In [1]:
```
import talk.config as con

%matplotlib inline

con.config_mosek()
con.config_matplotlib()
con.config_configManager()
```

```
Set MOSEKLM_LICENSE_FILE environment variable
Update ConfigManager
```

Constrained regression

Let $\mathbf{A} \in \mathbb{R}^{n \times m}$ and $\mathbf{b} \in \mathbb{R}^{n}$. We solve the constrained least squares problem:

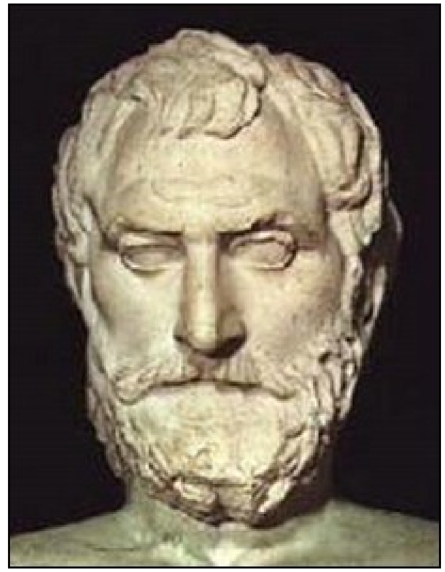
$$\begin{aligned} \mathbf{x}^* = \arg \min_{\mathbf{x} \in \mathbb{R}^m} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 \\ \text{s.t. } \Sigma\, x_i = 1 \\ \mathbf{x} \geq 0 \end{aligned}$$

Examples:

- Tracking an index (index in $\mathbf{b}$, assets in $\mathbf{A}$)
- Constructing an indicator, factor analyis, ...
- Approximation...
- ...

Regression is the **Swiss army knife** of professional quant finance.

Thales of Miletus (c. 624 BC - c. 546 BC).



The normal equations

As we (probably) all know

$$\mathbf{x}^* = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{x}$$

solves

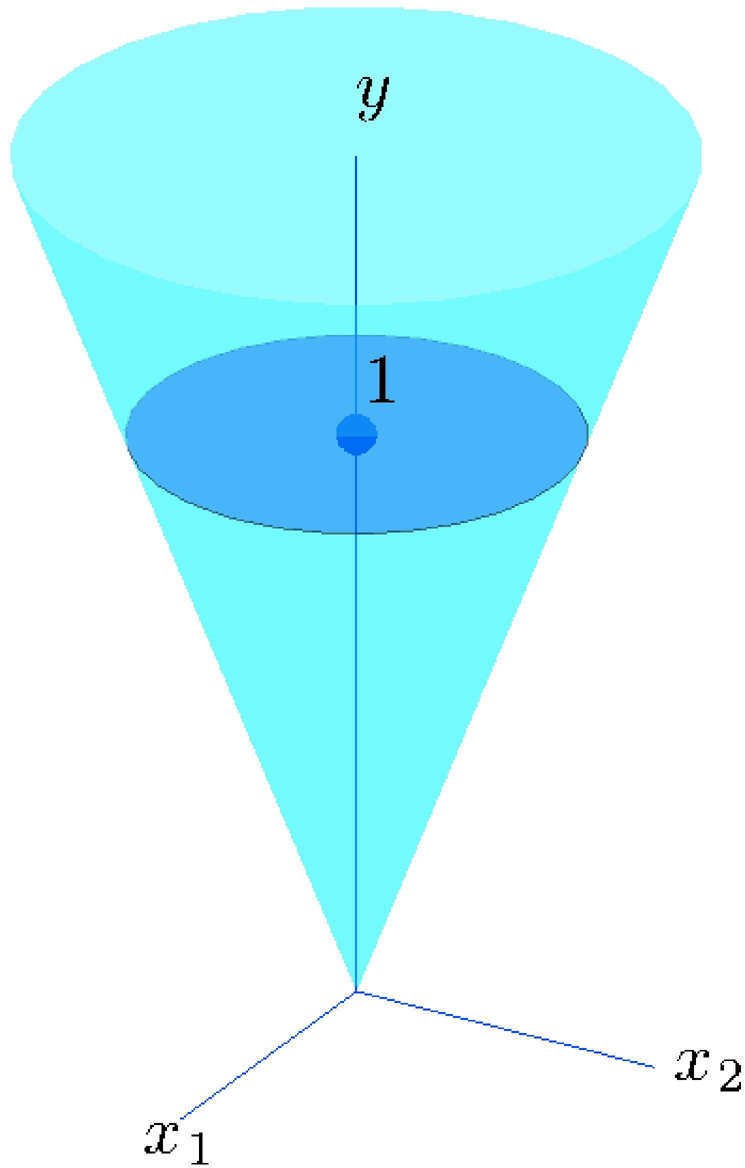
$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathbb{R}^m} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2$$

