20 przydatnych bibliotek w Pythonie

Kamil.Kwapisz.pl

Cześć!

Dziękuję Ci za zainteresowanie się tematyką mojego bloga i pobranie tego dokumentu.

Chciałbym przedstawić Ci zbiór ciekawych i wyjątkowo przydatnych bibliotek dostępnych w Pythonie.

Ich wykorzystanie nie tylko bardzo usprawni Twoją pracę, poprawi jakość kodu, lecz także sprawi, że będziesz programował ze znacznie większą **przyjemnością**, a to też bardzo ważny element 😊

Wszystkie biblioteki przetestowałem i sam bardzo często z nich korzystam, dlatego z czystym sumieniem mogę Ci je polecić.

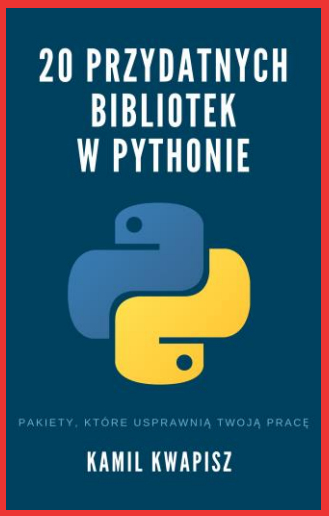
Celowo pomijam w tym dokumencie duże frameworki jak Django, Flask czy biblioteki w stylu Tensorflow.

Chcę przedstawić Ci pakiety, które nie wymagają przerabiania wielu godzin kursów jeszcze przed rozpoczęciem pracy, lecz w szybki i prosty sposób zwiększą jakość Twojego kodu i dadzą więcej frajdy z programowania.

Miłej lektury! 😊

Kamil Kwapisz

http://kamil.kwapisz.pl



1. Requests

Moim zdaniem jest to najlepsza biblioteka do obsługi protokołu HTTP. Wysyłanie żądań HTTP nigdy nie było tak proste i szybkie.

Jest to **must have** dla programistów korzystających z API lub wysyłających żądania do serwerów www.

W bardzo prosty sposób możemy wysłać żądanie, odebrać jego treść, sprawdzić kod odpowiedzi, historię przekierowań czy zmienić kodowanie treści.

Dokumentacja: https://2.python-requests.org/en/master/

2. BeautifulSoup

Parsowanie kodu HTML i XML dzięki tej bibliotece jest niesamowicie proste.

Pozwala ona wyciągnąć z pobranego kodu źródłowego strony interesujące nas informacje.

Idealnie sprawdza się w połączeniu z biblioteką Requests.

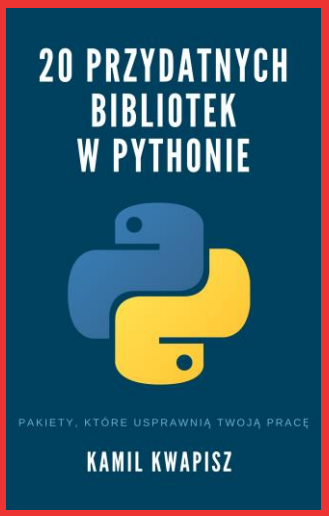
Dokumentacja:
https://www.crummy.com/software/BeautifulSoup/bs4/doc/

3. IPython

Znacznie ciekawszy i potężniejszy interpreter poleceń Pythona.

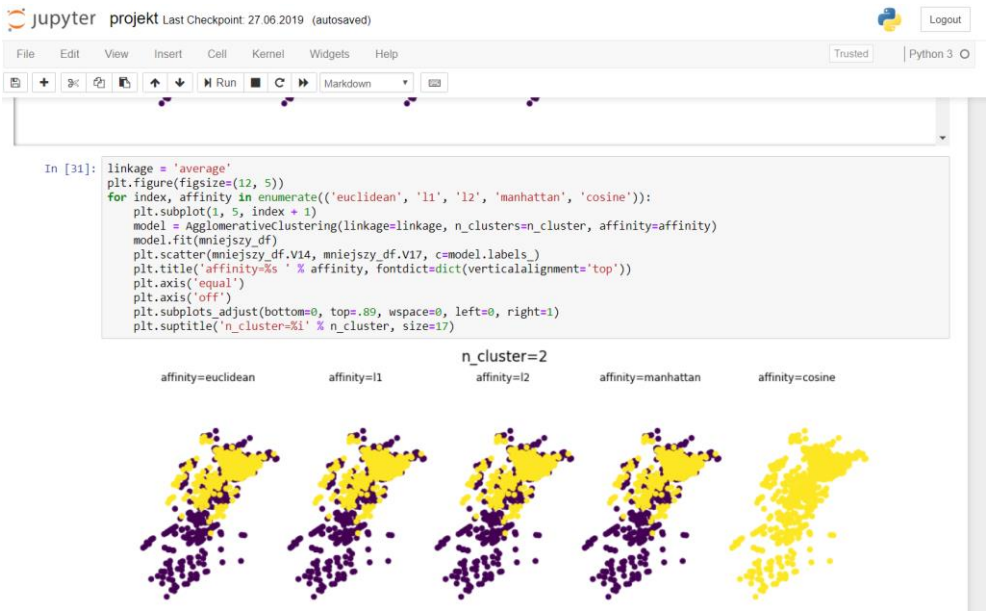
Jest rozszerzony o przydatną historię poleceń, autouzupełnianie oraz wiele dodatkowych narzędzi i usprawnień, jak na przykład znacznie prostsze mierzenie **czasu wykonania operacji** czy **aliasy**.

