In [1]:
```
import numpy as np
array = np.array([1, 2, 3, 4, 5])
array
```

Out[1]:
```
array([1, 2, 3, 4, 5])
```

In [2]:
```
np.log([1, 2, 3, 4, 5])
```

Out[2]:
```
array([0.        , 0.69314718, 1.09861229, 1.38629436, 1.60943791])
```

In [3]:
```
##Boolean Indexing
```

In [8]:
```
import random

rand_list = [random.random() * 10 for i in range (10)]
rand_list
```

Out[8]:
```
[7.028981291497293,
 7.1106314392123,
 0.31630605256362854,
 0.35024346675673446,
 5.529835385313997,
 0.8112849335571393,
 0.4347228146927351,
 6.762144848589107,
 9.284384987900832,
 4.466964888276025]
```

In [9]:
```
subset = [val for val in rand_list if val > 7]
subset
```

Out[9]:
```
[7.028981291497293, 7.1106314392123, 9.284384987900832]
```

In [ ]:
```

```

Boolean indexing with numpy

In [11]:
```
import numpy as np
rand_array = np.random.randn(7) * 10
print (rand_array)
print (rand_array > 0)
```

```
[ 10.71894661  -9.33946013 -10.3367762   -4.57375779  19.00943841
  -8.92423349 -17.87689006]
[ True False False False  True False False]
```

In [13]:
```
import numpy as np
rand_array = np.random.randn(7) * 10
print (rand_array)
print (rand_array > 0)

rand_array[rand_array > 0]
```

```
[ 11.10529327  -6.47739997 -18.46367499 -20.60471823 -26.59747285
   0.77400425 -17.85028925]
[ True False False False False  True False]
```

Out[13]:
```
array([11.10529327,  0.77400425])
```

Data in Dictionaries & Dataframes

In [14]:
```
data_dict = {"0 to 9" : np.arange(10),
             "ones" : np.ones(10),
             "zeros" : np.zeros(10)}
print(data_dict)
```

```
{'0 to 9': array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]), 'ones': array([1., 1., 1.,
1., 1., 1., 1., 1., 1., 1.]), 'zeros': array([0., 0., 0., 0., 0., 0., 0., 0.,
0., 0.])}
```

In [15]:
```
for key in data_dict:
    print(key)
```

```
0 to 9
ones
zeros
```

In [16]:
```
for key in data_dict:
    print(key)
    print(data_dict[key])
```

```
0 to 9
[0 1 2 3 4 5 6 7 8 9]
ones
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
zeros
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

In [17]:
```
for key in data_dict:
    print(key)
    print(data_dict[key])
data_dict.items()
```

```
0 to 9
[0 1 2 3 4 5 6 7 8 9]
ones
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
zeros
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

Out[17]:
```
dict_items([('0 to 9', array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])), ('ones', array
([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])), ('zeros', array([0., 0., 0., 0.,
0., 0., 0., 0., 0., 0.]))])
```

In [18]:
```
for key in data_dict:
    val = data_dict[key]
    print(key)
    print(val)
#       print(data_dict[key])

data_dict.items()
```

```
0 to 9
[0 1 2 3 4 5 6 7 8 9]
ones
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
zeros
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

Out[18]:
```
dict_items([('0 to 9', array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])), ('ones', array
([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])), ('zeros', array([0., 0., 0., 0.,
0., 0., 0., 0., 0., 0.]))])
```

In [19]:
```
for key, val in data_dict.items():
    print(key)
    print(val)
```

```
0 to 9
[0 1 2 3 4 5 6 7 8 9]
ones
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
zeros
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

In [20]:
```
for key, val in data_dict.items():
    print("Values from index 5", key)
    print(val[5:])
```

```
Values from index 5 0 to 9
[5 6 7 8 9]
Values from index 5 ones
[1. 1. 1. 1. 1.]
Values from index 5 zeros
[0. 0. 0. 0. 0.]
```

Pandas Dataframes

In [24]:
```
import pandas as pd

data = pd.Series(np.arange(2,12))
print(data)
```

```
0     2
1     3
2     4
3     5
4     6
5     7
6     8
7     9
8    10
9    11
dtype: int32
```