Almost there?

Shall we apply the sculptor method?

- We could delete the negative entries (really bad if they are all negative)
- We could scale the surviving entries to enforce the $\Sigma x_i = 1$.

Done?



$$y \geq \sqrt{x_1^2 + x_2^2} = \|\mathbf{x}\|_2$$

Conic Programming

We introduce an auxiliary scalar z:

$$
\begin{aligned}
\min_{z\in\mathbb{R},\mathbf{x}\in\mathbb{R}^m} \; & z \\
\text{s.t. } & z \geq \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 \\
& \Sigma\, x_i = 1 \\
& \mathbf{x} \geq 0
\end{aligned}
$$

We introduce an auxiliary vector $\mathbf{y} \in \mathbb{R}^n$:

$$\begin{aligned} \min_{z\in\mathbb{R},\mathbf{x}\in\mathbb{R}^m,\mathbf{y}\in\mathbb{R}^n} \quad & z \\ \text{s.t.} \quad & z \geq \|y\|_2 \\ & \mathbf{y} = \mathbf{A}\mathbf{x} - \mathbf{b} \\ & \Sigma\, x_i = 1 \\ & \mathbf{x} \geq 0 \end{aligned}$$

We **lifted** the problem from a m dimensional space into a $m + n + 1$ dimensional space.

Alternative notation:

$$z \geq \|y\|_2 \Leftrightarrow \ [z, y] \in \mathcal{Q}_{n+1}$$

Application: Implementing a minimum variance portfolio

The ith column of $\mathbf{R}$ is the time series of returns for the ith asset. Hence to minimize the variance of a portfolio (a linear combination of assets) we solve:

$$\begin{aligned}
\mathbf{w}^* = \arg \min_{\mathbf{w} \in \mathbb{R}^m} \|\mathbf{R}\mathbf{w} - \mathbf{0}\|_2 \\
\text{s.t. } \Sigma\, w_i = 1 \\
\mathbf{w} \geq 0
\end{aligned}$$

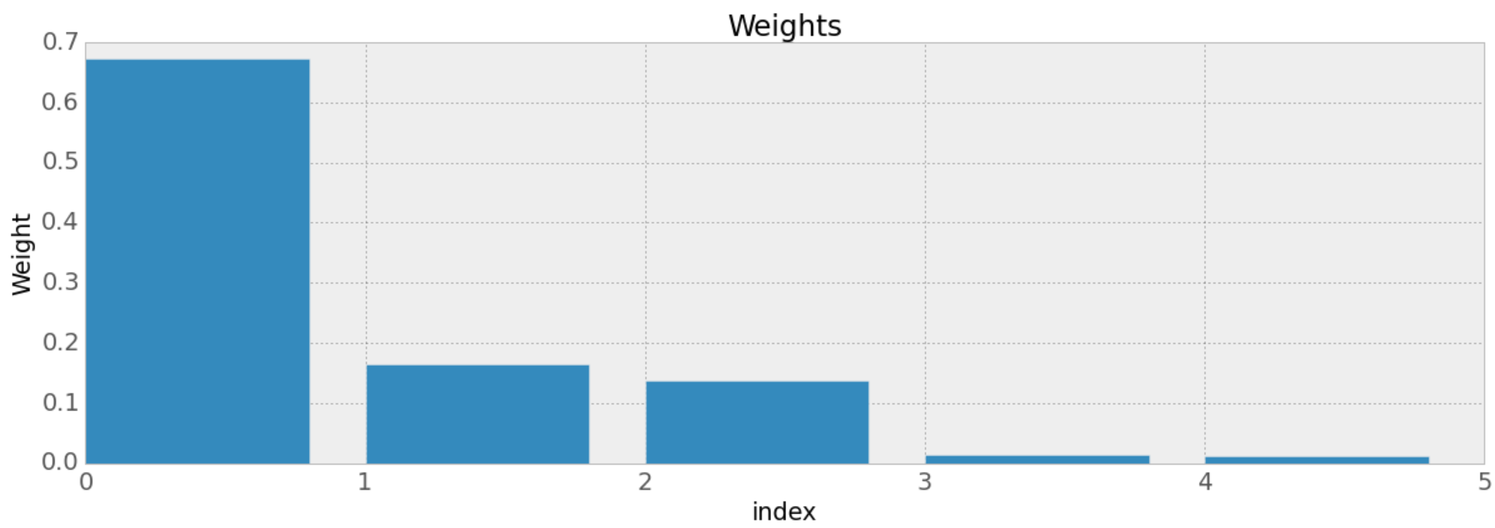
Attention: This is strictly speaking not a Minimum Variance portfolio as we interpret the variance as squared deviations from 0 rather than from the mean.

