#Python可视化

0. 数据可视化简介

在 Python 中，通常使用第三方库来进行可视化。常用的可视化库有 Matplotlib、Seaborn 和 Plotly 等。

Matplotlib 是 Python 中最古老、最流行的可视化库，提供了丰富的图表类型和高度自定义的绘图工具。使用 Matplotlib 可以生成静态图表，也可以使用其中的 pyplot 模块在交互式环境中绘图。

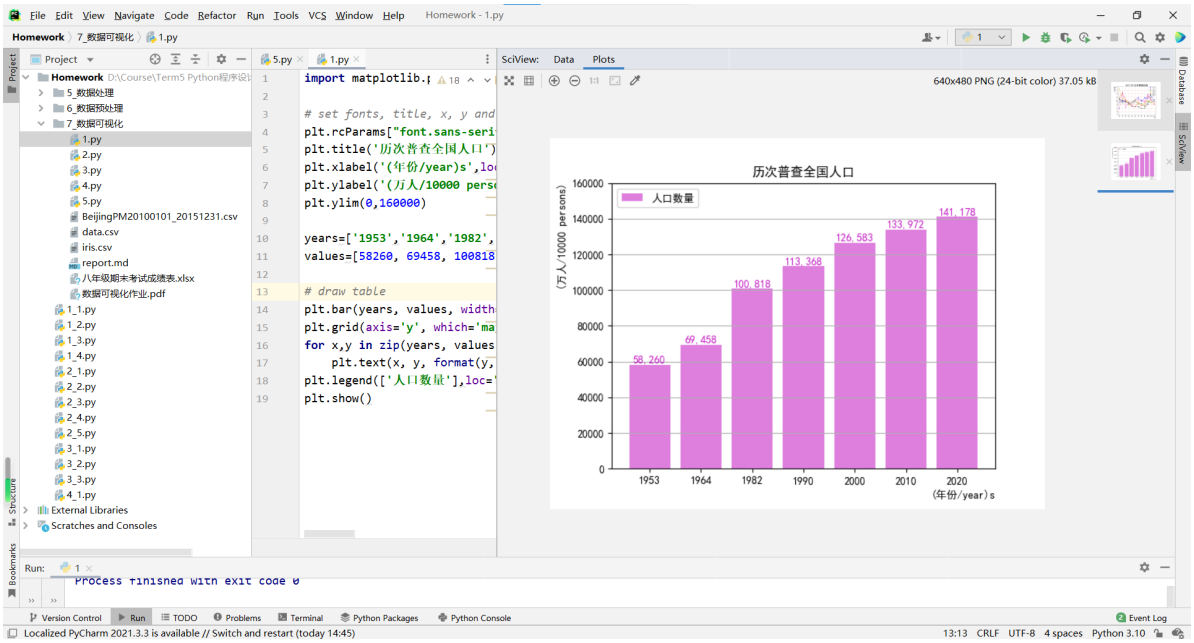
Seaborn 是基于 Matplotlib 的可视化库，提供了更简单、美观的 API，并自带许多实用的统计图表，如联合分布图、核密度图等。Seaborn 可以很方便地展示数据的分布情况和关系，是进行数据分析和可视化的首选库。

Plotly 是一个基于 Web 的可视化库，可以生成交互式、动态图表。Plotly 可以在线生成图表，也可以在本地使用其 Python API 进行绘图。Plotly 还提供了多种用于创建图表的布局和样式的选项，可以让图表更具吸引力。

在使用上，通常需要先导入所需的库，然后使用相应的函数或方法绘制图表。比如，在 Matplotlib 中，可以使用 `plot` 函数绘制折线图，使用 `scatter` 函数绘制散点图，使用 `bar` 函数绘制柱状图。

