

# SIG + Python

=

# Love Story

**1. SIG ?**

**2. Pourquoi Python**

**3. Données vecteur**

**4. Données raster**

**5. Des exemples**

# SIG ?



# SIG ?



## Systèmes d'Information Géographique

- capturer,
- créer,
- stocker,
- analyser,
- partager,
- visualiser

... des données géolocalisées

# Pourquoi Python ?

- Language de haut niveau

# Pourquoi Python ?

- Language de haut niveau
- Multi paradigme

# Pourquoi Python ?

- Language de haut niveau
- Multi paradigme
- Multi plateforme

# Pourquoi Python ?

- Language de haut niveau
- Multi paradigme
- Multi plateforme
- Rapide



# Pourquoi Python ?

- Language de haut niveau
- Multi paradigme
- Multi plateforme
- Rapide
- Dialogue facilement avec des langages bas niveau

# Pourquoi Python ?

- Language de haut niveau
- Multi paradigme
- Multi plateforme
- Rapide
- Dialogue facilement avec des langages bas niveau
- Orchestrer et chainer des traitements complexes

# Pourquoi Python ?

- Language de haut niveau
- Multi paradigme
- Multi plateforme
- Rapide
- Dialogue facilement avec des langages bas niveau
- Orchestrer et chainer des traitements complexes
- communauté & bibliothèques

# Python en maître SIG

Python a été adopté comme API/Scripting/binding dans la plupart des projets SIG

- QGIS
- Grass
- Mapnik
- GDAL
- pyproj
- shapely
- ArcGIS
- ...

# Types de données SIG

- Vecteur
- Raster

# Données vecteur

Binding python OGR du projet GDAL : [osgeo.org](http://osgeo.org)

# Données vecteur

Binding python OGR du projet GDAL : osgeo.ogr

Créer un point ?

```
# import OGR module for vector data
from osgeo import ogr

# create a geometry object of point type
point = ogr.Geometry(ogr.wkbPoint)

# set point coordinates
point.AddPoint(1198054.34, 648493.09)

# Get human readable format
print point.ExportToWkt()

# POINT(1198054.34 648493.09)
```

# Données vecteur

## Fiona

- Pythonique
- Rapide
- Basé sur Cython
- Par Sean Gillies

```
import fiona

# on ouvre les drivers de lecture
with fiona.drivers():
    # on ouvre un fichier shapefile
    with fiona.open('docs/data/test_uk.shp') as source:
        # on utilise les itérateurs pour lire les features
        # feature = géométrie + attributs
        for feature in source:
            # Les objets sont pur Python
            print feature['geometry']
```



# Analyse Vecteur

## Shapely

- Par Sean Gillies :-)
- ..
- Pour le traitement des géométries
- Basé sur GEOS

```
>>> # Calculs sur un polygone
>>> from shapely.geometry import Polygon
>>> # création du polygone
>>> polygon = Polygon([(0, 0), (1, 1), (1, 0)])
>>> # On accède à ses propriétés : surface
>>> polygon.area
0.5
>>> # longueur
>>> polygon.length
3.4142135623730949
```

# Dans Shapely

- Tests de relations
  - Intersecte, Touche, Contient, ...
- Opérations booléennes
  - Intersection, Union, ...
- Calculs
  - Distance
  - Buffer, Enveloppe convexe ...
- Accesseurs et propriétés
  - centroïde
  - Validité, Simplicité
  - Type
  - Coordonnées ...

# Données raster

## raster.io

- Binding GDAL
- Pythonique
- Rapide
- Par Sean Gillies...

```
# rasterio et numpy fonctionnent ensemble
import numpy
import rasterio

# On ouvre les drivers
with rasterio.drivers():
    # on lit un fichier geotiff
    with rasterio.open('tests/data/RGB.byte.tif') as src:
        # On récupère les valeurs des trois bandes de couleur
        b, g, r = src.read()
```

