Pobieranie danych z API NBP

Umiemy *rozmawiać* z usługami sieciowymi z poziomu Pythona. W poprzednich rozdziałach poznaliśmy API Narodowego Banku Polskiego. Połączmy zatem zdobytą wiedzę i napiszmy program, który pobierze nam kursy walut z konkretnego dnia!

> **UWAGA** Tutaj zaczyna się historia Twojego kodu. Spróbuj - podążając za niniejszym podręcznikiem - tworzyć swoje rozwiązanie. Niniejsza książka jest przewodnikiem, mapą i drogowskazem. **Ale to TY samodzielnie napisz swoją aplikację**, samodzielnie się z nią zmagaj i spróbuj zmieniać, poprawiać, modyfikować. Dostaniesz gotowe rozwiązanie, ale bez własnych prób niewiele się nauczysz. A przecież dla nauki poszedł hajs na ten materiał, prawda?

Pobranie konkretnego notowania

Zaczniemy od pobrania tabeli A z konkretnego dnia. Przypomnijmy: zgodnie z dokumentacją API NBP (na stronie api.nbp.pl) tabela kursów typu *{table}* z dnia *{date}* dostępna będzie pod adresem `https://api.nbp.pl/api/exchangerates/tables/{table}/{date}/`, a dodanie parametru `?format=json` zwróci nam odpowiedź w JSONie.

Data ma być w formacie `YYYY-MM-DD` i od tego zaczniemy.
Weźmy 7 lutego 2025 roku. W zapisie `YYYY-MM-DD` (często zobaczycie też notację ze znakami procent, czyli: `%Y-%m-%d` - uwaga małe `m` i `d`!) to `2024-02-07`, czyli dla dwójki i siódemki potrzebujemy dodać *wiodące zera*. W Pythonie łatwo to osiągnąć używając *f-string*:

```
print(f"{7:02}")
```

da w efekcie `07`. Zgodnie z zapisem: liczba 7 zostanie dopełniona zerami na początku, aby łącznie liczba znaków wyniku była równa co najmniej dwa. Dla porównania zobacz co da `print(f"{123:05}")` albo `print(f"{12345:03}")`.

Zbudujmy więc cały adres usługi, którą będziemy pytać:

```
day = 7
month = 2
year = 2025
table = 'A'

query_url = f"https://api.nbp.pl/api/exchangerates/tables/{table}/{year
    :04}-{month:02}-{day:02}/?format=json"

print(query_url)
```

Wynik działania powyższego kodu to:

`https://api.nbp.pl/api/exchangerates/tables/A/2025-02-07/?format=json`

i zapewnie kiedy to czytasz jest po 7 lutego, więc możesz ten adres otworzyć w przeglądarce, Postmanie lub Bruno.

Zanim przejdziemy dalej - zróbmy sobie z tego kodu funkcję, która na podstawie podanej daty zwróci nam dedykowany dla tej daty adres. Lekka modyfikacja (zamiast zmiennych z konkretnie zdefiniowanymi wartościami mamy parametry funkcji, zamiast printa - zwracamy wartość):

```
def make_query_url(year, month, day, table = "A"):
    return f"https://api.nbp.pl/api/exchangerates/tables/{table}/{year:04}-{
        month:02}-{day:02}/?format=json"
```

Pobierzmy więc tabelę z jednego dnia i zapiszmy ją do serii plików JSON - po jednym dla każdej waluty. Nazwę pliku zbudujemy z daty notowania i nazwy waluty.

Jak zapisać plik JSON w Pythonie? Korzystając z pakietu `json`, zbudujmy od razu małą funkcję:

```
import json

def save_data_to_file(data, file_name):
    with open(file_name, "w", encoding="utf-8") as fp:
        json.dump(data, fp)
```

Funkcja ta wymaga danych (słownika, listy, listy słowników) przekazanych w argumencie `data` oraz nazwy pliku (`file_name`) do którego dane zostaną zapisane.

Nazwę pliku zbudujemy podobnie jak adres usługi:

```
def make_filename(year, month, day, currency):
    return f"{currency}_{year:04}_{month:02}_{day:02}.json"
```

Pozostało jeszcze odpytanie naszej usługi - właściwie kluczowe w tym rozdziale zadanie:

```
import requests

def get_rates(year, month, day, table="A"):
    query_url = make_query_url(year, month, day, table)
    result = requests.get(query_url)

    if result.status_code != 200:
        return []

    table = result.json()[0]
    rates = table.get("rates", [])

    return rates
```

Tutaj sprawa jest bardziej złożona. Nasza funkcja oczekuje czterech parametrów:

- daty - rozbitej na rok, miesiąc i dzień

- tabeli, którą chcemy uzyskać; ale to parametr z wartością domyślną równą `A`, więc nie musimy go podawać. Warto jednak, jeśli to możliwe, *uogólnić* nasze narzędzia.

