Żonglowanie danymi JSON

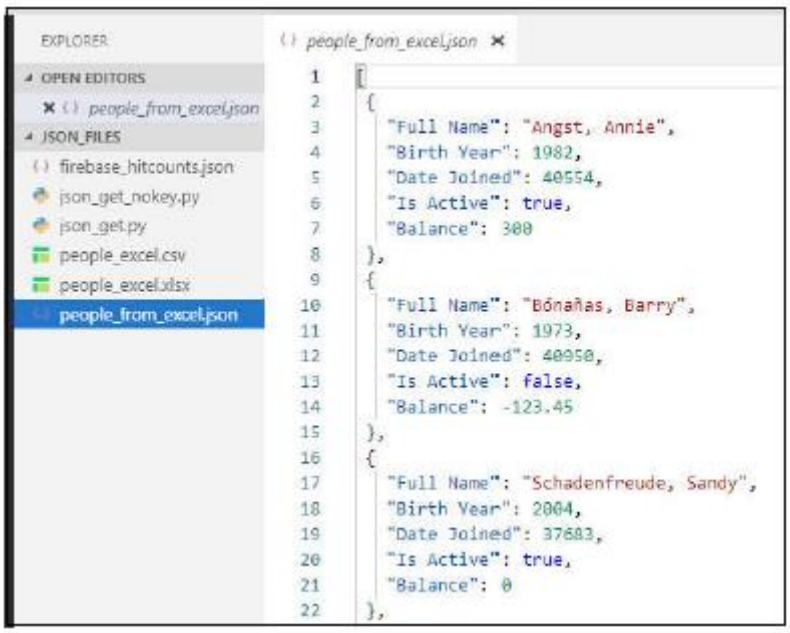
JSON (JavaScript Object Notation) to popularny format organizowania danych obiektowych. Ten termin, format marshalling, ogólnie oznacza format używany do przesyłania danych z jednego komputera do drugiego. Jednak niektóre bazy danych, takie jak bezpłatna baza danych czasu rzeczywistego w Google Firebase, faktycznie przechowują dane również w formacie JSON. Nazwa JavaScript na początku czasami trochę zniechęca ludzi, zwłaszcza gdy do pisania kodu używasz Pythona, a nie JavaScript. Ale nie martw się o to. Format właśnie rozpoczął swoją działalność w świecie JavaScript. Jest to obecnie szeroko znany format ogólnego przeznaczenia używany we wszystkich rodzajach komputerów i języków programowania. W tej części dowiesz się dokładnie, czym jest JSON, a także jak eksportować i importować dane do i z JSON. Jeśli stwierdzisz, że wszystkie modne słowa otaczające JSON sprawiają, że czujesz się niekomfortowo, nie martw się. Najpierw przejdziemy przez cały żargon. Jak zobaczysz, dane JSON są sformatowane prawie tak samo, jak słowniki danych Pythona. Więcnie będzie dużej ilości nowych rzeczy do nauczenia się. Ponadto masz już darmowy moduł Python JSON, który jeszcze bardziej ułatwia pracę z danymi JSON.

Organizowanie danych JSON

Dane JSON są mniej więcej odpowiednikiem słownika danych w Pythonie, dzięki czemu praca z plikami JSON jest dość łatwa. Prawdopodobnie najłatwiej to zrozumieć, porównując je z danymi tabelarycznymi. Na przykład na rysunku przedstawiono niektóre dane tabelaryczne w arkuszu programu Excel.

	A	B	C	D	E
1	**Full Name**	**Birth Year**	**Date Joined**	**Is Active**	**Balance**
2	Angst, Annie	1982	1/11/2011	TRUE	$300.00
3	Bónañas, Barry	1973	2/11/2012	FALSE	-$123.45
4	Schadenfreude, Sandy	2004	3/3/2003	TRUE	$0.00
5	Weltschmerz, Wanda	1995	4/24/1994	FALSE	$999,999.99
6	Malaise, Mindy	2006	5/5/2005	TRUE	$454.01
7	O'Possum, Ollie	1987	7/27/1997	TRUE	-$1,000.00
8					
9	Pusillanimity, Pamela	1979	8/8/2008	TRUE	$12,345.67

Rysunek przedstawia te same dane przekonwertowane do formatu JSON.



Każdy wiersz danych w arkuszu Excela został po prostu przekonwertowany na słownik par klucz:wartość w pliku JSON. I oczywiście jest wiele nawiasów klamrowych wskazujących, że są to dane słownikowe. To jeden ze sposobów tworzenia pliku JSON. Możesz także utworzyć plik JSON z kluczem, w którym każdy fragment danych ma pojedynczy klucz, który jednoznacznie go identyfikuje (żaden inny słownik w tym samym pliku nie może mieć tego samego klucza). Kluczem może być liczba lub tekst; tak naprawdę nie ma znaczenia, o ile jest unikalny dla każdego elementu. Kiedy pobierasz pliki JSON utworzone przez kogoś innego, nie jest niczym niezwykłym, że plik jest zaszyfrowany. Na przykład w osobistej witrynie Alana używa bezpłatnej bazy danych czasu rzeczywistego Google Firebase do liczenia odwiedzin na stronę i innych informacji o każdej stronie. Ta baza danych czasu rzeczywistego przechowuje dane, jak pokazano na rysunku.

