HowDoWeDoPython3Migration

On essaie au maximum d'avoir un code commun entre Python 2 et Python 3; souvent c'est possible de manière native, le reste du temps via six (pour les applications django six est dispo de base dans django.utils.six).

Éléments usuels :

- iteritems() n'existe plus en Python 3, vu les volumes de données qu'on traite, on peut partout utiliser items().
- le résultat de .values() sur un dictionnaire n'est pas de type liste, il faut donc le transformer si on veut juste en prendre un bout, genre get_wcs_services().keys()[0] → list(get_wcs_services().keys())[0].
- sort() utilise obligatoirement un paramètre key, plus moyen de faire avec cmp. ex: references.sort(lambda x, y: cmp(x[-1], y[-1])) → references.sort(key=lambda x: x[-1])
- les exceptions s'attrapent nécessairement avec la forme except Whatever as e, plus possible d'utiliser une virgule.
- on se trouvait parfois par erreur ou désir stylistique à préfixer des entiers avec un 0, genre date(2017, 01, 03), on ne peut plus faire ça.
- il est déconseillé d'utiliser file() pour ouvrir un fichier depuis Python 2.5, il faut utiliser open(), c'est désormais obligatoire.

Pour la gestion des encodages, utiliser force_text/smart_text/smart_bytes de Django, plutôt que des appels à unicode() ou str().

Via six :

La bibliothèque standard a été un petit peu rangée et certains modules ne se trouvent plus là où ils étaient; six assure la transition.

- urllib & urlparse, from django.utils.six.moves.urllib import parse as urlparse. (et/ou as urllib), ça donne accès à urlparse, urlunparse, parse_qs, quote, etc.
- from HTMLParser import HTMLParser → from django.utils.six.moves.html_parser import HTMLParser

Quand il faut nécessairement du code différent entre Python 2 et Python 3, plutôt que comparer les versions via sys.version_info il existe six.PY3 (vrai en Python 3) et six.PY2 (vrai en Python 2).