```
%load_ext autoreload
%autoreload 2
%matplotlib inline

from google.colab import drive
drive.mount('/content/drive')
```

```
import os, sys

DATAPATH = '/content/drive/My Drive/Coursera/EDHEC/investment-portfolio/data'
print(f"DATAPATH:{DATAPATH} contents:{os.listdir(DATAPATH)}")

MODULEPATH = '/content/drive/My Drive/Coursera/EDHEC/investment-portfolio/nb'
print(f"MODULEPATH:{MODULEPATH} contents:{os.listdir(MODULEPATH)}")

sys.path.append(MODULEPATH)
print(f"sys.path:{sys.path}")

import numpy as np
import pandas as pd

import edhec_risk_kit_108_BBI as erk
```

The Efficient Frontier - Part II

Let's start by loading the returns and generating the expected returns vector and the covariance ma

```
%load_ext autoreload
%autoreload 2
%matplotlib inline
```

```
#import edhec_risk_kit as erk

ind = erk.get_ind_returns(DATAPATH)
er = erk.annualize_rets(ind["1996":"2000"], 12)
cov = ind["1996":"2000"].cov()
```

As a first exercise, let's assume we have some weights, and let's try and compute the returns and v
weights, returns, and a covariance matrix.

The returns are easy, so let's add this to our toolkit

```
def portfolio_return(weights, returns):
    """
    Computes the return on a portfolio from constituent returns and weights
    weights are a numpy array or Nx1 matrix and returns are a numpy array or Nx1 matrix
    """
    return weights.T @ returns
```

The volatility is just as easy in matrix form:

```
def portfolio_vol(weights, covmat):
    """
    Computes the vol of a portfolio from a covariance matrix and constituent weights
    weights are a numpy array or N x 1 maxtrix and covmat is an N x N matrix
    """
    return (weights.T @ covmat @ weights)**0.5
```

```
l = ["Food", "Beer", "Smoke", "Coal"]
```

```
er[l]
```

```
cov.loc[l,l]
```

```
import pandas as pd
import numpy as np
ew = np.repeat(0.25, 4)
erk.portfolio_return(ew, er[l])
```

```
erk.portfolio_vol(ew, cov.loc[l,l])
```

The 2-Asset Case

In the case of 2 assets, the problem is somewhat simplified, since the weight of the second asset i

Let's write a function that draws the efficient frontier for a simple 2 asset case.

Start by generating a sequence of weights in a list of tuples. Python makes it easy to generate a lis
comprehension ... which you can think of as an efficient way to generate a list of values instead of

```
import numpy as np

n_points = 20
```

```
4 weights = [np.array([w, 1-w]) for w in np.linspace(0, 1, n_points)]
5
```

```
1 type(weights)
```

```
1 len(weights)
```

```
1 weights[0]
```

```
1 weights[4]
```

```
1 weights[19]
```

```
1 l = ["Games", "Fin"]
2 rets = [erk.portfolio_return(w, er[l]) for w in weights]
3 vols = [erk.portfolio_vol(w, cov.loc[l,l]) for w in weights]
```

```
ef = pd.DataFrame({"R": rets, "V": vols})
ef.plot.scatter(x="V", y="R", figsize = (12, 6))
```

```
DATAPATH:/content/drive/My Drive/Coursera/EDHEC/investment-portfolio/data cont
MODULEPATH:/content/drive/My Drive/Coursera/EDHEC/investment-portfolio/nb cont
sys.path:['', '/env/python', '/usr/lib/python36.zip', '/usr/lib/python3.6', '/
```

We can create function that plots the frontier:

```
def plot_ef2(n_points, er, cov):
    """
    Plots the 2-asset efficient frontier
    """
    if er.shape[0] != 2 or er.shape[0] != 2:
        raise ValueError("plot_ef2 can only plot 2-asset frontiers")
    weights = [np.array([w, 1-w]) for w in np.linspace(0, 1, n_points)]
    rets = [portfolio_return(w, er) for w in weights]
    vols = [portfolio_vol(w, cov) for w in weights]
    ef = pd.DataFrame({
        "Returns": rets,
        "Volatility": vols
    })
    return ef.plot.line(x="Volatility", y="Returns", style=".-")
```