1. 绘制历次人口普查全国人口数量柱状图

这一题考察的是柱状图的画图方法。



上图是可视化效果。代码和说明如下：

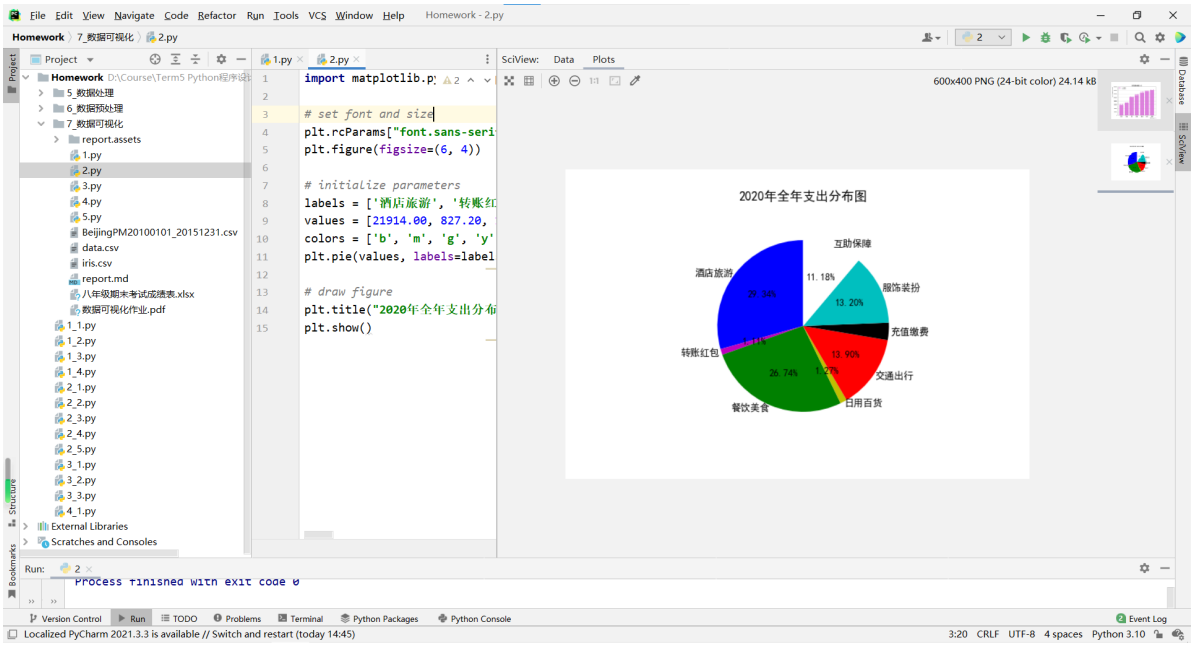
```
import matplotlib.pyplot as plt

# set fonts, title, x, y and y-limit
plt.rcParams["font.sans-serif"]=["SimHei"]
plt.title('历次普查全国人口')
```

```
plt.xlabel('(年份/year)s',loc='right')
plt.ylabel('(万人/10000 persons)',loc='top')
plt.ylim(0,160000)

years=['1953','1964','1982','1990','2000','2010','2020']
values=[58260, 69458, 100818, 113368, 126583, 133972, 141178]

# draw table
plt.bar(years, values, width=0.8,color='m',alpha=0.5,bottom=0.8)
plt.grid(axis='y', which='major')
for x,y in zip(years, values):
    plt.text(x, y, format(y, ','), ha='center', va='bottom', fontsize=10,
color='m', alpha=0.9)
plt.legend(['人口数量'],loc='upper left')
plt.show()
```

这里我们先进行图标参数的构造。主要有font, title, xlable, ylable, y-limit；在这之后，我们将图表中的数据写入，获取数据；最后，我们给出画图的适当参数，再进行plot。