Te dziwne rzeczy, które wyglądają jak -LAOqOxg6kmP4jhnjQXS, to wszystkie klucze, które Firebase generuje automatycznie dla każdego elementu danych, aby zagwarantować niepowtarzalność. Znak obok każdego klucza umożliwia rozwijanie i zwijanie informacji pod każdym kluczem.

KONWERSJA EXCELA DO JSONA

Jeśli zastanawiasz się, jak przekonwertować ten przykładowy arkusz kalkulacyjny Excel na format JSON, ustaw przeglądarkę na www.convertcsv.com/csv-to-json.htm i wykonaj następujące kroki:

1. W kroku 1 otwórz kartę Wybierz plik, ustaw kodowanie na UTF-8, kliknij przycisk Przeglądaj, wybierz plik programu Excel i kliknij Otwórz.

2. W kroku 2 upewnij się, że opcja Pierwszy wiersz to nazwy kolumn jest zaznaczona i ustaw opcję Pomiń liczbę linii na 1, aby pominąć wiersz nagłówków kolumn.

3. W kroku 5 kliknij przycisk CSV do JSON.

4. Obok opcji Zapisz swój wynik wpisz nazwę pliku, a następnie kliknij przycisk Pobierz wynik.

Plik powinien znaleźć się w folderze Pobrane (lub w innym miejscu, z którego normalnie pobierasz) z rozszerzeniem .json. Jest to zwykły plik tekstowy, więc można go otworzyć w dowolnym edytorze tekstu lub edytorze kodu, takim jak VS Code. Konwerter automatycznie pomija puste wiersze w plikach Excela, dzięki czemu Twój plik JSON nie będzie zawierał żadnych danych dla pustych wierszy w arkuszu kalkulacyjnym. Jeśli często pracujesz z plikami Excel, CSV, JSON i podobnymi typami danych, być może zechcesz poświęcić trochę czasu na zapoznanie się z wieloma narzędziami i możliwościami tej witryny http://www.convertcsv.com/. Jak widać na obrazku, Firebase ma również opcję Eksportuj JSON, która pobiera dane do pliku JSON na Twój komputer. Zrobiliśmy to. Rysunek pokazuje, jak wyglądają dane w tym pobranym pliku.

Można stwierdzić, że jest to plik JSON z kluczem, ponieważ każdy fragment danych jest poprzedzony unikalnym kluczem, takim jak -LAOqAyxxHrPw6pGXBMZ, po którym następuje dwukropek. W Pythonie możesz pracować zarówno z plikami JSON z kluczem, jak i bez klucza. Niektórzy czytelnicy mogli zauważyć, że pole Date Joined w pliku JSON nie wygląda jak zwykła data mm/dd/rrrr. Pole ostatniej wizyty z bazy danych Firebase to data/godzina, mimo że nie wygląda na datę ani godzinę. Ale nie martw

się o to. W dalszej części dowiesz się, jak przekonwertować te dziwnie wyglądające daty seryjne (jak się je nazywa) na format czytelny dla człowieka.

Zrozumienie serializacji

Jeśli chodzi o JSON, pierwszym modnym słowem, którego musisz się nauczyć, jest serializacja. Serializacja to proces przekształcania obiektu (takiego jak słownik Pythona) w strumień bajtów (znaków), które można przesyłać przewodowo, przechowywać w pliku lub bazie danych albo przechowywać w pamięci. Głównym celem jest zapisanie wszystkich informacji zawartych w obiekcie w sposób, który można łatwo odzyskać na dowolnym innym komputerze. Proces przekształcania go z powrotem w obiekt nazywa się deserializacją. Aby uprościć sprawę, możesz po prostu rozważyć użycie następujących definicji:

Serializacja: Konwertuj obiekt na łańcuch.

Deserializacja: Konwertuj ciąg znaków na obiekt.

Standardowa biblioteka Pythona zawiera moduł json, który pomaga pracować z plikami JSON. Ponieważ jest to część standardowej biblioteki, wystarczy umieścić import json w górnej części kodu, aby uzyskać dostęp do jego możliwości. Cztery główne metody serializacji i deserializacji JSON zostały podsumowane w tabeli

Metoda: Cel

json.dump() : Zapisuje (serializuje) dane Pythona do pliku JSON (lub strumienia).

json.dumps() : Zapisuje (serializuje) obiekt Pythona do łańcucha JSON.

json.load() : Ładuje (deserializuje) JSON z pliku lub podobnego obiektu.

json.loads(): ładuje (deserializuje) dane JSON z łańcucha.

Typy danych w JSON są nieco podobne do typów danych w Pythonie, ale nie są dokładnie takie same. Tabela 2 zawiera listę sposobów konwertowania typów danych między dwoma językami podczas serializacji i deserializacji.

