Beton

February 19, 2018

In [16]:
```
%matplotlib inline
import pandas as pd

# De data staan in CSV formaat, met tabs ('\t') als separator
# Decimale getallen zijn met een komma genoteerd
df = pd.read_csv('DataConcrete.txt',sep='\t',decimal=',')

df.head() # We bekijken de eerste vijf rijen
```

Out[16]:

	Cement	Blast furnace slags	Fly ashes	Water	Superplasticizers	Coarse aggregates	Fine aggregates	Age	Compressive strength
0	540.0	0.0	0	162	2.5	1040.0	676.0	28	79.99
1	540.0	0.0	0	162	2.5	1055.0	676.0	28	61.89
2	332.5	142.5	0	228	0.0	932.0	594.0	270	40.27
3	332.5	142.5	0	228	0.0	932.0	594.0	365	41.05
4	198.6	132.4	0	192	0.0	978.4	825.5	360	44.30

In [17]:
```
df.describe() # We bekijken wat statistieken van elke kolom
```

Out[17]:

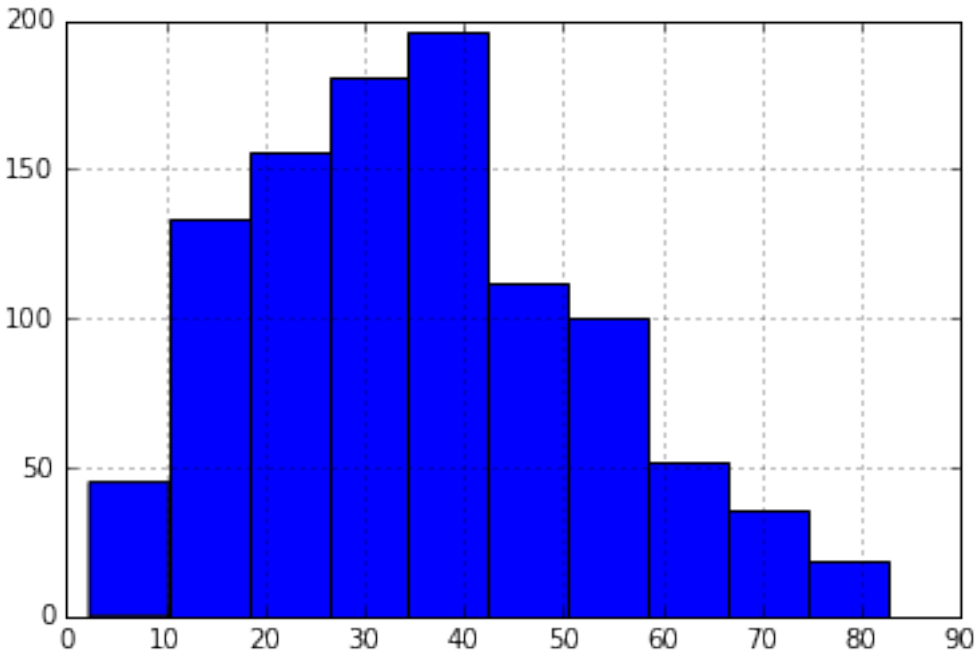
	Cement	Blast furnace slags	Fly ashes	Water	Superplasticizers	Coarse aggregates	Fine aggregates	Age
count	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000
mean	281.167864	73.895825	54.188350	181.567282	6.204660	972.918932	773.580485	45.662136
std	104.506364	86.279342	63.997004	21.354219				
min	102.000000	0.000000	0.000000	121.800000				
25%	192.375000	0.000000	0.000000	164.900000				
50%	272.900000	22.000000	0.000000	185.000000				
75%	350.000000	142.950000	118.300000	192.000000				
max	540.000000	359.400000	200.100000	247.000000				

```
std              5.973841          77.753954          80.175980     63.169912
min              0.000000         801.000000         594.000000      1.000000
25%              0.000000         932.000000         730.950000      7.000000
50%              6.400000         968.000000         779.500000     28.000000
75%             10.200000        1029.400000         824.000000     56.000000
max             32.200000        1145.000000         992.600000    365.000000
```

```
       Compressive strength
count           1030.000000
mean              35.817961
std               16.705742
min                2.330000
25%               23.710000
50%               34.445000
75%               46.135000
max               82.600000
```

