



# Python : langage homogène, explicite, et efficace

Pycon FR 2011 Rennes

Présenté par

**Victor Stinner**

<[victor.stinner@haypocalc.com](mailto:victor.stinner@haypocalc.com)>

# Sommaire

---

1. Avertissement

2. Homogène

3. Explicite

4. Efficace

Avertissement

# Avertissement

---

- Compare Python aux langages que je connais
- Compare le langage et non la bibliothèque standard
- Utilise les autres langages comme illustration



# Mon parcours

---

- QBASIC
- Turbo Pascal, assembleur Intel x86
- Turbo C, Borland C++ Builder, Delphi
- PHP, HTML, Javascript
- Bash, Python
- *What else?*

Homogène

# Orienté objet

---

- C
  - `fgets(buffer, size, file)`
- C
  - `read(file, buffer, size)`
- PHP
  - `trim(" abc ")`

# Orienté objet

---

- Python
  - `file.readline()`
  - `file.read(size)`
  - `" abc ".strip()`



# item in set

---

- PHP
  - `in_array($a, $b)`
  - `array_key_exists($a, $b)`
- C
  - `strstr(a, b)`
- Python
  - `a in b`

# item in set

---

- PHP
  - `in_array($set, $item)`
  - `array_key_exists($item, $set)`
- C
  - `strstr(set, item)`
- Python
  - `item in set`

# item in set

---

- PHP : `isset($set[$item])`
- Python
  - `item in list`
  - `item in tuple`
  - `item in dict`
  - `item in set`
  - `item in obj`    `# obj.__contains__(item)`
  - ...



# for x in y (foreach)

- C++

```
std::vector<int> liste;  
std::vector<int>::const_iterator it;  
for (it=liste.begin(); it != liste.end(); ++it)  
    std::cout << *it << std::endl;
```

- Python

```
for item in liste:  
    print(item)
```



# for x in y (foreach)

- C++0x

```
int liste[5] = {1, 2, 3, 4, 5};  
foreach(int x: liste)  
    ...  
foreach(int &x: liste)  
    x *= 2;
```

- Python : tuple, list, dict, set, fichier (ligne par ligne), ... (`__iter__`, `__next__`)

# Module, classe, attribut

- Perl
  - `Module::fonction`, `Class->methode`,  
`$objet->attribut`
- C
  - `struct.attribut`, `ptr->attribut`
- C++
  - `Classe::methode`, `this->attribut`,  
`objet->attribut`

# Module, classe, attribut

- Python
  - Module.fonction
  - Classe.methode
  - objet.methode
  - self.attribut
  - objet.attribut



# Callback

---

- PHP
  - `func($data)`
  - `process('func', $data) : func($data)`
- Perl
  - `func($data)`
  - `process(func, $data) : func->($data)`
- Python
  - `func(data)`
  - `process(func, data) : func(data)`



Explicite

# Variables magiques

- Bash, Perl et Ruby : conventions

- Perl

```
if( $texte =~ /cle=(.*)/ )  
    $valeur = $1;
```

- Bash

```
echo "pid=$$"
```

- Bash

```
$command = $1  
$filename = $2
```

# Variables magiques

- Python : explicite

```
import re
m = re.match("cle=(.*)")
if m:
    valeur = m.group(1)
```

```
import os
print("pid=%s" % os.getpid())
```

```
import sys
command = sys.argv[1]
filename = sys.argv[2]
```

# Formattage chaîne

- Bash, Perl, PHP

"Bonjour **\$prenom**"

- Python

"Bonjour **%s**" % **prenom**

"Bonjour **{}**".format(**prenom**)



# Pas d'ASCII Art

---

- C, PHP, Perl

test ? a : b

- Bash, C

a && b, a || b, !a

- PHP

dico = Array('cle' => 'valeur')

# Pas d'ASCII Art

---

- Python
  - a **if** test **else** b
  - a **and** b, a **or** b, **not** a
  - dico = {'cle': 'valeur'}
- Python
  - @decorateur
- PHP 5.4
  - liste = [1, 2, 3, 4, 5]

# Gestion d'erreur

- PHP (et Perl)

```
$f = open("file.txt") or die("Aaaarg");  
content = stream_get_contents($f);
```

- PHP (et C)

```
$f = @fopen("file.txt");  
if (isset($f))  
    $content = stream_get_contents($f);  
else  
    echo "impossible d'ouvrir file.txt\n";
```



# Gestion d'erreur

---

- Python

**try:**

```
f = open("file.txt")  
content = f.read()
```

**except IOError, err:**

```
print("impossible d'ouvrir file.txt")
```



# Gestion d'erreur

- C

```
f = fopen("file.txt", "r");  
fread(buffer, 4096, 1, f);  
# plantage !
```

- Python

```
f = open("file.txt")  
# lève une exception IOError  
content = f.read()
```

Efficace

# List comprehension

- `pairs = (x for x in xrange(1000000)  
          if (x % 2) == 0)`
- `doubles = (x*2 for x in pairs)`
- `for x in doubles:  
    if x > 1000:  
        break  
    print(x)`
- Un seul élément en mémoire
- Rapide



# List comprehension

---



```
for x in (x*2 for x in xrange(1000000))
    if (x % 2) == 0:
        if x > 1000:
            break
        print(x)
```

# Générateur

---

```
def fibonacci():  
    u, v = 0, 1  
    while True:  
        yield u  
        u, v = v, u+v  
  
for n in fibonacci():  
    if n > 100:  
        break  
    print(n)
```

# Générateur

---

- Condition d'arrêt gérée par le consommateur
- Code simple
- Un seul élément en mémoire
- Rapide



Bonus

# Défauts de Python

- `a = "abc",` # a est un tuple
- `print "abc",` # sans retour à la ligne
- `func((a,))` # pas très lisible
- `len(obj)` # au lieu de `obj.len()`
- Pas d'enum -> bibliothèques
- Pas de constante

# Mot clé `with`

---

- Fichier

```
with open("document.txt") as f:  
    content = f.read()  
    # f.close()
```

- Verrou

```
lock = threading.Lock()  
with lock:  
    # lock.acquire()  
    ... # section critique  
    # lock.release()
```



# Argument par mot clé

- Python

```
def open(filename, mode="r",  
          encoding="utf-8",  
          errors="strict"):  
    ...
```

```
open("document.txt")  
open("document.txt", encoding="latin1")  
open("document.txt", "w", errors="replace")
```

# Argument par mot clé

- PHP

```
function open($filename, $args)
{
    $defaults = Array(
        "mode" => "r",
        "encoding" => "utf-8",
    );
    $args = array_merge($defaults, $args);
    ...
}
open("document.txt",
    Array("encoding"=>"latin1"));
```

# Questions & Discussion



Contact:

victor.stinner@haypocalc.com



Merci à David Malcom pour le  
modèle LibreOffice

<http://dmalcolm.livejournal.com/>