PENSANDO EN APIS

CONSEJOS Y REFLEXIONES SOBRE EL DISEÑO DE APIS EN PYTHON

Juan B Cabral <jbc.develop@gmail.com> - http://jbcabral.wordpress.com

-START-

* Mucho de esto esta sacado de Java
* Java sirve para algo: Te enseña como hacer cosas feas pensando en que si no te esforzás van a quedar feas.
* Surge como una duda personal de como saber si lo que hago esta bien.
* Un api malo no deja de funcionar, solo es malo.
* Recomiendo un libro:
Practical API Design - Jaroslav Tulach (http://goo.gl/q8d7j)

STAGE 1 - API VS SPI

API

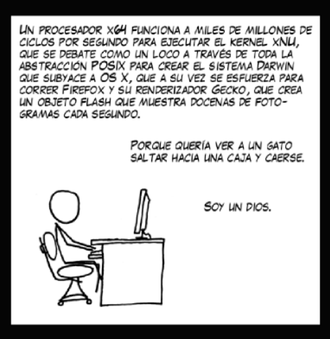
* Es la interfaz de un programa con el mundo.
* The API is the description of object... that you call and use to achieve a goal and
* Un coso externo le pide algo a nuestro programa y luego el coso recupera el control.

SPI

* The SPI is the description of object... that you extend and implement to achieve a goal
* API subset.
* Es la interfaz de un programa con un plugin.
* Nuestro programa le pide algo a un coso externo y luego nuestro programa recupera el control.

STAGE 2 - CLUELESS

* Clueless o "Falta de Idea" es el concepto de que encaramos un problema sabiendo todo lo necesario para resolverlo. En algún punto esto es inecesario por que
hay partes de nuestro enigma que ya lo resuelven librerías de terceros.
* Nadie sabe todo lo necesario para volar un avión.
* La ignorancia es un beneficio.
* Nos ayudan a enfocar en un problema.
* Esta para quedarse.
* No significa "no saber".
* Python es altamente "clueless" ya que nos aleja de conceptos como tipos, manejo de memoria y cosas así.



STAGE 3 - ZEN VS ZEN

Las librerías almenos contradicen de alguna manera el "zen" de python:

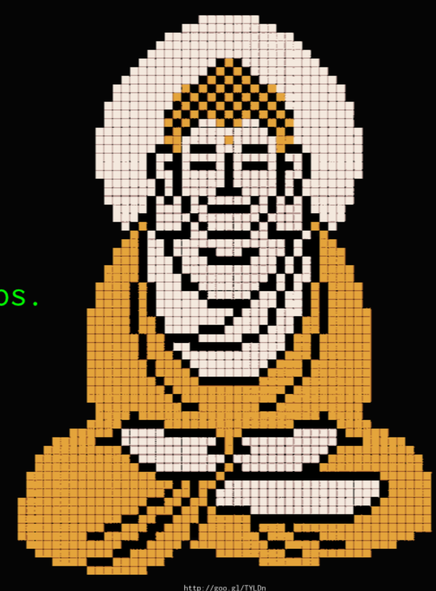
* Explicit is better than implicit.
* Flat is better than nested.
* Special cases aren't special enough to break the rules.
- There should be one --and preferably only one-- obvious way to do it.

Recordar

* Although practicality beats purity.
* Namespaces are one honking great idea -- let's do more of those!

BONUS STAGE

Como no me da la cara para decir "esta es la posta"; agrupe el resto en una sucesión de consejos.



http://goo.gl/TYLDn

STAGE 4 - CONSEJO: TAMAÑOS DE API

* Exponer solo los métodos necesarios
* Tamaño de un API: Mientras más Chico Mejor (len(dir(obj)) debería ser un número chico.)
* No exponer jerarquías profundas: No es lo mismo diseñar para la API que para reusar código.

STAGE 5 - CONSEJO: COOPERACIÓN CON OTRAS APIS

* Compatibilidad con las pilas.
* Seguir la PEP 8.
* Ojo con retornar objetos de otras APIs (disminuye el clueless).
* Ojo con redefinir comportamiento de otras APIs (aumenta el acoplamiento).

STAGE 6 - CONSEJO: TIPOS

* De preferencia NO exponer objetos propios como resultados de operaciones.
* Son preferibles los tipos nativos.
* Aplica también a mundo web:
* XML: NO
* JSON/YAML: SI (Da tipos nativos)
* Los controles de tipos deben hacerse en el nivel de API.
* Los Controles de tipos llevan tiempo.
* Los assert son buenas ideas para validar tipos.
* Ojo con el retorno de valores nulos (None != default).

STAGE 7 - CONSEJO: INMUTABILIDAD

* Si van a definir objetos intentar que sean inmutables (aumenta bastante la estabilidad de la librería... bueno no realmente)
* Darle muchos derechos al constructor para asignar atributos.

* Si un objeto es inmutable:
TRATAR de redefinir __repr__, __str__, __hash__, __cmp__, __eq__ y __ne__.

* Si un objeto es mutable:
* controlar mucho lo que llega por las API.
* Redefinir: __repr__, __str__, __cmp__, __eq__ y __ne__.

STAGE 8 - CONSEJO: DISEÑO

* Siempre planeen primero la funcionalidad.
* TDD.
* Primero el controller (MVC).
* Plantear inicialmente el nivel de excelencia que se quiere llegar.

STAGE 9 - CONSEJO: PORTANDO

* Respetar pep8: assertTrue -> assert_true/ asserttrue
* Utilizar funciones de python: obj.length() -> len(obj)
* Utilizar métodos de python: obj.appendChild(aobj) -> obj.append(aobj)

STAGE 10 - CONSEJO: PUBLICACIÓN

* No publiquen sin tests.
* TDD se merece una oportunidad.
* Publiquen de manera comunes a los developers python (pypi es mejor ppa).
* No publiquen sin documentación.
* Vean la pagina de la gente de Pocoo (http://www.pocoo.org/)
* Dar identidad gráfica al proyecto.

FINAL STAGE - CONSEJOS FINALES

* Hacer que nuestras cosas internas cumplan muchas de las cosas que dijimos.
* Las APIs simétricas son buena idea (si un API tiene load que tenga dump).
* Tratar de cumplir en su totalidad el zen de python.
* Compatibilidad para atrás es algo a tener en cuenta.
* No abusar de los patrones.
* Todo es cuestión de diseño (DRY or not DRY)
* Evitar el monkeypatch.