Python : JSON

dict : object

list, tuple : array

str : string

int and foat : number

True : true

False : false

None : null

Ładowanie danych z plików JSON

Aby załadować dane z plików JSON, upewnij się, że zaimportowałeś plik json w górnej części kodu. Następnie możesz użyć zwykłej metody open() pliku, aby otworzyć plik. Podobnie jak w przypadku innych rodzajów plików, możesz dodać encoding = "utf-8", jeśli uważasz, że dane zawierają obce znaki,

które należy zachować. Możesz także użyć newline="", aby uniknąć wstawiania znaku nowej linii na końcu każdego wiersza, który tak naprawdę nie jest częścią danych. To tylko ukryty znak kończący linię podczas wyświetlania danych na ekranie. Aby załadować dane JSON do Pythona, wymyśl nazwę zmiennej do przechowywania danych (użyjemy ludzi), a następnie użyj json.load() do załadowania zawartości pliku do zmiennej, jak poniżej:

```
import json

# This is the Excel data (no keys)

filename = 'people_from_excel.json'

# Open the file (standard file open stuff)

with open(filename, 'r', encoding='utf-8', newline='') as f:

# Load the whole json file into an object named people

people = json.load(f)
```

Uruchomienie tego kodu nie wyświetla niczego na ekranie. Obiekt people można jednak eksplorować na wiele sposobów, korzystając z instrukcji print() bez wcięć poniżej tego ostatniego wiersza. Na przykład to

```
print(people)
```

. . . wyświetla wszystko, co znajduje się w zmiennej p. Na wyjściu widać, że zaczyna się i kończy nawiasami kwadratowymi ([]), co oznacza, że ludzie to lista. Aby to zweryfikować, możesz uruchomić ten wiersz kodu:

```
print(type(people))
```

Gdy to zrobisz, Python wyświetli <class 'list'>, co oznacza, że obiekt jest instancją klasy list. Innymi słowy, jest to obiekt listy, chociaż większość ludzi nazwałaby go po prostu listą.

```
<class 'list'>
```

Ponieważ jest to lista, możesz przeglądać ją w pętli. W pętli możesz wyświetlić typ każdego elementu, na przykład:

```
for p in people:

print(type(p))
```

Wyjście z tego to:

```
<class 'dict'>

<class 'dict'>

<class 'dict'>

<class 'dict'>

<class 'dict'>

<class 'dict'>

<class 'dict'>
```

To przydatna informacja, ponieważ informuje, że każda z „ludzi” (którą w tym kodzie oznaczyliśmy skrótem p) jest słownikiem Pythona. Tak więc w pętli możesz wyizolować każdą wartość według jej klucza. Na przykład spójrz na ten kod:

for p in people:

print(p['Full Name'], p['Birth Year'], p['Date Joined'], p['Is Active'],

p['Balance'])

Uruchomienie tego kodu powoduje wyświetlenie wszystkich danych w pliku JSON, jak pokazano poniżej. Dane te pochodziły z arkusza kalkulacyjnego programu Excel pokazanego na rysunku 1.

Angst, Annie 1982 40554 True 300

Bónañas, Barry 1973 40950 False -123.45

Schadenfreude, Sandy 2004 37683 True 0

Weltschmerz, Wanda 1995 34448 False 999999.99

Malaise, Mindy 2006 38477 True 454.01

O'Possum, Ollie 1987 35638 True -1000

Pusillanimity, Pamela 1979 39668 True 12345.67

Konwertowanie daty programu Excel na datę JSON

Być może myślisz „Hej, czekaj. . . o co chodzi z tymi numerami 40554, 40950, 37683 w kolumnie Data dołączenia?” Cóż, to są daty seryjne, ale z pewnością możesz je przekonwertować na daty Pythona. Będziesz musiał zaimportować moduły xlrd (czytnik programu Excel) i datetime. Następnie, aby przekonwertować tę liczbę całkowitą w kolumnie p['Date Joined'] na datę Pythona, użyj tego kodu:

y, m, d, h, i, s = xlrd.xldate_as_tuple(p['Date Joined'],0)

joined = dt.date(y, m, d)

Aby wyświetlić tę datę w znanym formacie, możesz użyć ciągu f takiego jak ten:

print(f"{joined:%m/%d/%Y}")

Oto cały kod, w tym niezbędne importy na górze pliku:

import json, xlrd

import datetime as dt

This is the Excel data (no keys)

filename = 'people_from_excel.json'

Open the file (standard file open stuff)

with open(filename, 'r', encoding='utf-8', newline='') as f:

Load the whole json file into an object named people

people = json.load(f)

```
# Dictionaries are in a list, loop through and display each dictionary.

for p in people:

name=p['Full Name']

byear = p['Birth Year']

# Excel date pretty tricky, use xlrd module.

y, m, d, h, i, s = xlrd.xldate_as_tuple(p['Date Joined'],0)

joined = dt.date(y, m, d)

balance = '$' f"{p['Balance']:,.2f}"

print(f"{name:<22} {byear} {joined:%m/%d/%Y} {balance:>12}")
```

Oto dane wyjściowe tego kodu, które, jak widać, są dość starannie sformatowane i bardziej przypominają oryginalne dane programu Excel niż dane JSON. Jeśli chcesz wyświetlić dane w formacie dd/mm/rrrr, wystarczy zmienić wzór w ostatnim wierszu na %d/%m/%Y.

```
Angst, Annie 1982 01/11/2011 $300.00

Bónañas, Barry 1973 02/11/2012 $-123.45

Schadenfreude, Sandy 2004 03/03/2003 $0.00

Weltschmerz, Wanda 1995 04/24/1994 $999,999.99

Malaise, Mindy 2006 05/05/2005 $454.01

O'Possum, Ollie 1987 07/27/1997 $-1,000.00

Pusillanimity, Pamela 1979 08/08/2008 $12,345.67
```

