18

Kontrolowanie klawiatury i myszy za pomocą automatyzacji GUI

Znajomość różnych modułów Pythona przeznaczonych do edycji arkuszy kalkulacyjnych, pobierania plików i uruchamiania innych programów jest użyteczna. Jednak czasami po prostu nie ma żadnego modułu przeznaczonego dla aplikacji, z którą chcesz pracować. Wtedy narzędziami przeznaczonymi do automatyzacji zadań w komputerze są utworzone przez Ciebie programy, które przejmują bezpośrednią kontrolę nad klawiaturą i myszą. Tego rodzaju programy mogą kontrolować działanie innych aplikacji przez wysyłanie im sygnałów wirtualnych naciśnięć klawiszy i przycisków myszy, tak jakbyś sam siedział przed komputerem i pracował z daną aplikacją. Wymieniona technika jest znana jako *automatyzacja graficznego interfejsu użytkownika* lub krócej *automatyzacja GUI*. W automatyzacji GUI tworzone programy mogą zrobić to wszystko, co jest możliwe dla człowieka siedzącego przed komputerem. Program nie może jedynie wylać kawy na klawiaturę.

Automatyzację GUI można potraktować jak zaprogramowane ramię robota. Możesz je zaprogramować w taki sposób, aby ramię naciskało określone klawisze i poruszało myszą za Ciebie. Taka technika okazuje się szczególnie użyteczna w zadaniach wymagających wielu bezmyślnych kliknięć lub wypełnienia formularza.

Moduł o nazwie `pyautogui` zawiera funkcje przeznaczone do symulacji ruchu myszą, kliknięć przyciskami myszy oraz poruszania kółkiem myszy. W rozdziale przedstawię zaledwie ułamek funkcji oferowanych przez `pyautogui`. Pełną dokumentację tego modułu znajdziesz na stronie *http://pyautogui.readthedocs.org/*.

Instalacja modułu pyautogui

Moduł `pyautogui` może wysyłać sygnały wirtualnych naciśnięć klawiszy i kliknięć myszą w systemach Windows, macOS i Linux. W zależności od używanego systemu operacyjnego, może zaistnieć konieczność instalacji pewnych modułów dodatkowych (nazywanych *zależnościami*) przed instalacją `pyautogui`.

- W systemie Windows nie ma konieczności instalacji żadnych dodatkowych modułów.
- W systemie macOS należy wydać polecenia **`sudo pip3 install pyobjc-framework-Quartz`**, **`sudo pip3 install pyobjc-core`**, a następnie **`sudo pip3 install pyobjc`**.
- W systemie Linux należy wydać polecenia **`sudo pip3 install python3-xlib`**, **`sudo apt-get install scrot`**, **`sudo apt-get install python3-tk`** i **`sudo apt-get install python3-dev`**. (Scrot to używany przez `pyautogui` program do wykonywania zrzutów ekranu).

Po zainstalowaniu wymienionych zależności należy wydać polecenie **`pip install pyautogui`** (w systemach macOS i Linux zamiast `pip` użyj `pip3`) w celu instalacji modułu `pyautogui`.

W dodatku A znajdziesz więcej informacji szczegółowych na temat instalacji modułów opracowanych przez firmy trzecie. Aby sprawdzić, czy moduł `pyautogui` został prawidłowo zainstalowany, w powłoce interaktywnej Pythona wydaj polecenie `import pyautogui` i zobacz, czy nastąpiło wygenerowanie jakichkolwiek komunikatów błędów.

Pozostajemy na kursie

Zanim przejdziesz do tematu automatyzacji GUI, powinieneś wiedzieć, jak poradzić sobie z problemami, które mogą się pojawić. Python potrafi przesuwać kursor myszy i wirtualnie naciskać klawisze z ogromną szybkością. Szybkość może okazać się za duża dla innych programów, które po prostu nie będą nadążały. Jeśli ponadto wystąpi jakikolwiek problem, a Twój program nadal będzie poruszał

myszą, wówczas trudno ustalić, co tak dokładnie ten program robi oraz jak usunąć problem. Podobnie jak zaczarowane miotły w bajce Disneya zatytułowanej *Uczeń czarnoksiężnika* nieustannie zamiatają, a następnie przepełniają kosz na śmieci, który ma uprzątnąć myszka Mickey — Twój program może wymknąć się spod kontroli nawet wtedy, kiedy ściśle stosuje się do Twoich poleceń. Zatrzymanie takiego programu może być trudne, zwłaszcza gdy kursor myszy porusza się sam. W efekcie niemożliwe może być kliknięcie przycisku zamykającego okno środowiska IDLE. Na szczęście istnieje kilka sposobów pozwalających uniknąć kłopotów lub poradzić sobie z ewentualnymi problemami związanymi z automatyzacją GUI.

Zamknięcie wszystkiego przez wylogowanie się

Prawdopodobnie najprostszym sposobem na zatrzymanie działania programu automatyzacji GUI, który wymknął się spod kontroli, jest wylogowanie się użytkownika, co powinno zakończyć wszystkie uruchomione przez niego procesy. W systemach Windows i Linux wylogowanie następuje po naciśnięciu klawiszy *Ctrl+Alt+Del*. Natomiast w systemie macOS naciśnij klawisze *Command+Shift+Option+Q*. Wprawdzie wylogowanie spowoduje utratę niezapisanych plików, ale przynajmniej nie trzeba będzie czekać na wykonanie pełnego ponownego uruchomienia komputera.

Pauzy i funkcja bezpiecznej awarii

W skrypcie można zdefiniować przerwy po każdym wywołaniu funkcji. W ten sposób masz chwilę czasu na odzyskanie kontroli nad myszą i klawiaturą w przypadku wystąpienia problemów. W tym celu zmiennej `pyautogui.PAUSE` przypisz liczbę sekund określających długość pauzy. Przykładowo polecenie `pyautogui.` ↪`PAUSE = 1.5` powoduje wstrzymanie działania skryptu na półtorej sekundy po wykonaniu każdego wywołania funkcji modułu `pyautogui`. Wywołania funkcji innych modułów będą wykonywane bez żadnych przerw.

Moduł `pyautogui` oferuje funkcję pozwalającą na bezpieczną awarię programu. Przesunięcie kursora myszy w lewy górny róg ekranu spowoduje zgłoszenie przez `pyautogui` wyjątku `pyautogui.FailSafeException`. Twój program może obsłużyć ten wyjątek za pomocą poleceń `try` i `except`, alternatywą jest pozwolenie programowi na ulegnięcie awarii. Niezależnie od przyjętego podejścia funkcja spowoduje zatrzymanie działania programu, o ile jak najszybciej potrafisz przesuniesz kursor myszy w lewy górny róg ekranu. Istnieje możliwość wyłączenia omówionej funkcji, co wymaga użycia polecenia `pyautogui.FAILSAFE = False`. W powłoce interaktywnej wprowadź przedstawione poniżej polecenia.

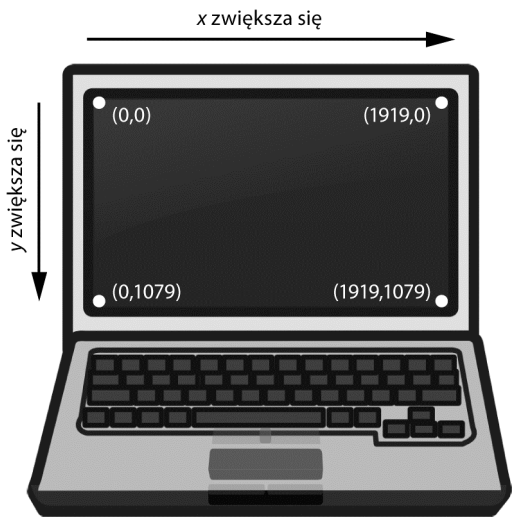
```
>>> import pyautogui
>>> pyautogui.PAUSE = 1
>>> pyautogui.FAILSAFE = True
```

W powyższym fragmencie kodu importujemy moduł `pyautogui` i za pomocą polecenia `pyautogui.PAUSE = 1` definiujemy jednosekundową przerwę po każdym wywołaniu funkcji. Z kolei polecenie `pyautogui.FAILSAFE = True` powoduje włączenie możliwości wystąpienia bezpiecznej awarii programu.

Kontrola poruszania myszą

W tym podrozdziale dowiesz się, jak za pomocą modułu `pyautogui` poruszać myszą i śledzić jej położenie na ekranie. Jednak najpierw musisz poznać sposób, w jaki moduł działa z układem współrzędnych.

Oferowane przez moduł `pyautogui` funkcje przeznaczone do kontrolowania myszy korzystają ze współrzędnych X i Y. Na rysunku 18.1 pokazałem układ współrzędnych na ekranie monitora. Ten układ współrzędnych jest podobny do używanego przez obrazy, który poznałeś w rozdziale 17. Punkt *początku układu współrzędnych*, gdzie współrzędne X i Y mają wartość zero, znajduje się w lewym górnym rogu ekranu. Współrzędne X zwiększają się w prawą stronę układu, natomiast współrzędne Y w dół układu. Wszystkie współrzędne są dodatnimi liczbami całkowitymi, nie istnieje ujemna współrzędna.



Rysunek 18.1. Współrzędne na ekranie komputera o rozdzielczości 1920×1080 pikseli

Rozdzielczość ekranu informuje o wyrażonej w pikselach szerokości i wysokości ekranu. Jeżeli rozdzielczość ekranu wynosi 1920×1080, punkt w jego lewym górnym rogu ma współrzędne (0, 0), natomiast punkt w prawym dolny rogu ma współrzędne (1919, 1079).

Funkcja `pyautogui.size()` zwraca składającą się z dwóch elementów krotkę przedstawiającą wyrażoną w pikselach szerokość i wysokość ekranu. W powłoce interaktywnej wprowadź przedstawione poniżej polecenia.

```
>>> import pyautogui
>>> pyautogui.size()
(1920, 1080)
>>> width, height = pyautogui.size()
```

W omawianym przykładzie funkcja `pyautogui.size()` zwraca `(1920, 1080)` w komputerze wyposażonym w ekran o rozdzielczości 1920×1080. W zależności od rozdzielczości ekranu ta wartość zwrotna może być inna. Otrzymane na skutek działania funkcji `pyautogui.size()` liczby określające szerokość i wysokość można przechowywać w zmiennych, takich jak `width` i `height`, aby tym samym zwiększyć czytelność kodu źródłowego programu.

Poruszanie kursorem myszy

Skoro poznałeś układ współrzędnych ekranu, możemy zająć się przesuwaniem kursora myszy. Funkcja `pyautogui.moveTo()` natychmiast przenosi kursor do wskazanego położenia na ekranie. Wartości w postaci liczb całkowitych dla współrzędnych X i Y definiują — odpowiednio — pierwszy i drugi argument tej funkcji. Opcjonalna liczba całkowita lub zmiennoprzecinkowa w argumencie w postaci słowa kluczowego `duration` określa liczbę sekund, które powinna zająć operacja przesunięcia kursora myszy do wskazanego położenia. Jeżeli pominiesz ten argument, jego wartością domyślną jest `0`, co oznacza natychmiastowe przejście. (W module `pyautogui` wszystkie argumenty w postaci słowa kluczowego `duration` są opcjonalne). W powłoce interaktywnej wprowadź przedstawione poniżej polecenia.

```
>>> import pyautogui
>>> for i in range(10):
      pyautogui.moveTo(100, 100, duration=0.25)
      pyautogui.moveTo(200, 100, duration=0.25)
      pyautogui.moveTo(200, 200, duration=0.25)
      pyautogui.moveTo(100, 200, duration=0.25)
```

W powyższym fragmencie kodu przesuwamy kursor myszy w kierunku zgodnym z ruchem wskazówek zegara między czterema punktami tworzącymi na ekranie kwadrat o podanych współrzędnych. Operacja jest powtórzona dziesięciokrotnie. Każdy ruch zabiera 1/4 sekundy, zgodnie z użytym argumentem w postaci słowa kluczowego `duration=0.25`. Jeżeli nie przekażemy trzeciego argumentu żadnemu z wywołań `pyautogui.moveTo()` w omawianym fragmencie kodu, wówczas kursor będzie natychmiast teleportowany między poszczególnymi punktami.

