# Tworzenie modułów

## 1. Wprowadzenie

pakiet

**moduł**

funkcja

Moduł to typ kontenera wypełnionego funkcjami - do pojedynczego modułu można włożyć dowolną ilość funkcji, jednakże mieszanie funkcji z różnych zastosowań (obszarów) nie jest dobrym pomysłem. Jeśli mamy wiele modułów to można je zgrupować w pakiet.

## 2. Tworzenie modułu

Plik o nazwie **mpmodul_1**

```
# to jest pierwszy moduł
if __name__ == "__main__":
    print("Wysołanie bezpośrednie(nie jako moduł)")
    print(__name__)
else:
    print("Jestem modułem")
    print(__name__)
```

Plik o nazwie **test_module_1**

```
import mpmodul_1
```

**Przeanalizuj**

Plik o nazwie **mpmodul_2**

Plik o nazwie **test_module_2**

Kiedy moduł jest importowany, jego zawartość jest wykonywana przez język Python. Daje to modułowi ‘okazję’, by zapoczątkować pewne swoje wewnętrzne procesy (np. może przypisać niektórym zmiennym wartości).

**Uwaga:** inicjalizacja ma miejsce tylko jeden raz, wraz z pierwszym importem, więc przypisania wykonane przez moduł nie są niepotrzebnie powtarzane.

Przykład:

- istnieje moduł o nazwie **test1**
- istnieje moduł o nazwie **test2** i zawiera on instrukcję **import test1**;
- istnieje też główny plik, zawierający instrukcje:
    - import **test1**
    - import **test2**

moduł **test1** zostanie zaimportowany tylko raz. Język Python pamięta importowane moduły i dyskretnie pomija wszelkie kolejne importy.

Załóżmy, że główny skrypt języka Python znajduje się w C:\Users\user\py\progs i nosi nazwę **main.py**

moduł, który mamy zamiar zaimportować znajduje się w **C:\Users\user\py\modules**

```
import sys
for p in sys.path:
    print(p)

path.append('C:\\Users\\user\\py\\modules')
```

została użyta metoda append - w rezultacie, nowa ścieżka zajmie ostatnie element w liście ścieżek; jeśli nie podoba się tobie ten pomysł, możesz w zamian skorzystać z metody insert() .