2. 绘制某人2020年支付宝年展示各类型支出占总支出比例的支出情况饼图

这一题考察的是饼图的绘制。



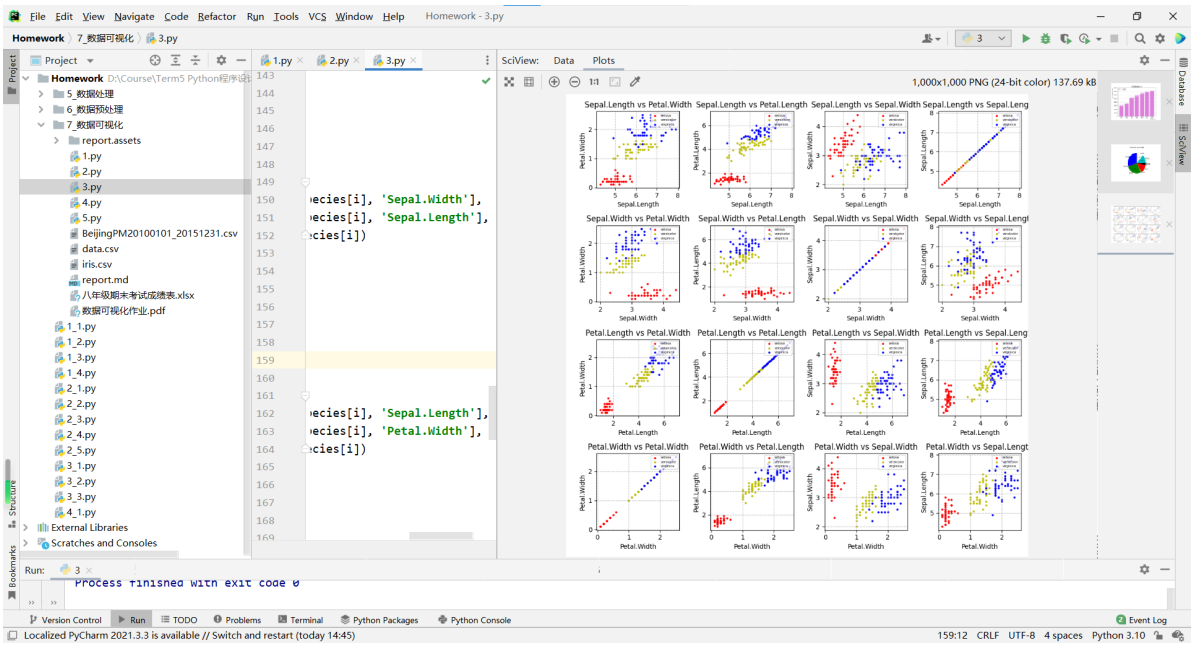
上图是可视化效果。代码如下：

```
import matplotlib.pyplot as plt

# set font and size
plt.rcParams["font.sans-serif"] = ["SimHei"]
plt.figure(figsize=(6, 4))

# initialize parameters
labels = ['酒店旅游', '转账红包', '餐饮美食', '日用百货', '交通出行', '充值缴费', '服饰装扮', '互助保障']
```

```
values = [21914.00, 827.20, 19973.20, 950.83, 10379.59, 2428.54, 9859.93,
8351.35]
colors = ['b', 'm', 'g', 'y', 'r', 'k', 'c', 'w']
plt.pie(values, labels=labels, colors=colors, labeldistance=1.02,
autopct='%.2f%%', startangle=90, radius=0.9, center=(0.2, 0.2), textprops=
{'fontsize': 9, 'color': 'k'})

# draw figure
plt.title("2020年全年支出分布图")
plt.show()
```