Funkcja `pyautogui.moveRel()` przenosi kursor do położenia ustalanego *względem* bieżącego. W pokazanym poniżej fragmencie kodu przenosimy kursor według tego samego wzorca kwadratu z wyjątkiem faktu, że kwadrat rozpoczyna się w miejscu, w którym znajdował się kursor myszy w chwili uruchomienia tego fragmentu kodu.

```
>>> import pyautogui
>>> for i in range(10):
      pyautogui.moveRel(100, 0, duration=0.25)
      pyautogui.moveRel(0, 100, duration=0.25)
      pyautogui.moveRel(-100, 0, duration=0.25)
      pyautogui.moveRel(0, -100, duration=0.25)
```

Funkcja `pyautogui.moveRel()` również pobiera trzy argumenty: liczbę pikseli, o które kursor ma być przesunięty w poziomie w kierunku prawej strony, liczbę pikseli, o które kursor ma być przesunięty w pionie do dołu oraz (opcjonalnie) liczbę sekund, które ma trwać ta operacja. Ujemna liczba całkowita dla pierwszego lub drugiego argumenty powoduje ruch kursora myszy w — odpowiednio — lewą stronę i w górę.

Pobranie informacji o położeniu kursora myszy

Istnieje możliwość ustalenia bieżącego położenia kursora myszy za pomocą wywołania funkcji `pyautogui.position()`. Wartością zwrotną tej funkcji jest krotka współrzędnych X i Y kursora myszy wskazujących punkt, w którym się znajdował w chwili wywołania tej funkcji. W powłoce interaktywnej wprowadź przedstawione poniżej polecenia, przesuń kursor myszy po każdym wywołaniu funkcji.

```
>>> pyautogui.position()
(311, 622)
>>> pyautogui.position()
(377, 481)
>>> pyautogui.position()
(1536, 637)
```

Oczywiście otrzymane wyniki powyższych wywołań funkcji `pyautogui.posi`
`↪tion()` będą różne, w zależności od miejsca, w którym znajduje się kursor myszy.

Projekt — gdzie teraz jest kursor myszy?

Możliwość ustalenia bieżącego położenia kursora myszy jest bardzo ważnym etapem podczas konfiguracji skryptów automatyzacji GUI. Jednak praktycznie niemożliwe jest podanie z dokładnością do piksela położenia kursora myszy, gdy tylko patrzymy na ekran. Pomocne byłoby opracowanie programu nieustannie wyświetlającego współrzędne X i Y bieżącego położenia kursora myszy.

Poniżej wymieniłem w punktach sposób działania programu na wysokim poziomie.

- Wyświetlenie współrzędnych X i Y bieżącego położenia kursora myszy.
- Uaktualnienie tych współrzędnych, gdy kursor myszy porusza się na ekranie.

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

- Wywołanie funkcji `pyautogui.position()` w celu pobrania bieżących współrzędnych.
- Usunięcie poprzednio wyświetlonych współrzędnych przez symulację użycia znaku Backspace (`\b`) względem znaków wyświetlonych na ekranie.
- Obsłużenie wyjątku `KeyboardInterrupt`, aby użytkownik mógł zakończyć działanie programu przez naciśnięcie klawiszy *Ctrl+C*.

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

Etap 1. Import modułu

Na początku programu umieść przedstawiony poniżej fragment kodu.

```
#! python3
# mouseNow.py — Wyświetla bieżące położenie kursora myszy.
import pyautogui
print('Naciśnij klawisze Ctrl+C, aby zakończyć działanie programu.')
#TODO: Pobranie i wyświetlenie współrzędnych kursora myszy.
```

Na początku omawianego programu importujemy moduł `pyautogui` i wyświetlamy użytkownikowi komunikat z przypomnieniem o możliwości zakończenia działania programu przez naciśnięcie klawiszy *Ctrl+C*.

Etap 2. Konfiguracja kodu zamykającego program oraz pętli działającej w nieskończoność

Do nieustannego wyświetlania dostarczonych przez funkcję `pyautogui.position()` współrzędnych bieżącego położenia kursora myszy możemy użyć pętli działającej w nieskończoność. Dla kodu kończącego działanie programu konieczne jest przechwycenie wyjątku `KeyboardInterrupt`, który jest zgłaszany po naciśnięciu przez użytkownika klawiszy *Ctrl+C*. Jeżeli nie zapewnisz obsługi tego wyjątku, na ekranie zostanie wyświetlony niezrozumiały dla użytkownika komunikat dotyczący ostatnich wywołań wraz z komunikatem błędu. W programie umieść przedstawiony poniżej nowy fragment kodu.

```
#! python3
# mouseNow.py — Wyświetla bieżące położenie kursora myszy.
import pyautogui
print('Naciśnij klawisze Ctrl+C, aby zakończyć działanie programu.')
```

```
try:
    while True:
        # TODO: Pobranie i wyświetlenie współrzędnych kursora myszy.
except KeyboardInterrupt: ❶
    print('\nGotowe!') ❷
```

W celu zapewnienia obsługi wyjątku `KeyboardInterrupt` działającą w nieskończoność pętlę `while` umieść w poleceniu `try`. Kiedy użytkownik naciśnie klawisze *Ctrl+C*, program przejdzie do klauzuli `except` ❶, a następnie w nowym wierszu zostanie wyświetlony komunikat `Gotowe!` ❷.

Etap 3. Pobranie i wyświetlenie bieżących współrzędnych kursora myszy

Kod zdefiniowany wewnątrz pętli `while` powinien pobrać bieżące współrzędne kursora myszy, sformatować je elegancko, a następnie wyświetlić. Wewnątrz pętli `while` programu umieść przedstawiony poniżej nowy fragment kodu.

```
#! python3
# mouseNow.py — Wyświetla bieżące położenie kursora myszy.
import pyautogui
print('Naciśnij klawisze Ctrl+C, aby zakończyć działanie programu.')
--cięcie--
        # Pobranie i wyświetlenie współrzędnych kursora myszy.
        x, y = pyautogui.position()
        positionStr = 'X: ' + str(x).rjust(4) + ' Y: ' + str(y).rjust(4)
--cięcie--
```

Przy użyciu sztuczki wielokrotnego przypisania zmienne `x` i `y` otrzymują wartości dwóch liczb całkowitych znajdujących się w krotce zwróconej przez wywołanie funkcji `pyautogui.position()`. Przekazanie zmiennych `x` i `y` do funkcji `str()` powoduje, że otrzymujemy współrzędne liczbowe w postaci ciągów tekstowych. Metoda ciągu tekstowego o nazwie `rjust()` powoduje dosunięcie wartości do prawej strony, aby zabierała taką samą ilość miejsca, niezależnie od tego, ile cyfr (jedną, dwie, trzy lub cztery) zawiera liczba wyrażająca współrzędną. Konkatenacja wyrównanego do prawej strony ciągu tekstowego współrzędnej wraz z etykietą `'X: '` i `'Y: '` daje nam elegancko sformatowany ciąg tekstowy, który będzie przechowywany w zmiennej `positionStr`.

Na końcu programu umieść przedstawiony poniżej nowy fragment kodu.

```
#! python3
# mouseNow.py — Wyświetla bieżące położenie kursora myszy.
--cięcie--
        print(positionStr, end='')
        print('\b' * len(positionStr), end='', flush=True) ❶
```

Polecenia te odpowiadają za rzeczywiste wyświetlenie wartości zmiennej `posi ↪tionStr` na ekranie. Argument w postaci słowa kluczowego `end=''` w wywołaniu funkcji `print()` uniemożliwia dodanie domyślnego znaku nowego wiersza na końcu wyświetlanego. Istnieje możliwość usunięcia tekstu już wyświetlonego na ekranie, ale dotyczy to jedynie ostatniego wiersza tekstu. Po użyciu znaku nowego wiersza nie można usunąć niczego, co zostało wyświetlone przed tym znakiem.

W celu usunięcia tekstu należy użyć znaku sterującego Backspace (`\b`). To znak specjalny, którego działanie polega na usunięciu znaku znajdującego się na końcu bieżącego wiersza wyświetlanego na ekranie. W poleceniu oznaczonym ❶ używamy replikacji ciągu tekstowego w celu wygenerowania ciągu tekstowego wraz z liczbą znaków `\b` odpowiadającą długości ciągu tekstowego przechowywanego w zmiennej `positionStr`. W ten sposób uzyskujemy efekt usunięcia ostatnio wyświetlonej wartości zmiennej `positionStr`.

Z powodów technicznych, których wyjaśnienie wykracza poza zakres tematyczny tej książki, zawsze powinieneś przekazywać argument `flush=True` w wywołaniu `print()` używającym znaku sterującego `\b`. W przeciwnym razie tekst na ekranie może nie zostać uaktualniony w oczekiwany sposób.

Ponieważ działanie pętli `while` odbywa się bardzo szybko, użytkownik w rzeczywistości nie zauważy operacji usunięcia i ponownego wyświetlenia całego ciągu tekstowego na ekranie. Jeśli na przykład współrzędna X pokazuje wartość 563 i kursor myszy przesunie się o jeden piksel w prawą stronę, wówczas będzie można odnieść wrażenie, że nastąpiła jedynie zmiana cyfry 3 na cyfrę 4 w liczbie 563.

Po uruchomieniu programu zostaną wyświetlone jedynie dwa wiersze tekstu, które będą wyglądały podobnie do pokazanych poniżej.

```
Naciśnij klawisze Ctrl+C, aby zakończyć działanie programu.
X:  290 Y:  424
```

W pierwszym wierszu mamy komunikat informujący o możliwości zakończenia działania programu przez naciśnięcie klawiszy *Ctrl+C*. Natomiast w drugim wierszu są wyświetlane bieżące współrzędne kursora myszy, które zmieniają się, gdy kursor myszy porusza się po ekranie. Za pomocą tego programu będziesz w stanie ustalić współrzędne kursora myszy niezbędne do podania w skryptach automatyzacji GUI.

Kontrola działania myszy

Skoro już wiesz, jak poruszać kursorem myszy po ekranie i odczytywać jego bieżące współrzędne, jesteś gotowy do rozpoczęcia klikania, przeciągania i przewijania myszą.

Kliknięcie myszą

Aby wysłać sygnał symulujący wirtualne kliknięcie myszą, należy wywołać metodę o nazwie `pyautogui.click()`. Domyślnie symulowane jest kliknięcie lewym przyciskiem myszy i odbywa się w bieżącym położeniu kursora myszy. Istnieje możliwość przekazania współrzędnych X i Y jako opcjonalnego pierwszego i drugiego argumentu metody, jeśli chcesz kliknąć inny punkt niż wskazywany przez bieżące położenie kursora myszy.