Funkcja korzystając z podanych parametrów, wywołuje naszą funkcję pomocniczą `make_query_url()`, aby uzyskać dokładny adres usługi do odpytania. Żądanie jest wysłane i powinno się udać. Jeśli mamy połączenie z internetem to dostaniemy odpowiedź z NBP. Kod tej odpowiedzi inny niż 200 oznaczać będzie dla nas błąd - zwracamy więc **pustą listę**.

`Status code` równy 200 to sukces - mamy kwotowania! Dekodujemy więc odpowiedź z JSONa (bo spodziewamy się, że takowy przyjdzie) i - wiedząc, że to lista kolejnych dni z notowaniami - sięgamy do pierwszego elementu na liście. Dlaczego? Bo prosimy o notowanie z jednego dnia - ta lista powinna mieć tylko jeden element: kwotowanie z tego konkretnego dnia. Na taki **kontrakt z usługą** jesteśmy umówieni, tego się spodziewamy.
Ten jeden dzień zapisujemy w zmiennej `table` (bo to tabela) - jest to słownik (przynajmniej powinien zgodnie z kontraktem). W kolejnym kroku *wyjmujemy* z naszej tabeli (pythonowego słownika) **listę kwotowań kolejnych walut**. Listę tą znajdziemy w polu o kluczu `rates`. Wyjmujemy poprzez metodę `.get()`, aby nie spowodować błędu jeśli akurat zabrakło takiego klucza. Jeśli zabrakło - będziemy mieć pustą listę.
To co pozyskaliśmy z `rates` zwracamy poza funkcję `get_rates()`.

Przystańmy na chwilę i podsumujmy co mamy:

- z daty (rok, miesiąc i dzień) możemy utworzyć adres usługi w NBP
- odpytujemy tą usługę i jeśli nie ma błędów - dostaniemy listę obiektów (pythonowych słowników):
  - z nazwą waluty (klucz `currency`),
  - jej kodem (w `code`)
  - oraz kursem z konkretnego dnia (`mid`)

Każde z kwotowań waluty chcemy zapisać w pliku JSON o nazwie np. `EUR_2025_02_07.json`, zatem potrzebujemy:

- zbudować nazwę pliku (mamy już funkcję `make_filename()`)
- do tego pliku zapisać nasze dane (korzystając z gotowej już funkcji `save_data_to_file()`)

Skoro `get_rates()` zwróci nam listę, to wystarczy przez nią *przebiec* i zapisać każdy jej element:

```
## przykladowe parametry - data
req_year = 2024
req_month = 10
req_day = 7

nbp_rates = get_rates(req_year, req_month, req_day)
```

```
for currency in nbp_rates:
    currency_code = currency["code"]
    filename = make_filename(req_year, req_month, req_day, currency_code)
    save_data_to_file(currency, filename)
```

Co tu się dzieje?

- ustalamy sobie datę notowania (zmienne `req_...`)- trochę umyślnie mamy tutaj dwucufrowy miesiąc i jednocyfrowy dzień - aby sprawdzić czy nasze funkcje składające ciągi znaków poradzą sobie i dodadzą gdzie trzeba wiodące zera
- odpytujemy NBP o tabelę (domyślnie A) z kursami
- w pętli:
  - wyciągamy ze zwróconych przez `get_rates()` elementów kod waluty (pole `code`)
  - budujemy z daty i kodu waluty nazwę pliku używając `make_filename()` - na przykład dla japońskiego jena będzie to w efekcie `JPY_2024_10_07.json`
  - mając dane (kolejne `currency` w kolejnych przebiegach pętli) i nazwę pliku (`filename`) możemy użyć `save_data_to_file()` do zapisu danych na dysku

Efekt jest taki, że powstało nam wiele plików, z których ten przykładowy jen wygląda tak:

```
{"currency": "jen (Japonia)", "code": "JPY", "mid": 0.026542}
```

Ćwiczenia

- Zobacz co się stanie jak poprosisz o notowania z Nowego Roku, Bożego Narodzenia albo dowolnej niedzieli. Samodzielnie zmodyfikuj odpowiednie parametry i sprawdź wynik. Powstaną jakieś pliki?
- Spróbuj zmienić funkcję `make_filename()` tak, aby wszystkie pliki trafiały do dedykowanego (może według nowego parametru tej funkcji?) katalogu

Pobranie notowań z wielu dni

Zbierając cały kod z poprzednich stron mamy taki skrypcik:

```
import requests
import json


def make_query_url(year, month, day, table="A"):
    return f"https://api.nbp.pl/api/exchangerates/tables/{table}/{year:04}-{
        month:02}-{day:02}/?format=json"
```