Python 3

Андрей Светлов

@andrew_svetlov
andrew.svetlov@gmail.com
http://asvetlov.blogspot.com

Строки

Только unicode

Utf-8 — кодировка по умолчанию

Неявные преобразования str↔bytes запрещены.

Числа

Unified int

Long — отсутствует

Дерево числовых типов

Decimal — теперь на C (x 20-120)

Function annotations

```
def f(a: int, b: float) -> str:
    return "{}:{}".format(a, b)

>>> f.__annotations__
{'a': builtins.int, 'b': builtins.float,
     'return': builtins.str}
```

Nonlocal

```
def f():
    a = 0
    def g():
        nonlocal a
        a += 1
    g()
    return a
```

Keyword only

```
>>> def f(a, b=0, *, c, d='Smith'):
...     pass
>>> f(1, c='John', d='Doe')
>>> f(1, c='Mary')
>>> f(1)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: f() needs keyword-only argument c
```

Extended iterable unpacking

```
a, *rest = range(5)
a, *rest, b = range(5)
```

ABC and new `super()`

```
class Base(metaclass=abc.ABCMeta):
    @abc.abstractmethod
    def f(self, a):
        """Comprehensive doc"""
        pass

class A(Base):
    def f(self, a):
        super().f(a)
```

Exception chain

```
def f():
    try:
        1 / 0
    except Exception as ex:
        raise RuntimeError("Oops") from ex
f()
```

Exception chain 2

```
Traceback (most recent call last):
  File "<string>", line 3, in f
    1/0
ZeroDivisionError: division by zero

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "<string>", line 7, in <module>
    f()
  File "<string>", line 5, in f
    raise RuntimeError("Division by zero") from ex
RuntimeError: Division by zero
```

Yield from

```
>>> def g(x):
...     yield from range(x, 0, -1)
...     yield from range(x)
...
>>> list(g(5))
[5, 4, 3, 2, 1, 0, 1, 2, 3, 4]
```

Новые метаклассы

```
class OrderedClass(type):
    @classmethod
    def __prepare__(metacls, name, bases, **kwds):
        return collections.OrderedDict()
    def __new__(cls, name, bases, classdict):
        result = type.__new__(cls, name, bases, dict(classdict))
        result.members = tuple(classdict)
        return result
class A(metaclass=OrderedClass):
    def one(self): pass
    def two(self): pass
    def three(self): pass
>>> A.members
('__module__', 'one', 'two', 'three')
```

Незаметные вкусности

yield from

New metaclasses

Новый GIL

Importlib

Stable ABI

PYC repository dirs

ABI version tagged .so files

Shared Dict

Вопросы?

@andrew_svetlov
andrew.svetlov@gmail.com
http://asvetlov.blogspot.com