In [2]:
```python
from mosek.fusion import Expr, Domain, Model, DenseMatrix, ObjectiveSense
import numpy as np

def __two_norm(model, v):
    t = model.variable(1, Domain.greaterThan(0.0))
    model.constraint(Expr.vstack(t, v), Domain.inQCone())
    return t

def min_var(matrix, lamb=0.0):
    """
    min 2-norm (matrix*w) - lamb*2-norm(w)
    s.t. e'w = 1, w >= 0
    """
    # define model
    model = Model('lsqPos')

    # introduce non-negative weight variables
    w = model.variable("w", int(matrix.shape[1]), Domain.inRange(0.0, 1.0))

    # e'*w = 1
    model.constraint(Expr.sum(w), Domain.equalsTo(1.0))

    # introduce the cones
    z = __two_norm(model, Expr.mul(DenseMatrix(matrix), w))
    t = __two_norm(model, w)

    # minimization of the residual
    model.objective(ObjectiveSense.Minimize, Expr.add(z, Expr.mul(t, lamb)))
    model.solve()

    return np.array(w.level())
```

In [5]:

```python
import numpy as np
import matplotlib.pyplot as plt
random_data = np.dot(np.random.randn(250,5), np.diag([1,2,3,4,5]))
data = min_var(random_data)
print(data)

plt.bar(range(0,5),data)
plt.ylabel("Weight"), plt.xlabel("index"), plt.title("Weights")
plt.show()
```

```
[ 0.67308696  0.16379912  0.13756761  0.01321287  0.01233341]
```



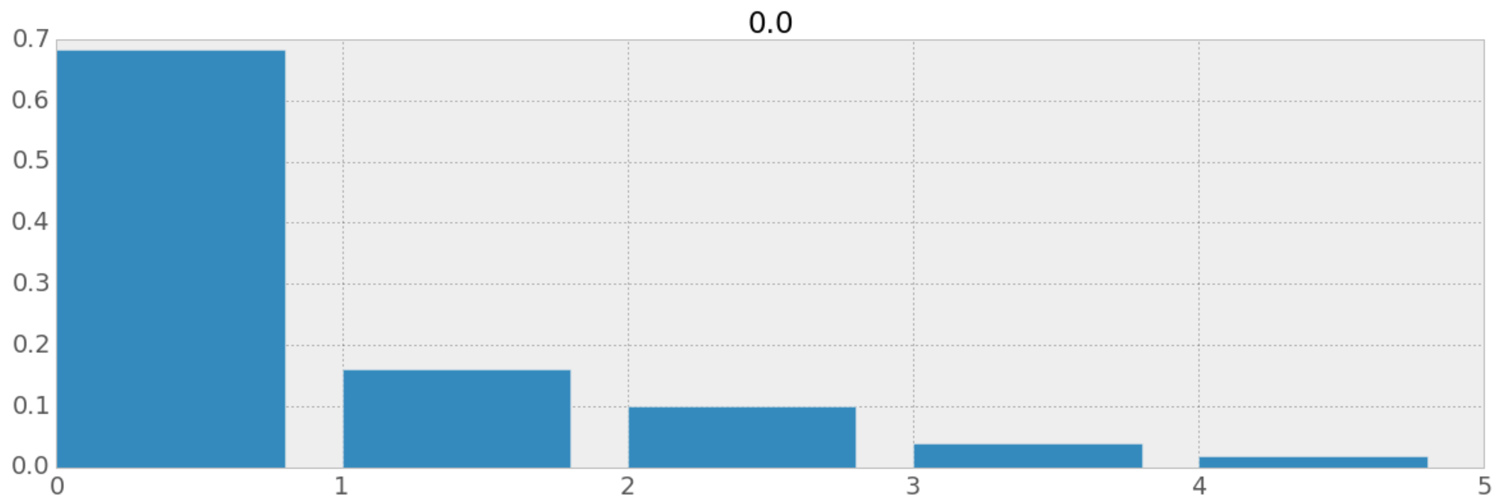
Balance?