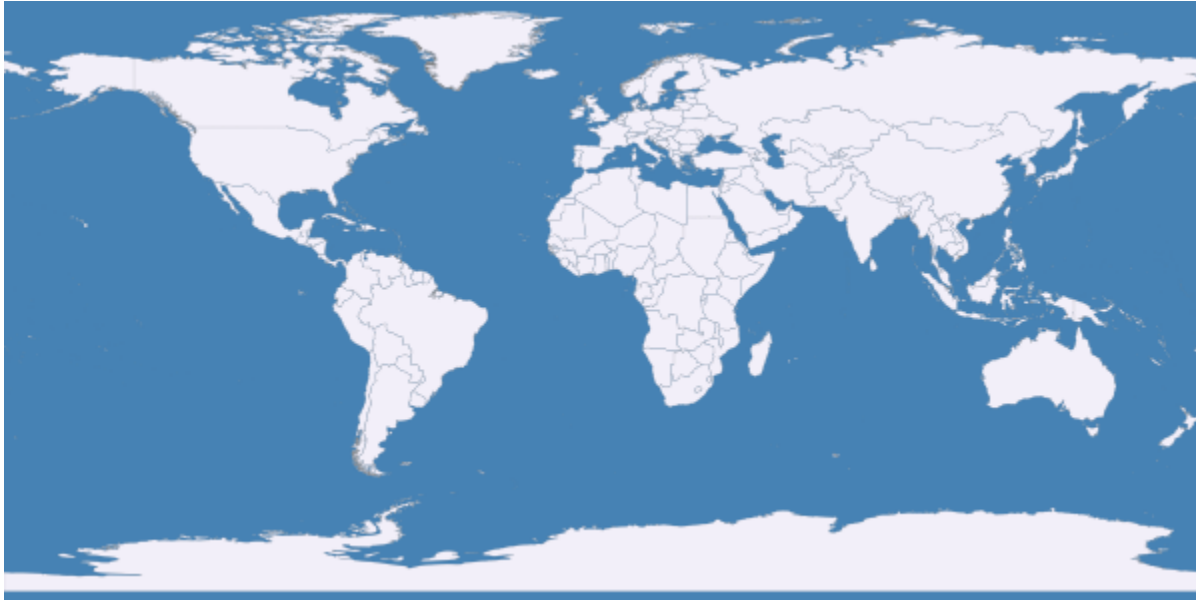
# Créer des cartes

## Mapnik

- Moteur de rendu (cf OpenStreetMap)
- Bibliothèque C++
- Binding Python

```
import mapnik
m = mapnik.Map(600,300)
m.background = mapnik.Color('steelblue')
s = mapnik.Style()
r = mapnik.Rule()
polygon_symbolizer = mapnik.PolygonSymbolizer(mapnik.Color('#f2eff9'))
r.symbols.append(polygon_symbolizer)
line_symbolizer = mapnik.LineSymbolizer(mapnik.Color('rgb(50%,50%,50%)'),0.1)
r.symbols.append(line_symbolizer)
s.rules.append(r)
m.append_style('My Style',s)
ds = mapnik.Shapefile(file='ne_110m_admin_0_countries.shp')
layer = mapnik.Layer('world')
layer.datasource = ds
layer.styles.append('My Style')
m.layers.append(layer)
m.zoom_all()
```

# Créer des cartes : résultat



```

import mapnik
# Création d'une carte
m = mapnik.Map(600,300)
# on règle la couleur de fond
m.background = mapnik.Color('steelblue')
# On crée un style, et sa règle d'application
s = mapnik.Style()
r = mapnik.Rule()
# Un symbolizer est ce qui détermine comment est représentée une feature
# Les polygones en gris clair et les lignes en gris
polygon_symbolizer = mapnik.PolygonSymbolizer(mapnik.Color('#f2eff9'))
r.symbols.append(polygon_symbolizer)
line_symbolizer = mapnik.LineSymbolizer(mapnik.Color('rgb(50%,50%,50%)'),0.1)
r.symbols.append(line_symbolizer)
# On ajoute ces règles au style, et le style à la carte
s.rules.append(r)
m.append_style('My Style',s)
# Chargement d'un datasource : nos données
ds = mapnik.Shapefile(file='ne_110m_admin_0_countries.shp')
# création d'un layer pour ce datasource
layer = mapnik.Layer('world')
layer.datasource = ds
# Application du style au layer (par son nom)
layer.styles.append('My Style')
# ajout du layer à la carte
m.layers.append(layer)
# On zoom sur l'ensemble des données
m.zoom_all()
# on fait le rendu dans une image
mapnik.render_to_file(m,'world.png', 'png')

