Programowanie obiektowe

2021L

Co było ostatnio?

(25.02.2021)

Co było ostatnio?

- Cykl życia oprogramowania (od specyfikacji do utrzymania);
- modele / metodyki (ciężkie; zwinne) [Scrum!];
- manifest zwinnego wytwarzania oprogramowania;
- Python – hello world;
  - import this;
  - historia (Guido van Rossum, Monty Python, pull-request);
  - narzędzia (python, ipython, Colab, *.py, PyCharm).

Python – wprowadzenia ciąg dalszy

Python...

PVM

- Wydajność *Python* vs *C*. Dlaczego tak jest?
- Kod źródłowy (src.py) >>>kod bajtowy (src.pyc) >>> PVM (Python Virtual Machine).

Python...

PVM

Moduły

```
# module mod1.py

def foo(x):
    print(x)

def bar(x):
    print(x)
```

```
from mod1 import foo

foo("Zaskocz mnie")
```

```
import mod1

mod1.foo("Zaskocz mnie")
```

Python...

PVM
Moduły
Pakiety... (1)

```
user@host:~$ tree
.
├── my_pkg
    ├── __init__.py
    ├── mod1.py
    └── mod2.py
```

Struktura drzewa katalogu (kluczowy plik **__init__.py**).

Określenie, do których modułów jest dostęp po importowaniu pakietu (z gwiazdką):

```
# __init__.py
__all__ = ["mod1"]
```

Wywołanie:

```
from my_pkg import *

mod1.foo("Zaskocz mnie jeszcze raz")
# mod2.bar("A Ty też mnie zaskoczysz?")
```

Python...

Należy pamiętać:

- **`import *`** importuje wszystko, co jest wyszczególnione w **`__all__ = []`**;
- możemy importować moduły z wybranego pakietu, nawet jeśli nie są dodane do listy **`__all__`**.

```
from my_pkg import mod1

mod1.foo("Czuję się zaskoczony")
```

Python...

PVM
Moduły
Pakiety... (1)
Pakiety... (2)
Typy danych

Za pomocą kodu „na żywo” zapoznamy się z typami danych i operacjami jakie na zmiennych różnych typów możemy wykonywać.

Powiemy o:

- rzutowaniu; `# x = int(x)`
- listach; `# ["Adam", "Bartek"]`
- krotkach; `# (3.14, 15.92)`
- słownikach; `# {1:"Bartek", 2: "Wiktor"}`
- zbiorach (set; frozenset). `# {"jeden", "2", 3, 4, 1, 2, 3}`

Python... zaawansowane listy

List comprehension

Bez *if*

Zamiast pisać tak:

```
numbers = []

for i in range(10):
    numbers.append(i)

print(numbers)
#[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Można napisać tak:

```
numbers = [i for i in range(10)]
print(numbers)
#[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Python... zaawansowane listy

List comprehension

Bez *if*

Z *if*

Zamiast pisać tak:

```
numbers = []
for i in range(10):
    if i % 2 == 0:
        numbers.append(i)

print(numbers)

#[0, 2, 4, 6, 8]
```

Można napisać tak:

```
numbers = [i for i in range(10) if i % 2 == 0]
print(numbers)

#[0, 2, 4, 6, 8]
```

Python... zaawansowane listy

List comprehension

Bez *if*

Z *if*

Trudniej...

Zamiast pisać tak:

```
numbers = []

for i in range(10):
    for j in range(5):
        if i % 2 == 0 and j % 2 == 1:
            numbers.append(i + j)

print(numbers)

#[1, 3, 3, 5, 5, 7, 7, 9, 9, 11]
```

Można napisać tak:

```
numbers = [i + j for i in range(10) for j in range(5) \
        if i % 2 == 0 and j % 2 == 1]
print(numbers)

#[1, 3, 3, 5, 5, 7, 7, 9, 9, 11]
```

Ciekawostka

Ciekawostka

Nazwa pliku

Skrypt nie musi mieć rozszerzenia ***py***;

```
python script1
```

```
python script2.py
```

Moduł, aby można go było importować, musi być zapisany z rozszerzeniem ***py***.

Ciekawostka

Nazwa pliku
UTF-8

W starszych wersjach (2.x), Python nie pozwalał używać znaków z kodowania UTF-8 (np. polskich znaków diakrytycznych), dlatego należało każdy plik poprzedzić informacją o używanym kodowaniu (jeżeli chcieliśmy pisać: *zażółć gęślą jaźń*).

```
# -*- coding: utf-8 -*-
print("Zażółć gęślą jaźń")
```

Dziękuję za uwagę.