

Quo vadis Python 3.8?

Dominik Kozaczko



2019-06-26



Plan prezentacji

- Kto-zacz-ko?
- Python 3.8
 - `:=`
 - `/`
 - runtime audit
 - type-hinting & annotations
- Podsumowanie

Kto-zacz-ko?

- Senior Backend Engineer w **sunscrapers**
- Wkład w OpenSource: djoser, django-getpaid, hedju
- Programujący nauczyciel -> uczący programista
- Python na maturze
- kursy online
- Dni Wolnego Oprogramowania

Python 3.8

nowości



sunscrapers

Py3.8 - PEP 572 - mors, który wygrzył Guido

- dyskusja wokół Assignment Expressions (wyrażeń przypisujących?) sprawiła, że Guido ustąpił ze stanowiska BDFL

- składnia

NAZWA := expr

- zwykle - w szerszym kontekście, otaczane nawiasami:

```
if (y := f(x)) is not None:  
    process_more(y)
```

Py3.8 - PEP 572 - przykłady

Upraszcza obsługę regex

```
if (match := pattern.search(data)) is not None:  
    # zrób coś ze znajdką
```

Znakomicie upraszcza niektóre pętle

```
while chunk := file.read(8192):  
    process(chunk)
```

upraszczanie zapisu złożeń

```
results = [(x, y, x/y) for x in input_data if (y := f(x)) > 0]
```

```
stuff = [[y := f(x), x/y] for x in range(5)]
```

Py3.8 - PEP 572 - przykłady

```
# copy.py ; Py<=3.7
```

```
reductor = dispatch_table.get(cls)
if reductor:
    rv = reductor(x)
else:
    reductor = getattr(x, "__reduce_ex__", None)
    if reductor:
        rv = reductor(4)
    else:
        reductor = getattr(x, "__reduce__", None)
        if reductor:
            rv = reductor()
        else:
            raise Error(
                "un(deep)copyable object of type %s" % cls)
```

```
# copy.py ; Py>=3.8
```

```
if reductor := dispatch_table.get(cls):
    rv = reductor(x)
elif reductor := getattr(x, "__reduce_ex__", None):
    rv = reductor(4)
elif reductor := getattr(x, "__reduce__", None):
    rv = reductor()
else:
    raise Error("un(deep)copyable object of type %s" % cls)
```

Py3.8 - PEP 572 - przykłady

```
# sysconfig.py ; Py<=3.7
```

```
while True:
    line = fp.readline()
    if not line:
        break
    m = define_rx.match(line)
    if m:
        n, v = m.group(1, 2)
        try:
            v = int(v)
        except ValueError:
            pass
        vars[n] = v
    else:
        m = undef_rx.match(line)
        if m:
            vars[m.group(1)] = 0
```

```
# sysconfig.py ; Py>=3.8
```

```
while line := fp.readline():
    if m := define_rx.match(line):
        n, v = m.group(1, 2)
        try:
            v = int(v)
        except ValueError:
            pass
        vars[n] = v
    elif m := undef_rx.match(line):
        vars[m.group(1)] = 0
```


Py3.8 - PEP 570 - argumenty wyłącznie pozycyjne



imgflip.com

foto: Drake / imgflip

składnia:

```
def name(  
    positional_only_parameters, /,  
    positional_or_keyword_parameters, *,  
    keyword_only_parameters):
```



sunscrapers

Py3.8 - PEP 570 - przykłady

poprawne nagłówki funkcji

```
def name(p1, p2, /, p_or_kw, *, kw):
```

```
def name(p1, p2=None, /, p_or_kw=None, *, kw):
```

```
def name(p1, p2=None, /, *, kw):
```

```
def name(p1, p2=None, /):
```

```
def name(p1, p2, /, p_or_kw):
```

```
def name(p1, p2, /):
```

```
def name(p_or_kw, *, kw):
```

```
def name(*, kw):
```

Py3.8 - PEP 570 - ciekawostka

```
def foo(name, **kws):  
    return 'name' in kws
```

```
>>> foo(1, **{'name': 2})
```

Py3.8 - PEP 570 - ciekawostka

```
def foo(name, **kws):  
    return 'name' in kws
```

```
>>> foo(1, **{'name': 2})
```

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: foo() got multiple values for argument 'name'

Py3.8 - PEP 570 - ciekawostka

```
def foo(name, /, **kws):  
    return 'name' in kws
```

```
>>> foo(1, **{'name': 2})
```

Py3.8 - PEP 570 - ciekawostka

```
def foo(name, /, **kws):  
    return 'name' in kws
```

```
>>> foo(1, **{'name': 2})
```

True

Py3.8 - PEP 578 - runtime audit hooks



foto: Reza Estakhrian/Getty Images



sunscrapers

Py3.8 - type-hinting i anotacje

- PEP 585 - uporządkowanie roli typów wbudowanych w anotacjach
- PEP 578 - literal types - definicja zachowania w zależności od parametrów
- PEP 544 - kaczce protokoły ;)
- PEP 589 - TypedDict - ustalanie typów dla konkretnych kluczy
- wspólna pamięć w multiprocessing

DO YOU HAVE ANY

"QUESTIONS"?

DZIĘKI!