Dokumentacja: https://ipython.org/



4. Jupyter

Jest to zbiór narzędzi, które bazują na IPythonie. Dzięki nim możesz tworzyć przejrzysty i atrakcyjny wizualnie kod Pythona, umieszczając w nim wykresy, tabele czy własny tekst.



Narzędzie to idealnie sprawdzi się do analizy danych, jednak ma również wiele innych zastosowań.

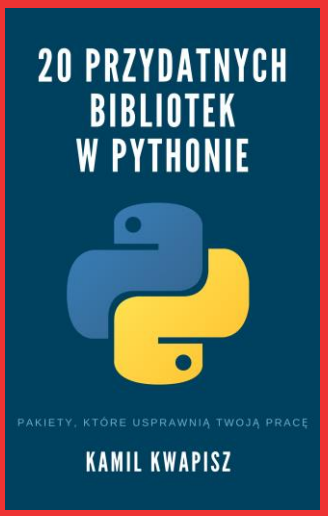
Dokumentacja: https://jupyter.org/

5. Black

W Pythonie jest wiele zasad określających w jakim stylu powinniśmy tworzyć nasz kod.

Głównym dokumentem to określającym jest PEP8, o którym możesz przeczytać w moim artykule.

Pilnowanie poprawności stylistycznej kodu może być czasochłonne, dlatego też wymyślono **formatery** – specjalne programy formatujące nasz kod tak, aby spełniał sugerowane wymogi.



Właśnie tym jest **Black**, formaterem kodu, który za nas myśli o stylistyce kodu. Wykorzystywanie tego narzędzia poprawi strukturę i wygląd Twojego kodu.

Dokumentacja: https://black.readthedocs.io/

6. SQLAlchemy

Korzystasz z bazy danych? Masz dość pisania długich kwerend SQL? To koniecznie sprawdź **SQLAlchemy**.

To biblioteka do **mapowania obiektowo-relacyjnego** (**ORM**).

Dzięki niej możesz tworzyć i obsługiwać infrastrukturę bazodanową programując obiektowo.

Zamiast poleceń SQL tworzysz klasy reprezentujące tabele, których pola są kolumnami.

Zamiast SELECT'ów wykonujesz odpowiednie metody na obiektach.

Jeżeli nie zamierzasz korzystać ze skomplikowanych opcji SQLa lub szczególnie optymalizować zapytań to zdecydowanie polecam Ci przetestowanie tej biblioteki.

Dokumentacja: https://www.sqlalchemy.org/

7. Records

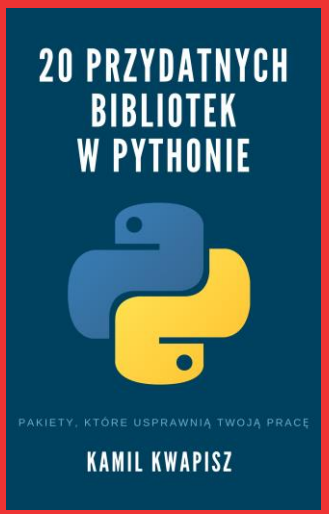
Jeśli jednak chcesz pisać polecenia SQL samemu, to mogę Ci polecić bibliotekę **Records**, która to usprawnia.

Połącz się z bazą, stwórz polecenie, wykonaj je, a wyniki otrzymasz zapisane w strukturach Pythona.

Zwrócone wiersze tabeli przechowywane są w strukturze, po której możemy iterować.

Narzędzie zapewnia Ci także prosty eksport danych.

Dokumentacja: https://pypi.org/project/records/



8. Pipenv

Pewnie już znasz **pip** – narzędzie służące do instalowania pakietów.

