems = soup.select('.package-snippet')
```

Gdy rozejrzysz się nieco w okolicach elementów `<a>`, zauważysz, że łączy w wynikach wyszukiwania mają atrybut `class="package-snippet"`. Po przejrzeniu pozostałej części kodu źródłowego HTML można dojść do wniosku, że klasa `package-snippet` jest używana jedynie dla łączy zawierających wyniki wyszukiwania. Nie musisz wiedzieć, czym jest klasa CSS o nazwie `package-snippet` 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 `'.package-snippet'` do odszukania wszystkich elementów `<a>` znajdujących się wewnątrz elementów o przypisanej klasie CSS `package-snippet`. Należy pamiętać, że w przypadku zmiany projektu witryny internetowej PyPI trzeba będzie uaktualnić program, aby do metody `soup.select()` był przekazywany odpowiedni selektor CSS. Pozostała część programu nie powinna wymagać modyfikacji.

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
# searchpypi.py — Otwiera kilka wyników wyszukiwania w Python Package Index.
import requests, sys, webbrowser, bs4
```

--cięcie--

```
# Otworzenie karty przeglądarki WWW dla każdego wyniku wyszukiwania.
linkElems = soup.select('.package-snippet')
numOpen = min(5, len(linkElems))
for i in range(numOpen):
    urlToOpen = 'https://pypi.org' + linkElems[i].get('href')
    print('Otwieranie', urlToOpen)
    webbrowser.open(urlToOpen)
```

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 `'.package-snippet'`. 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 `https://` → `pypi.org`, więc trzeba przeprowadzić konkatenację podanego fragmentu i ciągu tekstowego będącego wartością atrybutu `href`.

Teraz możemy natychmiast otworzyć pięć pierwszych wyników wyszukiwania w PyPI, na przykład wyrażenia *boring stuff*, przez wywołanie w wierszu poleceń `searchpypi boring stuff!` (W dodatku A znajdziesz informacje o tym, jak można łatwo uruchamiać programy Pythona w różnych systemach operacyjnych).

Pomysły na podobne programy

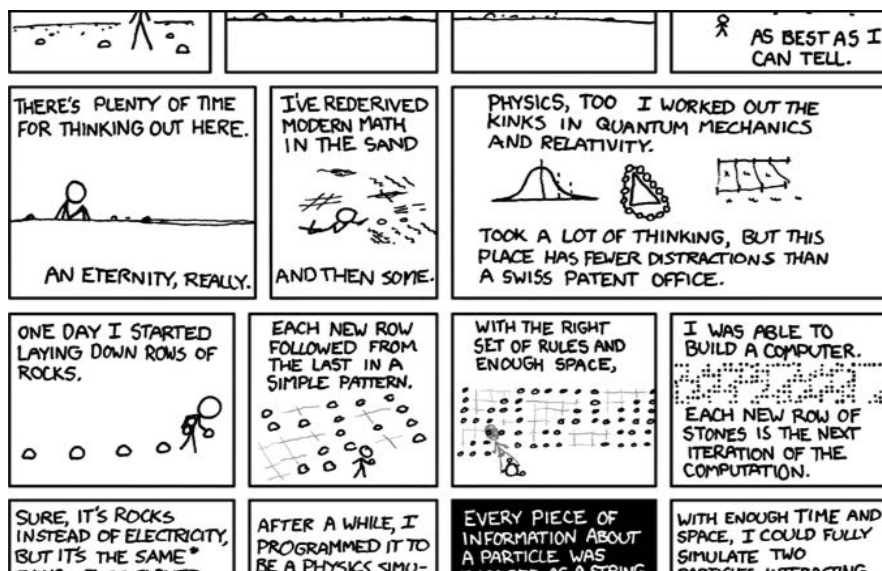
Zaletą kart w przeglądarce WWW jest możliwość łatwego otwierania łączy w nowych kartach, z którymi będzie można zapoznać się później. Program automatycznie otwierający jednocześnie kilka łączy może być wygodnym skrótem dla niektórych zadań. Oto one.

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



Rysunek 12.6. XKCD to witryna internetowa publikująca komiksy o różnej tematyce, na przykład romantyzm, sarkazm, matematyka, język i tak dalej

Poniżej wymienilem w punktach sposób działania programu.

