Python
the good parts

Miguel Araujo

@maraujop

import this

El zen de Python - PEP20

- Explícito es mejor que implícito
- Plano es mejor que anidado
- La legibilidad cuenta
- Debería haber una-- y preferiblamente solo una-- forma obvia de hacer algo.

Pythonic / Pythonico

No pythonico

```
names = ['Juan', 'Pedro', 'Isabel']
i = 0

while (i < len(names)):
    print names[i]
    i += 1
```

No pythonico

```
names = ["Juan", "Pedro", "Isabel"]


for i in range(len(names)):
    print names[i]
```

Pythonico

```python
names = ["Juan", "Pedro", "Isabel"]

for name in names:

    print name
```

No pythonico

```
if name == "Juan" or name == "Pedro" or name == "Maria":
```

Pythonico

```
if name in ("Juan", "Pedro", "Maria"):
```

No pythonico

```
numbers = [1, 2, 3, 4]
total = 0
for number in numbers:
    total += number
```

Pythonico

```
total = sum(numbers)
```

Coding style

Utiliza siempre un coding style

Coding style - PEP8

“*Code is read much more often than is written*
Guido Van Rossum

Coding style - PEP8

- Indentar con 4 espacios
- Columnas de 80 ó 100 caracteres
- Separar clases y funciones de primer nivel por 2 espacios
- Imports en lineas diferentes

```
import sys

import os
```

Coding style - Espacios

- **Sí:** spam(ham[1], {eggs: 2})

  No: spam(ham[1], { eggs: 2 })
- **Sí:** x, y = 4, 5

  No: x , y = 4 , 5
- **Sí:** def publish(message=None)

  No:def publish(message = None)

Coding style - Espacios

Sí:

```
x = 1
long_variable = 4
```

No:

```
x             = 1
long_variable = 4
```

Coding style - indentar

```
people = {
    "nombre": "Juan",
    "nombre": "Pedro",
    "nombre": "Maria"
}
```

```
people = {
    "nombre": "Juan",
    "nombre": "Pedro",
    "nombre": "Maria"
    }
```

Coding style - nomenclatura

```
CONSTANTS


def function_names_in_lowercase


class CapWords
    def method(self, ...):
    def _private_method(self, ...):
    def class_method(cls, ...):
```

Coding style - paréntesis

```
def search(number_of_files=0):
    if (number_of_files > 0):
        [...]
        return (number_of_files, files_found)
```

Coding style - Herramientas

- pep8
- pylint

Patrones

Expansión de listas

```
def print_args(*args):
    for arg in args:
        print arg


random_list = [1, "hi", [2, 3]]
print_args(*random_list)
```

Expansión de diccionarios

```
def search_feed(num_messages, min_favs, user_handle):
    [...]


search_feed(num_messages=20, min_favs=5, user_handle='maraujop')
search_feed(**{
    'num_messages': 20,
    'min_favs': 5,
    'user_handle': 'maraujop'
})
```

Posicional vs Keyword

```
search_feed(40, 5, 'maraujop')

search_feed(num_messages=40, min_favs=5, user_handle='maraujop')
```

Decoradores - PEP 318

```
from decorator import decorator

@decorator

def trace(f, *args, **kwargs):

    print "calling %s with args %s, %s" % (f, args, kwargs)

    return f(*args, **kwargs)


@trace

def search_feed(num_messages, min_favs, user_handle):

    print "Searching feed..."
```

Decoradores

```
search_feed(num_messages=20, min_favs=5, user_handle="maraujop")
```

```
calling <function search_feed at 0x105d491b8> with args
(20, 5, 'maraujop'), {}
Searching feed...
```

setattr / hasattr

```
person = Person()
variable_name = 'age'
value = 26


setattr(person, variable_name, value)


hasattr(person, variable_name)
hasattr(person, 'age')
```

isinstance / issubclass

```
isinstance(2, int)

True


isinstance(2, float)

False


value = {'name': "Juan"}

isinstance(value, dict)
```

Clases base abstractas

```
import abc

class Example:
    __metaclass__ = abc.ABCMeta

    @abc.abstractmethod
    def abstract_method(self):
        pass

    @abc.abstractproperty
    def abstract_property(self):
        pass
```

Métodos magicos

```
class Agenda:
    def __init__(self, description):
        self.people = [ ]
        self.description = description
    def __getattr__(self, name):
        return object.__getattribute__(self.people, name)


office = Agenda('People in the office')
office.append({'name': 'Miguel', 'age': 26})
print office.description
print [person['age'] for person in office]
```