Zapętlanie pliku JSON z kluczem

Otwieranie i ładowanie pliku JSON z kluczem jest takie samo, jak otwieranie pliku bez klucza. Jednak po załadowaniu dane są zwykle pojedynczym słownikiem, a nie listą słowników. Na przykład oto kod otwierający i ładujący dane, które wyeksportowaliśmy z Firebase. Te dane zawierają liczbę odwiedzin stron w witrynie, w tym nazwę strony, liczbę dotychczasowych odwiedzin, ostatnią odesłaną (ostatnią stronę, która wysłała kogoś na tę stronę) oraz datę i godzinę ostatniej wizyty. Jak widać, kod do otwierania i ładowania danych JSON jest w zasadzie taki sam. Dane JSON ładują się do obiektu, który nazwaliśmy hits:

```
import json

import datetime as dt

# This is the Firebase JSON data (keyed).

filename = 'firebase_hitcounts.json'

# Open the file (standard file open stuff).

with open(filename, 'r', encoding='utf-8', newline='') as f:

# Load the whole json file into an object named people
```

```
hits = json.load(f)

print(type(hits))
```

Po uruchomieniu tego kodu ostatnia linia wyświetla typ danych obiektu hits, do którego załadowano dane JSON, jako <class 'dict'>. Oznacza to, że obiekt hits jest jednym dużym słownikiem, a nie listą poszczególnych słowników. Możesz przechodzić przez ten słownik, używając prostej pętli, tak jak zrobiliśmy to w przypadku pliku JSON bez klucza, na przykład:

```
for p in hits:

print(p)
```

Rezultatem tego jest jednak to, że nie widzisz zbyt wielu danych. W rzeczywistości wszystko, co widzisz, to klucz do każdego podsłownika zawartego w większym słowniku trafień:

-LAOqAyxxHrPw6pGXBMZ

-LAOqOxg6kmP4jhnjQXS

-LAOrwciIQJZvuCAcyLO

-LAOs2nsVVxbjAwxUXxE

-LAOwqJsjfuoQx8WISlX

-LAQ7ShbQPqOANbDmm3O

-LAQrS6avlv0PuJGNm6P

-LI0iPwZ7nu3IUgiQORH

-LI2DFNAxVnT-cxYzWR

To nie jest błąd ani problem. Tak to działa ze słownikami zagnieżdżonymi. Ale nie martw się, dostęp do danych w każdym słowniku jest dość łatwy. Możesz na przykład użyć dwóch zmiennych pętli, które nazwiemy k (dla klucza) i v (dla wartości), aby wykonać pętlę hits.items(), jak poniżej:

```
for k, v in hits.items():

print(k,v)
```

Daje to inny widok, w którym widzisz każdy klucz, po którym następuje słownik dla tego klucza ujęty w nawiasy klamrowe (nawiasy klamrowe informują, że dane wewnątrz znajdują się w słowniku). Rysunek przedstawia wynik tego działania.

Wartości dla każdego podsłownika znajdują się w obiekcie v tej pętli. Jeśli chcesz uzyskać dostęp do poszczególnych elementów danych, użyj v, po którym następuje para nawiasów kwadratowych z nazwą klucza (dla pola) w środku. Na przykład v['count'] zawiera wszystko, co jest zapisane jako liczba: w danym wierszu. Spójrz na ten kod, w którym nawet nie zawracamy sobie głowy wyświetleniem klucza:

for k, v in hits.items():

Store items in variables.

key = k

hits = v['count']

last_visit=v['lastvisit']

page = v['page']

came_from=v['lastreferrer']

print(f"{hits} {last_visit} {page:<28} {came_from}")

Wynikiem tego są dane z każdego słownika, sformatowane w sposób nieco łatwiejszy do odczytania, jak pokazano na rysunku.

```
9061 1545316328750 /etg/downloadpdf.html          https://difference-engine.com/Courses/tml-5-1118/
3896 1545312678263 /                              http://m.facebook.com
3342 1545311601815 /html_css/index.html           https://alansimpson.me/
2220 1545280814480 /html_css/codequickies/        http://alansimpson.me/html_css/codequickies/dropdownmenu.html
2194 1545308609977 /index.html                    https://alansimpson.me/firebase/hitcounter/
1154 1544974827730 /javascript/code_quickies/     https://alansimpson.me/javascript/code_quickies/clickdropdown/
1547 1545305365597 /how/
1439 1545315903301 /datascience/beginner/         https://www.youtube.com/
1643 1545226502089 /datascience/cheatsheets/
```

Możesz zauważyć, że napotkaliśmy kolejną dziwną sytuację z kolumną lastvisit. Data pojawia się w formacie 545316328750 zamiast bardziej znanego formatu mm/dd/rrrr. Tym razem nie możemy winić Excela, ponieważ te daty nigdy nie były w Excelu. To, co widzisz tutaj, to znacznik czasu Firebase, kiedy element danych został ostatnio zapisany w bazie danych. Ta data jest wyrażona jako data UTC, w tym czas do nanosekundy. Dlatego numer jest taki długi. Oczywiście, jeśli potrzebujesz, aby ludzie byli w stanie zrozumieć te daty, musisz przetłumaczyć je na daty Pythona, co omówimy dalej.