```

# Créer des cartes

## Mapnik

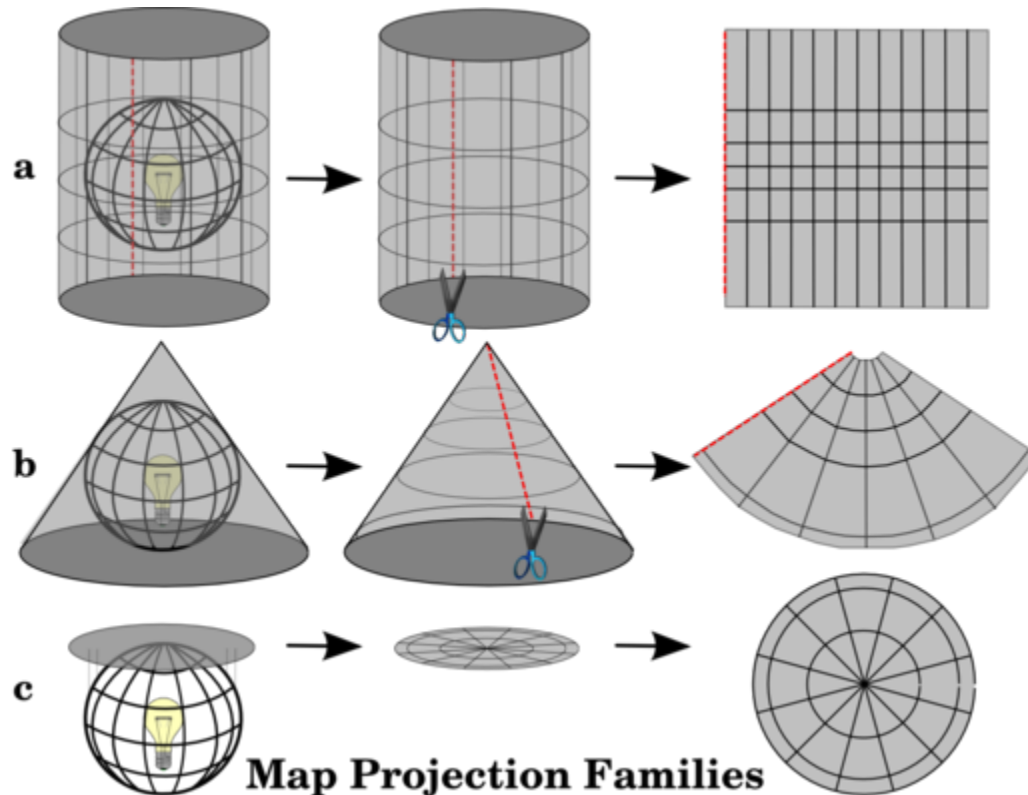
- Utilisation de styles XML ou Carto
- TileMill : éditeur de style avec rendu immédiat
- Rasters
- Vecteurs
- Formats multiples en entrée
  - PostGIS
- Formats multiples en sortie
  - Images
  - Vectoriel ( VectorTile... )
- Symbolologies avancées
- Serveurs de tuiles basés sur Mapnik





# Systèmes de coordonnées

La terre n'est pas plate !



# pyproj

- Binding Proj4

```
import pyproj
# Définition d'un système de coordonnées (ellipsoïdal)
wgs84 = pyproj.Proj('+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs')
# Un autre système de coordonnées (projection conique)
be31370 = pyproj.Proj('+proj=lcc +lat_1=51.16666723333334 ....')
# Les coordonnées d'un point en lat, lon
lat, long = wgs84(50.6701607, 4.6151754)
# transformation en coordonnées projetées
x2, y2 = pyproj.transform(wgs84, be31370, lat, lon, radians=True)
```

# Géocoder des données

Adresse => localisation

geopy

- Interface de géocodeurs en ligne
- Plusieurs Géocodeurs disponibles (Google, Nominatim...)
- /\ Conditions d'utilisation

```
>>> # On utilise Nominatim ( OSM )
>>> from geopy.geocoders import Nominatim
>>> geolocator = Nominatim()
>>> # Géocodage d'une adresse
>>> location = geolocator.geocode("175 5th Avenue NYC")
>>> # L'adresse complète correspondante est renvoyée
>>> print(location.address)
Flatiron Building, 175, 5th Avenue, Flatiron, New York, NYC, New York, ...
>>> # Ainsi que ses coordonnées en lat/lon
>>> print(location.latitude, location.longitude)
(40.7410861, -73.9896297241625)
>>> # Et des informations complémentaires
>>> print(location.raw)
{'place_id': u'9167009604', u'type': u'attraction', ...}
```