In [25]:
```
data_df = pd.DataFrame(data_dict)
data_df
```

Out[25]:

	0 to 9	ones	zeros
0	0	1.0	0.0
1	1	1.0	0.0
2	2	1.0	0.0
3	3	1.0	0.0
4	4	1.0	0.0
5	5	1.0	0.0
6	6	1.0	0.0
7	7	1.0	0.0
8	8	1.0	0.0
9	9	1.0	0.0

In [26]:
```
data_df["0 to 9"]
```

Out[26]:
```
0    0
1    1
2    2
3    3
4    4
5    5
6    6
7    7
8    8
9    9
Name: 0 to 9, dtype: int32
```

In [27]:
```
#df.loc[start:finish - 1] calls according to row name
data_df.loc[:5]
```

Out[27]:

	0 to 9	ones	zeros
0	0	1.0	0.0
1	1	1.0	0.0
2	2	1.0	0.0
3	3	1.0	0.0
4	4	1.0	0.0
5	5	1.0	0.0

In [28]:
```
#df.loc[start:finish - 1] calls according to row name
data_df.loc[:5]
```

Out[28]:

	0 to 9	ones	zeros
0	0	1.0	0.0
1	1	1.0	0.0
2	2	1.0	0.0
3	3	1.0	0.0
4	4	1.0	0.0
5	5	1.0	0.0

Create random data for a Dataframe

In [31]:
```
macro_dict = {"GDP" : {},
              "Money" : {},
              "Real GDP" : {},
              "Price Level" : {}}

for key in macro_dict:
    for i in range(1990, 2010):
        macro_dict[key][i] = np.random.random() * 10000

print(macro_dict)
```

```
{'GDP': {1990: 2271.750691338189, 1991: 9513.60902044082, 1992: 8388.183445583702, 1993: 35.856107211063026, 1994: 8545.438299876958, 1995: 7979.516924474199, 1996: 8422.740962689822, 1997: 4406.402965670648, 1998: 2806.8311800289457, 1999: 4839.358772776513, 2000: 9616.900851288445, 2001: 5006.09741842873, 2002: 7877.946577826013, 2003: 9295.1854283635, 2004: 9108.767172949389, 2005: 4353.690164286862, 2006: 5568.597777146455, 2007: 3815.535455065361, 2008: 5018.339258474879, 2009: 9608.540002680764}, 'Money': {1990: 8825.232320714023, 1991: 2804.6681771952462, 1992: 8198.205960719037, 1993: 8489.796547170612, 1994: 5892.307164214367, 1995: 543.142465722316, 1996: 9465.629604981475, 1997: 3815.836617798003, 1998: 8689.403601419866, 1999: 5888.029931169889, 2000: 6044.509538573879, 2001: 6157.379365257155, 2002: 5230.3224916655745, 2003: 5328.351950616334, 2004: 2307.8695394948513, 2005: 3295.9768732107896, 2006: 3958.2321420353796, 2007: 3011.3338529338794, 2008: 4079.930880593362, 2009: 477.78045978182917}, 'Real GDP': {1990: 9970.727843809884, 1991: 4953.976188730306, 1992: 342.24764606978874, 1993: 7234.991477080936, 1994: 4978.249817673688, 1995: 480.7157375687987, 1996: 6669.3832283764905, 1997: 8772.334713167276, 1998: 2544.610455136138, 1999: 3955.5864635650837, 2000: 2652.749846169843, 2001: 1820.6774468347553, 2002: 8916.766775915474, 2003: 7691.21853325068, 2004: 6897.130344972921, 2005: 7391.169377670905, 2006: 6982.293596094903, 2007: 1322.6236663560665, 2008: 7178.614980528144, 2009: 5910.8590658388885}, 'Price Level': {1990: 3775.014139865611, 1991: 4145.749971649274, 1992: 9906.233514707876, 1993: 3146.7385628262036, 1994: 220.70571134419015, 1995: 6237.1209436877425, 1996: 6892.911296467649, 1997: 6499.889118671621, 1998: 503.61585829035937, 1999: 7826.594521348771, 2000: 8350.059717932972, 2001: 7578.971237837253, 2002: 6310.089322673427, 2003: 4485.580784531886, 2004: 1081.2505257561666, 2005: 6332.280960816961, 2006: 4640.841167182533, 2007: 8363.273366452351, 2008: 6559.1004731980165, 2009: 6690.829980903699}}
```