Konwertowanie znaczników czasu Firebase na daty Pythona

Jak zawsze, pierwszą rzeczą, którą musisz zrobić podczas pracy z datami i godzinami w aplikacji Pythona, jest upewnienie się, że zaimportowałeś moduł datetime, co zwykle robimy przy użyciu kodu import datetime as dt, w którym dt jest opcjonalny alias (pseudonim łatwiejszy do wpisania niż imię i nazwisko). Ponieważ wiemy, że data/godzina Firebase jest oparta na UTC, wiemy, że możemy użyć metody datetime .utcfromtimestamp() do przekonwertowania jej na czas Pythona. Ale jest w tym haczyk. Jeśli postępowałeś ściśle według dokumentacji, spodziewałbyś się, że to zadziała:

last_visit = dt.datetime.utcfromtimestamp(v['lastvisit'])

Jednak w systemie Windows najwyraźniej ta rozdzielczość w nanosekundach to trochę za dużo, a ten kod zgłasza wyjątek błędu systemu operacyjnego. Na szczęście istnieje proste obejście. Podzielenie tej liczby ostatniej wizyty przez 1000 przycięć ostatnich kilku cyfr, co daje liczbę w dacie i godzinie o niższej rozdzielczości, którą system Windows może obsłużyć. Wszystko, na czym naprawdę zależy nam w tej aplikacji, to data ostatniej wizyty; w ogóle nie zależy nam na czasie. Możesz więc pobrać tylko datę i ominąć błąd, pisząc kod w następujący sposób:

last_visit = dt.datetime.utcfromtimestamp(v['lastvisit']/1000).date()

To, co otrzymasz w zmiennej last_visit, to prosta data Pythona. Możesz więc użyć standardowego ciągu f do sformatowania daty w dowolny sposób. Na przykład użyj tego w swoim łańcuchu f, aby wyświetlić tę datę:

{last_visit: %m/%d/%Y}

Daty będą w formacie mm/dd/rrrr w danych wyjściowych, tak jak poniżej:

20.12.2018

19.12.2018

17.12.2018

20.12.2018

30.11.2018

16.12.2018

20.12.2018

20.12.2018

19.12.2018

Ładowanie JSON bez klucza z łańcucha Pythona

Metoda load() użyta w poprzednich przykładach załadowała dane JSON z pliku. Jednak dane JSON są zawsze dostarczane w pliku tekstowym, więc możesz skopiować/wkleić całość do łańcucha Pythona. Zazwyczaj nadajesz całemu ciągowi nazwę zmiennej i ustawiasz ją jako równą pewnemu dokumentowi, który zaczyna się i kończy potrójnymi cudzysłowami. Umieść wszystkie dane JSON w potrójnych cudzysłowach, jak w poniższym kodzie. (Aby zachować krótki kod, uwzględniliśmy dane tylko dla kilku osób, ale przynajmniej możesz zobaczyć, jak dane są uporządkowane).

import json

```
# Here the JSON data is in a big string named json_string.
# It starts and the first triple quotation marks and extends
# down to the last triple quotation marks.
json_string = """
{
"people": [
{
"Full Name": "Angst, Annie",
"Birth Year": 1982,
"Date Joined": "01/11/2011",
"Is Active": true,
"Balance": 300
},
{
"Full Name": "Schadenfreude, Sandy",
"Birth Year": 2004,
"Date Joined": "03/03/2003",
"Is Active": true,
"Balance": 0
}
]
}
"""
```

Chociaż fajnie byłoby zobaczyć wszystkie dane z kodu w taki sposób, jest jedna duża wada: nie można przechodzić przez ciąg znaków, aby dostać się do poszczególnych elementów danych. Jeśli chcesz przejść przez pętlę, musisz załadować dane JSON z łańcucha do jakiegoś obiektu. Aby to zrobić, użyj json.loads() (gdzie s jest skrótem od ciągu znaków), jak w poniższym kodzie. Jak zwykle peep_data to tylko nazwa, którą wymyśliliśmy, aby odróżnić załadowane dane JSON od danych w łańcuchu:

```
# Load JSON data from the big json_string string.
peep_data = json.loads(json_string)
```

Teraz, gdy masz już obiekt do pracy (peep_data), możesz przechodzić przez pętlę i pracować z kodem po jednym fragmencie na raz, tak jak poniżej:

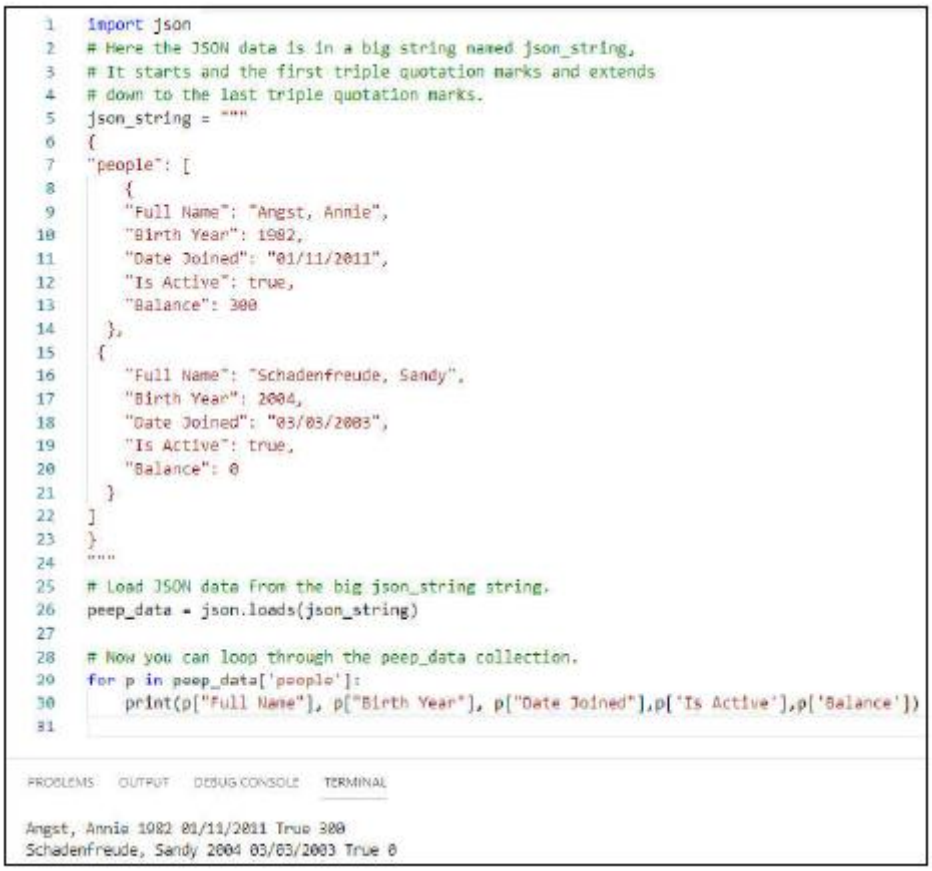
```
# Now you can loop through the peep_data collection.
```

for p in peep_data['people']:

print(p["Full Name"], p["Birth Year"], p["Date Joined"],p['Is

Active'],p['Balance'])

Rysunek pokazuje cały kod i wynik uruchomienia tego kodu w VS Code.



Ładowanie JSON z kluczem z łańcucha Pythona

Kluczowe dane mogą być również przechowywane w łańcuchu Pythona. W poniższym przykładzie ponownie użyliśmy json_string jako nazwy zmiennej, ale jak widać, dane wewnątrz łańcucha mają nieco inną strukturę. Pierwszy element ma klucz 1, a drugi element ma klucz 2. Ale znowu kod używa json.loads(json_string) do załadowania tej daty z łańcucha do obiektu JSON:

import json

Here the JSON data is in a big string named json_string,

It starts and the first triple quotation marks and extends

down to the last triple quotation marks.

json_string = """

{

"1": {

"count": 9061,

"lastreferrer": "https://difference-engine.com/Courses/tml-5-1118/",

"lastvisit": "12/20/2018",

"page": "/etg/downloadpdf.html"

},

"2": {

"count" : 3342,

"lastreferrer" : "https://alansimpson.me/",

"lastvisit" : "12/19/2018",

"page" : "/html_css/index.html"

}

}

"""

Load JSON data from the big json_string string.

hits_data = json.loads(json_string)

Now you can loop through the hits_data collection.

for k, v in hits_data.items():

print(f"{k}. {v['count']:>5} - {v['page']}")

Pętla na końcu drukuje klucz, liczbę trafień i nazwę strony z każdego elementu w formacie pokazanym w poniższym kodzie. Zauważ, że ta pętla wykorzystuje dwie zmienne o nazwach k i v do przechodzenia przez hits_data.items(), co jest standardową składnią przechodzenia przez słownik słowników:

1. 9061 - /etg/downloadpdf.html

2. 3342 - /html_css/index.html

Zmiana danych JSON

Gdy masz dane JSON w słowniku danych, możesz użyć standardowych procedur słownikowych do manipulowania danymi w słowniku. Przeglądając słownik danych ze zmiennymi klucz i wartość, możesz zmienić wartość dowolnej pary klucz:wartość, używając stosunkowo prostej składni:

wartość['klucz'] = nowe dane

Klucz i wartość to tylko zmienne k i v z pętli. Załóżmy na przykład, że przechodzisz przez słownik utworzony na podstawie bazy danych Firebase, który zawiera pole ostatniej wizyty pokazane jako numer znacznika czasu UTC. Chcesz zmienić ten znacznik czasu na ciąg w bardziej znanym formacie Pythona. Skonfiguruj pętlę jak w poniższym kodzie, w której pierwszy wiersz wewnątrz pętli tworzy nową zmienną o nazwie pydate, która zawiera datę jako datę Pythona. Następnie druga linia zastępuje treść v['lastvisit'] tą datą w formacie mm/dd/rr:

for k, v in hits.items():

Convert the Firebase date to a Python date.

pydate = dt.datetime.utcfromtimestamp(v['lastvisit']/1000).date()

In the dictionary, replace the Firebase date with string of Python date.

v['lastvisit']= f"{pydate:%m/%d/%Y})"

Po zakończeniu tej pętli wszystkie wartości w kolumnie „ostatnia wizyta” będą datami w formacie mm/dd/rrrr, a nie w formacie sygnatury czasowej Firebase.

