

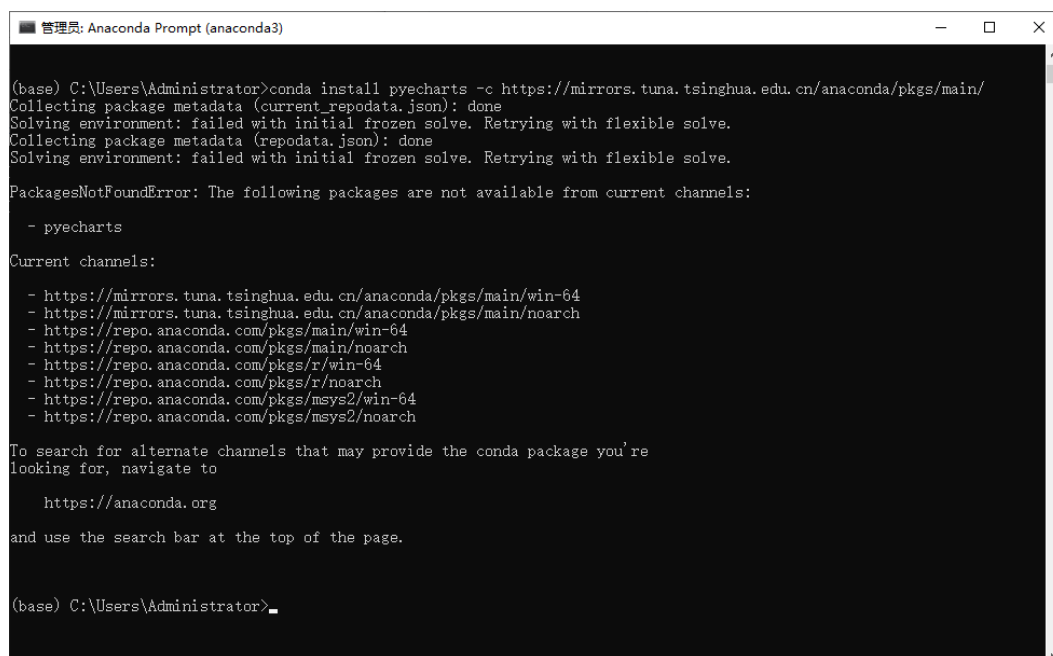
pyecharts 数据可视化

pyecharts 绘制柱状图

在 Python 中,使用 pyecharts 库导入 CSV 数据通常涉及到使用 `pandas` 库读取 CSV 文件,然后将数据转换为 pyecharts 能够处理的格式。

首先,确保安装了所需的库:

```
pip install pyecharts pandas
```



```
管理员: Anaconda Prompt (anaconda3)

(base) C:\Users\Administrator>conda install pyecharts -c https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/
Collecting package metadata (current_repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexible solve.
Collecting package metadata (repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexible solve.

PackagesNotFoundError: The following packages are not available from current channels:

- pyecharts

Current channels:

- https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/win-64
- https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/noarch
- https://repo.anaconda.com/pkgs/main/win-64
- https://repo.anaconda.com/pkgs/main/noarch
- https://repo.anaconda.com/pkgs/r/win-64
- https://repo.anaconda.com/pkgs/r/noarch
- https://repo.anaconda.com/pkgs/msys2/win-64
- https://repo.anaconda.com/pkgs/msys2/noarch

To search for alternate channels that may provide the conda package you're
looking for, navigate to

    https://anaconda.org

and use the search bar at the top of the page.

(base) C:\Users\Administrator>_
```

安装完成后,需要等待一会。

如果安装不成功

在命令行中输入:

```
conda install -c conda-forge pyecharts
```

从数据库中将表以 csv 格式文件导出

例 1 每年发生重大地震次数

```
import pyecharts.options as opts
from pyecharts.charts import Bar
import pandas as pd
```

```
# 读取CSV文件
```

```
df = pd.read_csv('countByYear.csv')
se = pd.Series(dtype='float64')
```

```
# 假设CSV文件有两列: 'Name'和'Value'
```

```
names = df['Year'].tolist()
values = df['count'].tolist()
```

```
# 创建柱状图
```

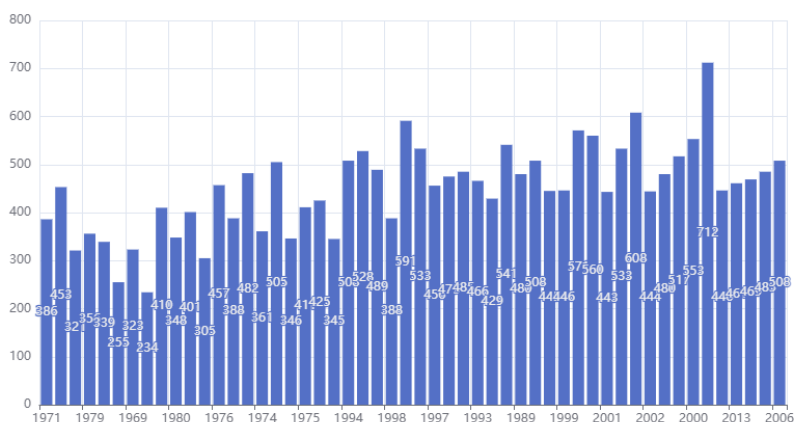
```
bar = (
    Bar()
    .add_xaxis(names)
    .add_yaxis('', values)
    .set_global_opts(title_opts=opts.TitleOpts(title="
1965-2016年地震次数"))
)
```

```
# 渲染图表到文件
```

```
bar.render('项目1-3.html')
```