In [32]:
```
data_df = pd.DataFrame(macro_dict)
data_df
```

Out[32]:

	GDP	Money	Real GDP	Price Level
1990	2271.750691	8825.232321	9970.727844	3775.014140
1991	9513.609020	2804.668177	4953.976189	4145.749972
1992	8388.183446	8198.205961	342.247646	9906.233515
1993	35.856107	8489.796547	7234.991477	3146.738563
1994	8545.438300	5892.307164	4978.249818	220.705711
1995	7979.516924	543.142466	480.715738	6237.120944
1996	8422.740963	9465.629605	6669.383228	6892.911296
1997	4406.402966	3815.836618	8772.334713	6499.889119
1998	2806.831180	8689.403601	2544.610455	503.615858
1999	4839.358773	5888.029931	3955.586464	7826.594521
2000	9616.900851	6044.509539	2652.749846	8350.059718
2001	5006.097418	6157.379365	1820.677447	7578.971238
2002	7877.946578	5230.322492	8916.766776	6310.089323
2003	9295.185428	5328.351951	7691.218533	4485.580785
2004	9108.767173	2307.869539	6897.130345	1081.250526
2005	4353.690164	3295.976873	7391.169378	6332.280961
2006	5568.597777	3958.232142	6982.293596	4640.841167
2007	3815.535455	3011.333853	1322.623666	8363.273366
2008	5018.339258	4079.930881	7178.614981	6559.100473
2009	9608.540003	477.780460	5910.859066	6690.829981

In [37]:
```
macro_df = pd.DataFrame(macro_dict)
macro_df
```

Out[37]:

	GDP	Money	Real GDP	Price Level
1990	2271.750691	8825.232321	9970.727844	3775.014140
1991	9513.609020	2804.668177	4953.976189	4145.749972
1992	8388.183446	8198.205961	342.247646	9906.233515
1993	35.856107	8489.796547	7234.991477	3146.738563
1994	8545.438300	5892.307164	4978.249818	220.705711
1995	7979.516924	543.142466	480.715738	6237.120944
1996	8422.740963	9465.629605	6669.383228	6892.911296
1997	4406.402966	3815.836618	8772.334713	6499.889119
1998	2806.831180	8689.403601	2544.610455	503.615858
1999	4839.358773	5888.029931	3955.586464	7826.594521
2000	9616.900851	6044.509539	2652.749846	8350.059718
2001	5006.097418	6157.379365	1820.677447	7578.971238
2002	7877.946578	5230.322492	8916.766776	6310.089323
2003	9295.185428	5328.351951	7691.218533	4485.580785
2004	9108.767173	2307.869539	6897.130345	1081.250526
2005	4353.690164	3295.976873	7391.169378	6332.280961
2006	5568.597777	3958.232142	6982.293596	4640.841167
2007	3815.535455	3011.333853	1322.623666	8363.273366
2008	5018.339258	4079.930881	7178.614981	6559.100473
2009	9608.540003	477.780460	5910.859066	6690.829981

In [50]:
```
macro_df["Velocity"] = macro_df["GDP"] / macro_df["Money"]
macro_df["Real GDP"] = macro_df["GDP"] / macro_df["Price Level"]
print(macro_df)
```

```
              GDP        Money   Real GDP  Price Level   Velocity
1990  2271.750691  8825.232321   0.601786  3775.014140   0.257415
1991  9513.609020  2804.668177   2.294786  4145.749972   3.392062
1992  8388.183446  8198.205961   0.846758  9906.233515   1.023173
1993    35.856107  8489.796547   0.011395  3146.738563   0.004223
1994  8545.438300  5892.307164  38.718700   220.705711   1.450270
1995  7979.516924   543.142466   1.279359  6237.120944  14.691388
1996  8422.740963  9465.629605   1.221942  6892.911296   0.889824
1997  4406.402966  3815.836618   0.677920  6499.889119   1.154767
1998  2806.831180  8689.403601   5.573357   503.615858   0.323018
1999  4839.358773  5888.029931   0.618322  7826.594521   0.821898
2000  9616.900851  6044.509539   1.151716  8350.059718   1.591014
2001  5006.097418  6157.379365   0.660525  7578.971238   0.813024
2002  7877.946578  5230.322492   1.248468  6310.089323   1.506207
2003  9295.185428  5328.351951   2.072237  4485.580785   1.744477
2004  9108.767173  2307.869539   8.424289  1081.250526   3.946829
2005  4353.690164  3295.976873   0.687539  6332.280961   1.320910
2006  5568.597777  3958.232142   1.199911  4640.841167   1.406840
2007  3815.535455  3011.333853   0.456225  8363.273366   1.267058
2008  5018.339258  4079.930881   0.765096  6559.100473   1.230006
2009  9608.540003   477.780460   1.436076  6690.829981  20.110785
```

