# Using Bash from python (ipython notebook)

In [3]:

```
tif = r"/Users/Kit/DEM_95.tif" # define a python variable
```

### The %%bash cell magic performs several lines of bash code. You can bring python variables in with you and bash variables out.

In [4]:

```
%%bash -s "$tif" --out output
## the -s shows the variables coming into bash
## the -out stores output in a variable called output
proj1=$(gdalinfo $1 | head -7 | tail -1 )
## the $1 means the first of the imported variables; you could have several.
echo $proj1
```

In [5]:

```
#back in python
print("Coord system is " + output[:-2] + "].")
```

```
Coord system is LOCAL_CS["NAD_1983_StatePlane_Connecticut_FIPS_0600_Feet"].
```

### Or, for a quick single line, you can use ! and store output in a textfile.

In [6]:

```
!gdalinfo /Users/Kit/DEM_95.tif > /Users/Kit/Deminfo.txt
```

In [25]:

```
#back in python
name = r"/Users/Kit/Deminfo.txt"
with open(name) as fp:
    for line in fp:
        if line[0:5]=='Coord':
            nextline = next(fp)
            print("Coord system is " + nextline[:-2] + "].")
```

```
Coord system is LOCAL_CS["NAD_1983_StatePlane_Connecticut_FIPS_0600_Feet"].
```

# Using R from python

### You must have the rpy2 module installed (pip install rpy2).

In [2]:

```
import rpy2, pandas ## pandas is nice, but not part of using R.
```

In [3]:

```
# make a small dataframe in python to have some data
firstfile = "/Users/Kit/Desktop/Overlapping/1_height_1st_all.txt"
first = pandas.read_csv(firstfile, header=0, delimiter=" ")
df = pandas.DataFrame(first)
df.columns = ['Xcoord', 'Ycoord', 'Zcoord', 'Angle']
df = df[0:10]
df
```

Out[3]:

| | Xcoord | Ycoord | Zcoord | Angle |
|---|---|---|---|---|
| **0** | 605999.02 | 6600330.48 | 0.01 | 2 |
| **1** | 605993.98 | 6600329.47 | 1.03 | 2 |
| **2** | 605997.64 | 6600330.62 | 0.01 | 2 |
| **3** | 605950.11 | 6600335.34 | -0.01 | 0 |
| **4** | 605975.87 | 6600331.05 | 11.94 | 1 |
| **5** | 605995.84 | 6600329.33 | -0.01 | 2 |
| **6** | 605996.39 | 6600330.72 | 0.01 | 2 |
| **7** | 605996.96 | 6600329.21 | -0.01 | 2 |
| **8** | 605956.74 | 6600333.15 | -0.01 | 1 |
| **9** | 605958.91 | 6600332.96 | 0.26 | 1 |

## Importing variables is similar to %%Bash, but in R you can use their python names.

In [4]:

```
%load_ext rmagic
## only need to load rmagic once
## this block is still speaking python
Y = df.Zcoord
X1 = df.Angle

## send variables to R:
%R -i Y,X1 ## must have no spaces between variables
```

```
The rmagic extension is already loaded. To reload it, use:
  %reload_ext rmagic
```

In [5]:

```
%%R
pretty.lousy.model <- lm(Y ~ X1)
summary(pretty.lousy.model)
```

```

Call:
lm(formula = Y ~ X1)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.1853 -1.4384 -0.7042 -0.6942 10.0002 
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    3.175      2.992   1.061    0.320
X1            -1.236      1.821  -0.679    0.517

Residual standard error: 3.862 on 8 degrees of freedom
Multiple R-squared:  0.05443, Adjusted R-squared:  -0.06377
F-statistic: 0.4605 on 1 and 8 DF,  p-value: 0.5165
```

## %R can run a single line of R code. Needs an extra step, %Rpull, to export R variables back to python.

In [27]:

```
%R coefs = coefficients(pretty.lousy.model)
%Rpull coefs
%R -o coefs
```

In [28]:

```
#back in python
coefs
```

Out[28]:

```
array([ 3.17533333, -1.23555556])
```