Elementy i składnia Pythona

Wiele języków programowania skupia się na rzeczach, które komputer robi i jak to robi, a nie na sposobie myślenia i pracy ludzi. Ten jeden prosty fakt sprawia, że większość języków programowania jest trudna do nauczenia się większości ludzi. Python jest inny, ponieważ opiera się na filozofii, że język programowania powinien być bardziej nastawiony na to, jak ludzie myślą, pracują i komunikują się, niż na to, co dzieje się wewnątrz komputera. Zen w Pythonie jest doskonałym przykładem tej ludzkiej orientacji, więc od tego zaczynamy ten rozdział.

Zen of Python

Zen of Python to lista zasad przewodnich dotyczących projektowania języka Python. Zasady te są w rzeczywistości ukryte w jajku wielkanocnym (slangowe określenie czegoś w języku programowania lub aplikacji, co nie jest łatwe do znalezienia i służy jako trochę wewnętrznego żartu dla ludzi, którzy nauczyli się Pythona na tyle, by móc go znaleźć). Aby dostać się do jajka wielkanocnego, wykonaj następujące kroki:

1. Uruchom VS Code z Anaconda Navigator i otwórz obszar roboczy Pythona 3.

2. Jeśli panel Terminal nie jest otwarty, wybierz opcję Wyświetl terminal z paska menu VS Code.

3. Wpisz python i naciśnij Enter, aby przejść do wiersza poleceń Pythona (>>>).

Jeśli po wprowadzeniu polecenia Pythona pojawi się komunikat o błędzie, nie panikuj. Musisz tylko przypomnieć VS Code, którego interpretera Pythona używasz. Wybierz polecenie Wyświetl paletę poleceń z menu, wpisz python, a następnie kliknij Python: Wybierz Interpreter i wybierz wersję Pythona 3 dołączoną do Anacondy.

4. Wpisz import this i naciśnij Enter.

Pojawia się lista 19 aforyzmów. Być może będziesz musiał przewijać w górę iw dół lub zwiększyć panel terminala, aby zobaczyć je wszystkie. Ale jak widać, aforyzmy są nieco żartobliwe w swojej retoryce filozoficznej. Ale ogólną ideą, którą wyrażają, jest zawsze próba uczynienia kodu bardziej czytelnym dla człowieka niż czytelnym dla komputera.

Zen jest czasami określany jako PEP 20, gdzie PEP jest akronimem propozycji ulepszeń Pythona. 20 prawdopodobnie odnosi się do 20 zasad Zen w Pythonie, z których tylko 19 zostało spisanych. Wszyscy możemy się zastanawiać lub wymyślać własną ostateczną zasadę. Ale to wszystko jest zabawne, więc nie martw się o dwudziesty (nieistniejący) Zen. . . nie będzie na żadnych testach. Istnieje wiele innych PEP-ów, z których wszystkie można znaleźć na stronie Python.org. Ten, o którym prawdopodobnie usłyszysz najczęściej, to PEP 8, który jest przewodnikiem po stylach kodu Pythona. Naczelną zasadą tych wytycznych jest „liczy się czytelność" - a to, co oznaczają, jest czytelne dla ludzi. Trzeba przyznać, że kiedy po raz pierwszy uczysz się kodu Pythona, kod większości innych ludzi będzie wydawał się jakimś bełkotem nabazgranym przez kosmitów i możesz nie mieć pojęcia, co to znaczy lub co robi. Ale gdy zdobędziesz doświadczenie z językiem, spójność stylu stanie się bardziej widoczna, a czytanie i zrozumienie kodu innych osób będzie coraz łatwiejsze, co jest doskonałym sposobem na samodzielne nauczenie się kodowania. W dalszej części tej książki przedstawimy Ci styl kodowania Pythona. Próbuję czytać o tym, zanim zaczniesz nad nim pracować, z pewnością znudzi cię do łez. Tak więc na razie za każdym razem, gdy usłyszysz wzmiankę o PEP, a zwłaszcza PEP 8, pamiętaj, że jest to odniesienie do wytycznych dotyczących stylu kodowania Pythona ze strony Python.org i możesz je znaleźć w dowolnym momencie, po prostu przeglądając PEP 8. Prawdę mówiąc, ten biznes PEP 8 może być rodzajem obosiecznego miecza dla uczniów. Z jednej strony nie chcesz nauczyć się mnóstwa złych

