



Pystart.pl

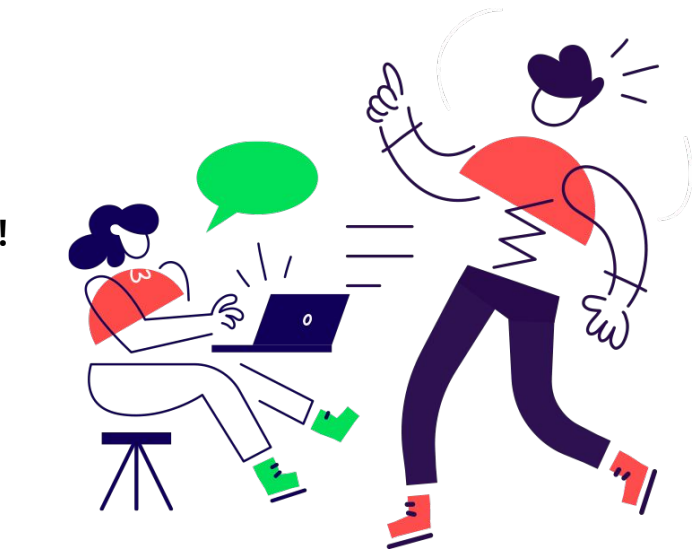
Podział projektu, pakiety, reużywalność.

lekcja dwudziesta ósma

PyStart #28 Podział projektu, reużywalność

Dlaczego dzielimy projekt?

- Separacja testów
- Grupowanie funkcji według zastosowania
- Przenoszenie pakietów między projektami
- Publikowanie pakietów open-source
- Łatwiejsza nawigacja po projekcie.. Ordnung muss sein!
- Praca zespołowa / dzielenie odpowiedzialności