In [44]:
```
macro_df.loc[2002:2009]
```

Out[44]:

	GDP	Money	Real GDP	Price Level	Velocity
2002	7877.946578	5230.322492	8916.766776	6310.089323	1.506207
2003	9295.185428	5328.351951	7691.218533	4485.580785	1.744477
2004	9108.767173	2307.869539	6897.130345	1081.250526	3.946829
2005	4353.690164	3295.976873	7391.169378	6332.280961	1.320910
2006	5568.597777	3958.232142	6982.293596	4640.841167	1.406840
2007	3815.535455	3011.333853	1322.623666	8363.273366	1.267058
2008	5018.339258	4079.930881	7178.614981	6559.100473	1.230006
2009	9608.540003	477.780460	5910.859066	6690.829981	20.110785

In [49]:
```
macro_df["Price Level"].loc[1998:2004]
```

Out[49]:
```
1998     503.615858
1999    7826.594521
2000    8350.059718
2001    7578.971238
2002    6310.089323
2003    4485.580785
2004    1081.250526
Name: Price Level, dtype: float64
```

In [51]:
```
macro_df["Price Level"].iloc[1998:2004]
```

Out[51]:
```
Series([], Name: Price Level, dtype: float64)
```

In [52]:
```
macro_df.loc[1998:2004, ["Real GDP", "Money"]]
```

Out[52]:

	Real GDP	Money
1998	5.573357	8689.403601
1999	0.618322	5888.029931
2000	1.151716	6044.509539
2001	0.660525	6157.379365
2002	1.248468	5230.322492
2003	2.072237	5328.351951
2004	8.424289	2307.869539

In [54]:
```
macro_df[["Real GDP", "Money"]]
```

Out[54]:

	Real GDP	Money
1990	0.601786	8825.232321
1991	2.294786	2804.668177
1992	0.846758	8198.205961
1993	0.011395	8489.796547
1994	38.718700	5892.307164
1995	1.279359	543.142466
1996	1.221942	9465.629605
1997	0.677920	3815.836618
1998	5.573357	8689.403601
1999	0.618322	5888.029931
2000	1.151716	6044.509539
2001	0.660525	6157.379365
2002	1.248468	5230.322492
2003	2.072237	5328.351951
2004	8.424289	2307.869539
2005	0.687539	3295.976873
2006	1.199911	3958.232142
2007	0.456225	3011.333853
2008	0.765096	4079.930881
2009	1.436076	477.780460

In [57]:
```
import matplotlib.pyplot as plt
%matpotlib inline
macro_df.plot.line()
```

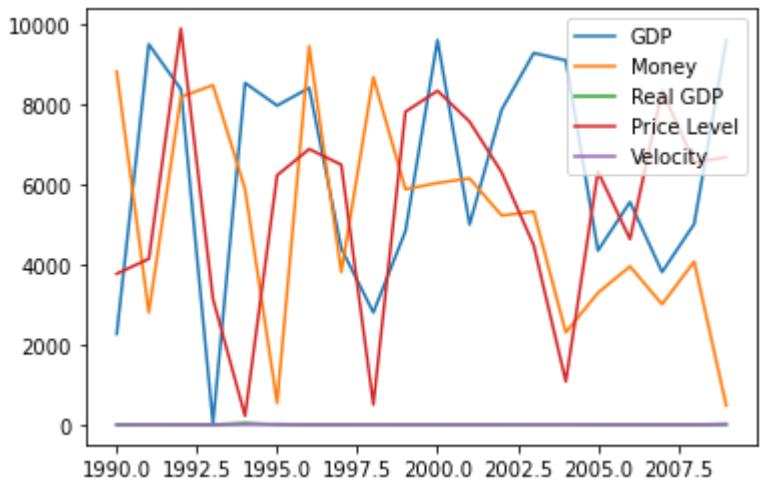
```
UsageError: Line magic function `%matpotlib` not found.
```