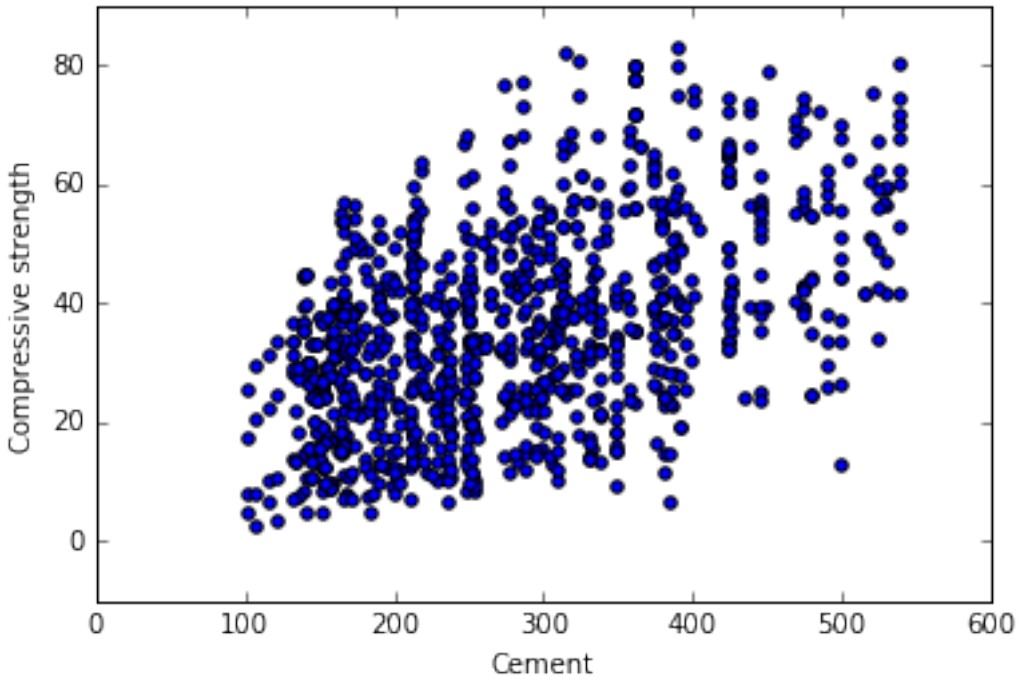
In [18]:
```
# De bedoeling is om de laatste kolom te voorspellen
# We bekijken eens of er veel variatie in deze waardes zit,
# door er een histogram van te maken
df['Compressive strength'].hist()
```