Być może zetknąłeś się już z tematyką **środowisk wirtualnych**.

Pipenv łączy te dwie rzeczy.

Automatycznie tworzy środowisko wirtualne i nim zarządza, co może znacznie usprawnić Twoją pracę.

Pipenv ułatwia również zarządzanie wymaganiami co do bibliotek używanych w projektach.

Dokumentacja: https://docs.pipenv.org/

9. pytest

Nie samym tworzeniem kodu żyje programista, co jakiś czas warto ten kod przetestować 😉

Do tego przyda Ci się właśnie biblioteka **pytest**.

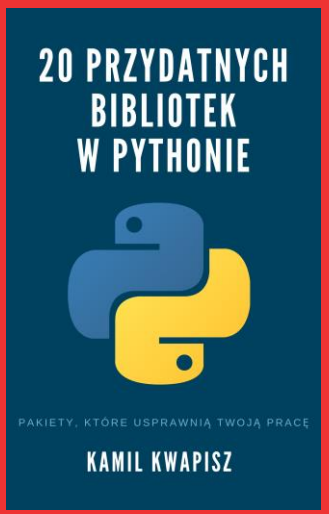
Narzędzie to szczególnie dobrze sprawdzi się do niewielkich testów jednostkowych, jednak jest ono kompatybilne z inną popularną biblioteką do testowania - **unittest**.

Dokumentacja: https://pytest.org/

10. Pendulum

Zarządzanie datami i strefami czasowymi bywa problematyczne. Rozwiązaniem tych problemów dla Ciebie może okazać się biblioteka **Pendulum**.

Narzędzie znacząco usprawnia obsługę zmiennych określających datę i czas.



Ułatwia również obsługę różnych stref czasowych i formatów zapisu dat.

Warto skorzystać z tego rozwiązania, jeżeli Twój projekt wymaga obsługi dat.

Dokumentacja: https://pendulum.eustace.io/

11. NumPy

Prawdopodobnie jedna z najpopularniejszych bibliotek w Pythonie. Wykorzystywana do wszelkich obliczeń numerycznych, jest genialną, darmową alternatywą dla MatLaba, z którego czerpie garściami.

Pokuszę się o stwierdzenie, że jeżeli wykonujesz operacje na dużej liczbie danych numerycznych to **koniecznie** używaj NumPy, a szczególnie jego struktur danych.

Zostały one zoptymalizowane specjalnie pod dane liczbowe, dzięki temu wydajność Twojego kodu wzrośnie.

Tablice NumPy wymagają znacznie mniej miejsca niż klasyczne listy wbudowane w Pythona.

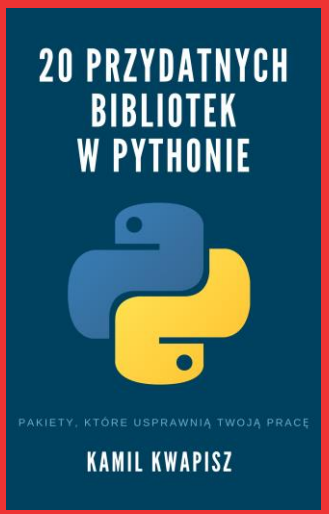
Dużo łatwiej wykonywać na nich również operacje matematyczne.

Dokumentacja: https://www.numpy.org/

12. Pandas

Pandas to biblioteka do przechowywania i wstępnej obróbki danych.

Najważniejszą strukturą dostarczaną przez pakiet jest **DataFrame** – czyli ramka danych przypominająca strukturą arkusze kalkulacyjnych (np. Excela).



Jest to tabela z danymi, w której wiersze są rekordami danych, a kolumny ich parametrami.

Pandas pozwala łatwo oczyścić tabelę, wypełnić braki lub usunąć wybrakowane wiersze czy też filtrować dane.

Co więcej, dane te bardzo łatwo przechowywać serializując je.

Zdecydowania zachęcam Cię do zapoznania się z tym narzędziem jeżeli obsługujesz nieco większe i bardziej skomplikowane dane nie tylko liczbowe.

Dokumentacja: https://pandas.pydata.org/

13. Dash

Gdy już zebrałeś i przetworzyłeś dane, warto je zwizualizować. Najatrakcyjniejszą formą ich wizualizacji według mnie zapewnia **Dash**.

Narzędzie pozwala na tworzenie interaktywnych aplikacji wizualizujących dane uruchamianych jako serwer lokalny.

Nie jest wymagana znajomość JavaScriptu ani nawet zaawansowanego HTMLa.