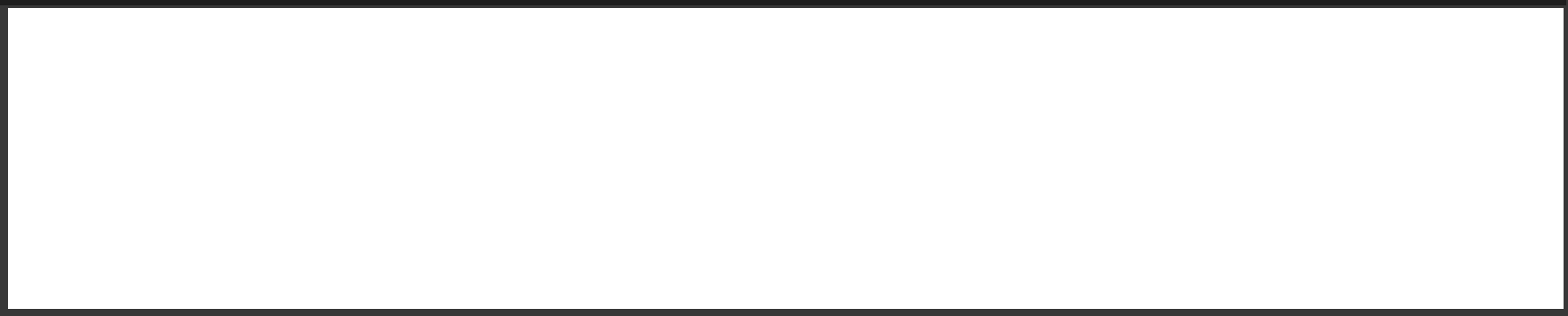
A useful summary of the visualization features in pandas is here

```
1 l = ["Fin", "Beer"]
2 erk.plot_ef2(25, er[l].values, cov.loc[l,l])
```

```
1
```

```
%load_ext autoreload
%autoreload 2
%matplotlib inline

from google.colab import drive
drive.mount('/content/drive')
```



```
import os, sys

DATAPATH = '/content/drive/My Drive/Coursera/EDHEC/investment-portfolio
```

```
print(f"DATAPATH:{DATAPATH} contents:{os.listdir(DATAPATH)}")

MODULEPATH = '/content/drive/My Drive/Coursera/EDHEC/investment-portfol
print(f"MODULEPATH:{MODULEPATH} contents:{os.listdir(MODULEPATH)}")

sys.path.append(MODULEPATH)
print(f"sys.path:{sys.path}")

import numpy as np
import pandas as pd

import edhec_risk_kit_108_BBI as erk
```

The Efficient Frontier - Part II

Let's start by loading the returns and generating the expected returns vector and the covariance matrix

```
%load_ext autoreload
%autoreload 2
%matplotlib inline
#import edhec_risk_kit as erk

ind = erk.get_ind_returns(DATAPATH)
er = erk.annualize_rets(ind["1996":"2000"], 12)
cov = ind["1996":"2000"].cov()
```

```
l = ["Food", "Beer", "Smoke", "Coal"]
```

```
er[l]
```

```
cov.loc[l,l]
```

```
import pandas as pd
import numpy as np
ew = np.repeat(0.25, 4)
erk.portfolio_return(ew, er[l])
```

```
erk.portfolio_vol(ew, cov.loc[l,l])
```


The 2-Asset Case

In the case of 2 assets, the problem is somewhat simplified, since the weight of the second asset is 1-the weight of the first asset.

Let's write a function that draws the efficient frontier for a simple 2 asset case.

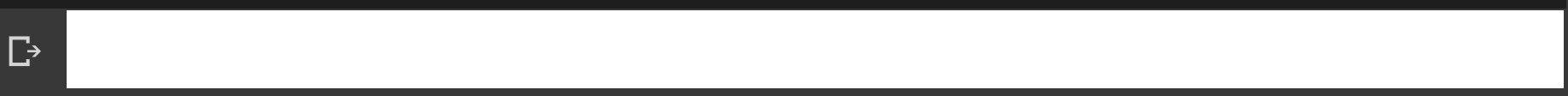
Start by generating a sequence of weights in a list of tuples. Python makes it easy to generate a list by using something called a *list comprehension* ... which you can think of as an efficient way to generate a list of values instead of writing a for loop.

```
import numpy as np

n_points = 20
```

```
4 weights = [np.array([w, 1-w]) for w in np.linspace(0, 1, n_points)]
5
```

```
1 type(weights)
```



```
1 len(weights)
```

```
1 weights[0]
```

```
1 weights[4]
```

```
1 weights[19]
```

```
1 l = ["Games", "Fin"]
2 rets = [erk.portfolio_return(w, er[l]) for w in weights]
3 vols = [erk.portfolio_vol(w, cov.loc[l,l]) for w in weights]
```

```
ef = pd.DataFrame({"R": rets, "V": vols})
ef.plot.scatter(x="V", y="R", figsize = (12, 6))
```

```
l = ["Fin", "Beer"]
erk.plot_ef2(25, er[l].values, cov.loc[l,l])
```

```
1
```