nawyków tylko po to, by odkryć, że musisz się ich później oduczyć. Z drugiej strony wymagania dotyczące formatowania PEP 8 są tak surowe, że wielu uczniów denerwuje się, próbując uruchomić te rzeczy bez martwienia się o puste miejsca, wielkie/małe litery i inne szczegóły. Aby uporać się z tym wszystkim, będziemy śledzić i wyjaśniać konwencje PEP 8 w miarę postępów. Jeśli chcesz, możesz pójść o krok dalej, konfigurując PyLint, aby Ci pomógł. PyLint to narzędzie dostarczane z Anacondą, które podpowiada kod podczas pisania. Wykonaj następujące kroki, aby włączyć Pylint i PEP 8 teraz, jeśli chcesz wziąć je na przejażdżkę:

1. W menu Mac kliknij Kod; w systemie Windows kliknij Plik w menu.

2. Wybierz Preferencje Ustawienia.

3. Kliknij Ustawienia obszaru roboczego.

4. Kliknij trzy kropki w prawym górnym rogu ustawień i wybierz Otwórz Settings.json.

5. Kliknij Ustawienia obszaru roboczego.

6. W polu Ustawienia wyszukiwania wpisz pylint.

7. Dotknij wskaźnikiem myszy python.linting.enabled, a jeśli ta opcja nie jest włączona, kliknij ją i wybierz True, aby włączyć linting.

8. Przewiń w dół i dotknij wskaźnikiem myszy python.linting.pep8Enabled, kliknij małą ikonę ołówka, która się pojawi i kliknij True.

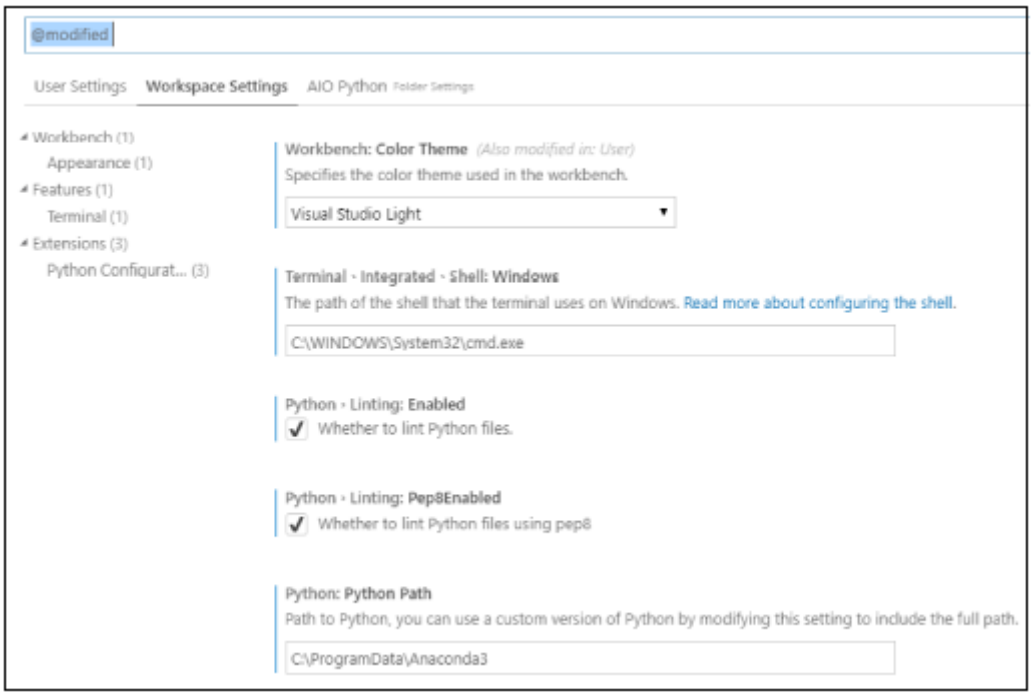
Zobaczysz kilka linii dodanych do ustawień obszaru roboczego, z których każda zaczyna się od python.linting, jak pokazano na rysunku