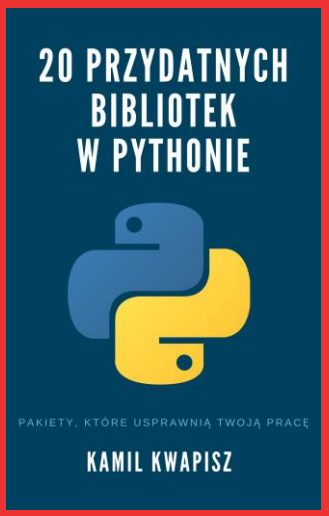
Jeśli chcesz uatrakcyjnić wizualizację danych, sprawdź tę bibliotekę.

Dokumentacja: https://plot.ly/dash

14. Pillow

Jeżeli przetwarzasz dane graficzne to **Pillow** zdecydowanie Ci się przyda.

Dzięki temu narzędziu wyciągniesz z obrazu jak najwięcej informacji, wyskalujesz go i przytniesz, zastosujesz wybrane filtry czy też obrócisz.



Pillow przyda Ci się także by nałożyć na obraz znak wodny.

Krócej mówiąc, jest to Twój **przybornik** do obróbki grafik.

Dokumentacja: https://pillow.readthedocs.io/

15. Selenium

Selenium jest narzędziem dostępnym dla wielu języków i technologii, jednak mimo wszystko postanowiłem umieścić je na tej liście.

Narzędzie to służy do automatyzowania przeglądarek.

Chcesz wykonywać testy elementów na swojej stronie? Żaden problem, zautomatyzuj proces za pomocą Selenium.

WebDriver Selenium otwiera „zdalnie sterowane" okno przeglądarki, która wykona instrukcje zawarte w kodzie.

Za pomocą tego narzędzia możesz kliknąć różne elementy strony, wypełnić formularze czy nawet zalogować się na swoje konto.

Dokumentacja: https://www.seleniumhq.org/

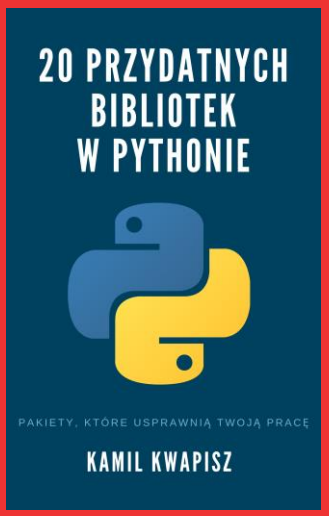
16. NLTK

Zestaw narzędzi przeznaczony do przetwarzania **tekstu naturalnego**, czyli takiego rozumianego dla człowieka, a niekoniecznie dla komputera.

Świetnie sprawdza się do podziału tekstu na mniejsze elementy jak np. słowa czy zdania. Sprawdzi się znacznie lepiej niż zwykła metoda **split**.

Przetwarzając teksty języka naturalnego to narzędzie jest obowiązkowe.

Dokumentacja: https://www.nltk.org/



17. Faker

Podczas tworzenia aplikacji bardzo często jesteśmy zmuszeni wykorzystywać losowe, nieprawdziwe dane.

Wymyślanie takich danych samemu jest żmudne i niepotrzebnie zajmuje dużo czasu.

Aby usprawnić ten proces możesz skorzystać z **Fakera** – generatora takich danych.

Możesz wygenerować fałszywe imiona i nazwiska, adresy, daty, czy fragmenty tekstu.

Ciekawą funkcją jest określanie szczegółów lokalizacyjnych. Dzięki temu można wygenerować polskie imiona i nazwiska czy adresy.

Dokumentacja: https://faker.readthedocs.io/

18. Validators

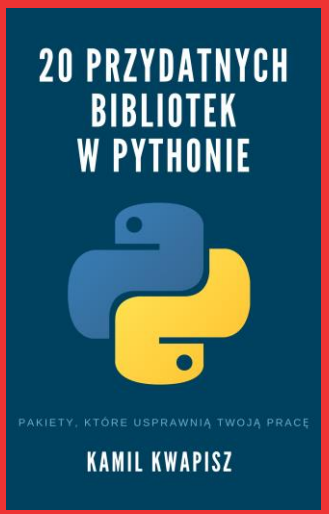
Programując aplikacje komunikujące się z użytkownikiem bardzo często potrzebujemy dokonać walidacji danych. W końcu ponoć kontrola jest najwyższą formą zaufania 😊

Do walidacji danych możesz wykorzystać pakiet **validators**, który gromadzi w sobie kilka przydatnych mechanizmów walidujących dane.