Jeżeli chcesz użyć innego przycisku myszy, w wywołaniu metody `pyautogui.click()` musisz umieścić argument w postaci słowa kluczowego `button`, który może przyjąć jedną z następujących wartości: `'left'`, `'middle'` lub `'right'`. Przykładowo wywołanie `pyautogui.click(100, 150, button='left')` spowoduje symulację kliknięcia lewym przyciskiem myszy w punkcie o współrzędnych (100, 150). Natomiast wywołanie `pyautogui.click(200, 250, button='right')` spowoduje symulację kliknięcia prawym przyciskiem myszy w punkcie o współrzędnych (200, 250).

W powłoce interaktywnej wprowadź przedstawione poniżej polecenia.

```
>>> import pyautogui
>>> pyautogui.click(10, 5)
```

Powinieneś zobaczyć przesunięcie kursora myszy niemal do lewego górnego rogu ekranu i jednokrotne kliknięcie lewym przyciskiem myszy. Pełne „kliknięcie" jest definiowane jako naciśnięcie przycisku myszy, a następnie jego zwolnienie bez przesunięcia kursora myszy. Istnieje również możliwość symulacji kliknięcia za pomocą funkcji `pyautogui.mouseDown()`, która jedynie symuluje naciśnięcie przycisku myszy, i `pyautogui.mouseUp()`, która jedynie symuluje zwolnienie przycisku myszy. Wymienione funkcje mają takie same argumenty jak `click()`. W rzeczywistości funkcja `click()` to po prostu wygodne opakowanie dla tych dwóch funkcji.

Kolejnym udogodnieniem jest funkcja `pyautogui.doubleClick()` symulująca dwukrotne kliknięcie lewym przyciskiem myszy. Z kolei funkcje `pyautogui.rightClick()` i `pyautogui.middleClick()` symulują kliknięcie — odpowiednio — prawym i środkowym przyciskiem myszy.

Przeciąganie myszą

Operacja przeciągania myszą oznacza przytrzymanie jednego z przycisków myszy podczas przesuwania kursora myszy. Możesz na przykład przeciągnąć pliki między katalogami przez przeciągnięcie ich ikon lub też możesz przeciągać w aplikacji kalendarza dane dotyczące spotkania.

Moduł `pyautogui` oferuje funkcje `pyautogui.dragTo()` i `pyautogui.dragRel()` przeznaczone do symulacji przeciągnięcia kursora myszy do nowego położenia lub położenia względem bieżącego. Argumenty funkcji `pyautogui.dragTo()` i `pyautogui.dragRel()` są takie same jak omówionych wcześniej funkcji `moveTo()` i `moveRel()`: współrzędna X dla ruchu poziomego, współrzędna Y dla ruchu pionowego

oraz opcjonalny czas trwania operacji. (W systemie macOS przeciągnięcie nie odbywa się prawidłowo, gdy kursor myszy porusza się zbyt szybko. Dlatego też zalecane jest użycie argumentu w postaci słowa kluczowego `duration`).

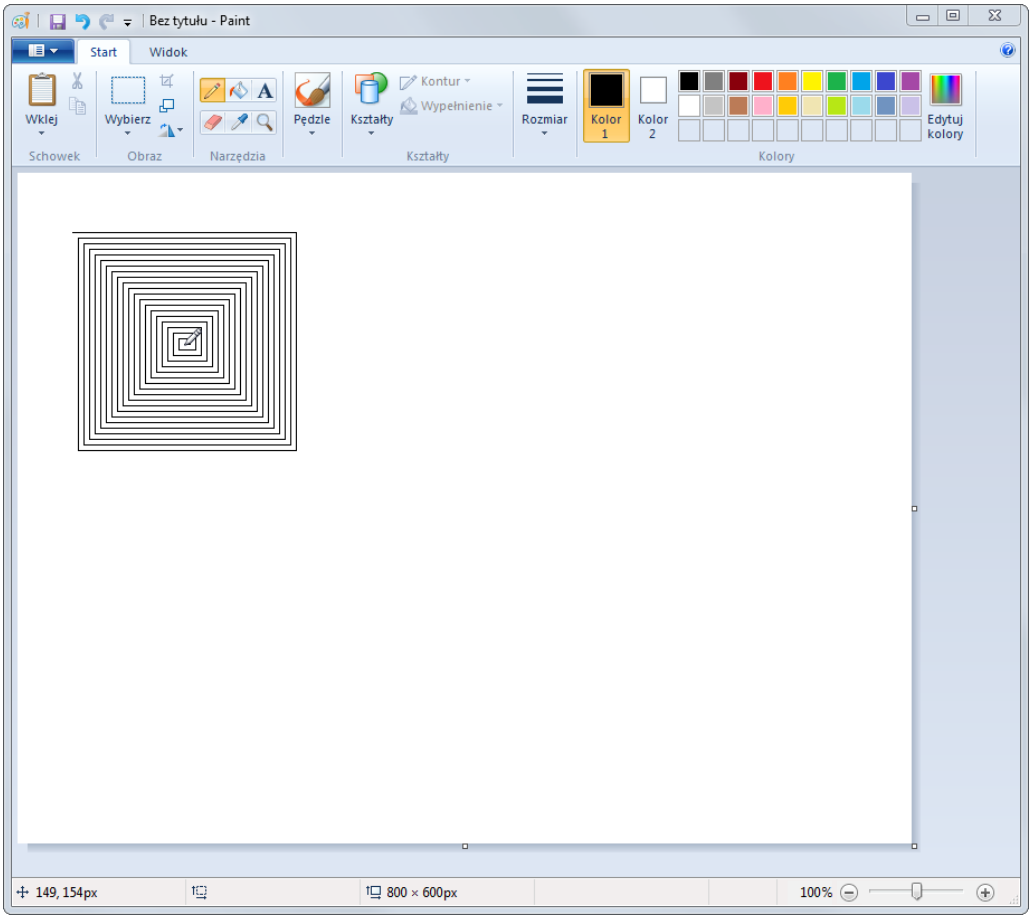
Aby wypróbować omówione powyżej funkcje, uruchom program graficzny, taki jak Paint w systemie Windows, Paintbrush w macOS lub GNU Paint w systemie Linux. (Jeżeli nie masz zainstalowanego żadnego edytora graficznego, możesz skorzystać z dostępnego w internecie, na przykład w witrynie *http://www.sumopaint.com/home/*). Moduł `pyautogui` wykorzystamy do rysowania w wymienionych aplikacjach.

Po umieszczeniu kursora myszy nad nowym obrazem w edytorze graficznym oraz po wybraniu narzędzia, takiego jak ołówek lub pędzel, wprowadź przedstawiony poniżej fragment kodu w nowym oknie edytora pliku i zapisz go pod nazwą *spiralDraw.py*.

```
import pyautogui, time
time.sleep(5) ❶
pyautogui.click()      # Kliknięcie w celu aktywacji programu.❷
distance = 200
while distance > 0:
    pyautogui.dragRel(distance, 0, duration=0.2)   # Przesunięcie kursora myszy w prawo. ❸
    distance = distance - 5 ❹// 4.
    pyautogui.dragRel(0, distance, duration=0.2)   # Przesunięcie kursora myszy w dół. ❺
    pyautogui.dragRel(-distance, 0, duration=0.2)  # Przesunięcie kursora myszy w lewo. ❻
    distance = distance - 5
    pyautogui.dragRel(0, -distance, duration=0.2)  # Przesunięcie kursora myszy w górę.
```

Po uruchomieniu programu nastąpi pięciosekundowa przerwa ❶ pozwalająca na umieszczenie kursora myszy nad oknem programu graficznego oraz wybór odpowiedniego narzędzia rysującego. Następnie program *spiralDraw.py* przejmuje kontrolę nad myszą i klika okno programu graficznego w celu jego aktywacji ❷. Okno jest uznawane za *aktywne*, gdy zawiera migający kursor. Wówczas wpisywane znaki lub — jak w omawianym przykładzie — operacja przeciągania kursorem myszy będzie miała wpływ na dane okno. Gdy edytor graficzny jest aktywny, program *spiralDraw.py* narysuje wzór spirali podobny do pokazanego na rysunku 18.2.

Wartość początkowa zmiennej `distance` wynosi `200`, więc podczas pierwszej iteracji pętli `while` pierwsze wywołanie `dragRel()` spowoduje przeciągnięcie kursora myszy o 200 pikseli w prawą stronę, a operacja potrwa 0,2 sekundy ❸. Wartość zmiennej `distance` zostaje zmniejszona do `195` ❹ i drugie wywołanie funkcji `dragRel()` przeciąga kursor myszy o 195 pikseli w dół ❺. Trzecie wywołanie funkcji `dragRel()` powoduje przeciągnięcie kursora myszy o 195 pikseli w poziomie, w lewą stronę ❻. Wartość zmiennej `distance` zostaje zmniejszona do `190` i ostatnie wywołanie `dragRel()` przeciąga kursor myszy o 190 pikseli w górę. Podczas każdej iteracji pętli kursor myszy jest przeciągany w prawo, w dół, w lewo i w górę, a zmienna `distance` ma nieco mniejszą wartość niż w poprzedniej iteracji. Dzięki umieszczeniu omówionego kodu w pętli kursorem myszy można poruszać w taki sposób, aby narysować kwadratową spiralę.



Rysunek 18.2. Wynik działania programu wykorzystującego funkcję pyautogui.dragRel()

Spiralę można narysować ręcznie (a dokładnie za pomocą myszy), ale będzie to dość wolna operacja wymagająca dużej precyzji. Moduł `pyautogui` zrobi w kilka sekund!

UWAGA *Do narysowania pokazanej spirali można by wykorzystać funkcje oferowane przez moduł `pillow`; więcej informacji na temat znajdziesz w rozdziale 17. Jednak zastosowanie automatyzacji GUI pozwala na wykorzystanie zaawansowanych narzędzi rysujących, które mogą być dostarczone przez edytor graficzny. Przykładem takich narzędzi są gradienty, różne pędzle i wypełnienie.*

Przewijanie myszą

Ostatnią funkcją dotyczącą myszy w module `pyautogui` jest `scroll()`. Argumentem tej funkcji jest liczba całkowita określająca liczbę jednostek, o które ma być przewinięta zawartość w górę lub w dół. Wielkość jednostki zależy od systemu

operacyjnego i aplikacji, więc trzeba poeksperymentować, aby dokładnie zobaczyć, jak przebiega przewijanie w danej sytuacji. Symulacja przewijania zachodzi w bieżącym położeniu kursora. Przekazanie funkcji `scroll()` wartości dodatniej powoduje przewinięcie w górę, natomiast — wartości ujemnej oznacza przewinięcie w dół. Poniższe polecenie wydaj w powłoce interaktywnej środowiska IDLE, gdy kursor myszy znajduje się nad oknem środowiska IDLE.

```
>>> pyautogui.scroll(200)
```

Zobaczysz, jak zawartość okna IDLE zostaje przewinięta w górę, a później powraca w dół. Przewinięcie w dół następuje, ponieważ środowisko IDLE automatycznie przewija zawartość w dół po wykonaniu polecenia. Wprowadź teraz poniższy fragment kodu.

```
>>> import pyperclip
>>> numbers = ''
>>> for i in range(200):
        numbers = numbers + str(i) + '\n'

>>> pyperclip.copy(numbers)
```

Powyższy fragment kodu importuje moduł `pyperclip` i definiuje zmienną o nazwie `numbers` zawierającą pusty ciąg tekstowy. Kod przeprowadza iterację przez 200 liczb i każdą z nich dodaje do zmiennej `numbers` wraz ze znakiem nowego wiersza. Po wykonaniu polecenia `pyperclip.copy(numbers)` schowek będzie zawierał 200 wierszy liczb. Otwórz nowe okno edytora pliku i wklej w nim tekst. W ten sposób otrzymasz wystarczająco duże okno, aby w nim przewijać zawartość. W powłoce interaktywnej wprowadź przedstawione poniżej polecenia.

```
>>> import time, pyautogui
>>> time.sleep(5); pyautogui.scroll(100)
```

W wierszu drugim mamy dwa polecenia rozdzielone średnikiem, który nakazuje Pythonowi wykonanie ich tak, jakby znajdowały się w oddzielnych wierszach. Jedyna różnica polega na tym, że powłoka interaktywna nie będzie oczekiwała podania danych wejściowych między tymi dwoma poleceniami. To jest bardzo ważne w omawianym przykładzie, ponieważ chcemy, aby wywołanie `pyautogui.` ↪`scroll()` następowało automatycznie po przerwie. (Wprawdzie umieszczenie tych dwóch poleceń w jednym wierszu będzie użyteczne w powłoce interaktywnej, ale w tworzonych programach powinieneś je umieścić w oddzielnych wierszach).