Métodos magicos

```
class Agenda:
    def __len__(self):
        return len(self.people)
    def __contains__(self, item):
        return item.lower() in [person['name'].lower() for person in self.people]

print len(office)
for name in ('Miguel', 'Juan'):
    if name in office:
        print "%s in office" % name
```

Métodos magicos

http://www.rafekettler.com/magicmethods.html

Antipatrones

__init__.py

getters / setters

import *

Idioms

Iterando una colección y sus índices

```
for i, name in enumerate(names):
```

PEP 279

Iterando varias colecciones

```
names = ["Juan", "Pedro", "Isabel", "Maria"]
numbers = [23, 31, 18]


for name, number in zip(names, numbers):
    print "name %s number %s" % (name, number)
```

[('Juan', 23), ('Pedro', 31), ('Isabel', 18)]

Crear diccionarios

```
names = ["Miguel", "Juan", "Pedro"]
ages = [21, 22, 34]


zip(names, ages)
[('Miguel', 21), ('Juan', 22), ('Pedro', 34)]
dict(zip(names, ages))
{'Juan': 22, 'Miguel': 21, 'Pedro': 34}
```

Contar con diccionarios

```
from collections import defaultdict


message = "Bienvenidos a la primera PyCon España, la primera"
count = defaultdict(int)
for word in message.split():
    count[word] += 1
```

Contar con Counter

```
from collections import Counter


message = "Bienvenidos a la primera PyCon España, la primera"

count = Counter(message.split())
```

Funcional

Funcional

- Ideal para trabajar estructuras de datos
- No debe abusarse de ella, ni obsesionarse con ella

Ordenar lista de diccionarios

```
people = [
    {'edad': 14, 'nombre': 'Juan'},
    {'edad': 26, 'nombre': 'Miguel'},
    {'edad': 30, 'nombre': 'Aria'}
]


sorted(people, key=lambda person: person['edad'])
```

Ordenar lista de diccionarios

- Las lambdas son lentas, itemgetter está implementado en C
- Para muchos más fácil de leer y más corto

```
from operator import itemgetter

sorted(people, key=itemgetter('edad'))
```

List comprehensions - PEP 202

- Las llamadas a función son cacheadas, son rápidas

```
words = ["hola", "que", "tal"]


[word.upper() for word in words]
```

filter

```
people = [
    {'edad': 24, 'nombre': 'Saul'},
    {'edad': 22, 'nombre': 'Juan'}
]


filter(lambda person: person['edad'] > 22, people)
[{'edad': 24, 'nombre': 'Saul'}]
```

Aplanando estructuras de datos

```python
from itertools import chain


list(chain.from_iterable([
    [{"nombre": "Jorge"}, {"nombre": "Pedro"}],
    [{"nombre": "Jesus"}]
]))
[{'nombre': 'Jorge'}, {'nombre': 'Pedro'}, {'nombre': 'Jesus'}]
```

Python rocks

Python rocks

- Un ecosistema robusto, maduro y de calidad
- Tests y documentación
- Entornos virtuales y pip

HTTP

Python requests

```
import requests


requests.get("http://example.com", params={'search_term': "foo"})

requests.post("http://example.com", data={'new_password': "bar"})
```

Python requests

- rauth OAuth 1.0/a, 2.0, y Ofly
- HTTPretty Mockear urls y sus respuestas
- httpie un cURL para humanos
- Muchos bindings para APIs

Fechas y tiempos

pytz

```
from datetime import datetime
import pytz
now = datetime.utcnow()
datetime.datetime(2013, 11, 17, 14, 13, 51, 641900)

now = now.replace(tzinfo=pytz.utc)
datetime.datetime(2013, 11, 17, 14, 13, 51, 641900, tzinfo=<UTC>)

now.astimezone(pytz.timezone("Europe/Madrid"))
datetime.datetime(2013, 11, 17, 15, 13, 51, 641900,
    tzinfo=<DstTzInfo 'Europe/Madrid' CET+1:00:00 STD>)
```

python-dateutil

```
import calendar
from dateutil.relativedelta import relativedelta


now + relativedelta(month=1, day=23)
now + relativedelta(months=-1, days=5)
now + relativedelta(weekday=calendar.FRIDAY)
```

Bases de datos

ORMs

- SQLAlchemy El ORM más potente de Python y mucho más
- peewee Similar al ORM de Django's pero con un API más consistente
- Django ORM

Más!!