1. Wczytanie strony głównej XKCD.
2. Zapisanie obrazu zawierającego komiks opublikowany na danej stronie.

3. Przejście na stronę wskazywaną przez łącze *Previous Comic*.
4. Powtarzanie operacji, dopóki nie zostanie pobrany najstarszy komiks.

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

1. Pobieranie stron za pomocą modułu `requests`.
2. Odszukanie za pomocą modułu `BeautifulSoup` adresu URL obrazu komiksu.
3. Pobranie i zapisanie na dysku twardym obrazu komiksu za pomocą metody `iter_content()`.
4. 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 przeglądarkę narzędzia dla programistów i przeanalizujesz elementy na stronie, będziesz mógł poczynić następujące ustalenia.

- Adres URL pliku obrazu komiksu jest zdefiniowany w atrybucie `href` elementu `<img>`.
- Element `<img>` 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 `https://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 = 'https://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 `'https://xkcd.com'`. Wartość tej zmiennej jest nieustannie uaktualniana (w pętli `while`) adresem URL łączą *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

Przechodzimy 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 = 'https://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, 'html.parser')

    # 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

--ciącie--

# Ustalenie adresu URL pliku obrazu komiksu.
comicElem = soup.select('#comic img')
if comicElem == []:
    print('Nie udało się odnaleźć pliku obrazu komiksu.')
else:
    comicUrl = 'https:' + 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 `<img>` pliku obrazu komiksu znajduje się wewnątrz elementu `<div>` zawierającego atrybut `id` o wartości `comic`. Dlatego też selektor `'#comic img'` pozwoli na wyodrębnienie właściwego elementu `<img>` 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 `<img>`. 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

--ciącie--

# 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 = 'https://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 `'https://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 https://xkcd.com...
Pobieranie obrazu https://imgs.xkcd.com/comics/phone_alarm.png...
Pobieranie strony https://xkcd.com/1358/...
Pobieranie obrazu https://imgs.xkcd.com/comics/nro.png...
Pobieranie strony https://xkcd.com/1357/...

```