Kiedy skończysz, wybierz Plik Zapisz wszystko. Następnie możesz zamknąć strony ustawień, klikając X na ich kartach. Jeśli kiedykolwiek w przyszłości poczujesz, że linting jest zbyt duży i utrudnia ci naukę, możesz wyłączyć te funkcje, wykonując następujące czynności:

1. Wybierz Kod -> Preferencje (na komputerze Mac) lub Plik -> Preferencje -> Ustawienia (w Windows)

2. Kliknij Ustawienia obszaru roboczego, a następnie kliknij trzy kropki i wybierz Pokaż zmodyfikowane ustawienia.

Zobaczysz ustawienia Workspace w formacie innym niż kod pokazany na rysunku



3. Jeśli chcesz wyłączyć tylko PEP 8 (który doprowadza większość początkujących do szaleństwa), kliknij znacznik wyboru Python Linting: Pep8Enabled.

Wypróbuj rzeczy z wyłączonym PEP 8. Jeśli linting nadal wydaje się być zbyt duży, możesz powtórzyć kroki i usunąć zaznaczenie obok Python -> Linting:Enabled.

Programowanie obiektowe

Ryzykując zbytnią technikę/informatykę, powinniśmy wspomnieć, że istnieją różne rodzaje języków i różne podejścia do projektowania języków. Być może najbardziej udanym i powszechnie używanym modelem jest tak zwane programowanie obiektowe lub w skrócie OOP. Jest to filozofia projektowania, która próbuje naśladować świat rzeczywisty w tym sensie, że składa się on z obiektów, z właściwościami i metodami (działaniami), które te obiekty wykonują. Weźmy na przykład samochód. Każdy samochód jest przedmiotem. Nie wszystkie samochody są dokładnie takie same. Różne samochody mają różne właściwości, takie jak marka, model, rok, kolor, rozmiar itd., które różnią się od siebie. A jednak wszystkie służą temu samemu podstawowemu celowi. . . aby przewieźć nas z punktu A do punktu B bez konieczności chodzenia lub korzystania z innego środka transportu. Wszystkie samochody mają pewne wspólne metody (rzeczy, które mogą zrobić). Możemy nimi jeździć, sterować, przyspieszać, spowalniać, kontrolować temperaturę wewnątrz i więcej, używając elementów sterujących w samochodzie, którymi możemy manipulować rękami. Obiekt w obiektowym języku programowania nie jest rzeczą fizyczną, jak samochód, ponieważ istnieje tylko wewnątrz komputera. Możesz jednak mieć klasę (którą możesz traktować jako twórcę obiektów, na przykład fabryka samochodów), która może produkować wiele różnych rodzajów obiektów (samochodów) do różnych celów (sportowych, terenowych, sedanów). Wszystkimi tymi obiektami można sterować za pomocą wspólnych elementów sterujących, podobnie jak wszystkimi samochodami steruje się za pomocą kierownicy, hamulców, przyspieszenia i zmiany biegów. Python jest w dużym stopniu językiem zorientowanym obiektowo. Może to nie być od razu widoczne, gdy zaczynasz naukę, ponieważ rdzeń języka składa się z „kontroli” (w postaci słów), które pozwalają ci kontrolować różne rodzaje obiektów. Musisz najpierw nauczyć się podstawowego języka, aby kiedy będziesz gotowy do używania obiektów innych ludzi, wiedziałeś, jak to zrobić. To trochę tak, że kiedy już wiesz, jak jeździć jednym

samochodem, prawie wiesz, jak jeździć wszystkimi. Nie musisz martwić się wynajmem samochodu tylko po to, aby odkryć, że pedał gazu jest na dachu, kierownica na podłodze i musisz używać poleceń głosowych zamiast hamulca, aby go spowolnić. Podstawowa umiejętność „jazdy” dotyczy wszystkich samochodów.