In [59]:
```
for key in macro_df:
    macro_df(key):plot.line()
    plt.show()
    plt.close()
```

```
  File "<ipython-input-59-920dab183c42>", line 2
    macro_df(key):plot.line()
    ^
SyntaxError: illegal target for annotation
```

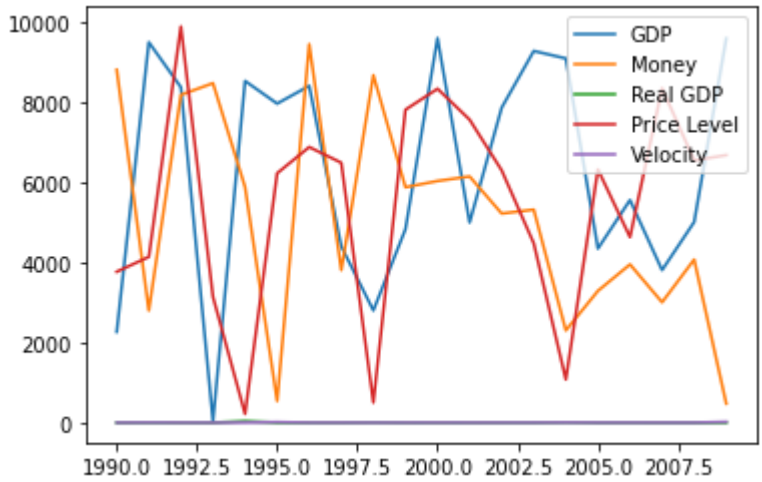
In [61]:
```
import matplotlib.pyplot as plt
%matplotlib inline
macro_df.plot.line(legend = "best")
```

