

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)

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"))
```

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



Pseudo-code :

```
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

Dérouler les boucles



```
for i in range(3):  
    print(i)
```

Dérouler les boucles



```
i = 0  
print(i)
```

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

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

Questions ?



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