# Slippy map

Simple avec folium (python + leaflet)

- Carte dynamique
- À la google map
- Génération de HTML

```
import folium
# On crée une carte en donnant le point centrale
map_osm = folium.Map(location=[45.5236, -122.6750])
# On écrit le fichier
map_osm.create_map(path='osm.html')
# C'est tout :-)
```

# Python et PostGIS

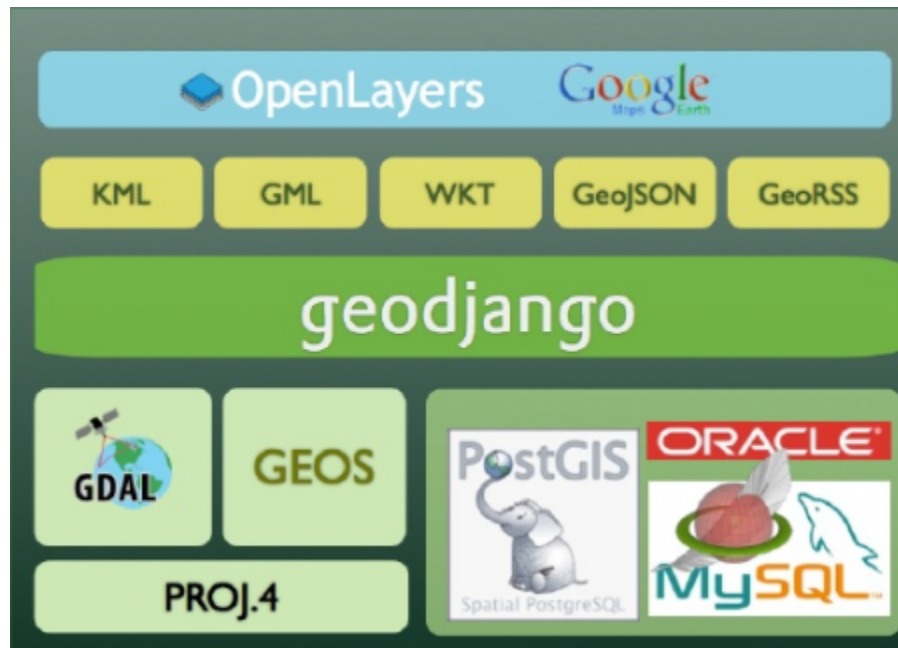
- Idem que PostgreSQL + SQL spatial

```
# Le module Python pour PostgreSQL
import psycopg2
# connection ( /\ exceptions )
conn=psycopg2.connect("dbname='foo' user='dbuser' password='mypass'")
# On initie un curseur
cur = conn.cursor()
# On exécute une requete - spatiale !
cur.execute("""
    select
        id, st_astext(geom) as t
        , st_distance(geom,
            st_transform(
                st_setsrid(
                    st_makepoint(45.5236, -122.6750)
                    , 4326), 2154)) as d
    from
        restaurants;""")
# Récupération des lignes
rows = cur.fetchall()
# On les affiche
print "\nRows: \n"
for row in rows:
```

# Framework GeoWeb

# geodjango

- django.contrib.gis
- fonctionne avec PostGIS / Spatialite de base
- ORM avec filtres spatiaux
- Fonctionnalités Géo intégrées dans Django
- Types géographiques + widgets





## Change world borders

[History](#)

## Country Attributes

Name: Population:   
Country wide population in 2005Country Codes ( [Show](#) )Area and Coordinates ( [Show](#) )

## Map View

## Geometry:

[Delete all Features](#)[Delete](#)[Save as new](#)[Save and continue editing](#)[Save](#)

## AUSTRALIA

AUSTRALIA POPULATION: 19,855,288

The country of Australia likes hot baths and chocolate cake.

**CheapOair® Cheap Flights** [CheapOair.com/Cheap-Flt](http://CheapOair.com/Cheap-Flt)  
CheapOair® Cheap Air Tickets Sale, Book Now & Get Up To 65% + \$15 Off!

**QANTAS flights to AU** [Qantas.com.au/US](http://Qantas.com.au/US)  
Can't miss fares to Australia. Book your flight on QANTAS now.

