

# Podział projektu na pliki

Moduły i pakiety, moduły standardowe

# Moduły

**Modułem** nazywamy plik tekstowy zawierający kod źródłowy napisany w języku Python. Mogą znajdować się w nim elementy takie jak definicje funkcji, klas, zmiennych.

**Nazwa modułu** to nazwa pliku bez rozszerzenia.

moduł

**utils**



nazwa pliku

**utils.py**



Podział projektu na pliki

# Moduły

Nazwy (np. funkcje, klasy, zmienne) znajdujące się w modułach mogą być importowane do modułu nad którym pracujemy. Zaimportować, czyli sprawić, aby dana nazwa stała się dostępna.

Sposoby importowania modułów i nazw:

1. zaimportowanie całego modułu

```
import MODUŁ
```

2. zaimportowanie całego modułu pod nową nazwą

```
import MODUŁ as NOWA_NAZWA
```

3. zaimportowanie wybranych nazw

```
from MODUŁ import NAZWA1, NAZWA2
```



Podział projektu na pliki

# Moduły

Nazwy (np. funkcje, klasy, zmienne) znajdujące się w modułach mogą być importowane do modułu nad którym pracujemy. Zaimportować, czyli sprawić, aby dana nazwa stała się dostępna.

Sposoby importowania modułów i nazw:

4. zaimportowanie wybranej nazwy pod nową nazwą

```
from MODUŁ import NAZWA as NOWA_NAZWA
```

5. zaimportowanie wszystkich nazw z danego modułu

```
from MODUŁ import *
```



Podział projektu na pliki

# Moduły

Sposoby importowania  
modułów i nazw:

1. **import MODUŁ**
2. **import MODUŁ \**  
    **as NOWA\_NAZWA**
3. **from MODUŁ \**  
    **import NAZWA1, NAZWA2**
4. **from MODUŁ \**  
    **import NAZWA \**  
    **as NOWA\_NAZWA**
5. **from MODUŁ import \***

**Plik utils.py:**

```
def czy_liczba(napis):  
    # ...  
    return True
```

**Plik kalkulator.py:**

```
import utils  
  
print("czy_liczba():", utils.czy_liczba("13"))
```



Podział projektu na pliki

# Moduły

Sposoby importowania  
modułów i nazw:

1. **import** MODUŁ
2. **import** MODUŁ \   
 **as** NOWA\_NAZWA
3. **from** MODUŁ \   
 **import** NAZWA1, NAZWA2
4. **from** MODUŁ \   
 **import** NAZWA \   
 **as** NOWA\_NAZWA
5. **from** MODUŁ **import** \*

**Plik *utils.py*:**

```
def czy_liczba(napis):  
    # ...  
    return True
```

**Plik *kalkulator.py*:**

```
import utils as narzedzia
```

```
print("czy_liczba():", narzedzia.czy_liczba("13"))
```



Podział projektu na pliki

# Moduły

Sposoby importowania  
modułów i nazw:

1. **import** MODUŁ
2. **import** MODUŁ \
- as** NOWA\_NAZWA
3. **from** MODUŁ \
- import** NAZWA1, NAZWA2
4. **from** MODUŁ \
- import** NAZWA \
- as** NOWA\_NAZWA
5. **from** MODUŁ **import** \*

**Plik *utils.py*:**

```
def czy_liczba(napis):  
    # ...  
    return True
```

**Plik *kalkulator.py*:**

```
from utils import czy_liczba  
  
print("czy_liczba():", czy_liczba("13"))
```



Podział projektu na pliki

# Moduły

Sposoby importowania  
modułów i nazw:

1. **import** MODUŁ
2. **import** MODUŁ \   
 **as** NOWA\_NAZWA
3. **from** MODUŁ \   
 **import** NAZWA1, NAZWA2
4. **from** MODUŁ \   
 **import** NAZWA \   
 **as** NOWA\_NAZWA
5. **from** MODUŁ **import** \*