Usuwanie danych ze słownika

Aby usunąć dane ze słownika podczas wykonywania pętli, użyj składni pop('nazwa_klucza', None). Zastąp „nazwa klucza” nazwą kolumny, którą chcesz usunąć. Na przykład, aby usunąć wszystkie nazwy kluczy lastreferrer i dane ze słownika utworzonego przez przykład JSON bazy danych Firebase, dodaj v.pop('lastreferrer', None) do pętli. Rysunek pokazuje przykład, w którym linie 1-8 importują dane Firebase do obiektu Pythona o nazwie hits.

```
1  import json
2  import datetime as dt
3  # This is the Firebase JSON data (keyed).
4  filename = 'firebase_hitcounts.json'
5  # Open the file (standard file open stuff).
6  with open(filename, 'r', encoding='utf-8', newline='') as f:
7      # Load the whole json file into an object named hits
8      hits = json.load(f)
9
10 for k, v in hits.items():
11     # Convert the Firebase date to a Python date.
12     pydate = dt.datetime.utcfromtimestamp(v['lastvisit']/1000).date()
13     # In the dictionary, replace the Firebase date with string of Python date.
14     v['lastvisit']= f"{pydate:%m/%d/%Y})"
15     # Remove the entire last referrer column.
16     v.pop('lastreferrer', None)
17
18 # Now look at the lastvisit date in the hits dictionary.
19 for k, v in hits.items():
20     print(k,v)
21
22
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

-LAOqAyxxHrPu6pGXBMZ {'count': 9261, 'lastvisit': '12/20/2018)', 'page': '/stg/downloadpdf.html'}
-LAOqOxg6kmP4jhnjQXS {'count': 3896, 'lastvisit': '12/20/2018)', 'page': '/'}
-LAOrwciIQJZvuCAcyLO {'count': 3342, 'lastvisit': '12/20/2018)', 'page': '/html_css/index.html'}
-LAOs2nsWVxbjAwoUXxE {'count': 2220, 'lastvisit': '12/20/2018)', 'page': '/html_css/codequickies/'}
-LAOuqJsjfuoQxBNISlX {'count': 2194, 'lastvisit': '12/20/2018)', 'page': '/index.html'}
-LAQ7ShbQPqOANbDmm3O {'count': 1154, 'lastvisit': '12/16/2018)', 'page': '/javascript/code_quickies/'}
-LAQrS6av1v8PuJGWm6P {'count': 1547, 'lastvisit': '12/20/2018)', 'page': '/how/'}
-LI0iPwZ7nu3IUgiQORH {'count': 1439, 'lastvisit': '12/20/2018)', 'page': '/datascience/beginner/'}
-LI2DFNAxVnT-cxYzNR- {'count': 1643, 'lastvisit': '12/19/2018)', 'page': '/datascience/cheatsheets/'}
```

Linia 10 rozpoczyna pętlę, która przechodzi przez każdy klucz (k) i wartość (v) w słowniku. Linia 12 konwertuje znacznik czasu na datę Pythona o nazwie pydate. Następnie linia 16 zastępuje ciąg czasu, który znajdował się w kolumnie lastvisit, tą datą Pythona jako ciąg znaków w formacie mm/dd/rrrr. Linia 16, v.pop('lastreferrer', None), usuwa całą parę lastreferrer klucz:wartość z każdego słownika. Pętla końcowa pokazuje zawartość słownika po dokonaniu tych zmian. Pamiętaj, że zmiany wprowadzone do słownika w Pythonie nie mają wpływu na plik lub łańcuch, z którego załadowałeś dane JSON. Jeśli chcesz utworzyć nowy ciąg JSON lub plik, użyj metod json.dumps() lub json.dump() omówionych dalej.

Zrzucanie danych Pythona do JSON

Do tej pory mówiliśmy o przenoszeniu danych JSON ze świata zewnętrznego do Twojej aplikacji, aby Python mógł z nich korzystać. Może się zdarzyć, że zechcesz pójść w przeciwnym kierunku, wziąć niektóre dane, które już znajdują się w Twojej aplikacji, w formacie słownikowym i wyeksportować je do formatu JSON, aby przekazać je innej aplikacji, ogółowi społeczeństwa lub cokolwiek innego. W tym miejscu do gry wchodzą metody json.dump() i json.dumps(). Metoda dumps() tworzy ciąg JSON danych, który nadal znajduje się w pamięci, gdzie można go wydrukować (), aby go zobaczyć. Na przykład poprzednie przykłady kodu importowały bazę danych Firebase do słownika Pythona, a następnie przechodziły przez ten słownik, zmieniając wszystkie znaczniki czasu na daty mm/dd/rrrr, a także usuwając wszystkie pary klucz:wartość lastreferrer. Załóżmy więc, że chcesz utworzyć ciąg JSON tego nowego słownika. Możesz użyć takich zrzutów, aby utworzyć ciąg o nazwie new_dict, a także wydrukować ten ciąg na konsoli. Ostatnie dwa wiersze kodu poza pętlą to:

#Looping is done, copy new dictionary to JSON string.

new_dict = json.dumps(hits)

print(new_dict)

Ciąg new_dict byłby wyświetlany w swoim natywnym, niezbyt czytelnym formacie, który wyglądałby mniej więcej tak:

{"-LAOqAyxxHrPw6pGXBMZ": {"count": 9061, "lastvisit": "12/20/2018)", "page":

"/etg/downloadpdf.html"}, "-LAOqOxg6kmP4jhnjQXS": {"count": 3896,

"lastvisit": "12/20/2018)", "page": "/"}, "-LAOrwciIQJZvuCAcyLO":

{"count": 3342, "lastvisit": "12/20/2018)", "page":

