Динамический мир Python - namespace, scope, LEGB rule, closure

Драль Алексей, study@bigdatateam.org
CEO at BigData Team, https://bigdatateam.org
https://www.facebook.com/bigdatateam

Дзен Python

- PEP 20 -- The Zen of Python

```
$ python -c "import this"
```

```
The Zen of Python, by Tim Peters
...
Namespaces are one honking great idea -- let's do more of those!
```

Окружения (namespace) в Python

1. Global namespace

BIGDATA TEAM

Окружения (namespace) в Python

1. Global namespace

```
$ ipython
```

Окружения (namespace) в Python

1. Global namespace

```
$ ipython
>> global_var = "global"
```

Окружения (namespace) в Python

1. Global namespace

```
$ ipython
>> global_var = "global"
>> "global_var" in globals()
True
```

BIGDATA TEAM

Окружения (namespace) в Python

1. Global namespace

```
>> dir() == sorted(globals().keys())
True
```

BIGDATA TEAM

Окружения (namespace) в Python

1. Global namespace

```
>> dir() == sorted(globals().keys())
True
>> isinstance(globals, object)
True
```

BIGDATA TEAM

Окружения (namespace) в Python

1. Global namespace

```
>> dir() == sorted(globals().keys())
True
>> isinstance(globals, object)
True
>> "globals" in dir()
```

BIGDATA TEAM

Окружения (namespace) в Python

1. Global namespace

```
>> dir() == sorted(globals().keys())
True
>> isinstance(globals, object)
True
>> "globals" in dir()
False
```

BIGDATA TEAM

Окружения (namespace) в Python

1. Global namespace
2. Built-in namespace

```
>> dir() == sorted(globals().keys())
True
>> isinstance(globals, object)
True
>> "globals" in dir()
False
>> "globals" in dir(__builtins__)
True
```

BIGDATA TEAM

Окружения (namespace) в Python

1. Global namespace
2. Built-in namespace

```
>> dir() == sorted(globals().keys())
True
>> isinstance(globals, object)
True
>> "globals" in dir()
False
>> "globals" in dir(__builtins__)
True
>> "global_var" in dir(__builtins__)
False
```

Built-in namespace

```
>> __builtins__
<module 'builtins' (built-in)>
```

Built-in namespace

```
>> __builtins__
<module 'builtins' (built-in)>
>> import builtins
>> builtins is __builtins__
True
```

Built-in namespace

```
>> __builtins__
<module 'builtins' (built-in)>
>> import builtins
>> builtins is __builtins__
True
>> vars(__builtins__)
{...
 'license': Type license() to see the
full license text,
...}
```

плюшки namespace

```
>> builtins.globals
<function globals()>
```

плюшки namespace

```
>> builtins.globals
<function globals()>
>> globals is builtins.globals is vars(builtins)["globals"]
True
```

плюшки namespace

```
>> builtins.globals
<function globals()>
>> globals is builtins.globals is vars(builtins)["globals"]
True
>> vars(builtins) == builtins.__dict__
True
```

плюшки **scope**

```
>> asdf
...
NameError: name 'asdf' is not defined
```

плюшки **scope**

```
>> asdf
...
NameError: name 'asdf' is not defined
>> import sys
>> sys.modules
{'sys': <module 'sys' (built-in)>, ...}
```

плюшки **scope**

```
>> asdf
...
NameError: name 'asdf' is not defined
>> import sys
>> sys.modules
{'sys': <module 'sys' (built-in)>, ...}
>> sys.builtin_module_names
{..., 'builtins', ..., 'sys', 'time', ...}
```

плюшки **scope**

```
>> asdf
...
NameError: name 'asdf' is not defined
>> import sys
>> sys.modules
{'sys': <module 'sys' (built-in)>, ...}
>> sys.builtin_module_names
{..., 'builtins', ..., 'sys', 'time', ...}
>> [x for x in sys.modules.keys() if "__" in x]
['__main__', '__future__']
```

Global namespace

```
>> vars(sys.modules["__main__"]) == globals()
True
```

Global namespace

```
>> vars(sys.modules["__main__"]) == globals()
True
>> sys.modules["__main__"].global_var
'global'
```

Global namespace

```
>> vars(sys.modules["__main__"]) == globals()
True
>> sys.modules["__main__"].global_var
'global'
>> sys.modules["__main__"].global_var is global_var
True
```