Out[61]: <AxesSubplot:>

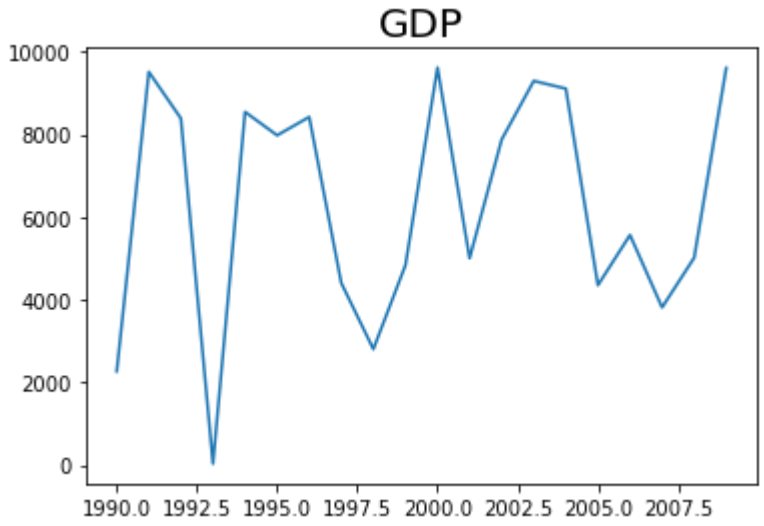


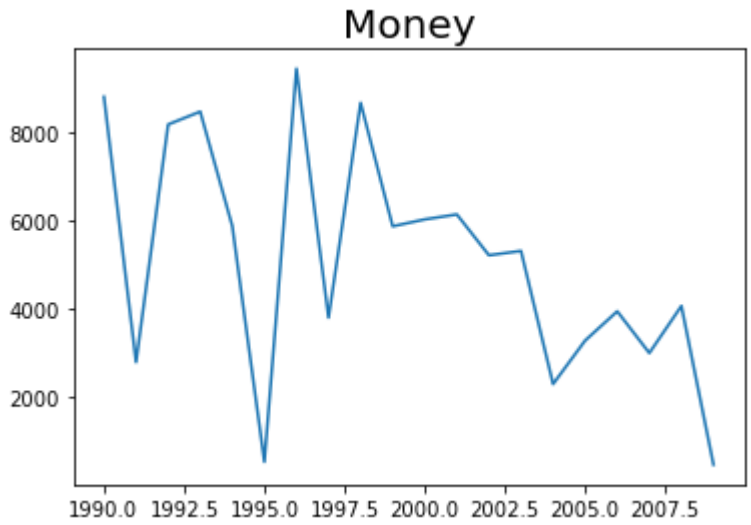
In [62]:
```
import matplotlib.pyplot as plt
%matplotlib inline
macro_df.plot.line()
```

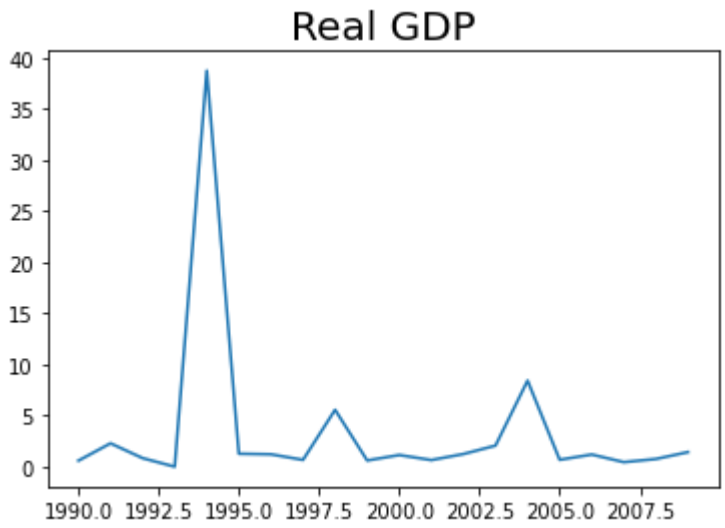
Out[62]: <AxesSubplot:>

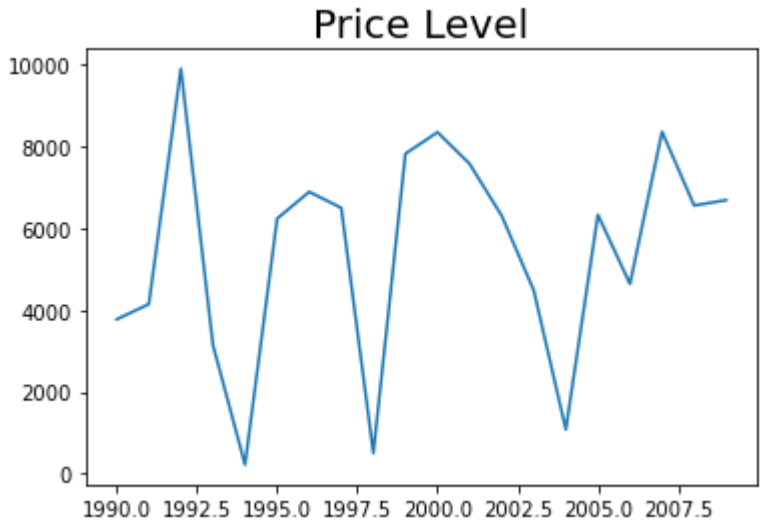


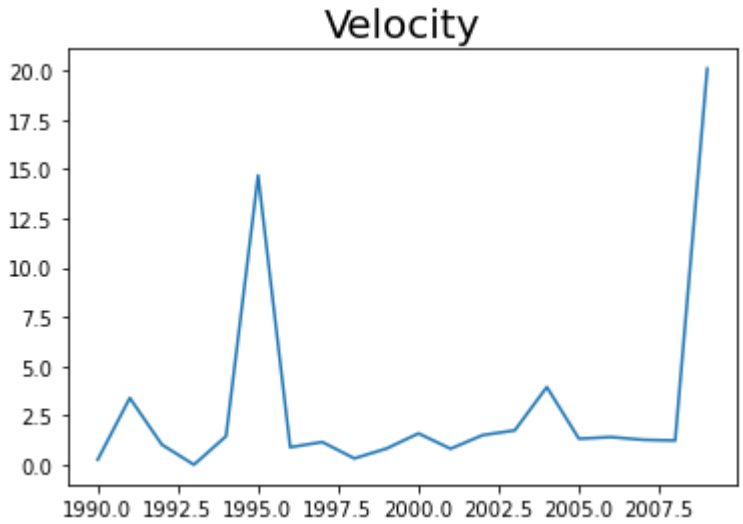
In [64]:
```
for key in macro_df:
    macro_df[key].plot.line()
    plt.title(key, fontsize = 20)
    plt.show()
    plt.close()
```