```
Pobieranie obrazu https://imgs.xkcd.com/comics/free_speech.png...
Pobieranie strony https://xkcd.com/1356/...
Pobieranie obrazu https://imgs.xkcd.com/comics/orbital_mechanics.png...
Pobieranie strony https://xkcd.com/1355/...
Pobieranie obrazu https://imgs.xkcd.com/comics/airplane_message.png...
Pobieranie strony https://xkcd.com/1354/...
Pobieranie obrazu https://imgs.xkcd.com/comics/heartbleed_explanation.png...
--cięcie--
```

Ten projekt jest dobrym przykładem programu, który może automatycznie podążać za łączami 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.

Jeżeli chcesz prowadzić interakcję ze stroną internetową w sposób na przykład zależny od kodu JavaScript uaktualniającego stronę, wówczas zamiast modułu `requests` powinieneś skorzystać z modułu `selenium`. W sporej części witryn internetowych typu e-commerce, na przykład Amazon, są używane systemy oprogramowania mające na celu rozpoznanie ruchu sieciowego, który może być wynikiem działania skryptów pobierających informacje z witryny internetowej lub zakładających wiele bezpłatnych kont. Takie witryny internetowe mogą odmawiać udostępniania stron po chwili działania skryptu i tym samym uniemożliwić jego funkcjonowanie zgodnie z oczekiwaniami. Moduł `selenium` zapewnia znacznie większe prawdopodobieństwo działania takich skryptów przez znacznie dłuższy czas niż w przypadku użycia modułu `requests`.

Wskazaniem dla witryny internetowej, że używany jest skrypt, będzie tak zwany ciąg tekstowy *user-agent* identyfikujący przeglądarkę WWW i umieszczany we wszystkich żądaniach HTTP. Przykładowo ciąg tekstowy *user-agent* dla modułu `requests` ma postać podobną do `'python-requests/2.21.0'`. Jeżeli odwiedzisz witrynę, taką jak <https://www.whatsmyua.info/>, zobaczysz używany ciąg tekstowy *user-agent*. Żądania wykonywane przez moduł `selenium` przypominają te, które są generowane, gdy człowiek samodzielnie przegląda zawartość strony, ponieważ ciąg tekstowy *user-agent* jest taki sam jak w zwykłej przeglądarce WWW (na przykład `'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:65.0) Gecko/20100101 Firefox/65.0'`). Jednak zastosowanie mają te same wzorce: kontrolowana przez `selenium` przeglądarka WWW będzie pobierała obrazy, reklamy, cookies i nierepektujące prywatności użytkownika programy szpiegujące, podobnie jak ma to miejsce podczas zwykłego używania przeglądarki WWW. Użycie modułu `selenium` może być wykryte przez witryny internetowe, a większość witryn internetowych typu e-commerce blokuje przeglądarki WWW kontrolowane przez `selenium`, aby uniemożliwiać pobieranie informacji z ich stron.

Uruchomienie przeglądarki WWW kontrolowanej przez moduł `selenium`

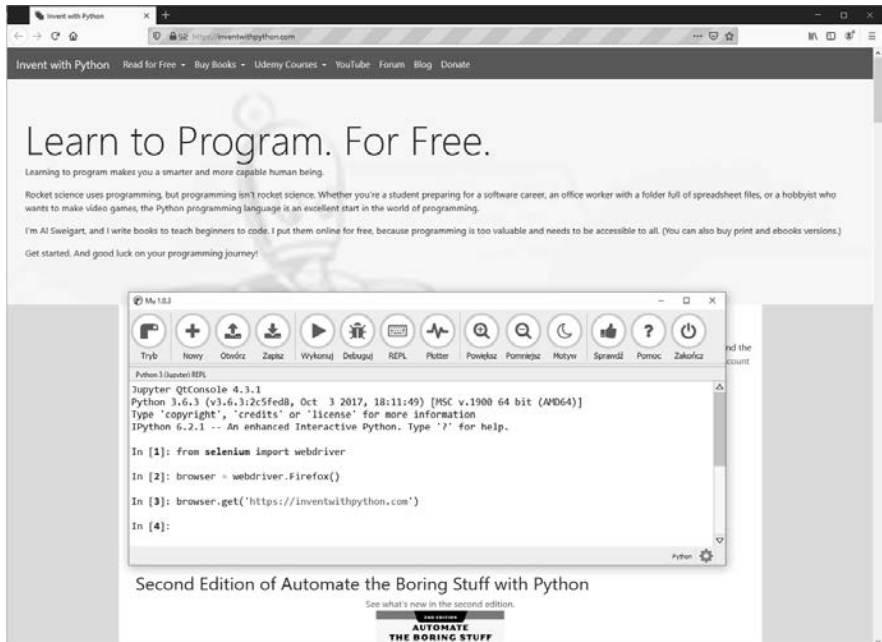
W omawianym tutaj przykładzie wykorzystamy przeglądarkę WWW o nazwie Firefox. Tę właśnie przeglądarkę będziemy kontrolować. Jeżeli jeszcze nie zainstalowałeś przeglądarki Firefox, możesz ją pobrać z witryny <https://getfirefox.com/>. Instalacja wymaga wydania polecenia `pip install --user selenium` z poziomu powłoki. Więcej informacji na ten temat znajdziesz w dodatku A.

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`. W powłoce interaktywnej wprowadź przedstawione poniżej polecenia.

```
>>> from selenium import webdriver
>>> browser = webdriver.Firefox()
```

```
>>> type(browser)
<class 'selenium.webdriver.firefox.webdriver.WebDriver'>
>>> browser.get('https://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('https://inventwithpython.com')` powoduje przekierowanie przeglądarki WWW do witryny *https://inventwithpython.com/*. W tym momencie okno przeglądarki Firefox powinno wyglądać tak, jak pokazałem na rysunku 12.7.



Rysunek 12.7. Po wywołaniach `webdriver.Firefox()` i `get()` w edytorze Mu na ekranie pojawia się okno wskazanej przeglądarki WWW (tutaj to Firefox)

Jeżeli nastąpi wygenerowanie komunikatu błędu o treści `geckodriver' executable needs to be in PATH`, oznacza to konieczność ręcznego pobrania sterownika dla przeglądarki WWW Firefox, zanim będzie mogła być kontrolowana przez `selenium`. Istnieje możliwość kontrolowania innych niż Firefox przeglądarek WWW, o ile zostaną zainstalowane dla nich odpowiednie sterowniki.

