

Projet FAT Python

Optimiseur statique pour Python



/dev/var, décembre 2015, Toulon

Distribué sous licence CC BY-SA

Victor Stinner
vstinner@redhat.com

Python est lent



- Plus lent que le **C**, langage « compilé »
- Plus lent que **Javascript** et ses compilateurs à la volée (JIT)
- **PyPy** reste peu utilisé (problème de l'API C de CPython)

Objectif



Remplacer

```
def func():  
    return len("abc")
```

par

```
def func():  
    return 3
```


Problème



- **Tout** est modifiable en Python
- Fonctions *builtins*
- Code d'une fonction
- Variables « locales » modifié par une autre fonction
- etc.

Problème



Remplacer la fonction len() :

```
builtins.len = lambda obj: "mock!"  
print(len("abc"))
```

Contraintes



- Respecter la « sémantique » du langage Python
- Ne pas casser les applications
- Ne pas avoir besoin de modifier le code source des applications

Gardes



- Tester des conditions à l'exécution
- Est-ce que `builtins.len` a été modifié ?
- Est-ce que `globals()['len']` a été modifié ?

Gardes



Pseudo-code :

```
if teste_gardes():  
    # rien n'a été modifié  
    fast_func()  
else:  
    # len() a été remplacé  
    func()
```


AST



- AST : *Abstract Syntax Tree*
Arbre syntaxique abstrait
- Code source \Rightarrow AST \Rightarrow Bytecode

AST



Appel `len("abc")` :

```
Call(func=Name(id='len', ctx=Load()),  
      args=[Str(s='abc')])
```

Nombre 3 :

```
Num(n=3)
```

Optimiseur AST



```
from ast import NodeTransformer  
  
class Optimizer(NodeTransformer):  
    def visit_Call(self, node):  
        return ast.Num(n=3)
```


Optimisations



- Appeler les fonctions *builtins*
- Dérouler les boucles
- Propager les constantes
- *Constant folding*
- Elimination du code mort
- Convertir les fonctions *builtins* en constantes

Propager constantes



```
def func():  
    x = 1  
    y = x + 1  
    return y
```

=>

```
def func():  
    x = 1  
    y = 1 + 1  
    return y
```

Constant folding



Propager les constantes et *constant folding*:

```
def func():          =>  def func():  
    x = 1             x = 1  
    y = x + 1         y = 2  
    return y         return 2
```


Dérouler les boucles



```
for i in range(3): => i = 0  
    print(i)           print(i)
```

```
    i = 1  
    print(i)
```

```
    i = 2  
    print(i)
```

Dérouler les boucles



Dérouler les boucles et propager les constantes :

```
for i in range(3): => i = 0  
    print(i)          print(0)
```

```
i = 1  
print(1)
```

```
i = 2  
print(2)
```

Dérouler les boucles



Dérouler les boucles, propager les constantes et supprimer les variables inutiles :

```
for i in range(3): => print(0)  
    print(i)
```

```
    print(1)
```

```
    i = 2  
    print(2)
```


Questions ?



http://faster-cpython.rtfld.org/fat_python.html