OpenStreetMap Sample Project Appendixes

Jay Teguh Wijaya

Appendix 1. audit-streets.py

```
import xml.etree.cElementTree as ET
from collections import defaultdict
import re
import pprint
import codecs

OSMFILE = "map"
street_prefix_re = re.compile(r'\b\S+\.?', re.IGNORECASE)

global_prefix = "Jalan"
mapping = [ {"jl.": global_prefix},
            {"Jl.": global_prefix},
            {"jalan": global_prefix},
            {"Jalan": global_prefix}, # Repeat just in case we have
concatenated names i.e. "JalanSomething"
            {"JLN.": global_prefix},
            {"JL.": global_prefix},
            {"Jln.": global_prefix},
            {"jln": global_prefix},
            {"J l .": global_prefix},
            {"J": global_prefix}]

# Street names having these matches must not be added prefix "Jalan"
outlier_regexii = [r'(?:[Ss]treet)$', r'(?:[Rr]oad)$']
remove_nextto = r'[/,-]'

def audit_street_prefix(street_prefixes, street_name):
    m = street_prefix_re.search(street_name)
    if m:
        street_prefix = m.group()
        if street_prefix not in street_prefixes:
            street_prefixes[street_prefix] = 1
        else:
            street_prefixes[street_prefix] += 1

def is_street_name(elem):
```

```
    return (elem.attrib['k'] == "addr:street")


def audit(osmfile):
    osm_file = open(osmfile, "r")
    street_prefixes = defaultdict(set)
    street_names = set()
    for event, elem in ET.iterparse(osm_file, events=("start",)):

        if elem.tag == "node" or elem.tag == "way":
            for tag in elem.iter("tag"):
                if is_street_name(tag):
                    audit_street_prefix(street_prefixes, tag.attrib['v'])
                    street_names.add(tag.attrib['v'])

    return street_names


# Uppercase first letter of each word, and do nothing else.
def upcaseplus(s):
    return " ".join([i[0].upper() + i[1:] for i in s.split()])

def update_street_name(name, mapping):
    replaced = False

    # Removes anything next to remove_nextto regex
    m = re.search(remove_nextto, name)
    if m is not None:
        name = name[0:m.start(0)].strip()
    # Replaces according to mapping, except for names matching
outlier_regexii.
    for regex in outlier_regexii:
        result = re.search(regex, name)
        if result is not None:
            string = result.group(0)
            # "sunsetroad" to "sunset road "
            name = name.replace(string, " {} ".format(string))
            # "sunset road " to "Sunset Road"
            name = upcaseplus(" ".join(name.split()))
            replaced = True
    if not replaced:
        for item in mapping:
            for prefix in item:
```

```
                if name.startswith(prefix) and not replaced:
                    name = upcaseplus(name.replace(prefix, "", 1).strip())
                    name = " ".join([item[prefix], name])
                    replaced = True
                    break

    # Add the rest with global prefix
    if not replaced:
        name = "{0} {1}".format(global_prefix, upcaseplus(name))

    return name


def test():
    street_names = audit(OSMFILE)
    # pprint.pprint(street_names)
    with codecs.open("names.txt", "w") as text_file:
        text_file.write("")
    for name in street_names:
        better_name = update_street_name(name, mapping)
        try:
            with codecs.open("names.txt", "a") as text_file:
                text_file.write(name + " => " + better_name + "\n")
            # print name, " => ", better_name
        except:
            pass
        if name == "jln flamboyan2 perumnas":
            assert better_name == "Jalan Flamboyan2 Perumnas"
        if name == "Popies II / Benesari, Kuta":
            assert better_name == "Jalan Popies II"
        if name == "By Pass Sunset Road":
            assert better_name == "By Pass Sunset Road"

if __name__ == '__main__':
    test()
```