Global namespace

```
>> vars(sys.modules["__main__"]) == globals()
True
>> sys.modules["__main__"].global_var
'global'
>> sys.modules["__main__"].global_var is global_var
True
>> if __name__ == "__main__":
>>     print("hi")
hi
```

BIGDATA TEAM

Local namespace

```
global_var = "global"

def local_func(arg):
    local_var = "local"
    print(locals())
    print(global_var)
    print(globals()["global_var"])
```

```
>> local_func(arg=2)
???
```

Local namespace

```
global_var = "global"

def local_func(arg):
    local_var = "local"
    print(locals())
    print(global_var)
    print(globals()["global_var"])
```

```
>> local_func(arg=2)
{'arg': 2, 'local_var': 'local'}
global
global
```

Local namespace

```
global_var = "global"

def local_func(arg):
    local_var = "local"
    print(locals())
    print(local_func.__dict__)
```

```
>> local_func(arg=2)
{'arg': 2, 'local_var': 'local'}
{}
>> local_func.__dict__
{}
```

Namespace vs Scope

1. Global namespace
2. Built-in namespace
3. Local namespace
4. ¯_(ツ)_/¯

Namespace vs Scope

1. Global namespace
2. Built-in namespace
3. Local namespace
4. ¯_(ツ)_/¯

Scope:

1. Лексический (статический)

Namespace vs Scope

1. Global namespace
2. Built-in namespace
3. Local namespace
4. ¯_(ツ)_/¯

Scope:

1. Лексический (статический)
2. Динамический

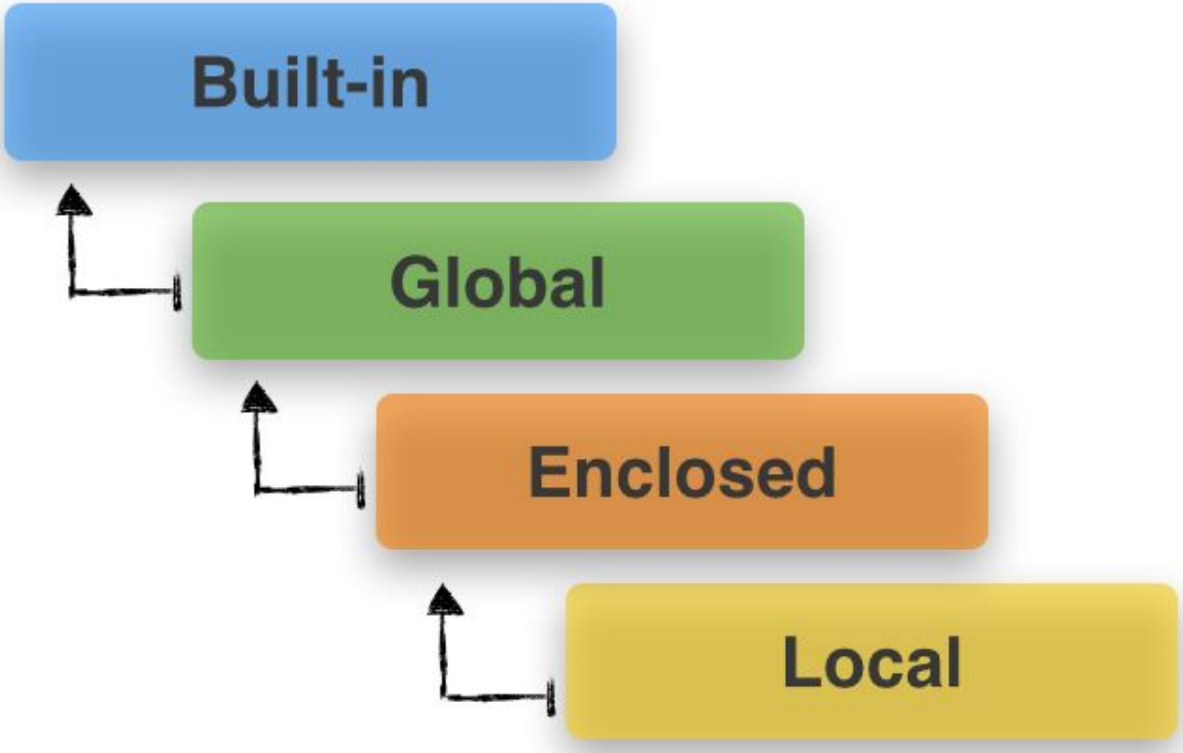
Local namespace

```
global_var = "global"

def local_func(arg):
    local_var = "local"
    print(global_var)
```

```
>> local_func(arg=2)
global
>> global_var = "xxx"
>> local_func(arg=2)
xxx
```