Chcesz sprawdzić czy podany napis jest emailem lub adresem URL? Nic prostszego.

A co ważniejsze, oszczędzasz czas jaki przeznaczyłbyś na tworzenie własnego walidatora.

Dokumentacja: https://validators.readthedocs.io/



19. Responder

Zdarza się tak, że chcemy w naszej aplikacji wystawić proste API. Potrzebne nam są jedynie metody pozwalające na wyświetlenie elementów z bazy lub dodanie ich.

Zamiast korzystać ze skomplikowanego **Django Rest Framework** bądź **Flaska** możesz sięgnąć po coś mniejszego – **Responder.**

Ten minimalistyczny framework pozwoli Ci na szybkie stworzenie prostego API.

Zachęcam Cię do przetestowania go jako alternatywa dla innych webowych frameworków.

Dokumentacja: https://python-responder.org/

20. Eventlet

Jest to biblioteka do wykonywania równoległych operacji w Pythonie.

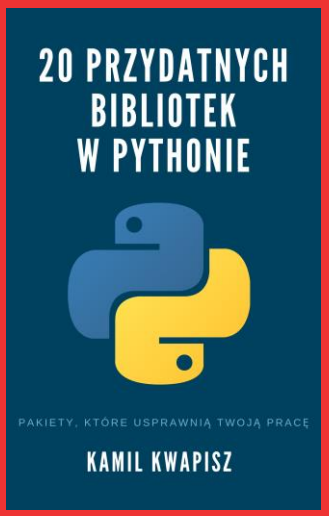
Jej największą zaletą jest to, że nie ingeruje w sposób pisania kodu, lecz jego uruchamiania.

Chcesz stworzyć scrapera, który jak najszybciej wykona zadanie? Wykorzystanie do tego biblioteki **eventlet** może cały proces przyspieszyć.

Dokumentacja: http://eventlet.net/

BONUS

Postanowiłem uzupełnić dokument o jeszcze dwie dodatkowe biblioteki, które mogą Ci się spodobać 😉



21. Requests-HTML

W 1 i 2 punkcie pisałem o bibliotekach **Requests** i **BeautifulSoup**. Wspomniałem również, że bardzo dobrze sprawdzają się w połączeniu.

Twórca pierwszej wymienionej biblioteki postanowił to wykorzystać i stworzył projekt łączący obie biblioteki.

Korzystając z niej możemy pobierać kod HTML ze strony i od razu go przetwarzać.

Narzędzie zostało również wyposażone w dodatkowe funkcjonalności jak np. znajdowanie linków na stronie, wyszukiwanie wzorca w tekście czy symulowanie wczytania kodu JavaScript.

Dokumentacja: https://html.python-requests.org/

22. Vader

Pisałem już wcześniej o przetwarzaniu języka naturalnego. Ludzie są emocjonalni, więc teksty tworzone przez nich bardzo często są nacechowane pozytywnie lub negatywnie.

Do badania takiego nacechowania, czyli **sentymentu,** może posłużyć Ci biblioteka **Vader**.

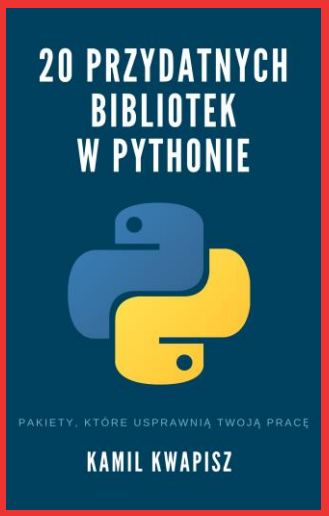
Analiza tekstu polega na określeniu jak bardzo jest on pozytywny, negatywny i neutralny.

Biblioteka sprawdza się w przypadku anglojęzycznych tekstów.

Dokumentacja: https://github.com/cjhutto/vaderSentiment

Cieszę się, że dotarłeś do końca tego dokumentu 😉

Mam nadzieję, że chociaż kilka z podpowiedzianych przeze mnie rozwiązań wykorzystasz w swoich projektach, lub już je wykorzystujesz.



Jeżeli chcesz odkryć jeszcze więcej przydatnych bibliotek możesz skorzystać z ciekawej listy, którą znalazłem na GitHubie: https://github.com/vinta/awesome-python/

Mam do Ciebie również prośbę.

Podziel się proszę linkiem do bloga http://kamil.kwapisz.pl/ z przynajmniej jednym znajomym, którego temat programowania zainteresuje. Z góry Ci za to dziękuję 😊

Dzięki i do zobaczenia! 😉

Kamil Kwapisz