Po naciśnięciu klawisza *Enter* w celu uruchomienia kodu masz pięć sekund na kliknięcie okna edytora pliku i tym samym aktywowanie go. Po upłynięciu podanego czasu funkcja `pyautogui.scroll()` będzie w odstępach pięciosekundowych przewijała zawartość tego okna w górę.

Praca z ekranem

Twoje programy stosujące automatyzację GUI nie muszą ślepo klikać i wprowadzać treści. Moduł `pyautogui` ma możliwość wykonania zrzutu ekranu, czyli utworzenia pliku obrazu na podstawie aktualnej zawartości ekranu. Funkcjonalność powoduje również zwrot obiektu typu `Image` przedstawiającego bieżący wygląd ekranu. Jeżeli tylko przeglądałeś zawartość tej książki, powinieneś zapoznać się z rozdziałem 17. i zainstalować moduł `pillow`, zanim będziesz kontynuować pracę i wykonywać przykłady przedstawione w tym podrozdziale.

W systemie Linux konieczne jest zainstalowanie programu `scrot`, aby można było używać oferowanej przez moduł `pyautogui` funkcji tworzenia zrzutu ekranu. Instalację tego programu możesz przeprowadzić w powłoce za pomocą polecenia **`sudo apt-get install scrot`**. Jeżeli używasz systemów Windows lub macOS, pomiń ten krok i kontynuuj lekturę.

Wykonanie zrzutu ekranu

W celu wykonania zrzutu ekranu w Pythonie należy wywołać funkcję `pyautogui.screenshot()`. W powłoce interaktywnej wprowadź przedstawione poniżej polecenia.

```
>>> import pyautogui
>>> im = pyautogui.screenshot()
```

W powyższym fragmencie kodu zmienna `im` zawiera obiekt typu `Image` przedstawiający wykonany zrzut ekranu. Teraz w zmiennej `im` możesz wywoływać metody obiektu `Image`, podobnie jak w każdym innym obiekcie tego typu. W powłoce interaktywnej wprowadź przedstawione poniżej polecenia.

```
>>> im.getpixel((0, 0))
(176, 176, 175)
>>> im.getpixel((50, 200))
(130, 135, 144)
```

Funkcji `getpixel()` przekazujemy krotkę współrzędnych, taką jak (0, 0) lub (50, 200), a wartością zwrotną będzie kolor piksela znajdującego się na obrazie w punkcie o podanych współrzędnych. Wartość zwrotna funkcji `getpixel()` to krotka RGB składająca się z trzech liczb całkowitych określających ilość składowej koloru czerwonego, zielonego i niebieskiego w danym pikselu. (Nie ma tutaj czwartej wartości kanału alfa, ponieważ obraz zrzutu ekranu nie zawiera przezroczystości). W taki sposób program może „zobaczyć", co aktualnie znajduje się na ekranie.

Analiza zrzutu ekranu

Przyjmujemy założenie, że jednym z kroków w programie stosującym automatyzację GUI jest kliknięcie szarego przycisku. Przed wywołaniem metody `click()` możesz wykonać zrzut ekranu i spojrzeć na piksel, który ma być kliknięty przez skrypt. Jeżeli nie będzie to taki sam kolor szary jak na przycisku do kliknięcia, wówczas program „wie", że coś jest nie w porządku. Być może okno zostało nieoczekiwanie przesunięte lub wyskakujące okno dialogowe przysłoniło przycisk. Na tym etapie zamiast kontynuować operację — i prawdopodobnie kliknąć nieprawidłowy element — program może „zobaczyć", że kliknięty będzie niepoprawny element i zatrzymać swoje działanie.

Funkcja `pixelMatchesColor()` modułu `pyautogui` zwróci wartość `True`, jeśli piksel w punkcie o podanych współrzędnych X i Y na ekranie ma określony kolor. Pierwszy i drugi argument tej funkcji to liczby całkowite definiujące współrzędne X i Y. Trzecim argumentem jest krotka trzech liczb całkowitych określających kolor RGB punktu na ekranie, który ma być dopasowany przez piksel. W powłoce interaktywnej wprowadź przedstawione poniżej polecenia.

```
>>> import pyautogui
>>> im = pyautogui.screenshot()
>>> im.getpixel((50, 200)) ❶
(130, 135, 144)
>>> pyautogui.pixelMatchesColor(50, 200, (130, 135, 144)) ❷
True
>>> pyautogui.pixelMatchesColor(50, 200, (255, 135, 144)) ❸
False
```

Po wykonaniu zrzutu ekranu i użyciu funkcji `getpixel()` w celu pobrania krotki RGB dla piksela znajdującego się w punkcie o wskazanych współrzędnych ❶ te same współrzędne i krotkę koloru RGB przekazujemy funkcji `pixel↪MatchesColor()` ❷, która powinna zwrócić wartość `True`. Następnie zmieniamy wartość w krotce RGB i ponownie wywołujemy funkcję `pixelMatchesColor()` wraz z tymi samymi współrzędnymi ❸. Wartością zwrotną powinno być `False`. Metoda okaże się użyteczna, gdy program stosujący automatyzację GUI będzie miał wywołać funkcję `click()`. Pamiętaj, że kolor w punkcie o podanych współrzędnych musi być *dokładnie* dopasowany. Jeżeli będzie nawet minimalnie inny, na przykład `(255, 255, 254)` zamiast `(255, 255, 255)`, wtedy wartością zwrotną funkcji `pixelMatchesColor()` będzie `False`.

Projekt — rozbudowa programu mouseNow.py

Możemy teraz rozbudować utworzony wcześniej w rozdziale program *mouseNow.py*, aby wyświetlał nie tylko współrzędne X i Y bieżącego położenia kursora myszy, ale również podawał kolor RGB piksela wskazywanego przez kursor myszy.

Kod zdefiniowany w pętli `while` programu *mouseNow.py* zmodyfikuj w pokazany poniżej sposób.

```
#! python3
# mouseNow.py — Wyświetla bieżące położenie kursora myszy.
--cięcie--
        positionStr = 'X: ' + str(x).rjust(4) + ' Y: ' + str(y).rjust(4)
        pixelColor = pyautogui.screenshot().getpixel((x, y))
        positionStr += ' RGB: (' + str(pixelColor[0]).rjust(3)
        positionStr += ', ' + str(pixelColor[1]).rjust(3)
        positionStr += ', ' + str(pixelColor[2]).rjust(3) + ')'
        print(positionStr, end='')
--cięcie--
```

Kiedy teraz uruchomisz program *mouseNow.py*, wygenerowane dane wyjściowe będą zawierały także wartość RGB piksela wskazywanego przez kursor myszy.

```
Naciśnij klawisze Ctrl+C, aby zakończyć działanie programu.
X:  406 Y:   17 RGB: (161,  50,  50)
```

Te informacje wraz z funkcją `pixelMatchesColor()` powinny ułatwić Ci dodanie do skryptów stosujących automatyzację GUI kodu sprawdzającego kolor piksela.

Rozpoznawanie obrazu

Co zrobić w sytuacji, jeśli wcześniej nie wiadomo, w którym punkcie trzeba będzie kliknąć? W takim przypadku można użyć rozpoznawania obrazu. Przekaż modułowi `pyautogui` obraz przeznaczony do kliknięcia i pozwól mu na ustalenie współrzędnych.

Jeżeli na przykład wcześniej wykonałeś zrzut ekranu w celu przygotowania przechowywanego w pliku *submit.png* obrazu przycisku *Prześlij*, wówczas funkcja `locateOnScreen()` zwróci współrzędne miejsca na ekranie, w którym został znaleziony ten obraz. Aby zobaczyć, jak działa funkcja `locateOnScreen()`, wykonaj zrzut niewielkiej części ekranu, zapisz plik obrazu na dysku, a następnie w powłoce interaktywnej wprowadź przedstawione poniżej polecenia. Nie zapomnij o zastąpieniu *submit.png* nazwą, pod którą zapisałeś plik.

```
>>> import pyautogui
>>> pyautogui.locateOnScreen('submit.png')
(643, 745, 70, 29)
```

Czteroelementowa krotka zwrócona przez funkcję `locateOnScreen()` zawiera informacje o współrzędnej X lewej krawędzi, współrzędnej Y górnej krawędzi, szerokości i wysokości pierwszego miejsca na ekranie, w którym został znaleziony

obraz. Po wypróbowaniu powyższego kodu we własnym komputerze z użyciem innego obrazu wygenerowane dane wyjściowe będą odmienne od przedstawionych w przykładzie.

Jeżeli obraz nie zostanie znaleziony na ekranie, wartością zwrotną `locateOn ↪Screen()` będzie `None`. Pamiętaj, że obraz na ekranie musi być doskonale dopasowany do podanego w kodzie, aby mógł zostać rozpoznany. Jeżeli obraz będzie się różnił chociaż jednym pikselem, wtedy funkcja `locateOnScreen()` zwraca wartość `None`.

W przypadku znalezienia obrazu w kilku miejscach na ekranie funkcja `locate ↪OnScreen()` zwraca obiekt `Generator`. Obiekt można później przekazać funkcji `list()`, która z kolei zwróci krotki składające się z czterech liczb całkowitych. Dla każdego wystąpienia obrazu na ekranie otrzymamy czteroelementową krotkę wskazującą miejsce danego wystąpienia. Kontynuujemy wcześniejszy przykład w powłoce interaktywnej. Wpisz poniższe polecenie i pamiętaj o zastąpieniu ciągu tekstowego `'submit.png'` ciągiem zawierającym nazwę używanego obrazu.

```
>>> list(pyautogui.locateAllOnScreen('submit.png'))
[(643, 745, 70, 29), (1007, 801, 70, 29)]
```

Każda z czteroelementowych krotek przedstawia pewien obszar na ekranie. Jeżeli szukany obraz będzie znaleziony tylko w jednym miejscu, wówczas wartością zwrotną funkcji `list()` i `locateAllOnScreen()` będzie po prostu lista zawierająca tylko jedną krotkę.