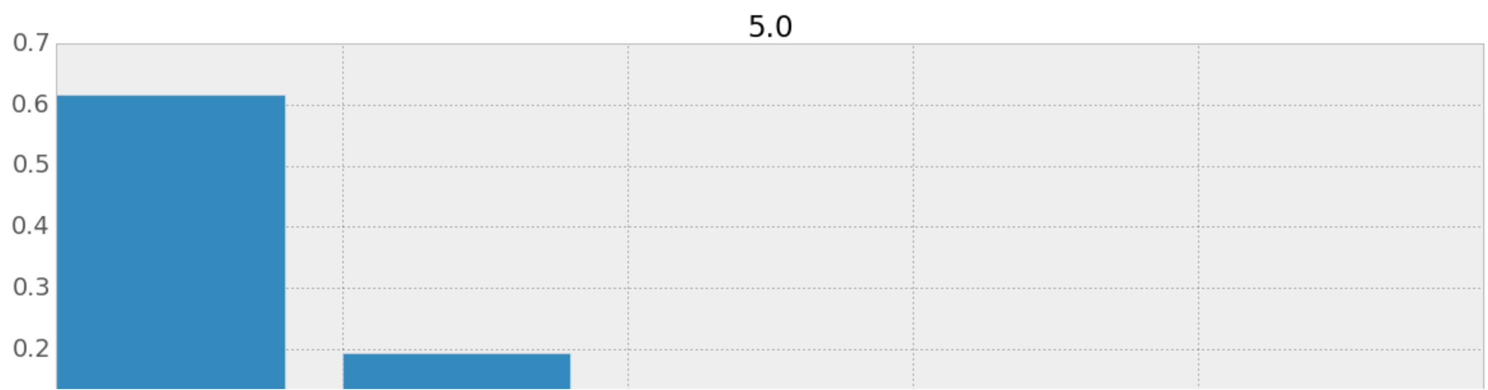
- Bounds
- **Tikhonov regularization** (penalty by the 2-norm of the weights in the objective), also known as **Ridge Regression** or **Shrinkage to the mean**

$$\mathbf{w}^* = \arg\min_{\mathbf{w}\in\mathbb{R}^m} \|\mathbf{R}\mathbf{w}\|_2 + \lambda\|\mathbf{w}\|_2$$
$$\text{s.t.} \; \Sigma\, w_i = 1$$
$$\mathbf{w} \geq 0$$

- The $1/N$ portfolio is the limit for $\lambda \to \infty$

In [4]:
```
import matplotlib.pyplot as plt
for lamb in [0.0, 5.0, 10.0, 20.0, 50.0, 100.0, 200.0]:
    data = min_var(random_data, lamb=lamb)
    plt.bar(range(0,5), data)
    plt.title(lamb)
    plt.show()
```





Summary

- We solve a constrained least squares problem by introducing $n + 1$ additional dimensions (n is the number of rows is the problem).
- We introduce a quadratic cone living in those new dimensions.
- We construct a minimum variance portfolio.
- Using Tikhonov regularization we can interpolate between the Minimum Variance portfolio and the $1/N$ portfolio.