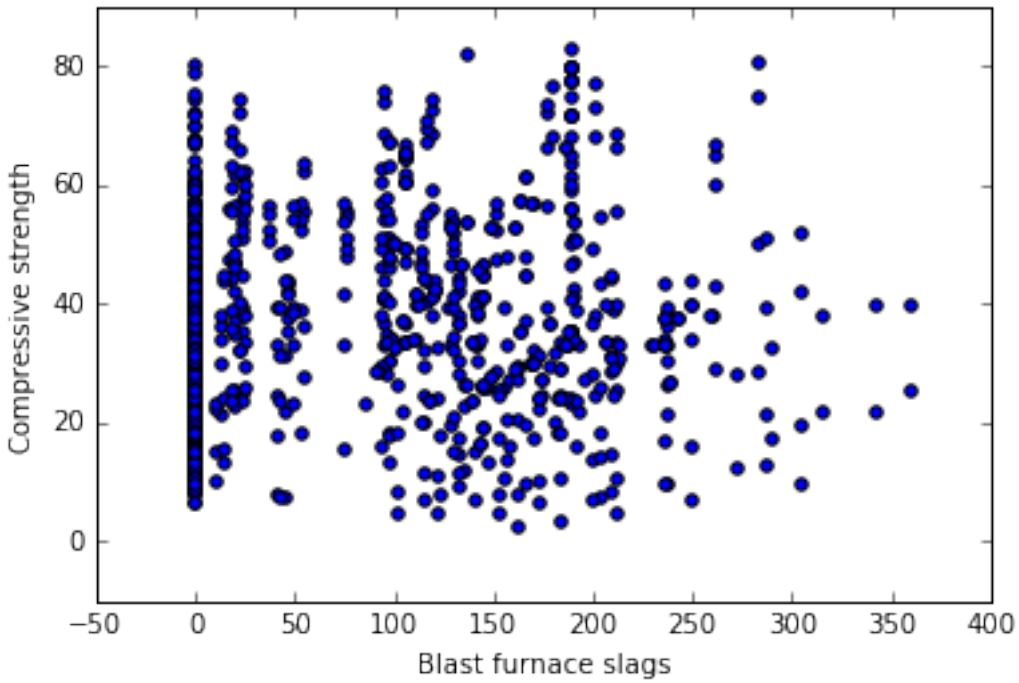
Out[18]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x10e4d55d0>
```

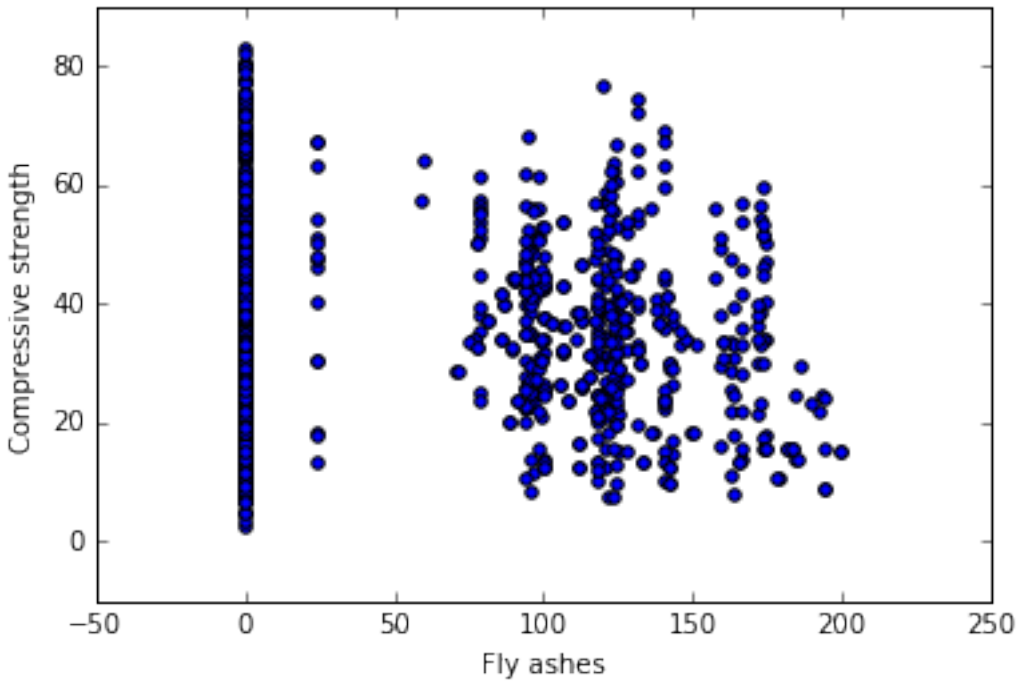


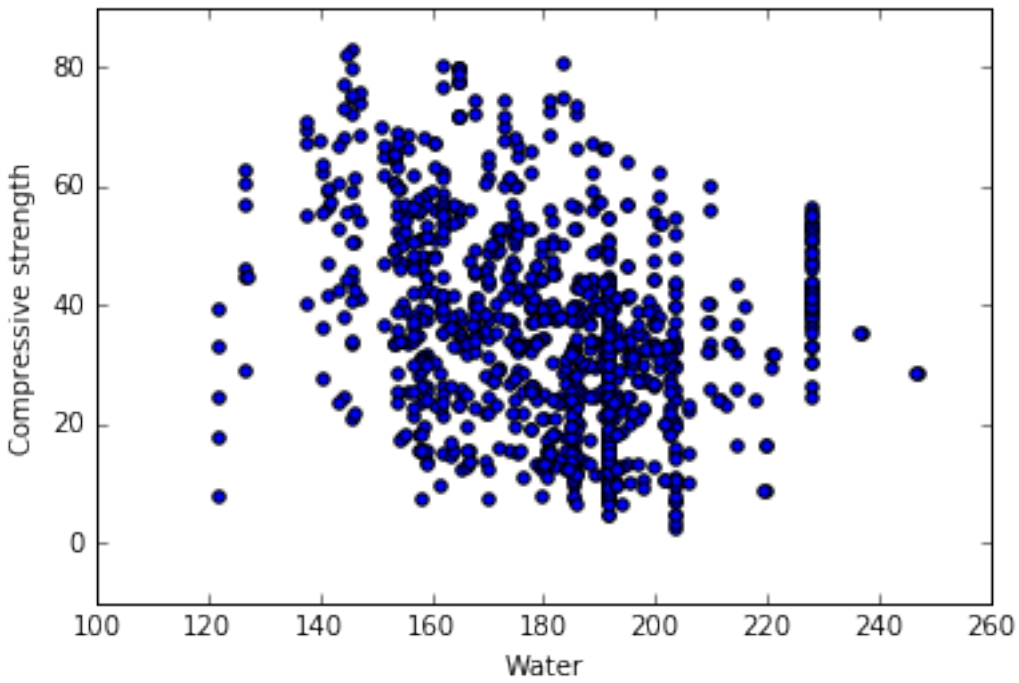
In [19]:
```
# Om een idee te krijgen van welke variabelen een invloed hebben
# op de doelkolom, maken we een aantal scatterplots
```

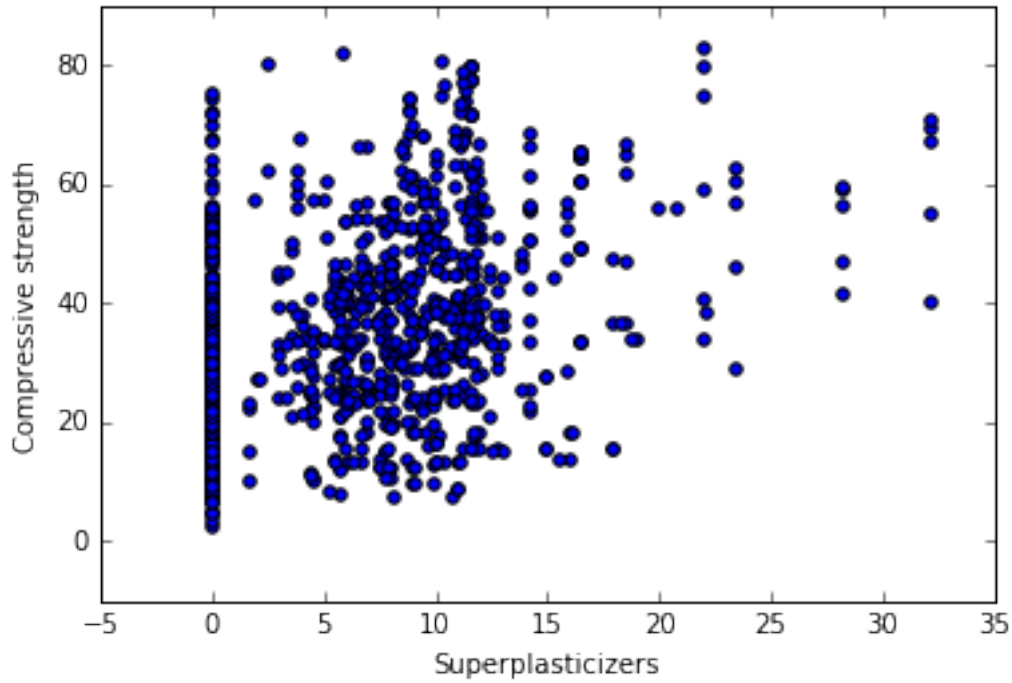
