# Programowanie
# *po 10 wykładzie*

Andrzej Giniewicz

17.05.2024

W tym tygodniu zajmujemy się tworzeniem własnych pakietów.

Przypomnijmy dotychczas poznaną organizację kodu w Pythonie. Możemy tworzyć funkcje, możemy też tworzyć klasy. Wiele klas i funkcji oraz inne fragmenty kodu, możemy zamknąć w moduły, które implementujemy jako plik z rozszerzeniem `.py` lub katalog z plikiem `__init__.py` wewnątrz. Alternatywą dla tworzenia bibliotek, jest tworzenie skryptów, które zamykamy w pliku o rozszerzeniu `.py` i w których kod uruchamiany jest od początku do końca. Dzięki specjalnej zmiennej `__name__` możemy mieć jeden plik `.py`, który jest zarazem biblioteką, jak i skryptem. Biblioteki, skrypty oraz inne pliki możemy zamknąć w **pakiet**.

Gdy używamy komendy `import`, importujemy bibliotekę. Aby mieć dostęp do niestandardowej biblioteki, jak robiliśmy to w przypadku biblioteki `hypothesis`, musimy zainstalować pakiet ją zawierający. Istnieje kilka rodzajów pakietów — pakiety źródłowe oraz binarne. Standardowe pliki binarne w Pythonie nazywamy **wheels**. Pliki binarne są dystrybuowane już skompilowane, zatem proces instalacji jest szybszy, ale mogą być one zależne od systemu operacyjnego oraz wersji Pythona, dla której je przygotowano.

Pakiet tworzymy, umieszczając kod źródłowy oraz pomocnicze pliki w jednym katalogu. Istnieje wiele narzędzi do tworzenia pakietów oraz wiele standardów. Można spotkać na przykład rekomendacje, aby tworzyć plik `setup.py`. Potem rekomendowanym formatem były pliki `setup.cfg`, natomiast teraz są to pliki z rozszerzeniem `.toml`. Łatwo natrafić na nieaktualne informacje, tym bardziej że większość projektów wciąż korzysta z formatu `setup.py`, który z jednej strony daje największe możliwości, ale z drugiej strony łatwo w nim popełnić błąd powodujący, że pakiet nie będzie działał prawidłowo. Najbardziej aktualne informacje zawsze są na stronie `pypa.io`[1].

Jedna ze stron PyPA posiada krótki tutorial przeprowadzający przez tworzenie paczek. Postaraj się prześledzić go krok po kroku[2]. Możesz spróbować wgrać na `https://test.pypi.org` pakiet z grą w kółko i krzyżyk przygotowany na podstawie projektu z ostatnich zajęć. Kod źródłowy znajduje się pod adresem `http://prac.im.pwr.edu.pl/~giniew/`

[1] PyPA to skrót od Python Packaging Authority.

[2] Patrz `https://packaging.python.org/en/latest/tutorials/packaging-projects/`.

`pwr_tic_tac_toe.zip`. Spośród komend rekomendowanych w tutorialu, unikaj jedynie aktualizowania komendy `pip`, przynajmniej jeśli była zainstalowana razem z Anacondą — wtedy nie należy jej aktualizować komendą `pip`, tylko komendą `conda`. Niemniej jednak tegoroczna dystrybucja Anacondy ma wystarczająco nowy `pip`, aby wykonać całe zadanie.

Pakiety wgrane w ten sposób dobrze działają, jeśli użytkownik ma zainstalowanego Pythona. Jeśli chcemy, aby nasza aplikacja mogła być używana również przez osoby nieposiadające całego środowiska dla programistów, można posłużyć się projektem takim, jak PyInstaller ze strony `https://pyinstaller.org/`. Jest to projekt, który zbiera z systemu wymagane części Pythona oraz bibliotek takich jak PyGame lub PyQt oraz przygotowuje niezależny plik wykonywany (na przykład `.exe` na Windowsie), który można uruchomić bez osobnego Pythona.