Сообщество LEGB



Правило LEGB

```
global_var = "global"

def local_func(arg):
    local_var = "local"
    global_var = "local_update"
    print(locals())
    print(global_var)
    print(globals()["global_var"])
```

```
>> local_func(arg=2)
{'arg': 2, 'local_var': 'local', 'global_var': 'local_update'}
local_update
global
>> global_var
'global'
```

BIGDATA TEAM

Правило LEGB

```
global_var = "global"

def local_func(arg):
    global global_var
    local_var = "local"
    global_var = "local_update"
    # print locals(), global_var, globals()["global_var"]
```

```
>> local_func(arg=2)
{'arg': 2, 'local_var': 'local', 'global_var': 'local_update'}
local_update
global → local_update
>> global_var
'global' → 'local_update'
```

Enclosing / Nonlocal namespace

```
def make_counter():
    count = 0  # enclosing var

    def inner():
        count = 1
        print(locals())
        return count

    print(locals())
    return inner
```

```
>> counter = make_counter()
{'count': 0, 'inner': <function
make_counter.<locals>.inner ...>}
>> counter()
{'count': 1}
1
>> counter()
{'count': 1}
1
```

BIGDATA TEAM

Enclosing / Nonlocal namespace

```
def make_counter():
    count = 0  # enclosing var

    def inner():
        count += 1
        print(locals())
        return count

    print(locals())
    return inner
```

```
>> counter = make_counter()
{'count': 0, 'inner': <function
make_counter.<locals>.inner ...>}
>> counter()
UnboundLocalError: local
variable 'count' referenced
before assignment
```

BIGDATA TEAM

Enclosing / Nonlocal namespace

```
def make_counter():
    count = 0  # enclosing var

    def inner():
        nonlocal count
        count += 1
        print(locals())
        return count

    print(locals())
    return inner
```

```
>> counter = make_counter()
{'count': 0, 'inner': <function
make_counter.<locals>.inner ...>}
>> counter()
{'count': 1}
1
>> counter()
{'count': 2}
2
>> counter()
{'count': 3}
3
```

Enclosing / Nonlocal namespace

```
def make_counter():
    count = 0

    def inner():
        nonlocal count
        count += 1
        print(locals())
        return count

    print(locals())
    return inner
```

```
>> counter = make_counter()
{'count': 0, 'inner': <function
make_counter.<locals>.inner ...>}
>> counter(), counter()
...
>> counter.__closure__
(<cell at ...: int object at ...>,)
```

Enclosing / Nonlocal namespace

```
def make_counter():
    count = 0

    def inner():
        nonlocal count
        count += 1
        print(locals())
        return count

    print(locals())
    return inner
```

```
>> counter = make_counter()
{'count': 0, 'inner': <function
make_counter.<locals>.inner ...>}
>> counter(), counter()
...
>> counter.__closure__
(<cell at ...: int object at ...>,)
>> counter.__closure__[0].cell_contents
3
```

BIGDATA TEAM

Enclosing / Nonlocal namespace

```
def make_counter():
    count = 0

    def inner():
        nonlocal count
        count += 1
        print(locals())
        return count

    print(locals())
    return inner
```

```
>> counter = make_counter()
{'count': 0, 'inner': <function
make_counter.<locals>.inner ...>}
>> counter(), counter()
...
>> counter.__closure__
(<cell at ...: int object at ...>,)
>> counter.__closure__[0].cell_contents
3
>> counter.__closure__[0].cell_contents = 5
>> counter()
{'count': 6}
6
```

Итого

Теперь вы умеете в:

- Python Namespace

Итого

Теперь вы умеете в:

- Python Namespace
- Python import

Итого

Теперь вы умеете в:

- Python Namespace
- Python import
- LEGB rule

Итого

Теперь вы умеете в:

- Python Namespace
- Python import
- LEGB rule
- Static & dynamic scope

Итого

Теперь вы умеете в:

- Python Namespace
- Python import
- LEGB rule
- Static & dynamic scope
- Closure

Итого

Теперь вы умеете в:

- Python Namespace
- Python import
- LEGB rule
- Static & dynamic scope
- Closure