```
doel = 'Compressive strength'
for kolom in df.columns:
    if kolom != doel:
        df.plot(kolom, doel, kind='scatter')
```

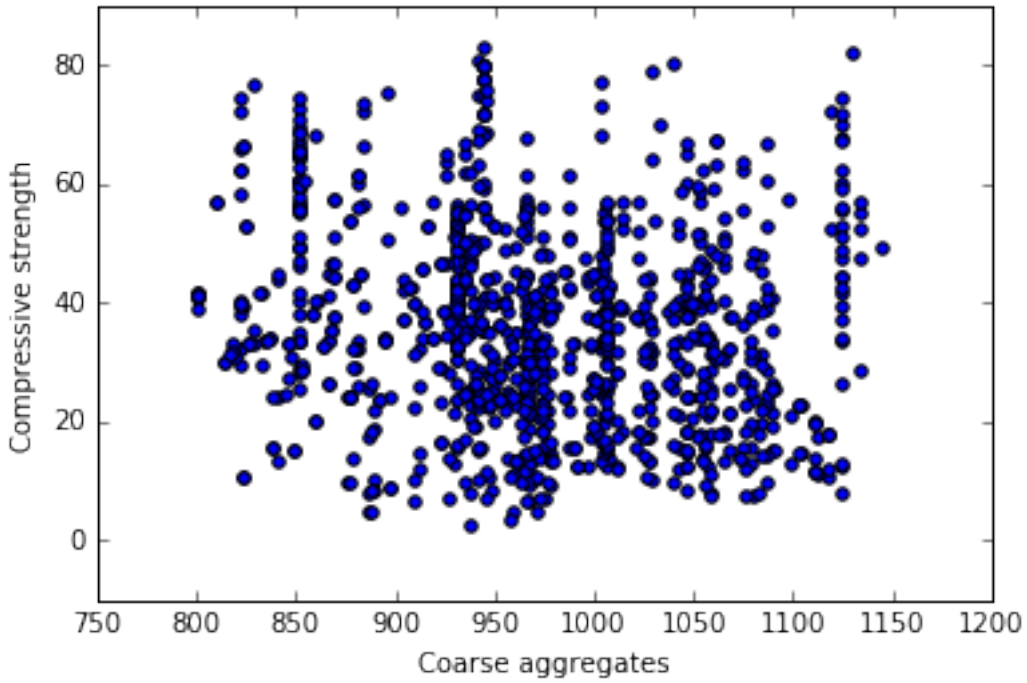


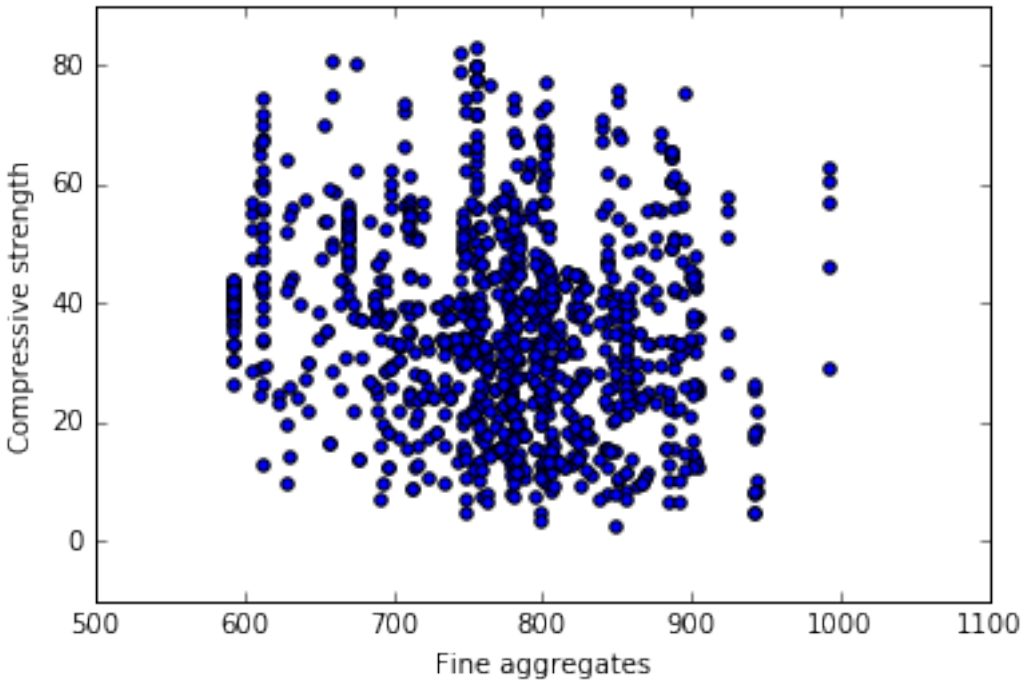


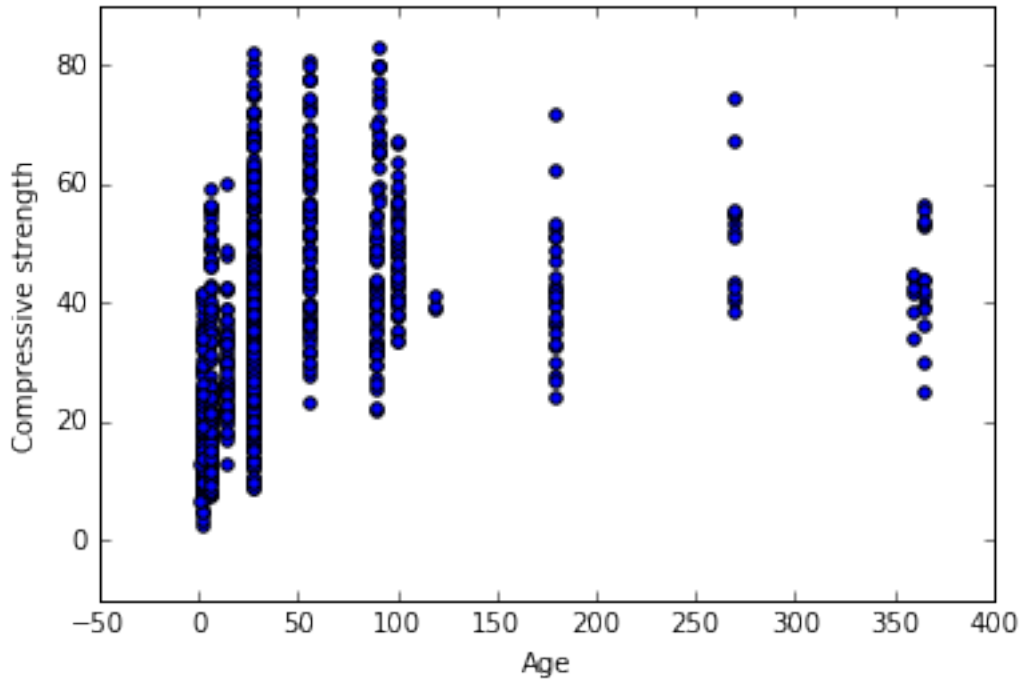

Compressive strength
Fly ashes
80
60
40
20
0
−50
0
50
100
150
200
250


Compressive strength
Water
80