Mając składającą się z czterech elementów krotkę przedstawiającą obszar na ekranie, gdzie został znaleziony obraz, możesz kliknąć środek tego obszaru. W tym celu krotkę przekaż do funkcji `center()`, której wartością zwrotną są współrzędne X i Y oznaczające środek danego obszaru. W powłoce interaktywnej wprowadź przedstawione poniżej polecenia. Nie zapomnij o zastąpieniu odpowiednimi wartościami nazwy pliku obrazu, czteroelementowej krotki i pary współrzędnych.

```
>>> pyautogui.locateOnScreen('submit.png')
(643, 745, 70, 29)
>>> pyautogui.center((643, 745, 70, 29))
(678, 759)
>>> pyautogui.click((678, 759))
```

Po otrzymaniu współrzędnych środka z funkcji `center()` przekazanie ich do funkcji `click()` powinno zasymulować kliknięcie na ekranie obszaru, który został dopasowany do obrazu podanego jako argument funkcji `locateOnScreen()`.

Kontrola klawiatury

Moduł `pyautogui` zawiera funkcje przeznaczone do symulowania naciśnięć klawiszy na klawiaturze. Dzięki temu zyskujesz możliwość wypełniania formularzy oraz wprowadzania tekstu w aplikacjach.

Przekazanie ciągu tekstowego z klawiatury

Funkcja `pyautogui.typewrite()` symuluje naciśnięcie klawisza na klawiaturze. Działanie tego wirtualnego naciśnięcia klawisza zależy od aktywnego okna oraz pola tekstowego. Być może najpierw trzeba będzie symulować kliknięcie myszą pola tekstowego, aby zostało wybrane i stało się aktywne.

Przechodzimy teraz do prostego przykładu. Pozwolimy Pythonowi na automatyczne wpisanie ciągu tekstowego *Witaj, świecie!* w oknie edytora pliku. Najpierw otwórz nowe okno edytora pliku i umieść je w lewym górnym rogu ekranu. Wtedy po symulacji przez moduł `pyautogui` kliknięcia odpowiedniego miejsca okno to stanie się aktywne. Następnie w powłoce interaktywnej wprowadź przedstawione poniżej polecenie.

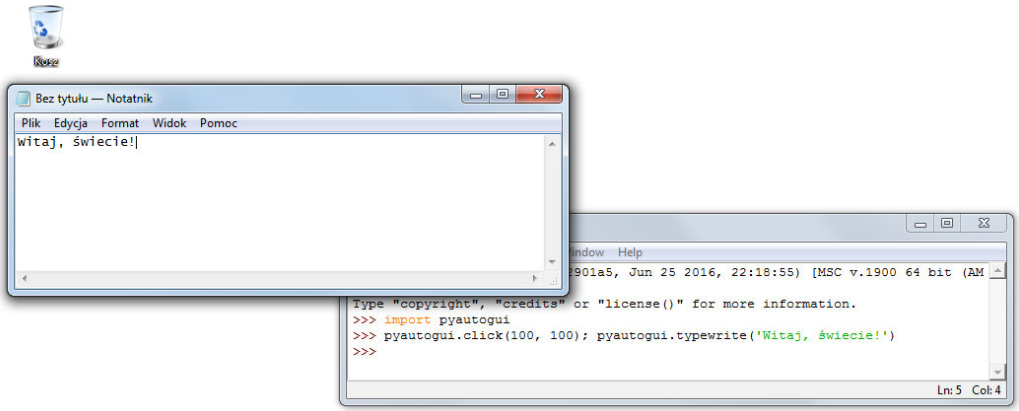
```
>>> pyautogui.click(100, 100); pyautogui.typewrite('Witaj, świecie!')
```

Zwróć uwagę na umieszczenie rozdzielonych średnikiem dwóch poleceń w tym samym wierszu. W ten sposób unikamy sytuacji, gdy powłoka interaktywna oczekuje na podanie danych wejściowych między wykonaniem obu poleceń. Takie rozwiązanie chroni również przed przypadkowym aktywowaniem innego okna między wykonaniem funkcji `click()` i `typewrite()`, co mogłoby namieszać w tym przykładzie.

Python najpierw symuluje wirtualne kliknięcie myszą punktu o współrzędnych (100, 100), który powinien wskazywać okno edytora pliku i tym samym je aktywować. Następnie wywołanie funkcji `typewrite()` przekazuje ciąg tekstowy *Witaj, świecie!* do okna, wyglądającego teraz tak, jak pokazałem na rysunku 18.3. W ten sposób dysponujesz kodem, który potrafi za Ciebie wpisywać tekst.

Domyślnie funkcja `typewrite()` natychmiast wpisze cały ciąg tekstowy. Jednak istnieje możliwość przekazania jej opcjonalnego drugiego argumentu określającego krótką przerwę między wpisywaniem poszczególnych znaków. Wartość tego drugiego argumentu to liczba całkowita lub zmiennoprzecinkowa definiująca wyrażoną w sekundach pauzę. Przykładowo polecenie `pyautogui.typewrite('Witaj, świecie!', 0.25)` określa przerwę trwającą 1/4 sekundy po wpisaniu znaku *W*, kolejną przerwę po znaku *i*, kolejną po *t* i tak dalej. Tego rodzaju możliwość wolniejszego wprowadzania tekstu może być użyteczna w aplikacjach, które nie mogą przetwarzać znaków wystarczająco szybko, aby nadążyć za modułem `pyautogui`.

W przypadku znaków, takich jak *A* lub *!*, moduł `pyautogui` automatycznie symuluje naciśnięcie także klawisza *Shift*.



Rysunek 18.3. Użycie modułu pyautogui w celu kliknięcia okna edytora pliku oraz wpisania w nim ciągu tekstowego Witaj, świecie!

Nazwy klawiszy

Nie wszystkie klawisze są łatwe do przedstawienia za pomocą pojedynczych znaków. Jak na przykład mógłbyś z użyciem jednego znaku przedstawić klawisz *Shift* lub strzałkę kursora w lewo? W module `pyautogui` te klawisze klawiatury są symulowane za pomocą krótkich ciągów tekstowych. I tak `'esc'` oznacza klawisz *ESC*, natomiast `'enter'` klawisz *Enter*.

Funkcji `typewrite()` można zamiast pojedynczego argumentu w postaci ciągu tekstowego przekazać listę ciągów tekstowych przedstawiających klawisze. Przykładowo w poniższym wywołaniu mamy symulację naciśnięcia klawisza *A*, później *B*, następnie dwukrotnie strzałki kursora w lewo oraz na końcu klawiszy *X* i *Y*.

```
>>> pyautogui.typewrite(['a', 'b', 'left', 'left', 'X', 'Y'])
```

Ponieważ naciśnięcie klawisza strzałki kursora w lewo powoduje przeniesienie kursora klawiatury, więc wygenerowane będą dane wyjściowe *XYab*. W tabeli 18.1 wymieniłem ciągi tekstowe klawiszy klawiatury, jakie można przekazać funkcji `typewrite()` modułu `pyautogui` w celu symulacji naciśnięć dowolnych kombinacji klawiszy.

Możesz również przeanalizować listę `pyautogui.KEYBOARD_KEYS` w celu poznania wszystkich możliwych ciągów tekstowych klawiszy, jakie są akceptowane przez `pyautogui`. Ciąg tekstowy `'shift'` odwołuje się do klawisza *Shift* i jest odpowiednikiem `'shiftLeft'`. To samo ma zastosowanie do ciągów tekstowych `'ctrl'`, `'alt'` i `'win'`, wszystkie odwołują się do klawiszy znajdujących się po lewej stronie klawiatury.

Tabela 18.1. Atrybuty PyKeyboard

Klawisz	Opis
`'a'`, `'b'`, `'c'`, `'A'`, `'B'`, `'C'`, `'1'`, `'2'`, `'3'`, `'!'`, `'@'`, `'#'` i tak dalej.	Klawisze poszczególnych znaków.
`'enter'` (lub `'return'` lub `'\n'`)	Klawisz *Enter*.
`'esc'`	Klawisz *Esc*.
`'shiftleft'`, `'shiftright'`	Lewy i prawy klawisz *Shift*.
`'altleft'`, `'altright'`	Lewy i prawy klawisz *Alt*.
`'ctrlleft'`, `'ctrlright'`	Lewy i prawy klawisz *Ctrl*.
`'tab'` (lub `'\t'`)	Klawisz *Tab*.
`'backspace'`, `'delete'`	Klawisze *Backspace* i *Delete*.
`'pageup'`, `'pagedown'`	Klawisze *Page Up* i *Page Down*.
`'home'`, `'end'`	Klawisze *Home* i *End*.
`'up'`, `'down'`, `'left'`, `'right'`	Klawisze strzałek kursora.
`'f1'`, `'f2'`, `'f3'` i tak dalej.	Klawisze od *F1* do *F12*.
`'volumemute'`, `'volumedown'`, `'volumeup'`	Klawisze wyciszenia dźwięku, zmniejszenia poziomu dźwięku i zwiększenia poziomu dźwięku. (Wprawdzie niektóre klawiatury nie mają tych klawiszy, ale system operacyjny i tak potrafi symulować ich naciśnięcia).
`'pause'`	Klawisz *Pause*.
`'capslock'`, `'numlock'`, `'scrolllock'`	Klawisze *Caps Lock*, *Num Lock* i *Scroll Lock*.
`'insert'`	Klawisz *Ins* lub *Insert*.
`'printscreen'`	Klawisz *Prtsc* lub *Print Screen*.
`'winleft'`, `'winright'`	Lewy i prawisz klawisz *Win* (w systemie Windows).
`'command'`	Klawisz *Command* (w systemie macOS).
`'option'`	Klawisz *Option* (w systemie macOS).

Naciskanie i zwalnianie klawiszy

Podobnie jak funkcje `mouseDown()` i `mouseUp()`, także funkcje `pyautogui.keyDown()` i `pyautogui.keyUp()` symulują wirtualne naciśnięcie i zwolnienie klawisza. Argumentem funkcji `pyautogui.keyDown()` i `pyautogui.keyUp()` jest dowolny z wymienionych w tabeli 18.1 ciąg tekstowy klawisza klawiatury. Dla wygody moduł `pyau` ↪`togui` oferuje funkcję `pyautogui.press()`, która wywołuje wymienione funkcje, aby symulować pełne naciśnięcie klawisza.

Wydaj poniższe polecenie, które spowoduje wpisanie znaku dolara (otrzymanego przez symulację naciśnięcia klawiszy *Shift*+*4*).

```
>>> pyautogui.keyDown('shift'); pyautogui.press('4'); pyautogui.keyUp('shift')
```

Powyższe polecenie symuluje naciśnięcie klawisza *Shift*, naciśnięcie (i zwolnienie) klawisza *4*, a następnie zwolnienie klawisza *Shift*. Jeżeli konieczne jest wpisanie ciągu tekstowego w polu tekstowym, funkcja `typewrite()` lepiej nadaje się do tego celu. Jednak w aplikacjach wykorzystujących polecenia w postaci pojedynczych naciśnięć klawiszy funkcja `press()` stanowi prostsze podejście.