**Plik *utils.py*:**

```
def czy_liczba(napis):  
    # ...  
    return True
```

**Plik *kalkulator.py*:**

```
from utils import czy_liczba as sprawdz  
from math import ceil as sufit, floor as podloga  
  
print("czy_liczba():", sprawdz("13"))  
print("sufit():", sufit(3.3))  
print("podloga():", podloga(3.3))
```



Podział projektu na pliki



# Moduły

Sposoby importowania  
modułów i nazw:

1. **import** MODUŁ
2. **import** MODUŁ \
- as** NOWA\_NAZWA
3. **from** MODUŁ \
- import** NAZWA1, NAZWA2
4. **from** MODUŁ \
- import** NAZWA \
- as** NOWA\_NAZWA
5. **from** MODUŁ **import** \*

**Plik *utils.py*:**

```
def czy_liczba(napis):  
    # ...  
    return True
```

**Plik *kalkulator.py*:**

```
from utils import *
```

```
print("czy_liczba():", czy_liczba("13"))
```



Podział projektu na pliki

# Moduły

Sposoby importowania  
modułów i nazw:

1. **import** MODUŁ
2. **import** MODUŁ \
- as** NOWA\_NAZWA
3. **from** MODUŁ \
- import** NAZWA1, NAZWA2
4. **from** MODUŁ \
- import** NAZWA \
- as** NOWA\_NAZWA
5. **from** MODUŁ **import** \*

**utils.py:**

```
def czy_liczba(napis):  
    # ...  
    return True
```

**kalkulator.py:**

```
from utils import *
```

```
def czy_liczba(x):  
    # ...  
    return False
```

```
print("czy_liczba():", czy_liczba("13"))?
```

Która funkcja  
zostanie wywołana?



Podział projektu na pliki

# Moduły

Sposoby importowania  
modułów i nazw:

1. **import** MODUŁ
2. **import** MODUŁ \
3. **from** MODUŁ \
4. **from** MODUŁ \
5. **from** MODUŁ **import** \*

**utils.py:**

```
def czy_liczba(napis):  
    # ...  
    return True
```

**kalkulator.py:**

```
from utils import czy_liczba
```

```
def czy_liczba(x):  
    # ...  
    return False
```

```
print("czy_liczba():", czy_liczba("13"))
```

Czy po zmianie  
sposobu  
importowania nazw  
problem nadal  
występuje?



Podział projektu na pliki

# Moduły

**Podczas importowania modułu jego kod źródłowy jest interpretowany i wykonywany.** Może to powodować niepożądane zachowanie.

Specjalna zmienna `__name__` przechowuje nazwę modułu w postaci napisu. Wartość `"__main__"` oznacza, że moduł ten został uruchomiony bezpośrednio przez użytkownika.

Sprawdzenie wartości zmiennej `__name__` w module pozwala uniknąć wykonywania kodu modułu podczas jego importowania.



Podział projektu na pliki

# Moduły

Uruchomienie modułu  
kalkulator spowoduje  
wykonanie kodu  
znajdującego się w module  
utils.

Na ekranie zostaną  
wyświetlone linie  
zaczynające się od utils:  
oraz kalkulator:.

utils.py:

```
def czy_liczba(napis):  
    pass
```

```
print("utils: Kod modułu.")
```

```
print("utils: Wartość zmiennej __name__ to:",  
      __name__)
```

kalkulator.py:

```
import utils
```

```
print("kalkulator: Kod głównego programu.")
```

```
print("kalkulator: Wartość zmiennej __name__ to:",  
      __name__)
```

wykonuje kod modułu utils



Podział projektu na pliki

# Moduły

Rozwiązaniem  
jest sprawdzenie wartości  
zmiennej `__name__`  
w module `utils`.

Za pomocą instrukcji  
warunkowej należy  
zdecydować, czy określony  
fragment kodu ma zostać  
wykonany.