60
40
20
0
100
120
140
160
180
200
220
240
260


Compressive strength
Superplasticizers
0
20
40
60
80
−5
0
5
10
15
20
25
30
35


Compressive strength
Coarse aggregates
0
20
40
60
80
750
800
850
900
950
1000
1050
1100
1150
1200

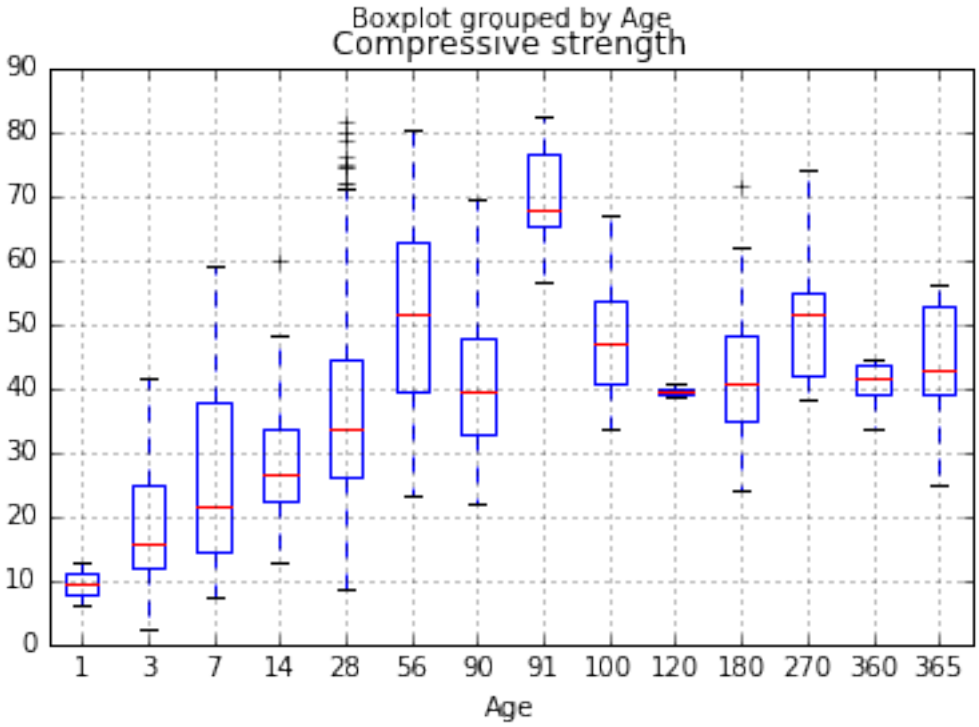




In [20]:
```
# Een eerste observatie is dat "jong" beton minder sterk
# lijkt dan oud beton.
```