Más

- Pandas Librería de análisis de datos
- pattern Data mining
- Boto Interfaz para AWS
- Pillow Librería para manejo de imágenes
- Path Manejo de rutas y ficheros
- Docopt Parser de línea de comandos
- bleach Sanitizador de HTML
- Jinja Motor de plantillas

APIs humanas
!=
APIs buenas

Simplificar la mecánica habitual al máximo

Ejemplo nltk

Natural Language Tool Kit

```
tokens = nltk.word_tokenize("I like eating chocolate with milk")

tagged = nltk.pos_tag(tokens)

[(u'I', u'PRP'), (u'like', u'VB'),

 (u'eating', u'VBG'), (u'chocolate', u'NN'),

 (u'with', u'IN'), (u'milk', u'NN')]
```

Ejemplo TextBlob

API sencilla para procesar datos textuales

```
blob = TextBlob("I like eating chocolate with milk")

blob.tags

[(u'I', u'PRP'), (u'like', u'VB'),

 (u'eating', u'VBG'), (u'chocolate', u'NN'),

 (u'with', u'IN'), (u'milk', u'NN')]
```

Debe tener conceptos de alto nivel

Ejemplo OpenCV

```
import cv2
import cv2.cv as cv

def camshift_tracking(img1, img2, bb):
    hsv = cv2.cvtColor(img1, cv.CV_BGR2HSV)
    mask = cv2.inRange(hsv, np.array((0., 60., 32.)), np.array((180., 255., 255.)))
    x0, y0, w, h = bb
    x1 = x0 + w -1
    y1 = y0 + h -1
    hsv_roi = hsv[y0:y1, x0:x1]
    mask_roi = mask[y0:y1, x0:x1]
    hist = cv2.calcHist( [hsv_roi], [0], mask_roi, [16], [0, 180] )
    cv2.normalize(hist, hist, 0, 255, cv2.NORM_MINMAX);
    hist_flat = hist.reshape(-1)
    prob = cv2.calcBackProject([hsv,cv2.cvtColor(img2, cv.CV_BGR2HSV)], [0], hist_flat, [0, 180], 1)
    prob &= mask
    term_crit = ( cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 1 )
    new_ellipse, track_window = cv2.CamShift(prob, bb, term_crit)
    return track_window
```

Ejemplo SimpleCV

```
camera, display = Camera(), Display()
bounding_box = (100, 200, 100, 100)
track_set = [ ]
previous = camera.getImage()
while display.isNotDone():
    frame = camera.getImage()
    track_set = frame.track('camshift', track_set, previous, bounding_box)
    track_set.drawBB()
    previous = frame
    frame.save(display)
```

Poder usarse sin conocer detalles internos

Ejemplo urllib2

```
import urllib2

gh_url = 'https://api.github.com'
req = urllib2.Request(gh_url)

password_manager = urllib2.HTTPPasswordMgrWithDefaultRealm()
password_manager.add_password(None, gh_url, 'user', 'pass')
auth_manager = urllib2.HTTPBasicAuthHandler(password_manager)
opener = urllib2.build_opener(auth_manager)

urllib2.install_opener(opener)
handler = urllib2.urlopen(req)

print handler.getcode()
print handler.headers.getheader('content-type')
```

Ejemplo requests

```
import requests


r = requests.get('https://api.github.com', auth=('user', 'pass'))


print r.status_code

print r.headers['content-type']
```

Debe generar código mantenible

Ejemplo argparse - PEP 389

```
if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('--run', dest='action', help='argparse example')
    args = parser.parse_args()
    if args.action == 'update':
        print "Updating..."
    elif args.action == 'watch':
        print "Watch..."
```

Ejemplo docopt

```
"""
Usage:
    example.py run (update|watch)
    example.py -h | --help
    example.py --version
Options:
    -h --help   Show help message.
    --version   Show version.
"""
```

```
if __name__ == '__main__':
    args = docopt(__doc__, version='1.0')
    if args['update']:
        print "Updating..."
    elif args['watch']:
        print "Watch..."
```

Python moderno

Python moderno

- Python 2.0 en el año 2000
- PEPs
- Recolector de basura / soporte de Unicode
- PEP 1 -- PEP Purpose and Guidelines

Python 3.4

Gracias,
¿preguntas?