3. 使用四种数据，两两组合绘制散点图

这一题考察的是散点图和子图。



上图是可视化效果。代码和说明如下：

```
import matplotlib.pyplot as plt
import pandas as pd

# get file
iris = pd.read_csv('iris.csv')


# initialize figure parameters
colors = ['r', 'y', 'b']
species = iris.Species.unique()
plt.figure(figsize=(10, 10))


# sub figures
# figure 1
plt.subplot(4, 4, 13)
for i in range(len(species)):
    plt.scatter(iris.loc[iris.Species == species[i], 'Petal.Width'],
                iris.loc[iris.Species == species[i], 'Petal.Width'],
```

```
                s=5, c=colors[i], label=species[i])
plt.title("Petal.Width vs Petal.Width")
plt.xlabel('Petal.Width')
plt.ylabel('Petal.Width')
plt.grid(True, linestyle='--', alpha=0.8)
plt.legend(loc='upper right', fontsize=5)

# figure 2
plt.subplot(4, 4, 14)
for i in range(len(species)):
    plt.scatter(iris.loc[iris.Species == species[i], 'Petal.Width'],
                iris.loc[iris.Species == species[i], 'Petal.Length'],
                s=5, c=colors[i], label=species[i])
plt.title("Petal.Width vs Petal.Length")
plt.xlabel('Petal.Width')
plt.ylabel('Petal.Length')
plt.grid(True, linestyle='--', alpha=0.8)
plt.legend(loc='upper right', fontsize=5)

# figure 3
plt.subplot(4, 4, 15)
for i in range(len(species)):
    plt.scatter(iris.loc[iris.Species == species[i], 'Petal.Width'],
                iris.loc[iris.Species == species[i], 'Sepal.Width'],
                s=5, c=colors[i], label=species[i])
plt.title("Petal.Width vs Sepal.Width")
plt.xlabel('Petal.Width')
plt.ylabel('Sepal.Width')
plt.grid(True, linestyle='--', alpha=0.8)
plt.legend(loc='upper right', fontsize=5)

# figure 4
plt.subplot(4, 4, 16)
for i in range(len(species)):
    plt.scatter(iris.loc[iris.Species == species[i], 'Petal.Width'],
                iris.loc[iris.Species == species[i], 'Sepal.Length'],
                s=5, c=colors[i], label=species[i])
plt.title("Petal.Width vs Sepal.Length")
plt.xlabel('Petal.Width')
plt.ylabel('Sepal.Length')
plt.grid(True, linestyle='--', alpha=0.8)
plt.legend(loc='upper right', fontsize=5)

# figure 5
plt.subplot(4, 4, 9)
for i in range(len(species)):
    plt.scatter(iris.loc[iris.Species == species[i], 'Petal.Length'],
                iris.loc[iris.Species == species[i], 'Petal.Width'],
                s=5, c=colors[i], label=species[i])
plt.title("Petal.Length vs Petal.Width")
plt.xlabel('Petal.Length')
plt.ylabel('Petal.Width')
plt.grid(True, linestyle='--', alpha=0.8)
plt.legend(loc='upper right', fontsize=5)
```

```
# figure 6
plt.subplot(4, 4, 10)
for i in range(len(species)):
    plt.scatter(iris.loc[iris.Species == species[i], 'Petal.Length'],
                iris.loc[iris.Species == species[i], 'Petal.Length'],
                s=5, c=colors[i], label=species[i])
plt.title("Petal.Length vs Petal.Length")
plt.xlabel('Petal.Length')
plt.ylabel('Petal.Length')
plt.grid(True, linestyle='--', alpha=0.8)
plt.legend(loc='upper right', fontsize=5)

# figure 7
plt.subplot(4, 4, 11)
for i in range(len(species)):
    plt.scatter(iris.loc[iris.Species == species[i], 'Petal.Length'],
                iris.loc[iris.Species == species[i], 'Sepal.Width'],
                s=5, c=colors[i], label=species[i])
plt.title("Petal.Length vs Sepal.Width")
plt.xlabel('Petal.Length')
plt.ylabel('Sepal.Width')
plt.grid(True, linestyle='--', alpha=0.8)
plt.legend(loc='upper right', fontsize=5)

# figure 8
plt.subplot(4, 4, 12)
for i in range(len(species)):
    plt.scatter(iris.loc[iris.Species == species[i], 'Petal.Length'],
                iris.loc[iris.Species == species[i], 'Sepal.Length'],
                s=5, c=colors[i], label=species[i])
plt.title("Petal.Length vs Sepal.Length")
plt.xlabel('Petal.Length')
plt.ylabel('Sepal.Length')
plt.grid(True, linestyle='--', alpha=0.8)
plt.legend(loc='upper right', fontsize=5)

# figure 9
plt.subplot(4, 4, 5)
for i in range(len(species)):
    plt.scatter(iris.loc[iris.Species == species[i], 'Sepal.Width'],
                iris.loc[iris.Species == species[i], 'Petal.Width'],
                s=5, c=colors[i], label=species[i])
plt.title("Sepal.Width vs Petal.Width")
plt.xlabel('Sepal.Width')
plt.ylabel('Petal.Width')
plt.grid(True, linestyle='--', alpha=0.8)
plt.legend(loc='upper right', fontsize=5)

# figure 10
plt.subplot(4, 4, 6)
for i in range(len(species)):
    plt.scatter(iris.loc[iris.Species == species[i], 'Sepal.Width'],
                iris.loc[iris.Species == species[i], 'Petal.Length'],
                s=5, c=colors[i], label=species[i])
plt.title("Sepal.Width vs Petal.Length")
```