Kombinacja klawiszy

Kombinacja klawiszy lub inaczej *skrót* to połączenie naciśnięcia kilku klawiszy, co powoduje wywołanie pewnej funkcji aplikacji. Do najczęściej stosowanych kombinacji klawiszy zaliczamy *Ctrl+C* (w systemach Windows i Linux) i *Command+C* (system macOS). W takim przypadku użytkownik naciska i przytrzymuje klawisz *Ctrl*, następnie naciska klawisz *C* i później zwalnia oba klawisze. Aby tę kombinację odtworzyć w module `pyautogui` za pomocą funkcji `keyDown()` i `keyUp()`, można użyć poniższego fragmentu kodu.

```
pyautogui.keyDown('ctrl')
pyautogui.keyDown('c')
pyautogui.keyUp('c')
pyautogui.keyUp('ctrl')
```

To jednak jest nieco skomplikowane. Zamiast powyższego fragmentu kodu lepiej wykorzystać funkcję `pyautogui.hotkey()`, która pobiera wiele argumentów w postaci ciągów tekstowych klawiszy, symuluje ich naciśnięcie w podanej kolejności, a następnie zwolnienie ich w odwrotnej kolejności. Dlatego też w przykładzie kombinacji klawiszy *Ctrl+C* możemy użyć poniższego wywołania.

```
pyautogui.hotkey('ctrl', 'c')
```

Funkcja `pyautogui.hotkey()` okazuje się szczególnie użyteczna w przypadku większych kombinacji klawiszy. W procesorze tekstu Microsoft Word kombinacja klawiszy *Ctrl+Alt+S* powoduje wyświetlenie panelu stylów. Zamiast używać sześciu różnych wywołań funkcji (po trzy `keyDown()` i `keyUp()`), lepiej będzie użyć funkcji `pyautogui.hotkey('ctrl', 'alt', 's')`.

Mając umieszczone nowe okno edytora pliku środowiska IDLE w lewym górnym rogu ekranu, w powłoce interaktywnej wprowadź poniższy fragment kodu. Jeżeli używasz systemu macOS, ciąg tekstowy `'alt'` zastąp przez `'ctrl'`.

```
>>> import pyautogui, time
>>> def commentAfterDelay():
      pyautogui.click(100, 100) ❶
      pyautogui.typewrite('W środowisku IDLE Alt-3 umieszcza wiersz w komentarzu.') ❷
      time.sleep(2)
      pyautogui.hotkey('alt', '3') ❸

>>> commentAfterDelay()
```

W powyższym fragmencie kodu zdefiniowaliśmy funkcję o nazwie `commentAf ↪terDelay()`, która po wywołaniu spowoduje kliknięcie okna edytora pliku, aktywując je tym samym ❶. Następnie wpisuje w nim tekst `W środowisku IDLE Alt-3 umieszcza wiersz w komentarzu` ❷, wstrzymuje działanie przez dwie sekundy, a później symuluje naciśnięcie kombinacji klawiszy *Alt+3* (w systemie macOS to będzie *Ctrl+3*) ❸. Ten skrót klawiszowy powoduje dodanie dwóch znaków `##` w bieżącym wierszu, co oznacza utworzenie komentarza. (To jest użyteczna sztuczka, gdy tworzysz kod Pythona w edytorze środowiska IDLE).

Przegląd funkcji modułu pyautogui

Ponieważ w tym rozdziale przedstawiłem wiele różnych funkcji, poniżej znajdziesz ich krótkie podsumowanie.

- `moveTo(x, y)`. Przenosi kursor myszy do punktu o podanych współrzędnych X i Y.
- `moveRel(xOffset, yOffset)`. Przenosi kursor myszy względem jego bieżącego położenia.
- `dragTo(x, y)`. Przenosi kursor myszy i jednocześnie symuluje przytrzymanie lewego przycisku myszy.
- `dragRel(xOffset, yOffset)`. Przenosi kursor myszy względem jego bieżącego położenia i jednocześnie symuluje przytrzymanie lewego przycisku myszy.
- `click(x, y, button)`. Symuluje kliknięcie przyciskiem myszy (domyślnie to lewy przycisk).
- `rightClick()`. Symuluje kliknięcie prawym przyciskiem myszy.
- `middleClick()`. Symuluje kliknięcie środkowym przyciskiem myszy.
- `doubleClick()`. Symuluje dwukrotne kliknięcie lewym przyciskiem myszy.
- `mouseDown(x, y, button)`. Symuluje naciśnięcie przycisku w punkcie wskazywanym przez współrzędne X i Y.
- `mouseUp(x, y, button)`. Symuluje zwolnienie przycisku w punkcie wskazywanym przez współrzędne X i Y.
- `scroll(units)`. Symuluje przewijanie kółkiem myszy. Argument w postaci liczby dodatniej oznacza przewinięcie w górę, natomiast w postaci liczby ujemnej — przewinięcie w dół.
- `typewrite(message)`. Powoduje wpisanie znaków znajdujących się w podanym ciągu tekstowym.
- `typewrite([klawisz1, klawisz2, klawisz3])`. Powoduje wpisanie ciągów tekstowych przedstawiających podane klawisze.
- `press(klawisz)`. Symuluje pełne naciśnięcie i zwolnienie wskazanego klawisza.
- `keyDown(klawisz)`. Symuluje naciśnięcie wskazanego klawisza.

- `keyUp(klawisz)`. Symuluje zwolnienie wskazanego klawisza.
- `hotkey([klawisz1, klawisz2, klawisz3])`. Symuluje naciśnięcie w podanej kolejności klawiszy wskazanych przez ciągi tekstowe, a następnie zwolnienie tych klawiszy w odwrotnej kolejności.
- `screenshot()`. Zwraca obiekt typu `Image` przedstawiający zrzut ekranu. (Więcej informacji na temat obiekt `Image` znajdziesz w rozdziale 17.).

Projekt — automatyczne wypełnianie formularzy

Ze wszystkich żmudnych zadań do wykonania za najbardziej uciążliwe można uznać wypełnianie formularzy. Na szczęście zajmujemy się tym zadaniem jedynie w ostatnim rozdziale książki. Przyjmujemy założenie, że masz ogromną ilość danych w arkuszu kalkulacyjnym i przed Tobą stoi żmudne zadanie przepisania tych danych w pewnej innej aplikacji za pomocą formularza sieciowego. Niestety aplikacja nie może wykonać tego zadania za Ciebie. Wprawdzie niektóre aplikacje mają funkcję importowania danych pozwalającą na wczytanie arkusza kalkulacyjnego z danymi, ale czasami wydaje się, że nie ma innego rozwiązania niż tylko bezsensowne klikanie i wpisywanie danych całymi godzinami. Skoro dotarłeś tak daleko, czytając tę książkę, więc już doskonale wiesz, że *oczywiście* istnieje inna droga.

Formularz wykorzystany w tym projekcie został utworzony w aplikacji Dokumenty Google i znajdziesz go na stronie *https://docs.google.com/forms/d/e/1FAIpQLScSVDFU76rZvbO_tiIwSt6d9sOK0CZyS9KKMCP6cP5O5W5lVQ/viewform*. Wygląd dokumentu pokazałem na rysunku 18.4.

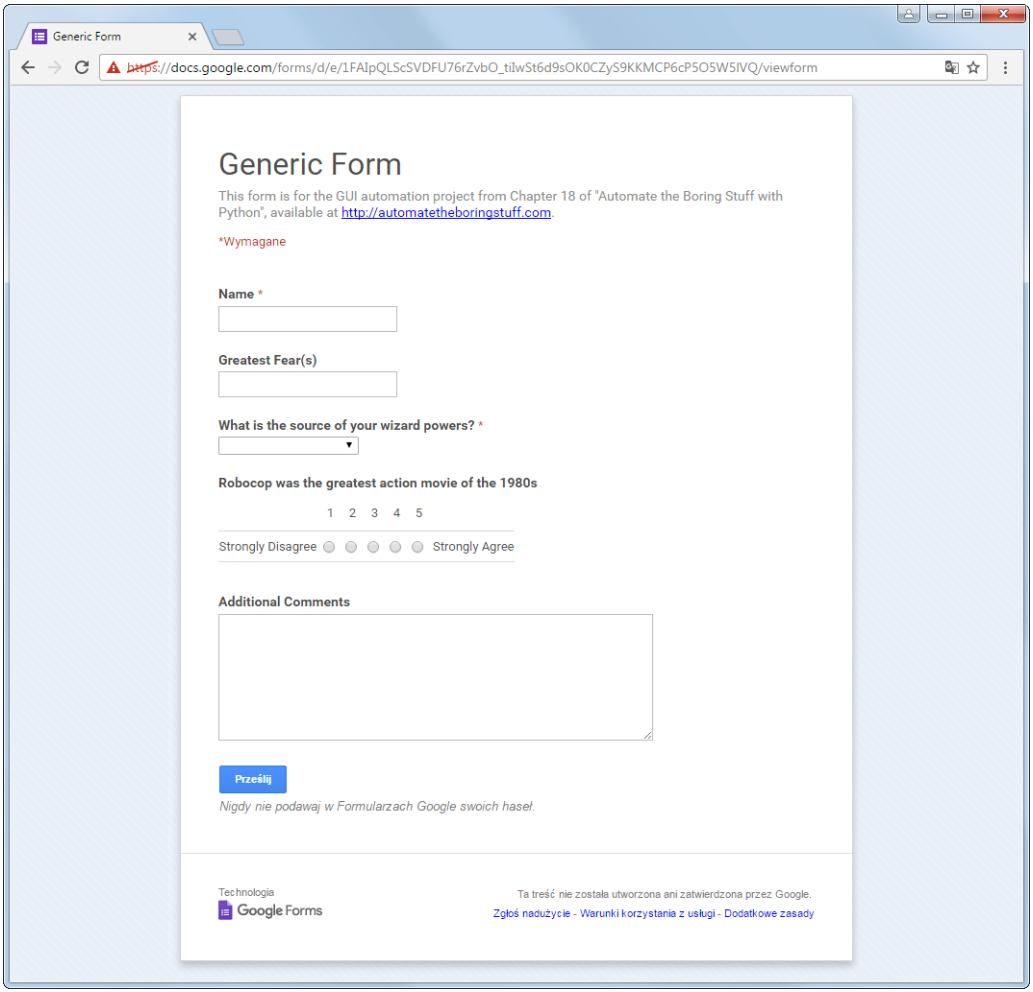
Poniżej wymieniłem w punktach sposób działania programu na wysokim poziomie.

- Kliknięcie pierwszego pola tekstowego formularza sieciowego.
- Przejście przez elementy formularza i wpisanie odpowiednich informacji w poszczególnych polach.
- Kliknięcie przycisku wysyłającego ten formularz sieciowy.
- Powtórzenie procesu wraz z kolejnym zbiorem danych.

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

- Wywołanie funkcji `pyautogui.click()` w celu kliknięcia formularza i przycisku *Prześlij*.
- Wywołanie funkcji `pyautogui.typewrite()` w celu wprowadzenia tekstu w polach.
- Obsługa wyjątku `KeyboardInterrupt`, aby użytkownik mógł zakończyć działanie programu przez naciśnięcie klawiszy *Ctrl+C*.