```
# We controleren of dit klopt d.m.v. een boxplot
df.boxplot(column='Compressive strength', by='Age')
```

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x108ec3a10>



In [21]:
```
# Op basis van deze boxplot, besluiten we dat beton
# minstens 50 dagen moet uitharden alvorens zijn
# maximale sterkte te bereiken.
# Voor de verdere analyse beperken we ons dus tot beton
# dat al minstens 50 dagen oud is.
df = df[df.Age >= 50]
df.describe()  # Nu hebben we nog 281 van de 1030 datapunten over
```

Out[21]:

	Cement	Blast furnace slags	Fly ashes	Water	\
count	281.000000	281.000000	281.000000	281.000000	
mean	294.698932	63.120996	49.131673	182.826690	
std	101.954243	79.322306	62.149342	25.763355	
min	102.000000	0.000000	0.000000	121.800000	
25%	213.500000	0.000000	0.000000	161.900000	
50%	277.200000	19.000000	0.000000	186.000000	
75%	362.600000	116.000000	118.200000	195.500000	
max	540.000000	305.300000	174.700000	228.000000	

	Superplasticizers	Coarse aggregates	Fine aggregates	Age	\

```
count         281.000000         281.000000         281.000000  281.000000
mean            5.851957         979.701423         773.964769  117.359431
std             6.354363          68.063585          91.254666   85.162213
min             0.000000         822.000000         594.000000   56.000000
25%             0.000000         932.000000         746.600000   56.000000
50%             5.500000         968.000000         781.000000   90.000000
75%            10.200000        1028.400000         845.000000  100.000000
max            32.200000        1134.300000         992.600000  365.000000