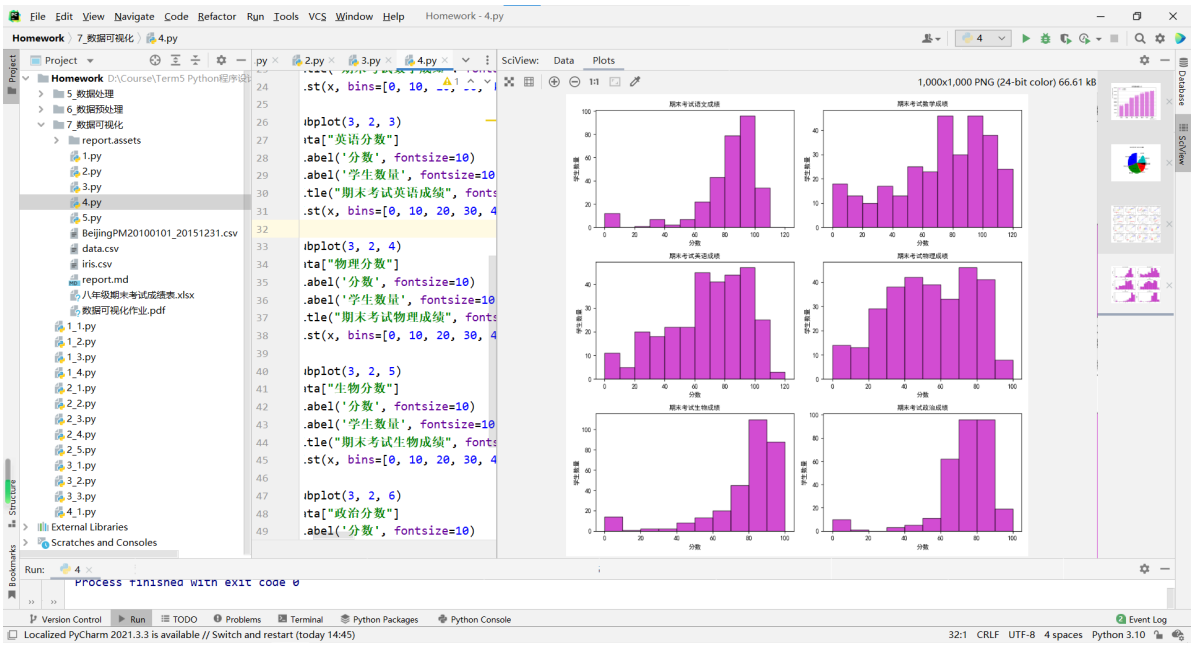
```
plt.xlabel('Sepal.Width')
plt.ylabel('Petal.Length')
plt.grid(True, linestyle='--', alpha=0.8)
plt.legend(loc='upper right', fontsize=5)

# figure 11
plt.subplot(4, 4, 7)
for i in range(len(species)):
    plt.scatter(iris.loc[iris.Species == species[i], 'Sepal.Width'],
                iris.loc[iris.Species == species[i], 'Sepal.Width'],
                s=5, c=colors[i], label=species[i])
plt.title("Sepal.Width vs Sepal.Width")
plt.xlabel('Sepal.Width')
plt.ylabel('Sepal.Width')
plt.grid(True, linestyle='--', alpha=0.8)
plt.legend(loc='upper right', fontsize=5)

# figure 12
plt.subplot(4, 4, 8)
for i in range(len(species)):
    plt.scatter(iris.loc[iris.Species == species[i], 'Sepal.Width'],
                iris.loc[iris.Species == species[i], 'Sepal.Length'],
                s=5, c=colors[i], label=species[i])
plt.title("Sepal.Width vs Sepal.Length")
plt.xlabel('Sepal.Width')
plt.ylabel('Sepal.Length')
plt.grid(True, linestyle='--', alpha=0.8)
plt.legend(loc='upper right', fontsize=5)

# figure 13
plt.subplot(4, 4, 1)
for i in range(len(species)):
    plt.scatter(iris.loc[iris.Species == species[i], 'Sepal.Length'],
                iris.loc[iris.Species == species[i], 'Petal.Width'],
                s=5, c=colors[i], label=species[i])
plt.title("Sepal.Length vs Petal.Width")
plt.xlabel('Sepal.Length')
plt.ylabel('Petal.Width')
plt.grid(True, linestyle='--', alpha=0.8)
plt.legend(loc='upper right', fontsize=5)

# figure 14
plt.subplot(4, 4, 2)
for i in range(len(species)):
    plt.scatter(iris.loc[iris.Species == species[i], 'Sepal.Length'],
                iris.loc[iris.Species == species[i], 'Petal.Length'],
                s=5, c=colors[i], label=species[i])
plt.title("Sepal.Length vs Petal.Length")
plt.xlabel('Sepal.Length')
plt.ylabel('Petal.Length')
plt.grid(True, linestyle='--', alpha=0.8)
plt.legend(loc='upper right', fontsize=5)

# figure 15
plt.subplot(4, 4, 3)
```

```
for i in range(len(species)):
    plt.scatter(iris.loc[iris.Species == species[i], 'Sepal.Length'],
                iris.loc[iris.Species == species[i], 'Sepal.Width'],
                s=5, c=colors[i], label=species[i])
plt.title("Sepal.Length vs Sepal.Width")
plt.xlabel('Sepal.Length')
plt.ylabel('Sepal.Width')
plt.grid(True, linestyle='--', alpha=0.8)
plt.legend(loc='upper right', fontsize=5)

# figure 16
plt.subplot(4, 4, 4)
for i in range(len(species)):
    plt.scatter(iris.loc[iris.Species == species[i], 'Sepal.Length'],
                iris.loc[iris.Species == species[i], 'Sepal.Length'],
                s=5, c=colors[i], label=species[i])
plt.title("Sepal.Length vs Sepal.Length")
plt.xlabel('Sepal.Length')
plt.ylabel('Sepal.Length')
plt.grid(True, linestyle='--', alpha=0.8)
plt.legend(loc='upper right', fontsize=5)


# draw figures
plt.tight_layout(pad=1.08)
plt.show()
```