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



Rysunek 18.4. Formularz użyty w tym projekcie

Etap 1. Ustalenie kroków do wykonania

Zanim przystąpisz do tworzenia kodu, najpierw musisz ustalić dokładne naciśnięcia klawiszy i kliknięcia myszą potrzebne do jednokrotnego wypełnienia formularza sieciowego. Przygotowany we wcześniejszej części rozdziału skrypt *mouseNow.py* pomoże w określeniu odpowiednich współrzędnych. Potrzebne są tylko współrzędne pierwszego pola tekstowego. Po jego kliknięciu można po prostu nacisnąć klawisz *Tab* i za jego pomocą przejść do kolejnego pola formularza. W ten sposób unikasz konieczności ustalania współrzędnych X i Y wszystkich pól formularza.

Poniżej wymieniłem kroki prowadzące do wprowadzenia danych w formularzu.

1. Kliknij pole *Name*. (Wykorzystaj skrypt *mouseNow.py* do ustalenia współrzędnych tego pola tekstowego po maksymalnym zwiększeniu okna przeglądarki WWW. W systemie macOS może wystąpić konieczność dwukrotnego kliknięcia: pierwszego w celu aktywowania przeglądarki WWW, a drugiego w celu przejścia do podanego pola).
2. Wpisz imię, a następnie naciśnij klawisz *Tab*.
3. Wpisz swoją największą obawę i naciśnij klawisz *Tab*.
4. Naciśnij klawisz strzałki kursora w dół w celu skorygowania wartości kolejnego pola. Klawisz trzeba nacisnąć raz dla opcji *wand*, dwa razy dla opcji *amulet*, trzy dla opcji *crystal ball* i cztery dla opcji *money*. Następnie naciśnij klawisz *Tab*. (Zwróć uwagę, że w systemie macOS klawisz strzałki kursora w dół trzeba będzie nacisnąć o raz więcej dla każdej opcji. W przypadku niektórych przeglądarek WWW może trzeba będzie nacisnąć również klawisz *Enter*).
5. Naciśnij klawisz strzałki w prawo, aby wybrać odpowiedź na pytanie. Naciśnij ten klawisz raz dla opcji *2*, dwukrotnie dla opcji *3*, trzykrotnie dla opcji *4* lub czterokrotnie dla opcji *5*. Ewentualnie naciśnij po prostu spację, aby zaznaczyć opcję *1*, która jest wybrana domyślnie. Następnie naciśnij klawisz *Tab*.
6. Wpisz dodatkowy komentarz i naciśnij klawisz *Tab*.
7. Naciśnij klawisz *Enter* w celu „kliknięcia" przycisku *Prześlij*.
8. Po wysłaniu formularza sieciowego przeglądarka WWW przeniesie Cię na stronę, na której trzeba kliknąć łącze, aby powrócić z powrotem do formularza.

Zauważ, że jeśli uruchomisz ten program później, być może będziesz musiał uaktualnić współrzędne kliknięcia myszą, ponieważ okno przeglądarki WWW mogło zmienić położenie. Rozwiązaniem będzie tutaj zawsze maksymalizacja okna przeglądarki WWW przed operacją ustalenia współrzędnych pierwszego pola formularza sieciowego. Ponadto różne przeglądarki w odmiennych systemach operacyjnych mogą działać nieco inaczej niż w przedstawionych powyżej krokach. Dlatego też przed uruchomieniem omawianego programu sprawdź działanie wymienionych kombinacji klawiszy w Twoim komputerze.

Etap 2. Przygotowanie współrzędnych

W przeglądarce WWW wczytaj pobrany przykładowy formularz (patrz rysunek 18.4) i zmaksymalizuj okno przeglądarki. Teraz otwórz okno narzędzia Terminal (systemy macOS i Linux) lub wiersza poleceń (system Windows), uruchom skrypt *mouseNow.py* i umieść kursor myszy nad polem *Name*, aby ustalić jego współrzędne X i Y. Te liczby będą później przypisane zmiennej `nameField` w programie. Ponadto ustal współrzędne X i Y oraz wartość krotki RGB dla niebieskiego przycisku *Prześlij*. Te dane będą później przypisane zmiennym — odpowiednio — `submitButton` i `submitButtonColor`.

Kolejnym krokiem jest podanie pewnych przykładowych danych w formularzu oraz kliknięcie przycisku *Prześlij*. Musisz zobaczyć, jak wygląda kolejna strona, aby za pomocą skryptu *mouseNow.py* ustalić współrzędne wyświetlanego na niej łącza *Submit another response*.

W pliku wprowadź przedstawiony poniżej kod źródłowy. Upewnij się, że zastąpiłeś wszystkie wartości podane kursywą wartościami wcześniej ustalonymi przez siebie.

```
#! python3
# formFiller.py — Automatycznie wypełnia formularz sieciowy.

import pyautogui, time

# Zdefiniowanie odpowiednich współrzędnych we własnym komputerze.
nameField = (648, 319)
submitButton = (651, 817)
submitButtonColor = (75, 141, 249)
submitAnotherLink = (760, 224)

# TODO: Umożliwienie użytkownikowi zakończenia działania skryptu.

# TODO: Zaczekanie na wczytanie strony formularza sieciowego.

# TODO: Wypełnienie pola Name.

# TODO: Wypełnienie pola Greatest Fear(s).

# TODO: Wypełnienie pola Source of Wizard Powers.

# TODO: Wypełnienie pola RoboCop.

# TODO: Wypełnienie pola Additional Comments.

# TODO: Kliknięcie przycisku Prześlij.

# TODO: Zaczekanie na wczytanie strony.

# TODO: Kliknięcie łącza "Submit another response".
```

Teraz potrzebujemy danych, które faktycznie zostaną wprowadzone w tym formularzu sieciowym. W rzeczywistości dane mogą pochodzić z arkusza kalkulacyjnego, pliku zwykłego tekstu lub witryny internetowej. W takim przypadku konieczne będzie użycie dodatkowego kodu wczytującego te dane do programu. Jednak na potrzeby omawianego projektu wszystkie dane umieszczamy po prostu w zmiennej. W programie umieść przedstawiony poniżej nowy fragment kodu.

```
#! python3
# formFiller.py — Automatycznie wypełnia formularz sieciowy.

--cięcie--

formData = [{'name': 'Alicja', 'fear': 'eavesdroppers', 'source': 'wand',
             'robocop': 4, 'comments': 'Powiedz Bobowi, że mówię mu cześć.'},
            {'name': 'Bob', 'fear': 'bees', 'source': 'amulet', 'robocop': 4,
            'comments': 'n/a'},
```

```
        {'name': 'Karol', 'fear': 'puppets', 'source': 'crystal ball',
        'robocop': 1, 'comments': 'Proszę zabrać kukiełki z pokoju.'},
        {'name': 'Alex Murphy', 'fear': 'ED-209', 'source': 'money',
        'robocop': 5, 'comments': 'Chronić niewinnych. Służyć społeczeństwu.
        Stać na straży prawa.'},
       ]

--cięcie--
```

Lista `formData` zawiera cztery słowniki informacji dla czterech różnych osób. Poszczególne słowniki mają klucze w postaci nazw pól tekstowych formularza i wartości w postaci danych przeznaczonych do wpisania w tych polach. Ostatnią czynnością konfiguracyjną jest określenie wartości zmiennej modułu `pyautogui` o nazwie `PAUSE`. Ustalamy, że po każdym wywołaniu funkcji ma nastąpić półsekundowa przerwa. Po poleceniu `formData` umieść polecenie przedstawione poniżej.

```
pyautogui.PAUSE = 0.5
```

Etap 3. Rozpoczęcie wpisywania danych

Pętla `for` przeprowadza iteracje przez wszystkie słowniki na liście `formData`, przekazuje wartości słownika do funkcji modułu `pyautogui`, które następnie będą je wprowadzać w polach tekstowych formularza sieciowego.

W programie umieść przedstawiony poniżej nowy fragment kodu.

```
#! python3
# formFiller.py — Automatycznie wypełnia formularz sieciowy.

--cięcie--

for person in formData:
    # Umożliwienie użytkownikowi zakończenia działania skryptu.
    print('>>> 5-SEKUNDOWA PRZERWA POZWALAJĄCA UŻYTKOWNIKOWI NACISNĄĆ CTRL-C <<<')
    time.sleep(5) ❶

    # Zaczekanie na wczytanie strony formularza sieciowego.
    while not pyautogui.pixelMatchesColor(submitButton[0], submitButton[1], ❷
    submitButtonColor):
        time.sleep(0.5)

--cięcie--
```

W skrypcie definiujemy małą funkcję bezpieczeństwa — pięciosekundową przerwę ❶ pozwalającą użytkownikowi na naciśnięcie klawiszy *Ctrl+C* (lub przesunięcie kursora myszy do lewego górnego rogu ekranu w celu zgłoszenia wyjątku `FailSafeException`) — aby zakończyć działanie, jeśli program wykonuje coś niezgodnego z oczekiwaniami. Następnie program czeka, aż stanie się widoczny kolor przycisku *Prześlij* ❷ oznaczający zakończenie wczytywania strony zawierającej ten formularz sieciowy. Pamiętaj, że współrzędne X i Y oraz kolor przycisku ustaliłeś

na etapie 2. pracy nad aplikacją, a te informacje są przechowywane w zmiennych — odpowiednio — `submitButton` i `submitButtonColor`. Aby użyć funkcji `pixelMatchesColor()`, konieczne jest przekazanie współrzędnych `submitButton[0]` i `submitButton[1]`, a także koloru `submitButtonColor`.

Po kodzie oczekującym na udostępnienie koloru przycisku *Prześlij* umieść przedstawiony poniżej nowy fragment kodu.

```
#! python3
# formFiller.py — Automatycznie wypełnia formularz sieciowy.

--cięcie--

    print('Wprowadzenie informacji o osobie %s...' % (person['name'])) ❶
    pyautogui.click(nameField[0], nameField[1]) ❷

    # Wypełnienie pola Name.
    pyautogui.typewrite(person['name'] + '\t') ❸

    # Wypełnienie pola Greatest Fear(s).
    pyautogui.typewrite(person['fear'] + '\t') ❹

--cięcie--
```

W kodzie umieściliśmy wywołanie `print()` przeznaczone do wyświetlenia w oknie narzędzia terminala lub w wierszu poleceń informacji o stanie działania programu, aby użytkownik wiedział, co się dzieje ❶.

Ponieważ program będzie „wiedział”, kiedy zakończy się wczytywanie strony zawierającej formularz sieciowy, wtedy wywoła funkcję `click()` symulującą kliknięcie pola tekstowego *Name* ❷, natomiast funkcja `typewrite()` wpisze ciąg tekstowy przechowywany w `person['name']` ❸. Na końcu ciągu tekstowego przekazywanego funkcji `typewrite()` będzie dodany jest znak `'\t'` symulujący naciśnięcie klawisza *Tab*, który powoduje przejście do kolejnego pola formularza, tutaj *Greatest Fear(s)*. Kolejne wywołanie funkcji `typewrite()` spowoduje wpisanie w wymienionym polu ciągu tekstowego przechowywanego w `person['fear']`, a następnie za pomocą tabulatora przejście do kolejnego pola formularza ❹.

Etap 4. Obsługa rozwijanych list i przycisków opcji

Rozwijane menu w polu *Wizard Powers* i przyciski opcji w polu *RoboCop* są nieco trudniejsze do obsługi niż zwykle pola tekstowe. W celu kliknięcia wspomnianych opcji kursorem myszy najpierw trzeba ustalić współrzędne X i Y każdej możliwej opcji. Znacznie łatwiejsze będzie dokonanie wyboru opcji z użyciem klawiszy strzałek kursora.