PyStart #28 Podział projektu, reużywalność

Dwa dodatkowe pojęcia

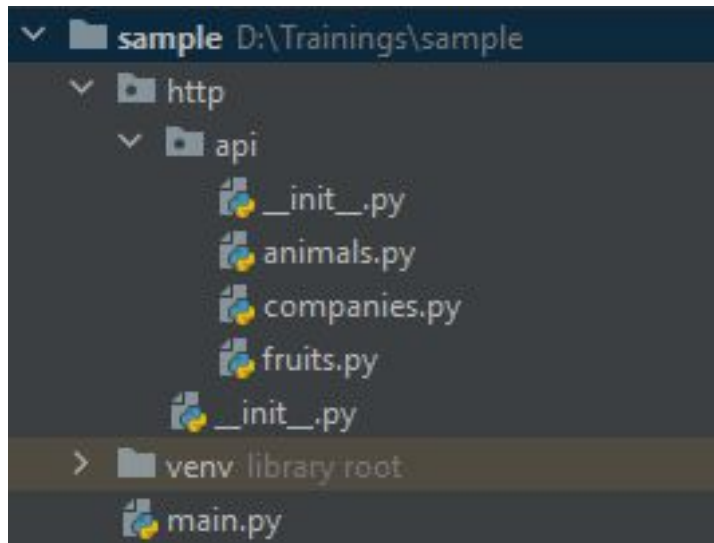
→ Moduł

→ Pakiet



PyStart #28 Podział projektu, reużywalność

Przykład



http to jest pakiet
api to jest pakiet

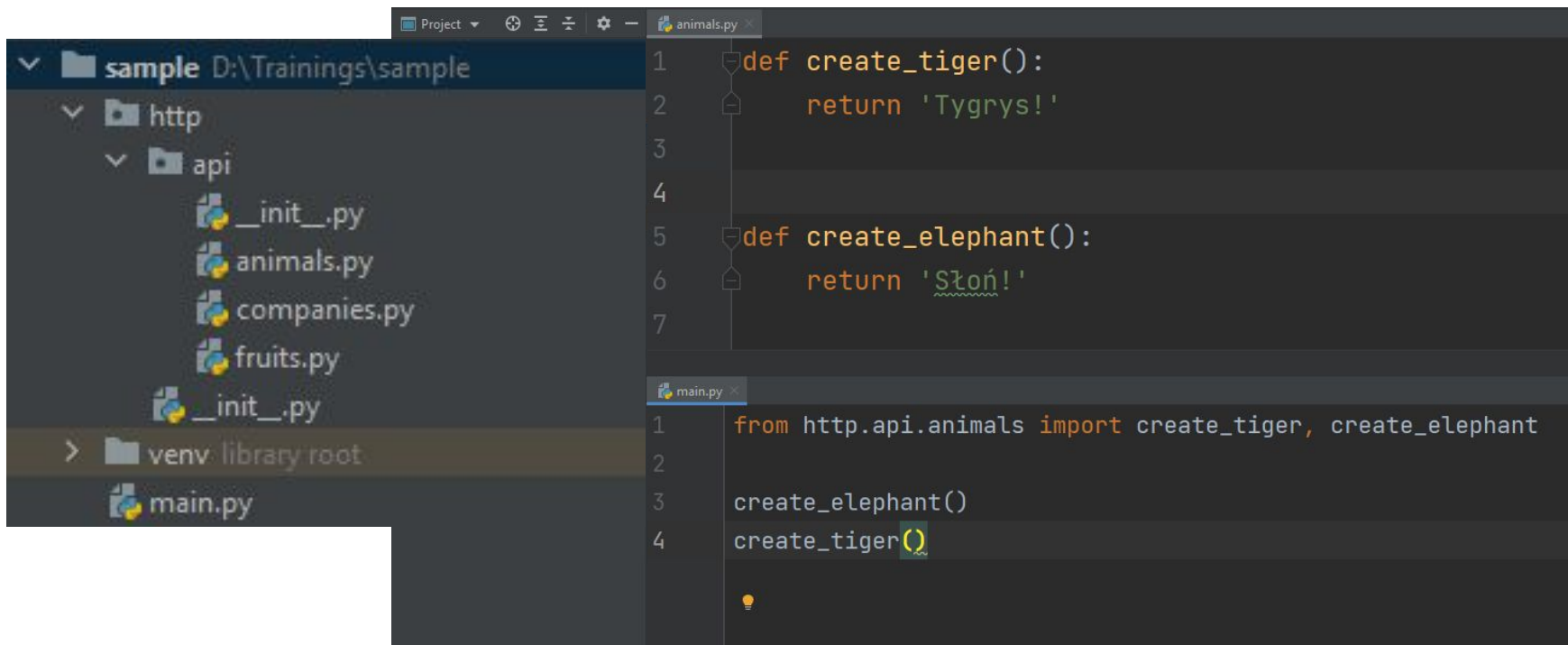
animals to moduł
companies to moduł
fruits to moduł



PyStart #28 Podział projektu, reużywalność

Jak to zaimportować?

Zawsze względem folderu w którym jest środowisko



The screenshot shows an IDE with a project structure on the left and two Python files open on the right. The project structure is as follows:

- sample D:\Trainings\sample
 - http
 - api
 - __init__.py
 - animals.py
 - companies.py
 - fruits.py
 - __init__.py
 - venv library root
 - main.py

The `animals.py` file contains the following code:

```
1 def create_tiger():
2     return 'Tygrys!'
3
4
5 def create_elephant():
6     return 'Słoń!'
7
```

The `main.py` file contains the following code:

```
1 from http.api.animals import create_tiger, create_elephant
2
3 create_elephant()
4 create_tiger()
```

PyStart #28 Podział projektu, reużywalność

Jak to zaimportować?

Zawsze względem folderu w którym jest środowisko

```
animals.py x
1 def create_tiger():
2     return 'Tygrys!'
3
4
5 def create_elephant():
6     return 'Słoń!'
7

main.py x
1 from http.api.animals import create_tiger, create_elephant
2
3 create_elephant()
4 create_tiger()
```

```
animals.py x
1 def create_tiger():
2     return 'Tygrys!'
3
4
5 def create_elephant():
6     return 'Słoń!'
7

main.py x
1 from http.api import animals
2
3 animals.create_elephant()
4 animals.create_tiger()
```

PyStart #28 Podział projektu, reużywalność

Jak to zaimportować?