这里我们在画图的时候，使用的子图index顺序应该是：13-14-15-16-9-10-11-12-5-6-7-8-1-2-3-4。这样才能实现从下到上从左到右铺开16个子图。

4. 绘制六门课程的成绩分段统计情况直方图

这一题考的是子图和直方图。

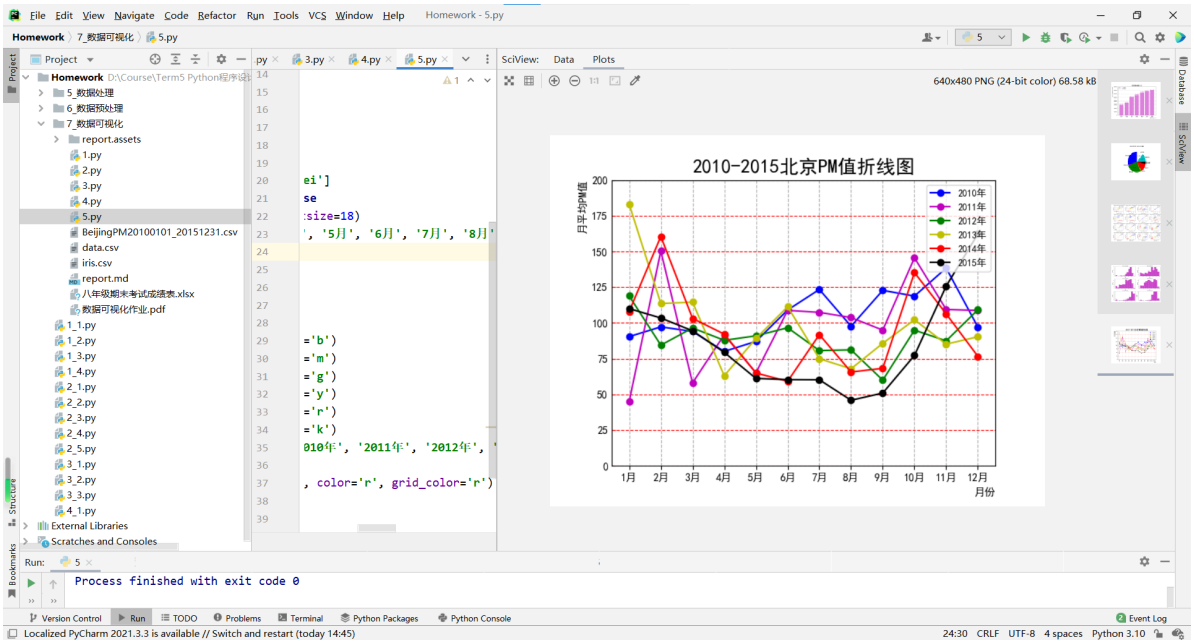


上图是可视化效果。代码如下：

```
import matplotlib.pyplot as plt
import pandas as pd

# get data
data = pd.read_excel('八年级期末考试成绩表.xlsx')

# set font and size
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.figure(figsize=(10, 10))

# draw figures
plt.subplot(3, 2, 1)
x = data["语文分数"]
plt.xlabel('分数', fontsize=10)
plt.ylabel('学生数量', fontsize=10)
plt.title("期末考试语文成绩", fontsize=10)
plt.hist(x, bins=[0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120],
facecolor='m', edgecolor="black", alpha=0.7)

plt.subplot(3, 2, 2)
x = data["数学分数"]
plt.xlabel('分数', fontsize=10)
plt.ylabel('学生数量', fontsize=10)
plt.title("期末考试数学成绩", fontsize=10)
plt.hist(x, bins=[0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120],
facecolor='m', edgecolor="black", alpha=0.7)

plt.subplot(3, 2, 3)
x = data["英语分数"]
plt.xlabel('分数', fontsize=10)
plt.ylabel('学生数量', fontsize=10)
plt.title("期末考试英语成绩", fontsize=10)
plt.hist(x, bins=[0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120],
facecolor='m', edgecolor="black", alpha=0.7)

plt.subplot(3, 2, 4)
x = data["物理分数"]
plt.xlabel('分数', fontsize=10)
plt.ylabel('学生数量', fontsize=10)
plt.title("期末考试物理成绩", fontsize=10)
plt.hist(x, bins=[0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100], facecolor='m',
edgecolor="black", alpha=0.7)

plt.subplot(3, 2, 5)
x = data["生物分数"]
plt.xlabel('分数', fontsize=10)
plt.ylabel('学生数量', fontsize=10)
plt.title("期末考试生物成绩", fontsize=10)
plt.hist(x, bins=[0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100], facecolor='m',
edgecolor="black", alpha=0.7)

plt.subplot(3, 2, 6)
