Processus de développement de CPython

Pycon FR 2012, Paris

Antoine Pitrou
<solipsis@pitrou.net>

Victor Stinner
<victor.stinner@gmail.com>

Sommaire

1. Aspect humain
2. Améliorations
3. Exemples de PEP
4. Côté technique

python™

Aspect humain

Communauté ouverte

- Développement de Python (langage et stdlib)
  ~= développement de CPython
- Discussions publiques et ouvertes à tous
- ... sauf sécurité et infrastructure

Liste python-ideas

- Brainstorm mondial
- Propositions maladroites dues à une mauvaise connaissance du langage
- Changements de syntaxe rarement acceptés
- Meilleures idées transformées en PEP

python™

Liste python-dev

- Propositions et questions concrètes sur l'implémentation de CPython
- Revue des commits
- Discussions sur les PEP en cours

python™

bugs.python.org

- Très actif
- Triage des tickets
- Reproduction et analyse d'un bug
- Choix parmis plusieurs solutions techniques
- Relecture des patchs

python™

Améliorations

Correction de bug

- Implémentation discutée sur le bug tracker
- Revue de code
- Commit dans Python 2.7, 3.2, 3.3 et 3.4

Ajout de fonctionnalité

- Proposition d'évolution de Python
  - *Python Enhancement Proposal (PEP)*
- ... ou pas PEP
- Implémentation discutée sur le bug tracker
- Revue de code
- Commit dans Python 3.4 (branche default)

python™

Améliorations

- Performances, consommation mémoire, stabilité
- Revue de code
- Souvent plusieurs versions d'un patch
- Commit dans Python 2.7, 3.2, 3.3 et 3.4

python™

Exemples de PEP

PEP 410 : nanosecondes

- Bug os.utime(dst, os.stat(src).st_mtime)
- Utiliser le type Decimal comme timestamp
- Précision d'une nanoseconde (10^{-9})
- La PEP liste 7 types possibles

python™

PEP 410 : rejetée

- Précision théorique, inaccessible en pratique
- Ajout d'une complexité injustifiée / inutile
- PEP rejetée par Guido van Rossum
- Patch juste pour os.utime() et os.stat() : nombre de nanosecondes (int)

Rejet d'une PEP

- Besoin trop spécifique
- Manque de cas d'utilisation
- Solution existante satisfaisante
- Refus servant à conserver un langage simple, homogène et cohérent
- PEP différées

python™

PEP 3151

- Refonte de la hiérarchie d'exceptions d'entrée-sortie
- IOError, OSError, EnvironmentError, ... fusionnés
- Nouvelles exceptions plus fines basées sur errno : FileNotFound (ENOENT), BlockingIOError (EAGAIN), ...

python™

PEP 3151 : acceptée

- Conception délicate : préserver la compatibilité
- Écriture longue : recenser les usages, argumentaire
- Discussion plus aisée que prévue : débats sur le nommage

python™

PEP 418

- Ajouts au module time de Python 3.3 :
- time.get_clock_info(name)
- time.monotonic()
- time.perf_counter()
- time.process_time()

python™

PEP 418 : débat houleux

- Débat de plusieurs semaines avec une dizaine d'intervenants
- Débat sur le vocabulaire : "accuracy", "monotonic", "steady"
- Débat sur monotonic() : fallback ou non ? (non!)
- Difficile définition des fonctions (doc)

PEP 418 : acceptée

- Après de nombreuses révisions de la PEP,
- PEP acceptée dans Python 3.3
- PEP avec de nombreuses annexes sur les horloges matérielles, systèmes d'exploitation, performances, etc.

python™

Côté technique

Suite de tests

Nombre et proportion de lignes de test (Lib/test)

- 2.0 : 11 000 lignes de test (4,5 %)
- 2.7 : 117 000 lignes (14 %)
- 3.3 : 167 000 lignes (21 %)

- Couverture des modules écrits en Python (Lib/) : 75% mesurés par Brett Cannon sur Python 3.3

python™

Suite de tests

- Tests unitaires
- Tests de stress (threads)
- Robustesse croissante (mais imparfaite)
  - Bugs sporadiques
  - Timeout
  - Problèmes externes
- Tests instables sur certains OS (theads et signaux sous BSD)

python™

Intégration continue

- Ferme de buildbots maintenus par la communauté
- Compilation et tests en debug
- Exécution des tests en série ou en parallèle

python™

Buildbots stables

- Bloquants pour une release
- Systèmes : FreeBSD 9, Gentoo, OpenIndiana, OS X Lion, RHEL 6, Ubuntu, Windows 7, Windows XP
- Architectures : x86, x86_64
- Compilateurs : gcc, clang, MSVC

Buildbots instables

- Indicatifs
- Pour développeurs courageux : échouent souvent

python™

Buildbots instables

- Systèmes : DragonflyBSD 3, Fedora, FreeBSD 6/7/8/9/10, Gentoo, NetBSD 5, OpenBSD 5, OpenIndiana, OS X Mountain Lion/Snow Leopard/Tiger, Solaris 10, Ubuntu, Windows 8/Server 2003/Server 2008
- Architectures : x86, x86_64, IA64, PA-RISC, SPARC

Buildbots spéciaux

- Compilation en mode release
- bigmem : 24 Go et 6 heures par build

python™

Contribuez !

Contribuer à Python

- Developer Guide écrit par Brett Canon
- http://docs.python.org/devguide/
- Liste core-mentorship
- http://www.python.org/dev/
- Pas besoin du droit de commit (push)

python™

Questions & Discussion

Contact:
solipsis@pitrou.net
victor.stinner@gmail.com

Merci à David Malcom pour le modèle LibreOffice

http://dmalcolm.livejournal.com/