       Compressive strength
count            281.000000
mean              48.565765
std               13.468959
min               21.860000
25%               39.000000
50%               46.930000
75%               56.340000
max               82.600000
```

In [22]:
```
# In onze scatterplots valt te zien dat er een (positief) verband
# lijkt tussen de hoeveelheid cement en de sterkte.
# We trainen een linear regressie model dat het verband tussen
# deze twee variabelen vat

from sklearn import linear_model
reg_cem = linear_model.LinearRegression()
X = df[['Cement']] # Twee paar vierkante haakjes omdat we een
                   # 2-dimensionale 281 x 1 matrix nodig hebben,
                   # in plaats van een 1-dimensionale vector van 281 lang.
y = df[doel]
reg_cem.fit(X, y)
# We bekijken eens hoe de geleerde parameters van het model eruit zien
print "Sterkte = "+str(reg_cem.coef_[0])+' * Cement + '+str(reg_cem.intercept_)
```

```
Sterkte = 0.0648078680775 * Cement + 29.466955592
```

In [23]:
```
# Als er bv. 100 kg/m^3 cement aanwezig is,
# zouden we een sterkte verwachten van:
cement = 100
print 0.0648078680775 * cement + 29.466955592
```

```
35.9477423997
```

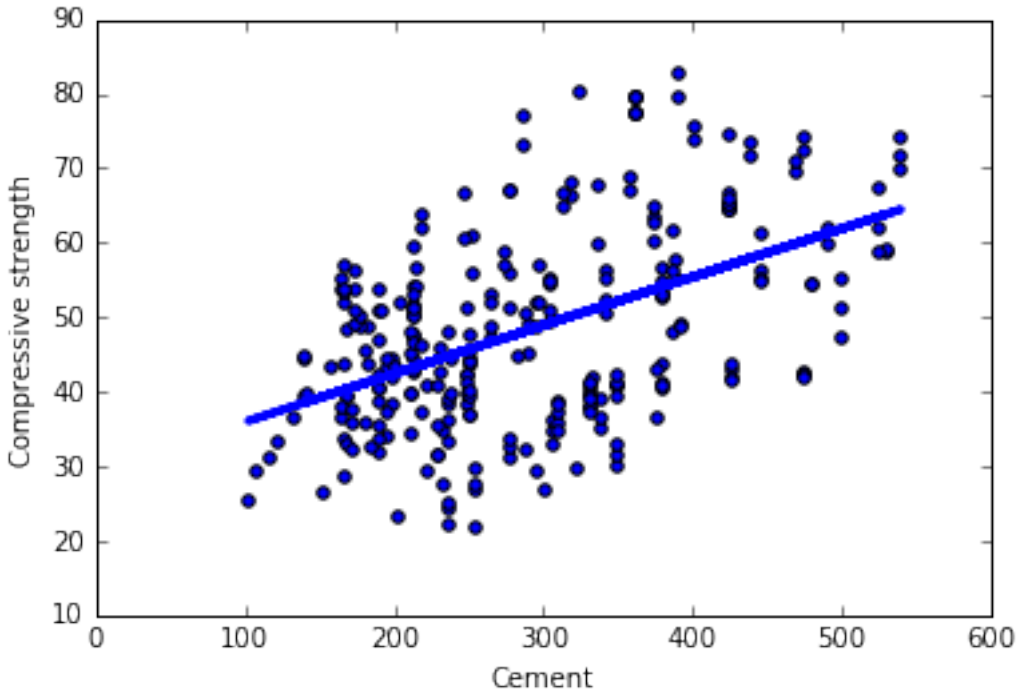
In [24]:
```
# Dit is ook de waarde die berekend wordt
# door de methode "predict" van het model.
print reg_cem.predict(cement)
```

```
[ 35.9477424]
```

In [26]:
```
# We tekenen even opnieuw een scatterplot
df.plot(x='Cement', y=doel, kind='scatter')