Appendix 2. Printed output of audit-streets.py

```
Jalan Hasanuddin => Jalan Hasanuddin
Jalan Uluwatu => Jalan Uluwatu
Sutoyo => Jalan Sutoyo
Jalan Raya Sesetan => Jalan Raya Sesetan
Jl. Bougenville Boulevard => Jalan Bougenville Boulevard
Jalan Ahmad Yani => Jalan Ahmad Yani
Jalan Uluwatu 2 => Jalan Uluwatu 2
Jl Surgiwa => Jalan L Surgiwa
Jl. Buni Sari => Jalan Buni Sari
Jalan Hassanudin => Jalan Hassanudin
Puputan 1 => Jalan Puputan 1
jalan Seroja => Jalan Seroja
Tukad Gangga 1 => Jalan Tukad Gangga 1
Jalan Pantai Batu Bolong 60 => Jalan Pantai Batu Bolong 60
Jalan Lanyahan, Br Nagi => Jalan Lanyaha
Jalan Diponegoro => Jalan Diponegoro
Sunset Road => Sunset Road
Angsoka => Jalan Angsoka
Jalan H.O.S. Cokroaminoto => Jalan H.O.S. Cokroaminoto
Jalan Wahidin => Jalan Wahidin
Taman Kanak-Kanak Kartika => Jalan Taman Kana
Bali Nusa Dua Tourism Complex, Lot N5 => Jalan Bali Nusa Dua Tourism Comple
…
Raya Puputan => Jalan Raya Puputan
Dewi Sartika => Jalan Dewi Sartika
Jl. Goa Gong => Jalan Goa Gong
Jalan Raya Candi Kuning => Jalan Raya Candi Kuning
Jl. Surapati, Buleleng, Bali => Jalan Surapat
Jalan Raya Ubud => Jalan Raya Ubud
Jalan Werkudara => Jalan Werkudara
JLN. LANYAHAN, BR NAGI, => Jalan LANYAHAN
SMA 6 => Jalan SMA 6
Kaliasem => Jalan Kaliasem
Jalan Veteran => Jalan Veteran
Jl Tukad Badung => Jalan L Tukad Badung
Jl. Gajahmada Seririt => Jalan Gajahmada Seririt
Jalan Tukad Banyu Poh => Jalan Tukad Banyu Poh
Jalan Goutama => Jalan Goutama
Jalan Teuku Umar Barat => Jalan Teuku Umar Barat
Pierre Tendean => Jalan Pierre Tendean
Jl. Raya => Jalan Raya
Jalan Jaya Pangus => Jalan Jaya Pangus
Jalan Imam Bonjol => Jalan Imam Bonjol
Tukad Musi 1 => Jalan Tukad Musi 1
Jalan Jaya Tirtha => Jalan Jaya Tirtha
Terompong => Jalan Terompong
Jalan Kajeng => Jalan Kajeng
Jl. Sriwedari , => Jalan Sriwedari
```

Appendix 3. data.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import xml.etree.ElementTree as ET
import pprint
import re
import codecs
import json
"""
The output should be a list of dictionaries
that look like this:

{
"id": "2406124091",
"type: "node",
"visible":"true",
"created": {
          "version":"2",
          "changeset":"17206049",
          "timestamp":"2013-08-03T16:43:42Z",
          "user":"linuxUser16",
          "uid":"1219059"
        },
"pos": [41.9757030, -87.6921867],
"address": {
          "housenumber": "5157",
          "postcode": "60625",
          "street": "North Lincoln Ave"
        },
"amenity": "restaurant",
"cuisine": "mexican",
"name": "La Cabana De Don Luis",
"phone": "1 (773)-271-5176"
}

In particular the following things should be done:
- you should process only 2 types of top level tags: "node" and "way"
- all attributes of "node" and "way" should be turned into regular
key/value pairs, except:
    - attributes in the CREATED array should be added under a key "created"
```

```
    - attributes for latitude and longitude should be added to a "pos"
array,
      for use in geospacial indexing. Make sure the values inside "pos"
array are floats
      and not strings.
- if second level tag "k" value contains problematic characters, it should
be ignored
- if second level tag "k" value starts with "addr:", it should be added to
a dictionary "address"
- if second level tag "k" value does not start with "addr:", but contains
":", you can process it
  same as any other tag.
- if there is a second ":" that separates the type/direction of a street,
  the tag should be ignored, for example:

<tag k="addr:housenumber" v="5158"/>
<tag k="addr:street" v="North Lincoln Avenue"/>
<tag k="addr:street:name" v="Lincoln"/>
<tag k="addr:street:prefix" v="North"/>
<tag k="addr:street:type" v="Avenue"/>
<tag k="amenity" v="pharmacy"/>

  should be turned into:

{...
"address": {
    "housenumber": 5158,
    "street": "North Lincoln Avenue"
}
"amenity": "pharmacy",
...
}

- for "way" specifically:

  <nd ref="305896090"/>
  <nd ref="1719825889"/>

should be turned into
"node_refs": ["305896090", "1719825889"]
"""
```

```
lower = re.compile(r'^([a-z]|_)*$')
lower_colon = re.compile(r'^([a-z]|_)*:([a-z]|_)*$')
problemchars = re.compile(r'[=\+/&<>;\'"\?%#$@\,\. \t\r\n]')

CREATED = [ "version", "changeset", "timestamp", "user", "uid"]

global_prefix = "Jalan"
mapping = [ {"jl.": global_prefix},
            {"Jl.": global_prefix},
            {"jalan": global_prefix},
            {"Jalan": global_prefix}, # Repeat just in case we have
concatenated names i.e. "JalanSomething"
            {"JLN.": global_prefix},
            {"JL.": global_prefix},
            {"Jln.": global_prefix},
            {"jln": global_prefix},
            {"J l .": global_prefix},
            {"J": global_prefix}]

# Street names having these matches must not be added prefix "Jalan"
outlier_regexii = [r'(?:[Ss]treet)$', r'(?:[Rr]oad)$']
remove_nextto = r'[/,-]'

def shape_element(element):
    node = {}
    if element.tag == "node" or element.tag == "way" :
        for key in element.attrib:
            if key in CREATED:
                if "created" not in node:
                    node["created"] = {}
                node["created"][key] = element.attrib[key]
            elif key == "lon":
                if "pos" not in node:
                    node["pos"] = [0,0]
                node["pos"][0] = float(element.attrib[key])
            elif key == "lat":
                if "pos" not in node:
                    node["pos"] = [0,0]
                node["pos"][1] = float(element.attrib[key])
            else:
                node[key] = element.attrib[key]
        node["type"] = element.tag
        for child_element in iter(element):
```

```
            if child_element.tag == "tag":
                key = child_element.attrib["k"]
                in_address = False
                if key.startswith("addr:"):
                    in_address = True
                    key = key.replace("addr:", "", 1)
                if (lower.search(key) or lower_colon.search(key)) and not
problemchars.search(key):
                    value = child_element.attrib["v"]
                    if in_address:
                        if "address" not in node:
                            node["address"] = {}
                        if ":" not in key:
                            if key == "street":
                                node["address"][key] =
update_street_name(value, mapping)
                            else:
                                node["address"][key] = value
                    else:
                        node[key] = value
            elif child_element.tag == "nd":
                if "node_refs" not in node:
                    node["node_refs"] = []
                node["node_refs"].append(child_element.attrib["ref"])
        return node
    else:
        return None

# Uppercase first letter of each word, and do nothing else.
def upcaseplus(s):
    return " ".join([i[0].upper() + i[1:] for i in s.split()])

def update_street_name(name, mapping):
    replaced = False

    # Removes anything next to remove_nextto regex
    m = re.search(remove_nextto, name)
    if m is not None:
        name = name[0:m.start(0)].strip()
    # Replaces according to mapping, except for names matching
outlier_regexii.
    for regex in outlier_regexii:
        result = re.search(regex, name)
```

```
        if result is not None:
            string = result.group(0)
            # "sunsetroad" to "sunset road "
            name = name.replace(string, " {} ".format(string))
            # "sunset road " to "Sunset Road"
            name = upcaseplus(" ".join(name.split()))
            replaced = True
    if not replaced:
        for item in mapping:
            for prefix in item:
                if name.startswith(prefix) and not replaced:
                    name = upcaseplus(name.replace(prefix, "", 1).strip())
                    name = " ".join([item[prefix], name])
                    replaced = True
                    break

    # Add the rest with global prefix
    if not replaced:
        name = "{0} {1}".format(global_prefix, upcaseplus(name))

    return name


def process_map(file_in, pretty = False):
    file_out = "{0}.json".format(file_in)
    data = []
    with codecs.open(file_out, "w") as fo:
        for _, element in ET.iterparse(file_in):
            el = shape_element(element)
            if el:
                data.append(el)
                if pretty:
                    fo.write(json.dumps(el, indent=2)+"\n")
                else:
                    fo.write(json.dumps(el) + "\n")

    return data

def store_data(data):
    from pymongo import MongoClient
    client = MongoClient("mongodb://localhost:27017")
    db = client.project2
    db.openstreet.remove()
```

```
    db.openstreet.insert(data)
    db.openstreet.ensureIndex({"pos":"2dsphere"})

def test():
    data = process_map('map', True)
    pprint.pprint(data[-1])

    store_data(data)

if __name__ == "__main__":
    test()
```

Appendix 4. Result of running variety.js script to data

key	types	occurrences	percents
_id	ObjectId	278378	100
created	Object	278378	100
created.changeset	String	278378	100
created.timestamp	String	278378	100
created.uid	String	278378	100
created.user	String	278378	100
created.version	String	278378	100
id	String	278378	100
type	String	278378	100
pos	Array	257450	92.48216453886442
node_refs	Array	20928	7.517835461135578
highway	String	13137	4.719122919196201
name	String	7074	2.541149085057009
building	String	5422	1.9477113852387762
created_by	String	4385	1.5751963158008178
source	String	3064	1.1006616902197732
surface	String	2471	0.8876419832026956
oneway	String	2468	0.886564311834987
amenity	String	2286	0.8211855821939953
is_in	String	1813	0.6512727298852639
address	Object	1286	0.46196179295777684
address.street	String	1144	0.41095201488623384
address.city	String	1066	0.3829325593258088
place	String	1018	0.3656898174424703
natural	String	976	0.35060241829454913
lanes	String	942	0.33838880946051775
is_in:country	String	873	0.3136023680032186
is_in:state	String	860	0.3089324587431478
tourism	String	717	0.2575634568823686
shop	String	690	0.2478644145729907
landuse	String	614	0.2205634065910381
is_in:county	String	593	0.21301970701707748
is_in:region	String	593	0.21301970701707748
service	String	536	0.19254395103061306
address.housenumber	String	520	0.18679637040283356
access	String	403	0.14476718706219602
bridge	String	398	0.14297106811601493
foot	String	373	0.13399047338510944
layer	String	353	0.12680599760038508
leisure	String	343	0.12321375970802291
waterway	String	343	0.12321375970802291
note	String	341	0.12249531212955046
smoothness	String	325	0.11674773150177097
religion	String	307	0.11028170329551905
barrier	String	299	0.1074079129816293
is_in:country_code	String	279	0.10022343719690494
bicycle	String	277	0.09950498961843249
cuisine	String	277	0.09950498961843249
gns:dsg	String	275	0.09878654203996007
website	String	255	0.09160206625523569
opening_hours	String	219	0.07867000984273183

address.postcode	String	193	0.06933019132259015
phone	String	184	0.06609717721946418
building:levels	String	176	0.06322338690557443
width	String	163	0.058553477645503595
operator	String	159	0.05711658248855873
office	String	153	0.05496123975314141
power	String	148	0.053165120806960314
man_made	String	144	0.05172822565001545
sport	String	144	0.05172822565001545
parking	String	142	0.051009778071543006
motor_vehicle	String	130	0.04669909260070839
address.housename	String	110	0.03951461681598402
aeroway	String	106	0.038077721659039145
fee	String	99	0.03556315513438562
atm	String	98	0.035203931345149396
tracktype	String	92	0.03304858860973209
smoking	String	91	0.03268936482049587
horse	String	87	0.031252469663551
motorcycle	String	87	0.031252469663551
supervised	String	87	0.031252469663551
description	String	86	0.03089324587431478
park_ride	String	84	0.030174798295842343
maxspeed	String	78	0.02801945556042503
noexit	String	68	0.02442721766806285
admin_level	String	65	0.023349546300354195
internet_access	String	62	0.02227187493264554
ele	String	54	0.019398084618755793
area	String	51	0.018320413251047135
wikipedia	String	51	0.018320413251047135
building:use	String	50	0.01796118946181092
name:en	String	50	0.01796118946181092
email	String	48	0.01724274188333848
history	String	48	0.01724274188333848
stars	String	46	0.016524294304866044
historic	String	45	0.016165070515629827
ref	String	44	0.015805846726393607
tunnel	String	42	0.015087399147921171
denotation	String	40	0.014368951569448734
fax	String	38	0.013650503990976297
alt_name	String	36	0.012932056412503862
payment:bitcoin	String	36	0.012932056412503862
fixme	String	35	0.012572832623267643
denomination	String	34	0.012213608834031425
sidewalk	String	34	0.012213608834031425
information	String	33	0.011854385044795208
wheelchair	String	33	0.011854385044795208