W przypadku przeglądarki WWW Firefox przejdź na stronę *https://github.com/mozilla/geckodriver/releases* i pobierz sterownik `geckodriver` dla używanego systemu operacyjnego. („Gecko” to nazwa silnika przeglądarki WWW użytego w programie Firefox). Przykładowo dla Windows pobierz plik *geckodriver-v0.24.0-win64.zip*, natomiast dla systemu macOS pobierz plik *geckodriver-v0.24.0-macos.tar.gz*. Jeżeli pojawią się nowsze wersje, będą miały nieco inne nazwy niż tutaj podane.

Pobrane archiwum ZIP zawiera plik *geckodriver.exe* (Windows) lub *geckodriver* (macOS i Linux), który należy umieścić w katalogu wymienionym w zmiennej systemowej PATH. Więcej informacji na temat zmiennej systemowej PATH znajdziesz w dodatku B, a także na stronie <https://stackoverflow.com/questions/40208051/selenium-using-python-geckodriver-executable-needs-to-be-in-path>.

W przypadku przeglądarki WWW Chrome należy przejść na stronę <https://sites.google.com/a/chromium.org/chromedriver/downloads> i pobrać archiwum ZIP dla używanego systemu operacyjnego. W archiwum znajdziesz plik *chromedriver.exe* (Windows) lub *chromedriver* (macOS i Linux), który należy umieścić w katalogu wymienionym w zmiennej systemowej PATH.

Dla innych ważnych przeglądarek WWW również istnieją sterowniki. Najczęściej można je dość łatwo znaleźć po wpisaniu w ulubionej wyszukiwarce internetowej wyrażenia *nazwa_przeglądarki_www_webdriver*.

Jeżeli mimo wszystko nadal będziesz mieć problemy z otwieraniem nowych okien przeglądarki WWW za pomocą modułu selenium, problem może być związany z tym, że bieżąca wersja przeglądarki WWW jest niezgodna z modulem selenium. Rozwiązaniem jest wówczas zainstalowanie starszej wersji przeglądarki WWW lub, co jest jeszcze łatwiejsze, starszej wersji modułu selenium. Listę numerów wersji modułu selenium znajdziesz na stronie <https://pypi.org/project/selenium/#history>. Niestety zgodność między wersjami modułu selenium i przeglądarek WWW bywa czasami problematyczna i być może będziesz musiał poszukać pewnych rozwiązań w internecie. Dodatek A zawiera więcej informacji na temat używania menedżera pakietów pip do instalowania określonych wersji pakietu, na przykład selenium. (Przykładowo być może będziesz musiał wydać polecenie typu `pip install --user -U selenium==3.14.1`).

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 12.3 wymieniałem 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 `NoSuchElementException`. 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 12.4.

Otwórz nową kartę edytora pliku i wpisz w nim poniższy fragment kodu.

Tabela 12.3. 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 12.4. 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.

```

from selenium import webdriver
browser = webdriver.Firefox()
browser.get('https://inventwithpython.com')
try:
    elem = browser.find_element_by_class_name('cover-thumb')
    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 'cover-thumb'. Jeżeli tego rodzaju element istnieje, wyświetlamy jego nazwę za pomocą atrybutu tag_name. Natomiast w przypadku niezalezienia elementu wyświetlony będzie zupełnie inny komunikat.

Program wygeneruje następujące dane wyjściowe.

Znaleziono element wraz z taką nazwą klasy!

Udało nam się znaleźć element o nazwie klasy 'cover-thumb' 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('https://inventwithpython.com')
>>> linkElem = browser.find_element_by_link_text('Read Online for Free')
>>> type(linkElem)
<class 'selenium.webdriver.remote.webelement.FirefoxWebElement'>
>>> linkElem.click()    # Podążanie za łączem zatytułowanym "Read Online for Free".
```

W powyższym fragmencie kodu przeglądarka Firefox przechodzi do witryny <https://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 podąża 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('https://login.metafilter.com')
```

```
>>> userElem = browser.find_element_by_id('user_name')
>>> userElem.send_keys('prawdziwa_nazwa_uzytkownika')