x = data["政治分数"]
plt.xlabel('分数', fontsize=10)
plt.ylabel('学生数量', fontsize=10)
```

```
plt.title("期末考试政治成绩", fontsize=10)
plt.hist(x, bins=[0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100], facecolor='m',
edgecolor="black", alpha=0.7)

plt.tight_layout(pad=1.08)
plt.show()
```

5. 展示北 京市2010~2015年PM2.5指数月平均数据六条折线图



上图是可视化效果。代码如下：

```
import matplotlib.pyplot as plt
import pandas as pd

# preprocessing data
df = pd.read_csv('BeijingPM20100101_20151231.csv', usecols=[1, 2, 6, 7, 8, 9])
df['PM_avg'] = df.iloc[:, 2:6].mean(axis=1)
df.groupby(['year', 'month'])['PM_avg'].mean().to_csv("data.csv")

# get data and cut into slice
data = pd.read_csv("data.csv")
x = data['month'][:12]
data_2010 = data['PM_avg'][:12]
data_2011 = data['PM_avg'][12:24]
data_2012 = data['PM_avg'][24:36]
data_2013 = data['PM_avg'][36:48]
data_2014 = data['PM_avg'][48:60]
data_2015 = data['PM_avg'][60:72]

# initialize figure parameters
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
plt.title('2010-2015北京PM值折线图', fontsize=18)
plt.xticks(x, ['1月', '2月', '3月', '4月', '5月', '6月', '7月', '8月', '9月', '10
月', '11月', '12月'])
```

```
plt.xlabel('月份', loc='right')
plt.ylabel('月平均PM値', loc='top')
plt.ylim(0, 200)

# draw figure with plot
p1, = plt.plot(x, data_2010, 'o-', color='b')
p2, = plt.plot(x, data_2011, 'o-', color='m')
p3, = plt.plot(x, data_2012, 'o-', color='g')
p4, = plt.plot(x, data_2013, 'o-', color='y')
p5, = plt.plot(x, data_2014, 'o-', color='r')
p6, = plt.plot(x, data_2015, 'o-', color='k')
plt.legend([p1, p2, p3, p4, p5, p6], ['2010年', '2011年', '2012年', '2013年',
'2014年', '2015年'], loc='upper right', fontsize=9)
plt.grid(linestyle='--')
plt.tick_params(axis='y', direction='in', color='r', grid_color='r')
plt.show()
```