"/html_css/index.html"}, ... }}

Wee zastąpiliśmy niektóre dane ... ponieważ nie musisz widzieć wszystkich elementów, aby zobaczyć, jak nieczytelne to wygląda. Na szczęście metoda .dumps() obsługuje opcję indent=, w której możesz określić, w jaki sposób chcesz wciąć dane JSON, aby były bardziej czytelne. Zazwyczaj wystarczają dwie spacje. Na przykład dodaj indent=2 do powyższego kodu w następujący sposób:

#Looping is done, copy new dictionary to JSON string.

new_dict = json.dumps(hits, indent=2)

print(new_dict)

Dane wyjściowe tej metody print() pokazują dane JSON w znacznie bardziej czytelnym formacie, jak pokazano tutaj:

{

"-LAOqAyxxHrPw6pGXBMZ": {

"count": 9061,

"lastvisit": "12/20/2018)",

"page": "/etg/downloadpdf.html"

},

"-LAOqOxg6kmP4jhnjQXS": {

"count": 3896,

"lastvisit": "12/20/2018)",

"page": "/"

},

...

}

Jeśli używasz znaków obcych lub specjalnych w swoim słowniku danych i chcesz je zachować, dodaj do kodu sure_ascii=False w następujący sposób:

new_dict = json.dumps(hits, indent=2, ensure_ascii=False)

W naszym przykładzie nazwy kluczy w każdym słowniku są już w porządku alfabetycznym (liczba, ostatnia wizyta, strona), więc nie musielibyśmy nic robić, aby umieścić je w ten sposób. Ale we własnym kodzie, jeśli chcesz mieć pewność, że klucze w każdym słowniku są w porządku alfabetycznym, dodaj sortkeys=True do swojej metody .dumps w następujący sposób:

new_dict = json.dumps(hits, indent=2, ensure_ascii=False, sort_keys=True)

Jeśli chcesz wypisać swój JSON do pliku, użyj json.dump() zamiast json.dumps(). Możesz użyć sure_ascii=False, aby zachować obce znaki, i sort_keys = True, aby ułożyć nazwy klawiszy alfabetycznie. Możesz również dołączyć opcję indent=, chociaż spowodowałoby to zwiększenie pliku i zazwyczaj chcesz zachować małe pliki, aby zaoszczędzić miejsce i zminimalizować czas pobierania. Załóżmy na przykład, że chcesz utworzyć plik o nazwie hitcounts_new.json (lub jeśli już istnieje, otwórz go, aby nadpisać jego zawartość). Chcesz zachować wszystkie obce znaki, które zapiszesz w pliku. Oto kod do tego; litera „w” jest wymagana, aby upewnić się, że plik otwiera się w celu zapisania do niego danych:

with open('hitcounts_new.json', 'w', encoding='utf-8') as out_file:

Następnie, aby skopiować słownik o nazwie hits jako JSON do tego pliku, użyj nazwy nadanej na końcu kodu w wierszu powyżej. Ponownie, aby zachować wszelkie obce znaki i być może ułożyć alfabetycznie nazwy kluczy w każdym słowniku, podążaj za tym wierszem z tym, upewniając się, że ten jest wcięty tak, aby był zawarty w bloku with:

json.dump(hits, out_file, ensure_ascii=False, sort_keys=True)

Rysunek przedstawia cały kod, zaczynając od danych wyeksportowanych z Firebase, przechodząc przez utworzony przez import słownik, zmieniając i usuwając część treści, a następnie zapisując nowy słownik do nowego pliku JSON o nazwie hitcounts_new.json.

Rysunek przedstawia zawartość pliku hitcounts_new.json po uruchomieniu aplikacji.

Nie wcięliśmy formatu JSON, ponieważ pliki są naprawdę do przechowywania lub udostępniania, a nie do przeglądania, ale nadal można zobaczyć, że wartości datevisited są w formacie mm/dd/rrrr, a para klucz:wartość lastreferrer nie jest w formacie tam, ponieważ wcześniejszy kod usunął tę parę klucz:wartość. JSON to bardzo szeroko stosowany format do przechowywania i udostępniania danych. Na szczęście Python ma wiele wbudowanych narzędzi do konsumowania i tworzenia danych JSON. Omówiliśmy tutaj najważniejsze możliwości. Ale nie wstydź się szukać w Google lub YouTube python json, jeśli chcesz dowiedzieć się więcej.

```
import json
import datetime as dt
# This is the Firebase JSON data (keyed).
filename = 'firebase_hitcounts.json'
# Open the file (standard file open stuff).
with open(filename, 'r', encoding='utf 8', newline='') as f:
    # Load the whole json file into an object named hits.
    hits = json.load(f)

# Loop through the new hits data dictionary.
for k, v in hits.items():
    # Convert the Firebase date to a Python date.
    pydate = dt.datetime.utcfromtimestamp(v['lastvisit']/1000).date()
    # In the dictionary, replace the Firebase date with string of Python date.
    v['lastvisit'] = f"{pydate:%m/%d/%Y})"
    # Remove the entire last referrer column.
    v.pop('lastreferrer', None)

# Write the modifed data to a JSON file named hitcounts_new.json.
with open('hitcounts_new.json', 'w', encoding='utf-8') as out_file:
    json.dump(hits, out_file, ensure_ascii=False, sort_keys=True)

print('Done')
```