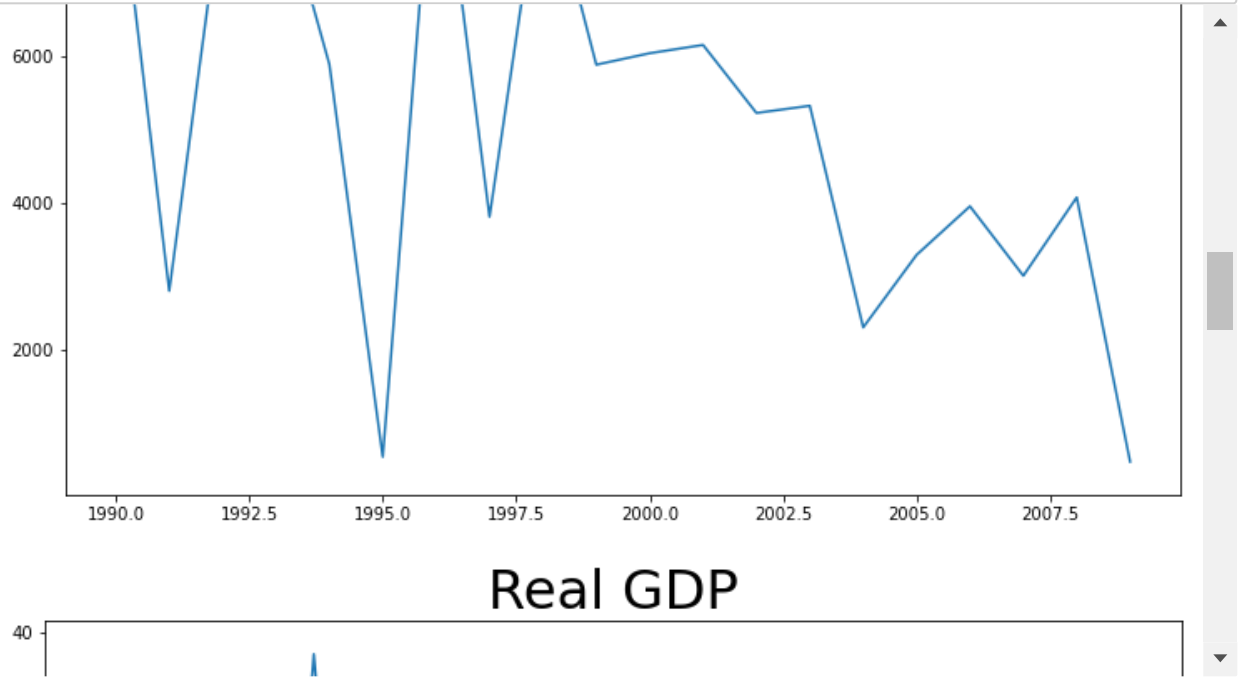

Price Level
10000
8000
6000
4000
2000
0
1990.0
1992.5
1995.0
1997.5
2000.0
2002.5
2005.0
2007.5


Velocity
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
1990.0
1992.5
1995.0
1997.5
2000.0
2002.5
2005.0
2007.5

In [67]:
```
for key in macro_df:
    # fig, ax = plt.subplots() allows you to adjust different features of the plo
    fig, ax = plt.subplots(figsize = (12,8))
    macro_df[key].plot.line(ax = ax)

    # we can make the display a scatter plot if we want
    macro_df[key].plot.line(ls = "",marker = "", ax = ax)

    plt.title(key, fontsize = 32)
    plt.show()
    plt.close()
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: