代码执行之后，cell 左侧的标签从 In [ ] 变成了 In [1]。In 代表输入，[]中的数字代表 kernel 执行的顺序，而 In [*]则表示代码 cell 正在执行代码。

In [ ]:
```
print("hello world!")
```
```
hello world!
```

cell 模式 有两种模式，编辑模式（edit mode）和命名模式（command mode）
编辑模式：enter 健切换，绿色轮廓 命令模式：esc 健切换，蓝色轮廓

In [ ]:
```
import time
time.sleep(3)
```

In [ ]:
```
import numpy as np
def square(x):
    return x * x
```

In [ ]:
```
x = np.random.randint(1, 10)
y = square(x)
print('%d squared is %d' % (x, y))
```
```
4 squared is 16
```

实现简单的 python 程序¶

完成基于 python 的选择排序算法 1.定义 selection_sort 函数执行选择排序功能。 2.定义 test 函数进行测试，执行数据输入，并调用 selection_sort 函数进行排序，最后输出结果。
导入相关的工具库

In [ ]:
```
%matplotlib inline
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

加载数据集

In [ ]:
```
df = pd.read_csv('fortune500.csv')
```

In [ ]:
```
df.head()
```

对数据属性列进行重命名，以便在后续访问

In [ ]:
```
df.columns = ['year', 'rank', 'company', 'revenue', 'profit']
```

检查数据条目是否加载完整

In [ ]:
```
len(df)
```

Out[ ]:
```
25500
```

检查属性列的类型

In [ ]:
```
df.dtypes
```

Out[ ]:
```
year         int64
rank         int64
company     object
revenue    float64
profit      object
dtype: object
```

其他属性列都正常，但是对于 profit 属性，期望的结果是 float 类型，因此其可能包含非数字的值，利用正则表达式进行检查

In [ ]:
```
non_numberic_profits = df.profit.str.contains('[^0-9.-]')
```

```
df.loc[non_numberic_profits].head()
```

Out[ ]:

	year	rank	company	revenue	profit
228	1955	229	Norton	135.0	N.A.
290	1955	291	Schlitz Brewing	100.0	N.A.
294	1955	295	Pacific Vegetable Oil	97.9	N.A.
296	1955	297	Liebmann Breweries	96.0	N.A.
352	1955	353	Minneapolis-Moline	77.4	N.A.

确实存在这样的记录，profit 这一列为字符串，统计一下到底存在多少条这样的记录。

In [ ]:

```
len(df.profit[non_numberic_profits])
```

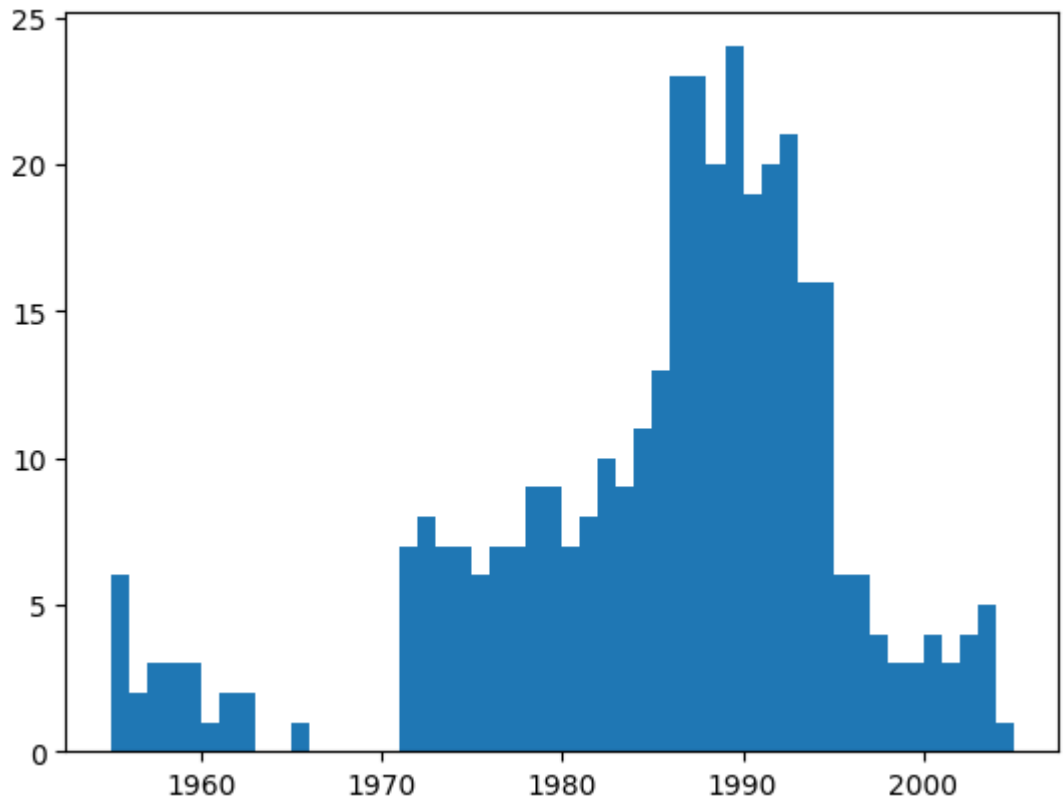
Out[ ]:

```
369
```

使用直方图显示一下按照年份的分布情况

In [ ]:

```
bin_sizes, _, _ = plt.hist(df.year[non_numberic_profits], bins=range(1955, 2006))
```



可见，单独年份这样的记录数都少于 25 条，即少于 4%的比例。这在可以接受的范围内，因此删除这些记录。

In [ ]:

```
df = df.loc[~non_numberic_profits]
df.profit = df.profit.apply(pd.to_numeric)
```

再次检查数据记录的条目数。

In [ ]:

```
len(df)
df.dtypes
```

Out[ ]:

```
year         int64
rank         int64
company     object
revenue    float64
profit     float64
dtype: object
```

使用 matplotlib 进行绘图¶

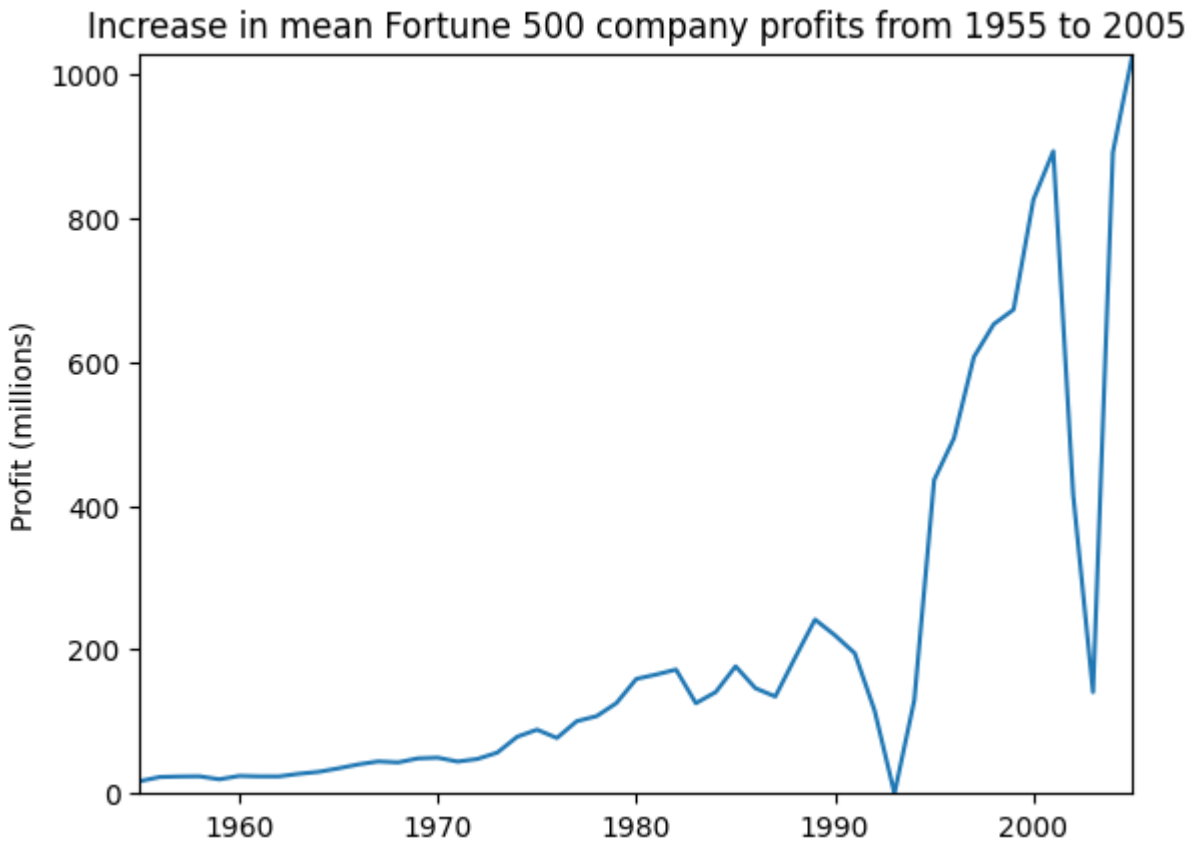
以年分组绘制平均利润和收入。首先定义变量和方法。

In [ ]:

```
group_by_year = df.loc[:, ['year', 'revenue', 'profit']].groupby('year')
avgs = group_by_year.mean()
x = avgs.index
y1 = avgs.profit
def plot(x, y, ax, title, y_label):
    ax.set_title(title)
    ax.set_ylabel(y_label)
    ax.plot(x, y)
    ax.margins(x=0, y=0)
```

In [ ]:

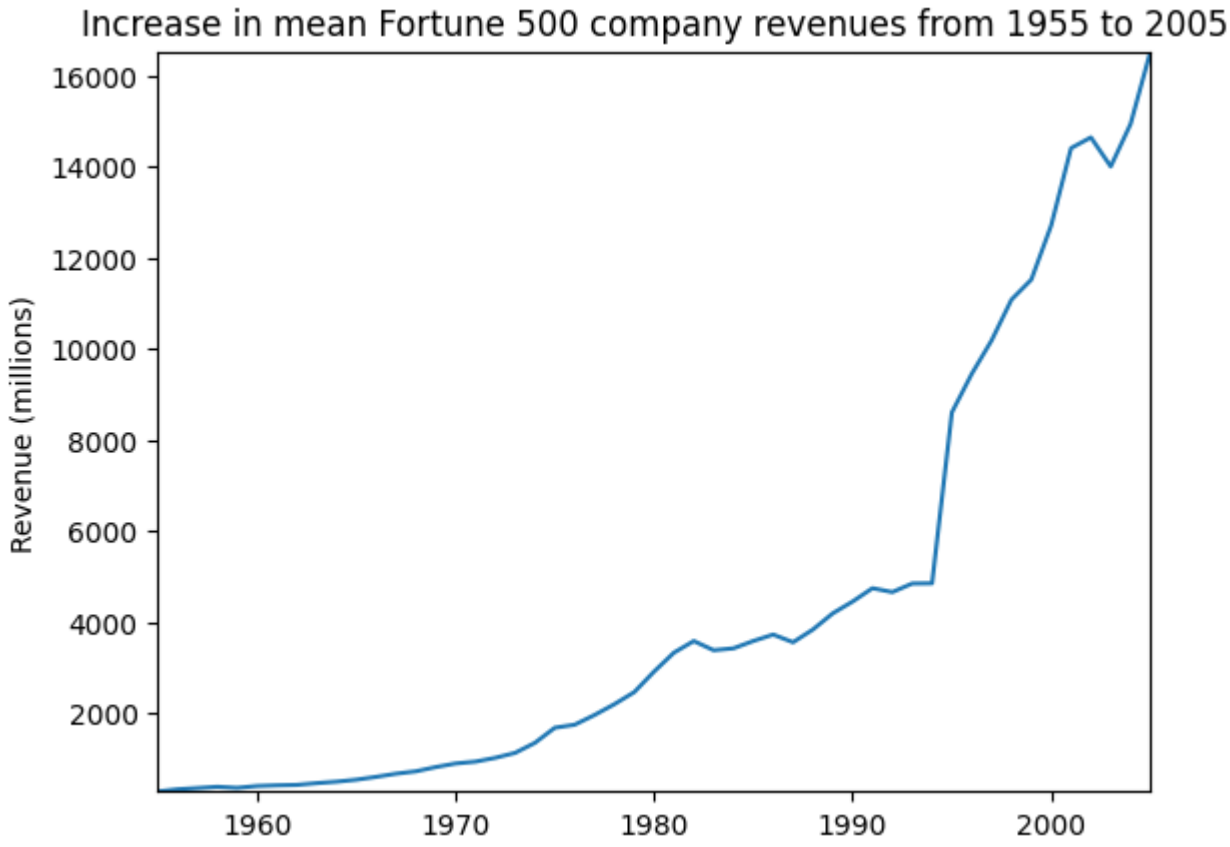
```
fig, ax = plt.subplots()
plot(x, y1, ax, 'Increase in mean Fortune 500 company profits from 1955 to 2005',
'Profit (millions)')
```



收入曲线

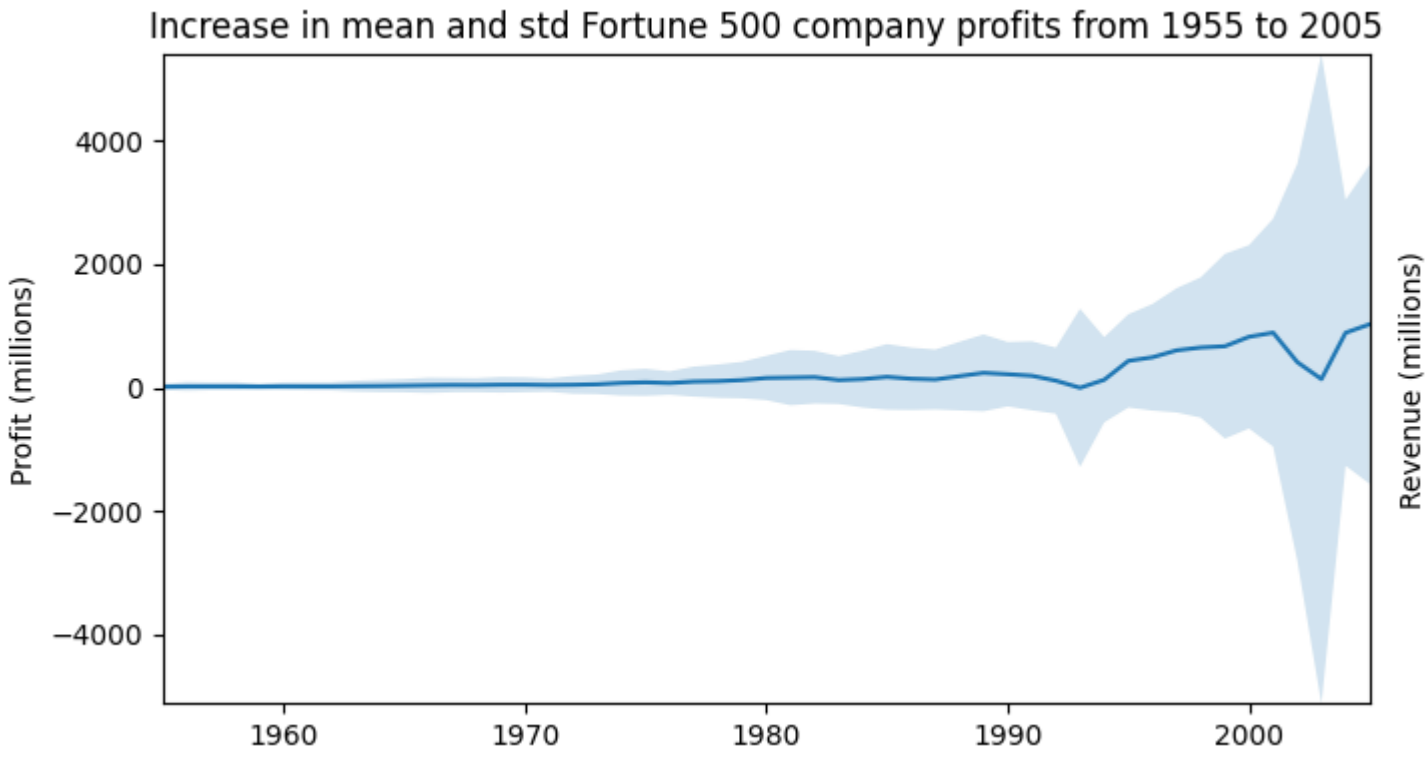
In [ ]:

```
y2 = avgs.revenue
fig, ax = plt.subplots()
plot(x, y2, ax, 'Increase in mean Fortune 500 company revenues from 1955 to 2005',
'Revenue (millions)')
```



公司收入曲线并没有出现急剧下降，可能是由于财务会计的处理。对数据结果进行标准差处理。
In [ ]:

```
def plot_with_std(x, y, stds, ax, title, y_label):
    ax.fill_between(x, y - stds, y + stds, alpha=0.2)
    plot(x, y, ax, title, y_label)
fig, (ax1, ax2) = plt.subplots(ncols=2)
title = 'Increase in mean and std Fortune 500 company %s from 1955 to 2005'
stds1 = group_by_year.std().profit.values
stds2 = group_by_year.std().revenue.values
plot_with_std(x, y1.values, stds1, ax1, title % 'profits', 'Profit (millions)')
plot_with_std(x, y2.values, stds2, ax2, title % 'revenues', 'Revenue (millions)')
fig.set_size_inches(14, 4)
fig.tight_layout()
```


Increase in mean and std Fortune 500 company profits from 1955 to 2005
Profit (millions)
Revenue (millions)
4000
2000
0
−2000
−4000
1960
1970
1980
1990
2000