>>> passwordElem = browser.find_element_by_id('user_pass')
>>> passwordElem.send_keys('prawdziwe_haslo')
>>> passwordElem.submit()
```

O ile serwis MetaFiler 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 *Wyślij* formularza zawierającego ten element. (Równie dobrze można użyć wywołania `emailElem.submit()`, a omawiany kod nadal będzie wykonywał to samo zadanie).

OSTRZEŻENIE *Unikaj umieszczania haseł w kodzie źródłowym. Bardzo łatwo można przypadkowo ujawnić innym osobom hasła pozostawione na dysku w postaci niezasyfrowanej. Jeżeli to możliwe, program powinien prosić użytkownika o wpisanie hasła za pomocą klawiatury, używając do tego omówionej w rozdziale 8. funkcji `pyinputplus.input` ➔ `Password()`.*

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 12.5 wymienilem najczęściej używane zmienne `Keys`.

Tabela 12.5. 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 powłoce 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('https://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 <https://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 umożliwia 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 otwarcie 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 korzystać 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.

PROGRAM PARTNERSKI

— GRUPY HELION —



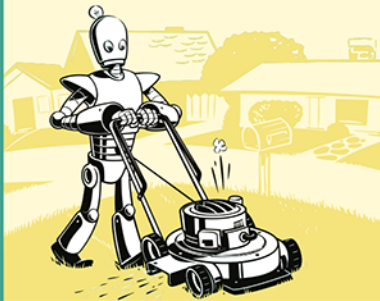
1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

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

GRUPA
Helion 



NUDNE I MĘCZĄCE? ZLEĆ TO PYTHONOWI!

Komputer jest wszechstronnym narzędziem, które szybko i dokładnie wykona wiele pracochłonnych zadań. Wystarczy przekazać mu instrukcje w zrozumiałym dla niego języku. Takim jak Python, który jest łatwy do nauczenia się, pozwala pisać wyrazisty, zwięzły kod i stwarza imponujące możliwości dzięki niezliczonej liczbie rozszerzeń i bibliotek. Tymczasem wciąż wiele osób wykonuje żmudne i męczące zadania ręcznie: pracując z arkuszem kalkulacyjnym, plikami albo pocztą e-mail. Czas to zmienić — pracę, nad którą ślęczysz kilka dni, komputer wykona dokładnie w ciągu najwyżej kilku sekund!

Ta książka jest drugim wydaniem nietypowego podręcznika programowania w Pythonie. Dzięki niej nie zostaniesz mistrzem świata w kodowaniu, za to nauczysz się tworzyć programy, które oszczędzą Ci mnóstwo czasu i wysiłku. Nawet jeśli nigdy nie programowałeś, błyskawicznie opanujesz podstawy i zapoznasz się z obszerną biblioteką Pythona przeznaczoną do automatyzacji takich zadań jak pobieranie danych z witryn internetowych, odczytywanie dokumentów oraz operacje wymagające klikania myszą i wpisywania tekstu. To wydanie zawiera nowy rozdział poświęcony weryfikacji danych wejściowych, a także samouczki dotyczące automatyzacji pracy z arkuszami Google i pocztą Gmail oraz podpowiedzi związane z automatycznym uaktualnianiem plików CVS.

Sprawdź, jak zautomatyzować:

- wyszukiwanie ciągu znaków w pliku lub wielu plikach
- tworzenie, uaktualnianie, przenoszenie i zmiany nazw plików oraz katalogów
- wyszukiwanie treści w sieci WWW oraz ich pobieranie
- podział, łączenie, nakładanie znaku wodnego i szyfrowanie dokumentów PDF
- wysyłanie powiadomień za pomocą wiadomości e-mail oraz SMS
- wypełnianie formularzy internetowych

Al Sweigart — programista i autor książek informatycznych. Pasjonat Pythona, opracował kilka popularnych modułów dla tego języka. Tworzone przez siebie oprogramowanie często udostępnia na zasadach open source. Jego ulubieńcem jest pięciokilogramowy kot.

Helion 	KOD KORZYŚCI Śięgnij po więcej! ▶ 
 helion.pl	ISBN 978-83-289-2386-7
 HELION S.A. ul. Kościuszki 1c 44-100 Gliwice tel.: 32 230 98 63 helion@helion.pl	 9 788328 923867
Cena: 119,00 zł	