trail_visibility	String	32	0.01149516125555899
motorcar	String	29	0.010417489887850332
name:id	String	29	0.010417489887850332
mtb:scale	String	28	0.010058266098614113
junction	String	27	0.009699042309377896
boat	String	26	0.009339818520141678
content	String	21	0.007543699573960586
designation	String	20	0.007184475784724367
lit	String	20	0.007184475784724367
water	String	20	0.007184475784724367
route	String	18	0.006466028206251931

seamark:type	String	18	0.006466028206251931
seamark:name	String	17	0.006106804417015712
brand	String	16	0.005747580627779495
internet_access:fee	String	16	0.005747580627779495
capacity	String	15	0.005388356838543275
entrance	String	15	0.005388356838543275
school:type_idn	String	15	0.005388356838543275
emergency	String	14	0.005029133049307057
embankment	String	13	0.004669909260070839
sac_scale	String	12	0.00431068547083462
wifi	String	12	0.00431068547083462
contact:phone	String	11	0.003951461681598402
outdoor_seating	String	11	0.003951461681598402
wetland	String	11	0.003951461681598402
building:roof	String	10	0.0035922378923621836
building:structure	String	10	0.0035922378923621836
building:walls	String	10	0.0035922378923621836
covered	String	10	0.0035922378923621836
int_name	String	10	0.0035922378923621836
intermittent	String	10	0.0035922378923621836
school	String	9	0.0032330141031259654
boundary	String	8	0.0028737903138897473
craft	String	8	0.0028737903138897473
is_in:beach	String	8	0.0028737903138897473
seamark:fixme	String	8	0.0028737903138897473
ski	String	8	0.0028737903138897473
snowmobile	String	8	0.0028737903138897473
contact:email	String	7	0.0025145665246535283
country	String	7	0.0025145665246535283
incline	String	7	0.0025145665246535283
seamark:longname	String	7	0.0025145665246535283
crossing	String	6	0.00215534273541731
cutting	String	6	0.00215534273541731
geological	String	6	0.00215534273541731
wpt_description	String	6	0.00215534273541731
address.country	String	5	0.0017961189461810918
comment	String	5	0.0017961189461810918
diet:vegetarian	String	5	0.0017961189461810918
hoops	String	5	0.0017961189461810918
name:ban	String	5	0.0017961189461810918
organic	String	5	0.0017961189461810918
population	String	5	0.0017961189461810918
role	String	5	0.0017961189461810918
shelter_type	String	5	0.0017961189461810918
status	String	5	0.0017961189461810918
tower:type	String	5	0.0017961189461810918
type_idn	String	5	0.0017961189461810918
construction	String	4	0.0014368951569448736
contact:website	String	4	0.0014368951569448736
district	String	4	0.0014368951569448736
parts	String	4	0.0014368951569448736
region	String	4	0.0014368951569448736
repair	String	4	0.0014368951569448736
road	String	4	0.0014368951569448736
seamark:information	String	4	0.0014368951569448736
shelter	String	4	0.0014368951569448736
takeaway	String	4	0.0014368951569448736

wpt_symbol	String	4	0.0014368951569448736
cargo	String	3	0.001077671367708655
contact:fax	String	3	0.001077671367708655
crop	String	3	0.001077671367708655
dispensing	String	3	0.001077671367708655
drive_in	String	3	0.001077671367708655
farm_auxiliary:type_idn	String	3	0.001077671367708655
fuel:diesel	String	3	0.001077671367708655
generator:source	String	3	0.001077671367708655
last_eruption	String	3	0.001077671367708655
material	String	3	0.001077671367708655
name:ru	String	3	0.001077671367708655
payment:litecoin	String	3	0.001077671367708655
sale	String	3	0.001077671367708655
source:road	String	3	0.001077671367708655
subdistrict	String	3	0.001077671367708655
wood	String	3	0.001077671367708655
address.full	String	2	0.0007184475784724368
address.province	String	2	0.0007184475784724368
artwork_type	String	2	0.0007184475784724368
canal:type_idn	String	2	0.0007184475784724368
central_line	String	2	0.0007184475784724368
clinic:type_id	String	2	0.0007184475784724368
cycleway	String	2	0.0007184475784724368
drive_through	String	2	0.0007184475784724368
ford	String	2	0.0007184475784724368
fuel:biodiesel	String	2	0.0007184475784724368
fuel:biogas	String	2	0.0007184475784724368
fuel:cng	String	2	0.0007184475784724368
fuel:electricity	String	2	0.0007184475784724368