W programie umieść przedstawiony poniżej nowy fragment kodu.

```
#! python3
# formFiller.py — Automatycznie wypełnia formularz sieciowy.

--cięcie--
```

```
    # Wypełnienie pola Source of Wizard Powers.
    if person['source'] == 'wand': ❶
        pyautogui.typewrite(['down', '\t']) ❷
    elif person['source'] == 'amulet':
        pyautogui.typewrite(['down', 'down', '\t'])
    elif person['source'] == 'crystal ball':
        pyautogui.typewrite(['down', 'down', 'down', '\t'])
    elif person['source'] == 'money':
        pyautogui.typewrite(['down', 'down', 'down', 'down', '\t'])

    # Wypełnienie pola RoboCop.
    if person['robocop'] == 1: ❸
        pyautogui.typewrite([' ', '\t']) ❹
    elif person['robocop'] == 2:
        pyautogui.typewrite(['right', '\t'])
    elif person['robocop'] == 3:
        pyautogui.typewrite(['right', 'right', '\t'])
    elif person['robocop'] == 4:
        pyautogui.typewrite(['right', 'right', 'right', '\t'])
    elif person['robocop'] == 5:
        pyautogui.typewrite(['right', 'right', 'right', 'right', '\t'])

--cięcie--
```

Po aktywowaniu rozwijanego menu (pamiętasz kod symulujący naciśnięcie klawisza *Tab* po wypełnieniu pola *Greatest Fear(s)*?) naciśnięcie klawisza strzałki kursora w dół spowoduje przejście do kolejnego elementu na liście. W zależności od wartości przechowywanej w zmiennej `person['source']`, program powinien wysłać odpowiednią liczbę naciśnięć klawisza przed przejściem do kolejnego pola formularza sieciowego. Jeżeli wartością klucza `'source'` w słowniku informacji o danym użytkowniku jest `'wand'` ❶, wówczas symulujemy tylko jednokrotne naciśnięcie klawisza strzałki kursora w dół (aby wybrać opcję *Wand*) i następnie symulujemy naciśnięcie klawisza *Tab* ❷. Jeżeli wartością klucza `'source'` jest `'amulet'`, symulujemy dwukrotne naciśnięcie klawisza strzałki kursora w dół, następnie klawisza *Tab* i tak dalej dla wszystkich pozostałych opcji rozwijanego menu.

Przyciski opcji dla pola *RoboCop* mogą być wybrane za pomocą klawisza kursora w prawo. Jeśli chcesz wybrać pierwszą opcję ❸, wystarczy naciśnięcie spacji ❹.

Etap 5. Wysłanie formularza i oczekiwanie

Pole *Additional Comments* formularza sieciowego można wypełnić za pomocą funkcji `typewrite()`, przekazując wartość `person['comments']` jako jej argument. Istnieje możliwość wpisania dodatkowej wartości `'\t'` w celu przejścia do kolejnego pola formularza, czyli tutaj przycisku *Prześlij*. Gdy ten przycisk jest aktywnym elementem, wywołanie funkcji `pyautogui.press('enter')` symuluje naciśnięcie klawisza *Enter* i tym samym wysłanie formularza. Po wysłaniu formularza program będzie czekał pięć sekund na wczytanie kolejnej strony.

Nowa strona po wczytaniu będzie zawierała łącze zatytułowane *Submit another response*, którego kliknięcie przenosi na nową stronę pustego formularza sieciowego. Współrzędne dla tego łącza na etapie 2. umieściliśmy w krotce o nazwie `submitAnotherLink`. Teraz przekazujemy te współrzędne funkcji `pyautogui.click()`, aby symulować kliknięcie łącza.

Kiedy nowy formularz jest gotowy do wypełnienia, zewnętrzna pętla `for` formularza kontynuuje działanie. W kolejnej iteracji w formularzu będą umieszczone informacje dotyczące następnej osoby.

Pracę nad programem zakończ przez dodanie przedstawionego poniżej nowego fragmentu kodu.

```
#! python3
# formFiller.py — Automatycznie wypełnia formularz sieciowy.

--cięcie--

    # Wypełnienie pola Additional Comments.
    pyautogui.typewrite(person['comments'] + '\t')

    # Kliknięcie przycisku Prześlij.
    pyautogui.press('enter')

    # Zaczekanie na wczytanie strony.
    print('Kliknięto przycisk Prześlij.')
    time.sleep(5)

    # Kliknięcie łącza "Submit another response".
    pyautogui.click(submitAnotherLink[0], submitAnotherLink[1])
```

Po zakończeniu działania zewnętrznej pętli `for` program wprowadził informacje o wszystkich osobach. W omawianym przykładzie mieliśmy tylko dane czterech osób. Jeżeli będziesz miał dane na przykład 4000 osób, wówczas opracowanie tego rodzaju programu może zaoszczędzić Ci naprawdę wiele czasu!

Podsumowanie

Automatyzacja GUI za pomocą modułu `pyautogui` pozwala na interakcję z zainstalowanymi w komputerze aplikacjami, które kontrolujemy, używając myszy i klawiatury. Wprawdzie przedstawione tutaj podejście jest na tyle elastyczne, aby wykonać wszystkie zadania, które w danej aplikacji może zrobić człowiek, ale wadą jest to, że tak przygotowany program na ślepo klika, gdzie mu każesz, i wpisuje, co mu każesz. Podczas pracy nad programami stosującymi automatyzację GUI postaraj się, aby tego rodzaju program szybko ulegał awarii, jeśli otrzyma nieprawidłowe polecenia. Wprawdzie awaria programu jest irytująca, ale jednocześnie stanowi znacznie lepsze rozwiązanie niż błędne działanie programu.

Wykorzystując moduł `pyautogui`, masz możliwość poruszania kursorem myszy po ekranie oraz symulowania kliknięć myszą, naciśnięć klawiszy i kombinacji klawiszy. Moduł `pyautogui` może również sprawdzić kolory na ekranie, co progra-

mowi stosującemu automatyzację GUI może dać wystarczającą ilość informacji dotyczących zawartości ekranu lub o ewentualnym zejściu z właściwej drogi. Modułowi `pyautogui` można nawet przekazać zrzut ekranu i pozwolić na ustalenie współrzędnych obszaru, który ma zostać kliknięty.

Wszystkie funkcje `pyautogui` można łączyć i automatyzować wszelkie nieustannie powtarzane zadania w komputerze. Naprawdę hipnotyczne może być obserwowanie ruchu kursora myszy i tekstu automatycznie pojawiającego się na ekranie. Dlaczego więc miałbyś nie poświęcić chwili czasu na wygodnie zajęcie miejsca przed monitorem i obserwację, jak program wykonuje za Ciebie całą ciężką pracę? Nie ulega wątpliwości, że prawdziwą satysfakcję przynosi obserwacja, jak Twój spryt zaoszczędził Ci konieczności wykonywania żmudnych zadań.

Pytania kontrolne

1. W jaki sposób za pomocą modułu `pyautogui` możesz doprowadzić do bezpiecznej awarii programu?
2. Która funkcja zwraca bieżącą rozdzielczość ekranu?
3. Która funkcja zwraca współrzędne kursora myszy w jego bieżącym położeniu?
4. Jaka jest różnica między funkcjami `pyautogui.moveTo()` i `pyautogui.moveRel()`.
5. Które funkcje mogą być używane do symulacji operacji przeciągania zawartości ekranu myszą?
6. Która funkcji wpisze ciąg tekstowy „Witaj, świecie!”?
7. W jaki sposób możesz symulować naciśnięcia klawiszy specjalnych, takich jak strzałki kursora w lewo i w prawo?
8. W jaki sposób można bieżącą zawartość ekranu zapisać w pliku obrazu o nazwie *screenshot.png*?
9. Jakie polecenie pozwala na zdefiniowanie dwusekundowej przerwy po każdym wywołaniu funkcji modułu `pyautogui`?

Projekty praktyczne

W celu zdobycia doświadczenia utwórz programy wykonujące omówione poniżej zadania.

Symulowanie zajętości

Wiele programów typu komunikator internetowy ustala to, czy jesteś zajęty lub z dala od komputera, przez wykrycie braku ruchu kursorem myszy przez pewien okres czasu, na przykład 10 minut. Być może chcesz się na chwilę oddalić od

biurka, ale jednocześnie nie chcesz, aby inni zobaczyli, że stan Twojego komunikatora internetowego zmienił się na „bezczynny". Utwórz skrypt, który będzie lekko poruszał kursorem myszy co 10 sekund. Zmiana położenia kursora myszy powinna być na tyle mała, aby kursor nie oddalił się zbyt daleko, gdy będziesz musiał ponownie skorzystać z komputera po dłuższej chwili działania skryptu.

Bot komunikatora internetowego

Google Talk, Skype, Yahoo Messenger, AIM oraz inne komunikatory internetowe bardzo często używają protokołów własnościowych, które innym programistom utrudniają opracowanie modułów Pythona pozwalających na interakcję z tego rodzaju programami. Jednak nawet te protokoły własnościowe nie mogą powstrzymać Ciebie od utworzenia programu stosującego automatyzację GUI.

Aplikacja Google Talk ma pasek wyszukiwania pozwalający na wprowadzenie nazwy użytkownika na liście przyjaciół i otworzenie nowego okna wiadomości po naciśnięciu klawisza *Enter*. To nowe okno automatycznie staje się aktywne. Inne aplikacje komunikatorów internetowych mają podobne sposoby otwierania nowych okien wiadomości. Utwórz program, który automatycznie wysyła powiadomienie do grupy osób wybranych z listy Twoich przyjaciół. Tego rodzaju program będzie być może musiał zmierzyć się z wyjątkowymi sytuacjami, takimi jak stan offline przyjaciela, okno czatu wyświetlane w różnych miejscach na ekranie, a także okna dialogowe zakłócające komunikację. Twój program będzie musiał wykonywać zrzuty ekranu i podpowiadać mechanizmowi interakcji z GUI oraz dostosowywać różne sposoby wykrywania, czy wirtualne naciśnięcia klawiszy zostały przekazane.

UWAGA *Być może będziesz musiał utworzyć kilka fikcyjnych kont testowych, aby przypadkiem nie zarzucić przyjaciół spamem podczas pracy nad tym programem.*

Samouczek dotyczący bota grającego w grę

Na stronie *http://inventwithpython.com/blog/2014/12/17/programming-a-bot-to-play-the-sushi-go-round-flash-game/* znajdziesz świetny samouczek zatytułowany „How to Build a Python Bot That Can Play Web Games". Wyjaśniono w nim temat tworzenia w Pythonie programu stosującego automatyzację GUI pozwalającą na granie w grę o nazwie Sushi Go Round, która została napisana we Flashu. Gra polega na klikaniu odpowiednich składników w celu przygotowania sushi zgodnego z zamówieniem klienta. Im szybciej zrealizujesz zamówienia bez błędów, tym więcej punktów zdobędziesz. To jest doskonałe zadanie dla programu stosującego automatyzację GUI i jednocześnie sposób dla oszustwo w grze oraz zdobycie naprawdę dużej ilości punktów! Samouczek prezentuje wiele tych samych tematów, które poruszyłem w tym rozdziale. Ponadto zawiera omówienie oferowanych przez moduł `pyautogui` podstawowych funkcji rozpoznawania obrazu.