# En gebruiken nu matplotlib om hier een blauwe lijn aan te voegen,
# die de door het model voorspelde waardes toont
import matplotlib.pyplot as plt
plt.plot(X, reg_cem.predict(X), color='blue', linewidth=3)
```

Out[26]:
```
[<matplotlib.lines.Line2D at 0x10edf92d0>]
```

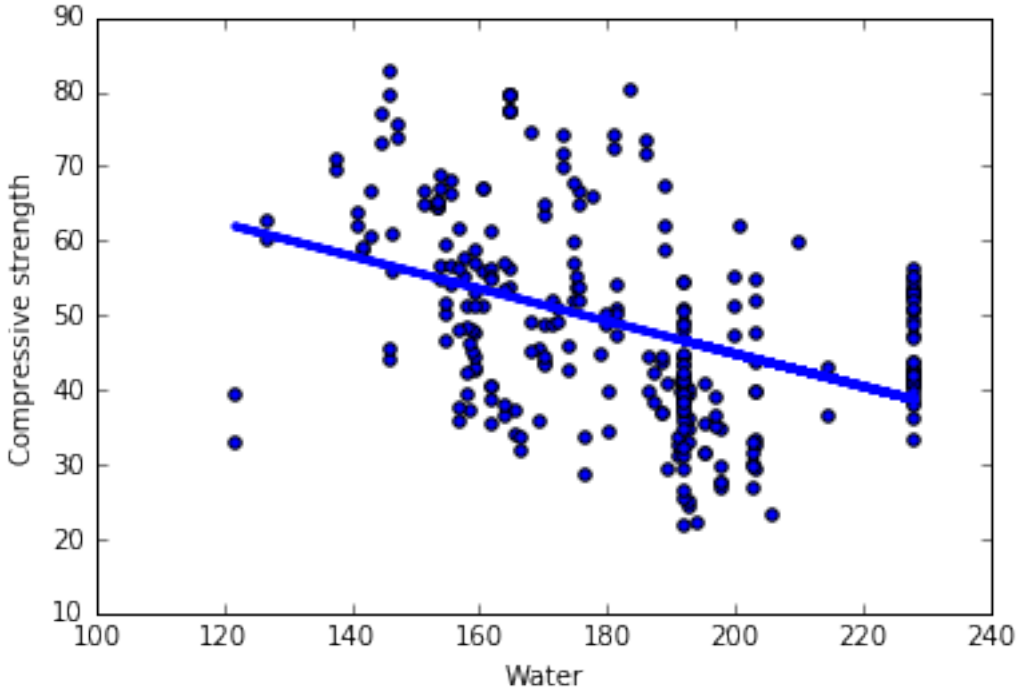


In [27]:
```
# In de scatterplots leek er ook een licht negatief verband te zijn tussen
# de hoeveelheid water en de sterkte.
reg_water = linear_model.LinearRegression()
reg_water.fit(df[['Water']], df[doel])
# We zien dat dit negatief verband inderdaad gevonden wordt:
print "Sterkte = "+str(reg_water.coef_[0])+' * Water + '+str(reg_water.intercept_)
```

```
Sterkte = -0.218466316259 * Water + 88.5072386882
```

In [28]:
```
# Ook deze rechte tekenen we op een scatterplot
df.plot(x='Water', y=doel, kind='scatter')
X = df[['Water']]
plt.plot(X, reg_water.predict(X), color='blue', linewidth=3)
```

Out[28]: [<matplotlib.lines.Line2D at 0x10ef11850>]



In [29]:
```
# Met het blote oog lijkt het erop dat deze lijn
# de data iets minder goed vat dan het model dat
# 'cement' als onafhankelijke variabele had.
# We kunnen eens controleren of dit klopt door
# naar de R^2 waarde te kijken
X = df[['Cement']]
y = df[doel]
print "R^2 cement: " + str(reg_cem.score(X, y))
X = df[['Water']]
print "R^2 water: " + str(reg_water.score(X, y))
```

```
R^2 cement: 0.240656732228
R^2 water: 0.174624727637
```

In [30]:
```
# We zien dat 'cement' inderdaad beter de data vat
# dan water.
# Het is natuurlijk ook mogelijk om een model te leren
# waarin beide variabelen gebruikt worden. Hiervan
# zouden we verwachten dat het beter scoort dan beide
# afzonderlijk.
reg_beide = linear_model.LinearRegression()
```

```
        X = df[['Cement','Water']]
        reg_beide.fit(X, y)
        print ("Sterkte = " + str(reg_beide.coef_[0]) + ' * Cement + ' +
                              str(reg_beide.coef_[1]) + ' * Water + ' +
                              str(reg_beide.intercept_))
        print "R^2 beide: " + str(reg_beide.score(X, y))
```

```
Sterkte = 0.0685745987997 * Cement + -0.235633409176 * Water + 71.4369804149
R^2 beide: 0.442990809855
```

```
In [31]: # Moesten we in het begin de datapunten jonger dan 50 dagen
         # niet hebben weggelaten, zouden we een minder sterk verband
         # gevonden hebben.
         df = pd.read_csv('DataConcrete.txt',sep='\t',decimal=',')
         X = df[['Cement','Water']]
         y = df[doel]
         reg_beide.fit(X,y)
         print "R^2 beide: " + str(reg_beide.score(X, y))
```

```
R^2 beide: 0.310261560311
```