fuel:lpg	String	2	0.0007184475784724368
full_name	String	2	0.0007184475784724368
generator:method	String	2	0.0007184475784724368
household	String	2	0.0007184475784724368
is_in:town	String	2	0.0007184475784724368
key	String	2	0.0007184475784724368
maritime	String	2	0.0007184475784724368
microbrewery	String	2	0.0007184475784724368
name:ja	String	2	0.0007184475784724368
name:uk	String	2	0.0007184475784724368
old_name	String	2	0.0007184475784724368
recycling_type	String	2	0.0007184475784724368
sloped_curb	String	2	0.0007184475784724368
	String	1	0.0003592237892362184
access:car	String	1	0.0003592237892362184
access:motorbike	String	1	0.0003592237892362184
access:roof	String	1	0.0003592237892362184
address.area	String	1	0.0003592237892362184
address.place	String	1	0.0003592237892362184
address.po_box	String	1	0.0003592237892362184
address.state	String	1	0.0003592237892362184
address.unit	String	1	0.0003592237892362184
aerialway	String	1	0.0003592237892362184
amadeus	String	1	0.0003592237892362184
backrest	String	1	0.0003592237892362184
bench	String	1	0.0003592237892362184
board_type	String	1	0.0003592237892362184

border_type	String	1	0.0003592237892362184
bus	String	1	0.0003592237892362184
class	String	1	0.0003592237892362184
clothes	String	1	0.0003592237892362184
colour	String	1	0.0003592237892362184
communication:microwave	String	1	0.0003592237892362184
delivery	String	1	0.0003592237892362184
description:en	String	1	0.0003592237892362184
diet:vegan	String	1	0.0003592237892362184
diplomatic	String	1	0.0003592237892362184
direction	String	1	0.0003592237892362184
fence_type	String	1	0.0003592237892362184
food	String	1	0.0003592237892362184
garden	String	1	0.0003592237892362184
harbour	String	1	0.0003592237892362184
hgv	String	1	0.0003592237892362184
iata	String	1	0.0003592237892362184
icao	String	1	0.0003592237892362184
image	String	1	0.0003592237892362184
is_in:district	String	1	0.0003592237892362184
is_in:island	String	1	0.0003592237892362184
is_in:province	String	1	0.0003592237892362184
is_in:subdistrict	String	1	0.0003592237892362184
landcover	String	1	0.0003592237892362184
landmark	String	1	0.0003592237892362184
loc_name	String	1	0.0003592237892362184
local_ref	String	1	0.0003592237892362184
maxheight	String	1	0.0003592237892362184
maxwidth	String	1	0.0003592237892362184
mtb:description	String	1	0.0003592237892362184
name:de	String	1	0.0003592237892362184
name:in	String	1	0.0003592237892362184
name:kn	String	1	0.0003592237892362184
name:zh	String	1	0.0003592237892362184
onr	String	1	0.0003592237892362184
pathway	String	1	0.0003592237892362184
payment:dogecoin	String	1	0.0003592237892362184
rent	String	1	0.0003592237892362184
rental	String	1	0.0003592237892362184
roof	String	1	0.0003592237892362184
seamark:topmark	String	1	0.0003592237892362184
seats	String	1	0.0003592237892362184
second_hand	String	1	0.0003592237892362184
tactile_paving	String	1	0.0003592237892362184
touism	String	1	0.0003592237892362184
tower	String	1	0.0003592237892362184
trail	String	1	0.0003592237892362184
url	String	1	0.0003592237892362184
walkway	String	1	0.0003592237892362184
wikipedia:en	String	1	0.0003592237892362184

Appendix 5. find-combined.py

```
# Find combined number of users making up 1% of all contributions.

def print_combined_users(db):
    contributions = db.openstreet.aggregate([
    {"$group":{"_id": "$created.user","count":{"$sum": 1}}},
    {"$sort": {"count": 1}}])['result']

    total = db.openstreet.count() / 100
    total_calc = 0
    comb_users = 0
    for c in contributions:
        total_calc += c['count']
        comb_users += 1
        if total_calc >= total:
            break

    percentage = round(float(comb_users) / float(len(contributions)) * 100,
2)
    print "Combined number of users making up only 1% ({0}) of posts: {1}
({2}% of all users)".format(total, comb_users, percentage)

def get_db():
    from pymongo import MongoClient
    client = MongoClient("mongodb://localhost:27017")
    db = client.project2
    return db

def test():
    db = get_db()
    print_combined_users(db)

if __name__ == "__main__":
    test()
```