Liczenie wcięć, wielki czas

Jeśli chodzi o podstawowy styl pisania kodu, jedyną rzeczą, która naprawdę odróżnia Pythona od innych języków, jest to, że używa wcięć zamiast nawiasów i nawiasów klamrowych, aby wskazać „bloki” lub „kawałki” kodu. Nie zakładamy, że znasz inne języki, więc nie martw się, jeśli to nic dla ciebie nie znaczy. Ale jeśli zdarzy ci się trochę zaznajomić z językiem takim jak JavaScript, wiesz, że jest sporo kłótni z nawiasami i takimi, aby kontrolować „co jest w czym”. Na przykład oto kod JavaScript. Jeśli znasz zabawkę Magic 8 Ball, możesz mieć pojęcie o tym, co robi ten program. Ale nie to jest ważne. Zauważ tylko, że jest tam dużo nawiasów, nawiasów klamrowych i średników:

```
document.addEventListener("DOMContentLoaded", function () {var question =

prompt("Ask magic 8 ball a question");var answer = Math.floor(Math.random()

* 8) 1; if (answer == 1) {alert("It is certain");} else if (answer == 2)

{alert("Outlook good");} else if (answer == 3) {alert("You may rely on

it");} else if (answer == 4) {alert("Ask again later");} else if (answer ==

5) {alert("Concentrate and ask again");} else if (answer == 6) {alert

("Reply hazy, try again");} else if (answer == 7) {alert("My reply is

no");} else if (answer == 8) {alert("My sources say no")} else {alert

("That's not a question");}alert("The end");})
```

Ten kod jest bałaganem i nie jest przyjemny do czytania. Możemy nieco ułatwić czytanie, dzieląc go na wiele wierszy i wcinając niektóre z nich. Nie jest to wymagane w JavaScript, ale można to zrobić, aby kod był trochę łatwiejszy do odczytania przez człowieka, jak pokazano poniżej:

```
document.addEventListener("DOMContentLoaded", funkcja () {

var question = prompt("Zadaj pytanie magicznej kuli 8");

var odpowiedź = Math.floor(Math.random() * 8) 1;

jeśli (odpowiedź == 1) {

alert("To pewne");

} inaczej jeśli (odpowiedź == 2) {

alert("Wygląd dobry");

} inaczej jeśli (odpowiedź == 3) {

alert("Możesz na tym polegać");

} inaczej jeśli (odpowiedź == 4) {

alert("Zapytaj ponownie później");
```

```
} inaczej jeśli (odpowiedź == 5) {

alert("Skoncentruj się i zapytaj ponownie");

} inaczej if (odpowiedź == 6) {

alert("Odpowiedz zamglona, spróbuj ponownie");

} inaczej jeśli (odpowiedź == 7) {

alert("Moja odpowiedź brzmi nie");

} else if (answer == 8) {

alert("My sources say no")

} else {

alert("That's not a question");}

alert("The end");

})
```

W JavaScript wymagane są wszystkie nawiasy i nawiasy klamrowe, ponieważ określają one, gdzie zaczynają się i kończą fragmenty kodu. Wcięcia dla czytelności są opcjonalne. W Pythonie jest zupełnie odwrotnie, ponieważ Python nie używa nawiasów klamrowych ani żadnych innych znaków specjalnych do oznaczania początków i końców bloków kodu. Zaznaczają je same wcięcia. Tak więc te wcięcia nie są wcale opcjonalne - w rzeczywistości są wymagane i mają znaczny wpływ na działanie kodu. Ale zaletą jest to, że kiedy czytasz kod (jako człowiek, nie jako komputer), stosunkowo łatwo jest zobaczyć, co się dzieje, i nie rozprasza cię tona dodatkowych cudzysłowów. Oto ten kod JavaScript napisany w Pythonie:

```
import random

question = input("Ask magic 8 ball a question")

answer = random.randint(1,8)

if answer == 1:

print("It is certain")

elif answer == 2:

print("Outlook good")

elif answer == 3:

print("You may rely on it")

elif answer == 4:

print("Ask again later")

elif answer == 5:

print("Concentrate and ask again")

elif answer == 6:
```

```
print("Reply hazy, try again")

elif answer == 7:

print ("My reply is no")

elif answer == 8:

print ("My sources say no")

else:

print("That's not a question")

print ("The end"
```

Możesz zauważyć, że na górze kodu Pythona znajduje się linia, która zaczyna się od słowa import. Są one bardzo powszechne w Pythonie, a w następnej sekcji dowiesz się dlaczego.