**Flinders View Cottage** [www.flindersviewcottage.com.au](http://www.flindersviewcottage.com.au)  
Flinders Ranges Accommodation Accommodation in Quorn

**Campervan Hire Tasmania** [Discovery-Campervans.com](http://Discovery-Campervans.com)  
Compare Cheap Campervan Hire Prices for Your Tasmanian Holiday

AdChoices ▶



## TOWNS

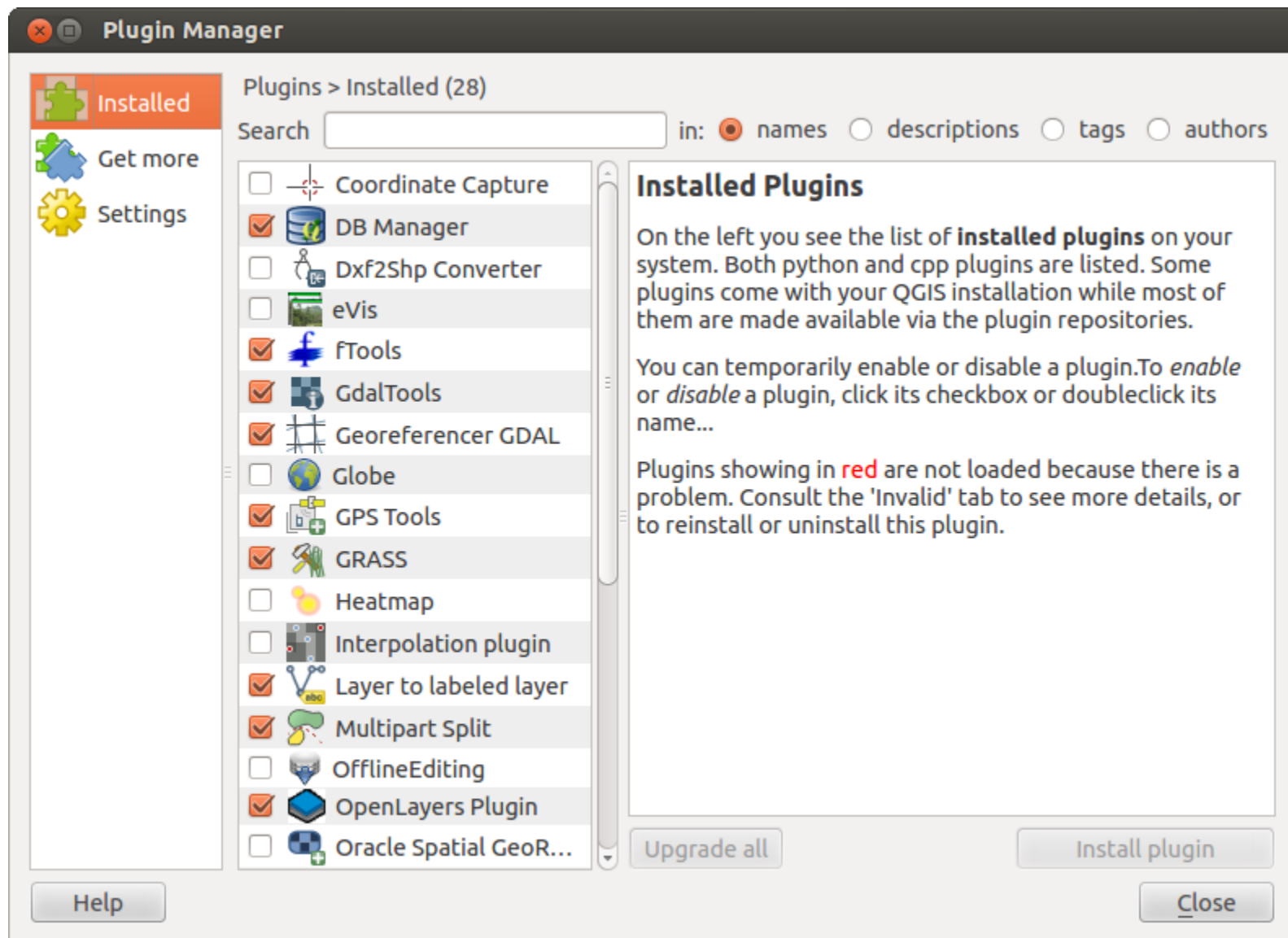
### HOBART

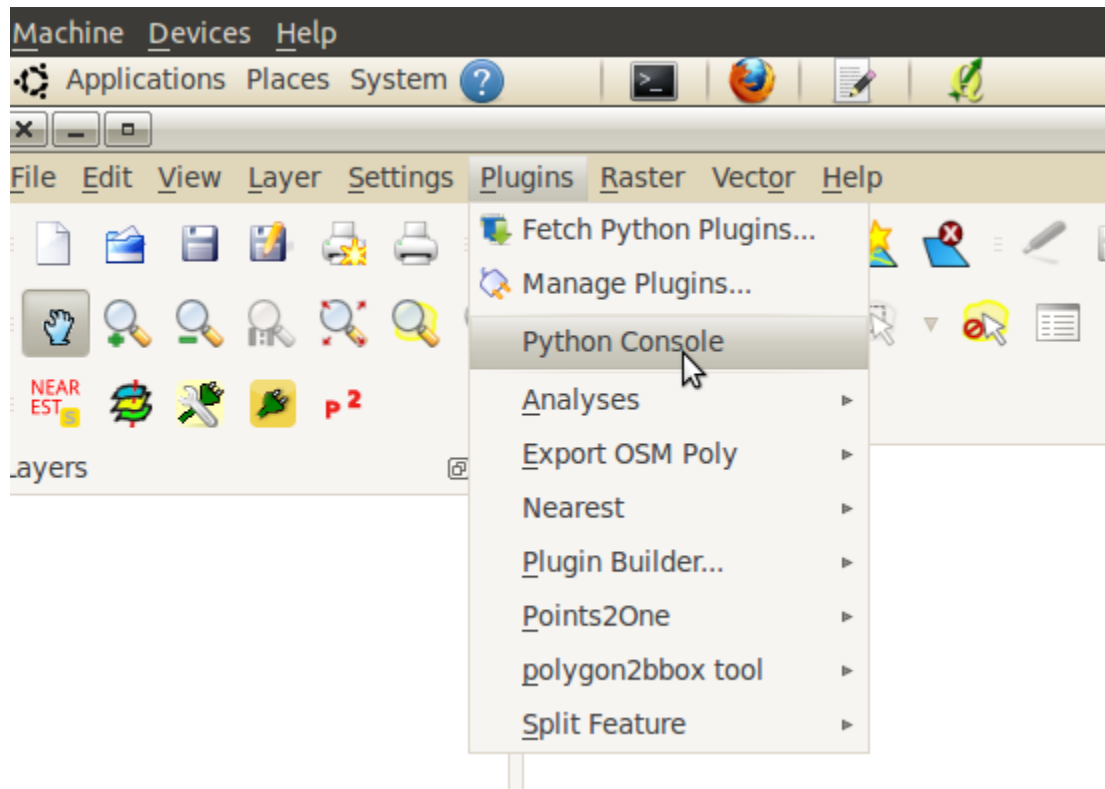
7000 1,745 HOBART TAS

The capital city of Tasmania is *Hobart*, which as a Central Business District is home to around 1,745 people, with the major industry being

# Bureautique : Étendre QGIS

- SIG bureautique
- C++
- Bindings Python
- => Étendre l'outil
  - Scripter
  - "Macros"
  - Modifier l'interface
  - Ajouter des fonctionnalités
  - Prototypage rapide
- => Développer des applications spécifiques
- Raison du succès de QGIS





# Étendre QGIS

- Basé sur Qt => PyQt4
- 2 modules pour QGIS

```
# Import des modules PyQt
from PyQt4.QtCore import *
from PyQt4.QtGui import *
# import des modules de qgis
from qgis.core import *
from qgis.gui import *



# Éviter si possible les imports * ! (ici préfixes)
```

- Accès à toute l'interface et pas de limites aux extensions
- Dépôt d'extensions





2011-11-19 14:43:59

Time Manager

 Settings  Export Video Time frame start: 2011-11-19 14:43:59 Time frame size: 2 hours

2011-10-23 21:13:53 2011-12-19 05:38:57

 Coordinate: 50.1,-27.9 Scale 1:148114599  ☒ Render EPSG:4326

# Bref : SIG ♥ Python :-)

- Beaucoup d'autres bibliothèques SIG
- SIG > Cartographie
  - Itinéraires
  - Analyse de graphes
  - 3D
  - Analyse d'image
  - Calcul scientifique
  - Simulation
  - Apprentissage artificiel
  - ...
- Bonne intégration
- Mangez en !



Fin^H^H^H

Ce n'est que le début

Questions ?

Vincent Picavet

@vpicavet

vincent.picavet@oslandia.com

[www.oslandia.com](http://www.oslandia.com)