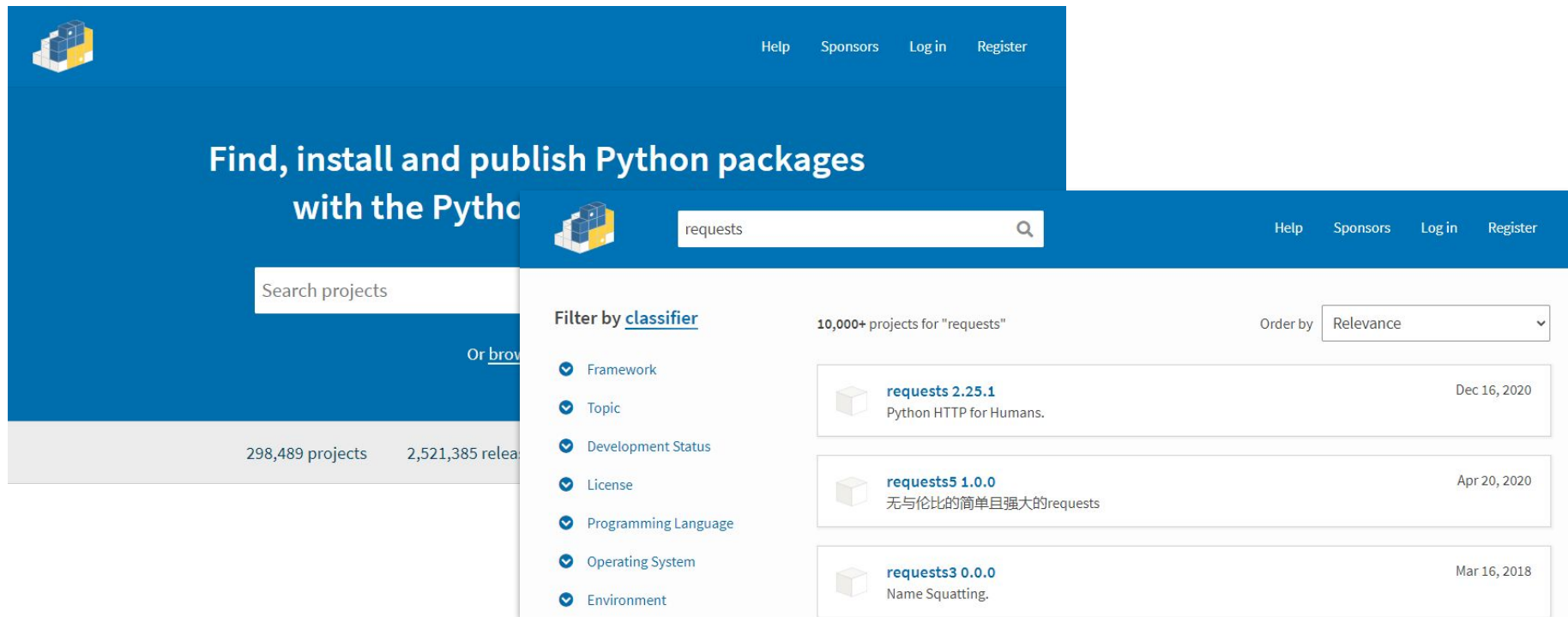
Tak samo z innymi funkcjami z venv / biblioteki standardowej

```
main.py x
1  from math import floor
2  from urllib.request import urlcleanup
3
4  floor(10)
5  urlcleanup()
6  |
7
8
```

PyStart #28 Podział projektu, reużywalność

Skąd brać pakiety?

→ Pypi.org



The screenshot displays the PyPI.org website interface. The main header is blue with the PyPI logo on the left and navigation links (Help, Sponsors, Log in, Register) on the right. Below the header, the text "Find, install and publish Python packages with the Python Package Index" is visible. A search bar contains the text "requests". Below the search bar, there are filters for "Framework", "Topic", "Development Status", "License", "Programming Language", "Operating System", and "Environment". The search results show 10,000+ projects for "requests", ordered by Relevance. The top results are:

- requests 2.25.1** (Dec 16, 2020): Python HTTP for Humans.
- requests5 1.0.0** (Apr 20, 2020): 无与伦比的简单且强大的requests
- requests3 0.0.0** (Mar 16, 2018): Name Squatting.

CZWARTA PRACA DOMOWA

UWAGA! UWAGA!

- Przygotuj funkcję, która będzie odbierała argumenty w postaci:

```
group_them(wall="red", tomato="red", light="yellow", lemon="yellow", grass="green")
```

W odpowiedzi funkcja powinna wyświetlić (w osobnych wierszach)

```
Wall is red \n Tomato is red \n Light is yellow itd.
```

- Korzystając z rekurencji wyświetl sumę wyrazów ciągu Fibonacciego

`fibonacci_sum(10)` powinno zwrócić:

```
1 + 1 + 2 + 3 + 5 + 8 + 13 + 21 + 34 + 55
```

- Korzystając z rekurencji znajdź największy wspólny dzielnik metodą Euklidesa.

- Przygotuj dekorator, który będzie wycinał listę według podanych przez użytkownika argumentów `@cut(1, 10)`
<- powinien wyciąć [1:10] z podanej listy