Korzystanie z modułów Pythona

Jednym z sekretów sukcesu Pythona jest to, że składa się ze stosunkowo prostego, czystego, podstawowego języka. To jest część, której naprawdę musisz się najpierw nauczyć. Oprócz tego podstawowego języka istnieje wiele, wiele modułów, które możesz pobrać za darmo i uzyskać dostęp z własnego kodu. Te moduły są również napisane w języku rdzenia, ale tak naprawdę nie musisz tego widzieć ani nawet o tym wiedzieć, ponieważ możesz uzyskać dostęp do całej mocy modułów z podstawowego języka rdzenia. Większość modułów jest przeznaczona do określonych zastosowań, takich jak nauka, liczby, sztuczna inteligencja lub praca z datami i godzinami lub . . . cokolwiek. Piękno tego polega na tym, że ktoś inny (prawdopodobnie wiele osób) spędził dużo czasu na tworzeniu, testowaniu i dostrajaniu tego modułu, abyś nie musiał. Wystarczy zaimportować dowolne moduły do własnego pliku .py i wykorzystać możliwości modułów zgodnie z instrukcjami w dokumentacji dostępnej dla każdego modułu. Poprzedni przykładowy program Magic 8 Ball zaczyna się od tego wiersza:

```
import random
```

Jak się okazuje, rdzeń języka Python nie ma nic wbudowanego, aby wygenerować liczbę losową. Chociaż prawdopodobnie moglibyśmy wymyślić jakiś sposób na jego utworzenie, nie ma takiej potrzeby, ponieważ ktoś inny już wymyślił, jak to zrobić i udostępnił kod za darmo. Uruchomienie naszego programu z import random mówi programowi, że chcemy wykorzystać możliwości tego modułu do wygenerowania liczby losowej. Następnie w dalszej części programu generujemy losową liczbę od 1 do 8 w tym konkretnym wierszu kodu:

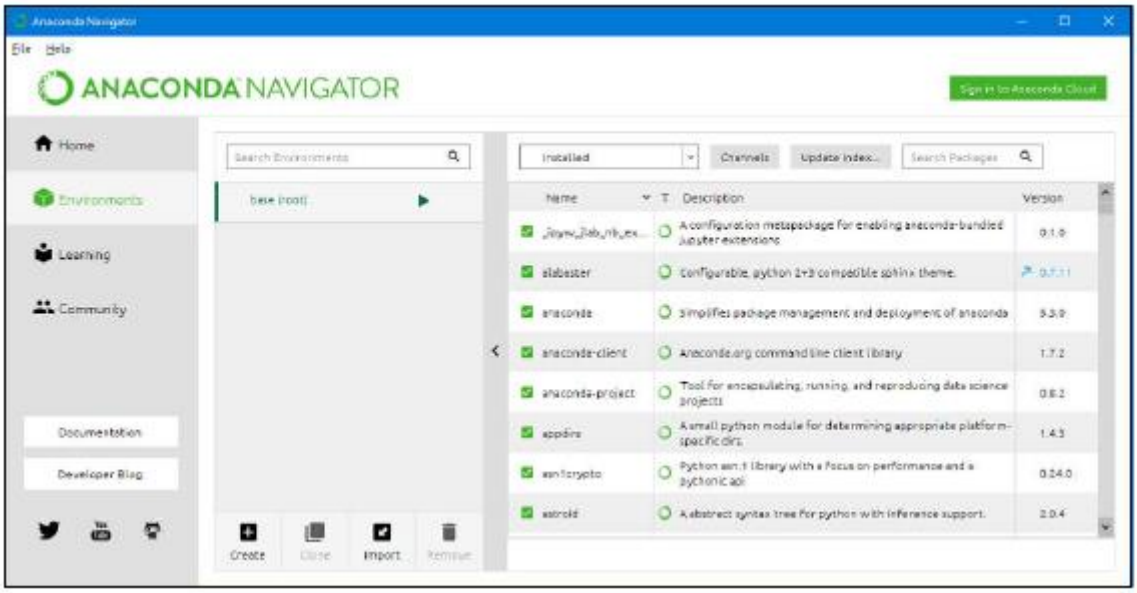
```
answer = random.randint(1,8)
```

Korzystanie z istniejącego modułu oszczędza mi konieczności ponownego wymyślania koła, próbując wymyślić, jak zrobić losową liczbę z języka rdzenia. Istnieją dosłownie setki darmowych modułów dla Pythona, co oznacza, że prawie nigdy nie musisz wymyślać żadnych kół. Musisz tylko wiedzieć, który moduł zaimportować do swojego programu. Teraz możesz się zastanawiać, gdzie znajdują się wszystkie moduły i gdzie można je zdobyć. Cóż, szczerze mówiąc, są wszędzie w Internecie. Ale są szanse, że nigdy nie będziesz musiał go pobierać, ponieważ masz już wszystkie najczęściej używane moduły na świecie. Masz je, ponieważ pojawiły się wraz z Anacondą, kiedy je pobrałeś i zainstalowałeś. Aby się przekonać, wykonaj następujące kroki:

1. Otwórz Anacondę w zwykły sposób na swoim komputerze.

2. Kliknij Środowiska w lewej kolumnie.

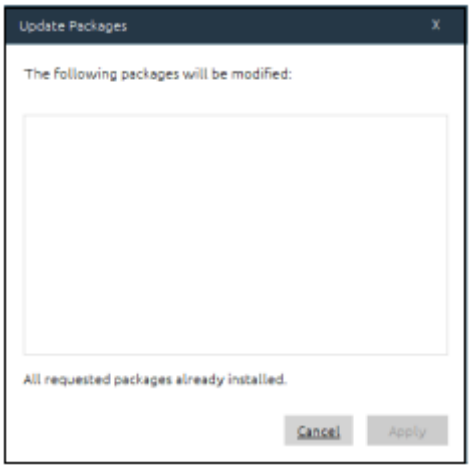
Te rzeczy, które widzisz po prawej, to w rzeczywistości moduły Pythona, które są już zainstalowane na twoim komputerze i gotowe do zaimportowania i użycia w razie potrzeby.



Przewijając nazwy, zobaczysz, że masz ich już mnóstwo. W prawej kolumnie zobaczysz nawet, którą wersję modułu posiadasz.

Możesz również zauważyć, że niektóre numery wersji w prawej kolumnie są kolorowe i pokazują strzałki. Reprezentują one moduły, które mogą mieć nowszą wersję dostępną do pobrania. Podobnie jak w przypadku języków programowania, moduły ewoluują w czasie i wersji, w miarę jak ich autorzy ulepszają rzeczy i dodają nowe możliwości. Nie musisz jednak zawsze mieć najnowszą wersję. Jeśli twoja wersja działa, możesz się tego trzymać.

Jedną z wielu fajnych rzeczy w Anacondzie jest to, że aby uzyskać najnowszą wersję, nie musisz wykonywać żadnych dziwnych poleceń pip, jak zaleca wiele starszych samouczków Pythona. Zamiast tego wystarczy kliknąć strzałkę lub numer wersji modułu, który chcesz pobrać. W rzeczywistości możesz kliknąć tyle, ile chcesz. Następnie kliknij Zastosuj w prawym dolnym rogu. Anaconda wykonuje całą brudną robotę polegającą na znalezieniu bieżącego modułu, ustaleniu, czy rzeczywiście jest dostępna nowsza wersja, a następnie pobraniu tej wersji, jeśli jest dostępna. Po zakończeniu wszystkich pobierania zobaczysz okno dialogowe, takie jak pokazane na rysunku



Jeśli na liście nie ma żadnych nazw, oznacza to, że wszystkie wybrane moduły są w rzeczywistości aktualne, więc możesz po prostu kliknąć Anuluj, a następnie kliknąć Strona główna w lewym okienku, aby powrócić do strony głównej Anacondy. Jeśli z drugiej strony widzisz niektóre nazwy wymienione w obszarze Następujące pakiety zostaną zmodyfikowane, kliknij Zastosuj, aby zainstalować najnowsze wersje.