**utils.py:**

```
def czy_liczba(napis):  
    pass
```

```
if __name__ == "__main__":  
    print("utils: Kod modułu.")  
    print("utils: Wartość zmiennej __name__ to:",  
        __name__)
```

**kalkulator.py:**

```
import utils
```

```
print("kalkulator: Kod głównego programu.")  
print("kalkulator: Wartość zmiennej __name__ to:",  
    __name__)
```

nadal wykonuje kod  
modułu `utils`



Podział projektu na pliki

# Pakiety

**Pakiety** są sposobem na uporządkowanie modułów, jest **to kolekcja modułów**.

Fizycznie pakiet jest katalogiem mogącym<sup>1</sup> zawierać:

- specjalny moduł o nazwie `__init__` - w najprostszym przypadku jest to pusty plik, może także zawierać kod inicjalizujący pakiet (np. ustawienie zmiennych),
- moduły,
- pakiety.

1. dla wtajemniczonych: od Pythona 3.3 plik `__init__.py` jest opcjonalny, zachowanie to opisano w PEP 420 -- Implicit Namespace Packages, <https://www.python.org/dev/peps/pep-0420/>



Podział projektu na pliki

# Pakiety

Przykładowa struktura katalogów projektu.

Moduł kalkulator  
jest głównym skryptem  
do uruchomienia  
przez użytkownika.

Pakiet utils  
jest wykorzystywany  
przez moduł kalkulator.

```
.  
|-- kalkulator.py  
'-- utils  
    |-- __init__.py      (opcjonalny)  
    |-- strings.py  
    '-- user.py
```



Podział projektu na pliki



# Pakiety

Sposoby importowania nazw  
z pakietów:

1. **from** **PAKIET.MODUL** \  
    **import** **NAZWA**
2. **import** **PAKIET.MODUL**
3. **import** **PAKIET**  
    *(wymaga pliku `__init__.py`)*

**utils/strings.py:**

```
def czy_liczba(napis):  
    # ...  
    return True
```

**kalkulator.py:**

```
from utils.strings import czy_liczba  
  
print("czy_liczba():",  
      czy_liczba("13"))
```



Podział projektu na pliki

# Pakiety

Sposoby importowania nazw  
z pakietów.

1. **from** PAKIET.MODUL \  
    **import** NAZWA
2. **import** PAKIET.MODUL
3. **import** PAKIET  
    *(wymaga pliku \_\_init\_\_.py)*

**utils/strings.py:**

```
def czy_liczba(napis):  
    # ...  
    return True
```

**kalkulator.py:**

```
import utils.strings  
  
print("czy_liczba():",  
      utils.strings.czy_liczba("13"))
```



Podział projektu na pliki

# Pakiety

Sposoby importowania nazw  
z pakietów.

1. **from** PAKIET.MODUL \  
    **import** NAZWA
2. **import** PAKIET.MODUL
3. **import** PAKIET  
    (wymaga pliku `__init__.py`)

**utils/strings.py:**

```
def czy_liczba(napis):  
    # ...  
    return True
```

**utils/ \_\_init\_\_ .py:**

```
from . import strings
```

**kalkulator.py:**

```
import utils
```

```
print("czy_liczba():",  
      utils.strings.czy_liczba("13"))
```



Podział projektu na pliki

# Pakiety

Sposób importowania nazw  
**w obrębie pakietu.**

*Ten sposób importowania  
nazw może być również  
stosowany w pliku  
\_\_init\_\_.py.*

**utils/strings.py:**

```
def czy_liczba(napis):  
    # ...  
    return True
```

**utils/user.py:**

```
from .strings import czy_liczba
```

```
def pobierz_liczbe():  
    napis = input("Podaj liczbę:")  
    if czy_liczba(napis):  
        return int(napis)  
    else:  
        return False
```



Podział projektu na pliki

# Moduły standardowe

Python jest dostarczany wraz z szeroką gamą modułów, ich pełną listę można znaleźć na stronie:

<https://docs.python.org/3/library/>



Podział projektu na pliki

# Pytania

1. Czym jest moduł?
2. Jaka jest różnica pomiędzy modułem a pakietem?
3. Jakie znasz sposoby importowania modułów?



Podział projektu na pliki

# Literatura

1. Moduły i pakiety, <https://docs.python.org/3/tutorial/modules.html>
2. Lista wbudowanych pakietów Pythona, <https://docs.python.org/3/library/>



Podział projektu na pliki

