Database Connection

SHIP enables you to use a variety of databases. It does this by using SQLAlchemy, an ORM which is independint from databses like PostgreSQL, MySQL, MSSQL, Oracle or Sqlite. SHIP uses a global session which may be hooked up to the database of your choosing.

In [17]:
```
from ship import config
config.connect('sqlite:///examples.db')
```

Before using a database, it needs to be filled with data. The premiums records for 2012 which are part of SHIP are csv files from the Bundesamt für Gesundheit BAG. The regions come from an excel file hosted on priminfo.ch and the insurers data are from another excel file hosted on said site. We will do a better job of explaining the details of acquiring the data for yourself once the 2013 premiums records are released.

Anywho, to fill your database, do the following, after you've connected of course.

In [18]:
```
from ship import load
load.all()
```

This will load all available premium records of all years into the database. Run multiple times, only new files are imported.

Querying

There are a number of queries and methods implemented to allow some basic data retrieval. We thought of doing an API for this, but without anyone using it we would just be doing that in an ivory tower kind of setting. So for now these methods are just something to play around with.

For example, take the following helper methods:

Years available in the database

In [19]:
```
from ship import db
db.years()
```

Out[19]:
```
[2012]
```

Possible insurance types

In [20]:
```
', '.join(db.insurance_types())
```

Out[20]:
```
u'Base, DIV, HAM_RDS,
HMO'
```

Possible cantons

In [21]:
```
', '.join(db.cantons())
```

Out[21]:
```
u'AG, AI, AR, BE, BL, BS, FR, GE, GL, GR, JU, LU, NE, NW, OW,
SG, SH, SO, SZ, TG, TI, UR, VD, VS, ZG, ZH'
```

Possible franchises by age

In [22]:
```
from pandas import DataFrame, Series
DataFrame({
    "Franchises": [
        ' / '.join(map(str, db.franchises(0))),
        ' / '.join(map(str, db.franchises(19)))
    ]}, index=["Under 18", "Over 18"]
)
```

Out[22]:

	Franchises
Under 18	0 / 100 / 200 / 300 / 400 / 500 / 600
Over 18	300 / 500 / 1000 / 1500 / 2000 / 2500

There's more!

Execute SQL

You may also use SQL directly if you feel uncomfortable with SQLAlchemy.

Let's see how many records per canton are in the database so far using an SQL query.

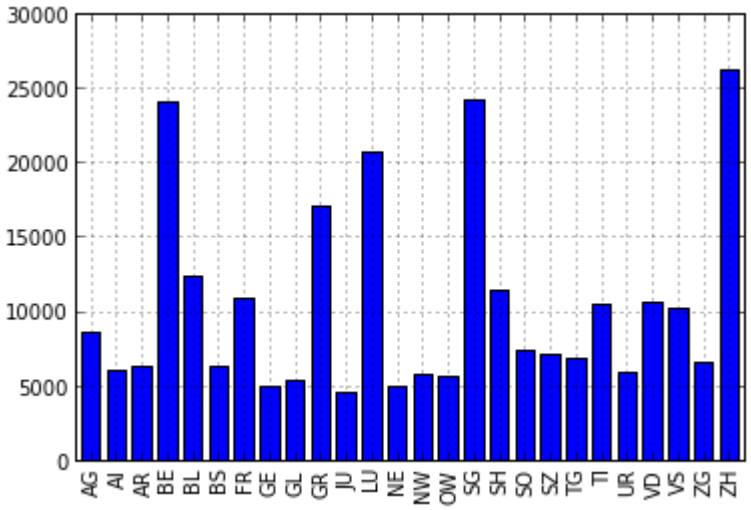
In [23]:
```
from ship import db
from pandas import DataFrame, Series

query = """
    SELECT   canton, COUNT(*) as records
    FROM     premiums
    WHERE    "group" == "CH"
    GROUP BY canton
    ORDER BY records DESC
"""

count_by_canton = dict()
for canton, count in db.execute(query):
    count_by_canton[canton] = count

Series(count_by_canton).plot(kind="bar")
```

Out[23]:
```
<matplotlib.axes.AxesSubplot at
0x1067e8bd0>
```



Query Objects

There are also two objects that help query the database without knowing it too well. The db.Premiums and db.Towns classes. These classes wrap around an SQLAlchemy Query instance and provide some predefined ways of filtering the query. The query may be accessed at any time for customization.

The following example uses the Premiums class to calculate the average premiums for each canton.

In [24]:
```
from ship.models import Premium
from ship.config import session
from sqlalchemy import func

averages = dict()

for canton in db.cantons():
    premiums = db.Premiums()

    # for the current year
    premiums = premiums.for_year(2012)

    # limit to the given canton
    premiums = premiums.for_canton(canton)

    # for adults only
    premiums = premiums.for_adults()

    # aggregate the query
    query = premiums.q.with_entities(func.avg(Premium.premium))
    average = query.all()

    # convert to Francs (the values are stored in Rappen to avoid
    averages[canton] = db.unpack(average)[0] / 100.0

Series(averages).order(ascending=False).plot(kind="bar")
```

Out[24]:
```
<matplotlib.axes.AxesSubplot at
0x107abb250>
```