Składnia importowania modułów

Jak już wspomnieliśmy, we własnym kodzie Pythona musisz zaimportować moduł, zanim będziesz mógł uzyskać dostęp do jego możliwości. Składnia do tego jest

import nazwa modułu [jako alias]

Kiedy widzisz taki wykres, zawsze pamiętaj o czterech rzeczach:

* W kodzie rozróżniana jest wielkość liter, co oznacza, że należy wpisać import i używać wszystkich małych liter, jak pokazano. Nie zadziała, jeśli użyjesz wielkich liter.

* Wszystko pisane kursywą to miejsce na konkretne informacje, które podasz we własnym kodzie. Na przykład dostępne są dziesiątki modułów. W swoim kodzie musisz zastąpić nazwa_modułu dokładną nazwą modułu, który chcesz zaimportować.

* Wszystko w nawiasach kwadratowych jest opcjonalne, co oznacza, że możesz wpisać polecenie z tą częścią lub bez niej.

* Nigdy nie wpisujesz nawiasów kwadratowych w swoim kodzie, ponieważ nie są one częścią języka Python. Są używane tylko do wskazania opcjonalnych części składni.

Możesz wpisać import w dowolnym miejscu, w którym wpisujesz kod Pythona: w wierszu poleceń Pythona (>>>), w pliku .py lub w notatniku Jupyter. W pliku .py należy zawsze umieszczać instrukcje importu jako pierwsze, aby ich możliwości były dostępne dla reszty kodu.

Używanie aliasu z modułami

Jak wspomniano w poprzedniej sekcji, możesz przypisać alias, „pseudonim” do dowolnego importowanego modułu, po prostu postępując po nazwie modułu spacją, słowem as, a następnie wybraną przez siebie nazwą. Zwykle jest to krótka nazwa, którą łatwo wpisać i zapamiętać, więc nie musisz wpisywać długiej nazwy za każdym razem, gdy chcesz uzyskać dostęp do możliwości modułu. Na przykład, zamiast wpisywać import random, aby zaimportować ten moduł, możemy go zaimportować i nadać mu pseudonim typu rnd, który jest krótszy:

```
import random as rnd
```

Następnie w kolejnym kodzie nie używałbyś pełnej nazwy, losowej, w odniesieniu do modułu. Zamiast tego użyjesz krótkiej nazwy rnd, jak w poniższym przykładzie:

```
answer = rnd.randint(1,8)
```

W tym krótkim przykładzie może to nie wydawać się wielkim problemem. Ale możesz natknąć się na moduły, które mają długie nazwy, a Twój kod wymaga odwoływania się do tego modułu w wielu różnych miejscach. Mając dostępną nazwę aliasu, możesz wpisać tylko tę krótką nazwę, nie musisz wpisywać pełnej nazwy. W poprzednich rozdziałach obiecaliśmy wiele praktycznych zadań w tej książce. Opowiedzenie ci wszystkich tych faktów bez wielu praktycznych ćwiczeń praktycznych może wydawać się oszukiwanie. Zrobiliśmy to jednak, ponieważ te fragmenty wiedzy podstawowej pomogą ci zrozumieć rzeczy podczas nauki Pythona i powinny pomóc w zobaczeniu dużego obrazu i zapamiętywaniu rzeczy. Ale teraz, bez zbędnych ceregieli, nadszedł czas, aby naprawdę zastosować wszystko, czego się do tej pory nauczyłeś i zacząć brudzić sobie ręce prawdziwym kodem Pythona.