自行添加 X 轴名称, Y 轴名称。

1965-2016年地震次数



1 快速绘制图表

【知识准备】

3.1 认识 Pyecharts

3.1.1 什么是 Pyecharts



pyecharts 是一个用于生成 Echarts 图表的类库，是一款将 Python 与 Echarts 相结合的强大的数据可视化工具。pyecharts 提供了丰富的图表类型，包括柱状图、折线图、散点图、饼图、地图等等，这些图表类型可以满足各种数据可视化的需求。

pyecharts 的使用非常简单，用户只需要几行代码就可以创建出高质量的图表。而且，pyecharts 还提供了丰富的扩展功能，用户可以根据自己的需要进行二次开发。总的来说，pyecharts 是一个功能强大、易用性高的数据可视化库，可以帮助用户更加高效地完成数据可视化工作。

3.1.2 Pyecharts 特性

pyecharts 是一个基于 Python 的开源可视化库，它的主要目的是让数据可视化变得更加简单、易用和高效。pyecharts 实际上是 Echarts 的 Python 封装，Echarts 是一个由百度开源的数据可视化 JS 库，它提供了各种各样的图表类型和交互功能。而 Python 是一门富有表达力的语言，很适合用于数据处理。当数据分析遇上数据可视化时，pyecharts 诞生了。可以理解为 pyecharts 是实现 Echarts 与 Python 对接的一个库。pyecharts 有以下特性：

1. 简洁的 API 设计，支持链式调用。
3. 囊括了 30+种常见图表。
3. 支持主流 Notebook 环境，Jupyter Notebook 和 JupyterLab。
4. 可轻松集成至 Flask, Django 等主流 Web 框架。
5. 高度灵活的配置项，可轻松搭配出精美的图表。
6. 详细的文档和示例，帮助开发者更快的上手项目。
7. 多达 400+地图文件以及原生的百度地图，为地理数据可视化提供强有力的支持。

3.2 安装 Pyecharts

目前 Pyecharts 一共有三个大版本，V0.X、V1.X、V3.X，版本之间差别比较大。V0.5.x 和 V1 间不兼容，V1 是一个全新的版本。V0.5x 版本仅支持 python3.7、3.4+，不再进行维护。新版本系列将从 V1.0.0 开始，V1.0x 版本，仅支持 python3.6+。V2 版本，支持 Python3.6~Python3.11，新版本系列从 Echarts4 切换到 Echarts5。

例 3.1 安装 pyecharts

使用 pyecharts 绘图时，需要在 Python 环境中安装 pyecharts 库。安装方法不依赖于你使用的 IDE（如 Anaconda 或 PyCharm），可以在任何 Python 环境中安装和使用 pyecharts。

1. 在 Anaconda 环境中安装 pyecharts

打开 Anaconda Prompt 工具，在提示符的后面输入如下命令：

```
conda install pyecharts
```

以上命令执行后，若 Anaconda Prompt 窗口中出现如下信息，表明 pyecharts 安装完成：

…省略行…

```
Installing collected packages: prettytable, simplejson, pyecharts
```

```
Successfully installed prettytable-0.7.2 pyecharts-1.5.1 simplejson-3.16.0
```

…省略行…

安装完成后，在命令提示符后面输入 python，之后输入如下导入语句：

```
from pyecharts.echarts import Bar
```

执行以上语句后，若 Anaconda Prompt 窗口没有出现任何错误信息，说明 pyecharts 安装成功，否则说明安装失败。

3. 在 PyCharm 中使用 pyecharts

打开 PyCharm，进入 Terminal（位于底部工具栏）或者（View -> Tool Windows -> Terminal），输入以下命令安装 pyecharts：

```
pip install pyecharts
```

3.3 快速绘制 pyecharts 图表

pyecharts 安装完成后，在 Python 代码中导入 pyecharts 开始绘图。使用 pyecharts 绘图的基本步骤如下。

1. 在.py 文件中，引入 pyecharts 库文件

使用 python 绘制 pyecharts 图表，首先在 py 文件中导入 pyecharts 库文件。pyecharts 是一个用于创建交互式图表的 Python 库，pyecharts 中的 options 模块提供了一系列的配置项，比如标题、图例、工具箱等，这些配置项可以用于定制图表（在任务 2 中详细介绍配置项）。

```
#导入 pyecharts 库
```

```
from pyecharts.charts import Bar
```

```
#从 pyecharts 库导入名为 options 的模块，并将其别名为 opts
```

```
from pyecharts.options import Opts
```

2. 创建图表对象

根据绘制的不同图表，使用不同的方法创建其图表对象。

```
#创建柱形图图表对象，Bar()创建柱状图
```

```
bar = Bar()
```

3. 添加数据到图表

add()方法用于添加散点图的数据系列。调用 Bar#add_xaxis()函数，设置 x 轴数据，实际数据放在列表中，作为参数传递给该函数；调用 Bar#add_yaxis()函数，设置 y 轴数据，第一个参数是柱状图标题，第二个参数是列表类型的容器变量，表示 y 轴的数据。

```
#添加数据
```

```
bar.add_xaxis([])
```

```
bar.add_yaxis("",[])
```

4. 设置图表的相关配置项

`set_global_opts` 是一个全局设置函数，用于设置图表的各种全局属性，其中 `title` 设置图表的标题。

```
#设置图表的标题
```

```
bar.set_global_opts(title_opts=opts.TitleOpts(title=""))
```

5. 渲染图表或者保存或输出图表

如果使用 Jupyter 绘制图表，则使用 `render_notebook()` 方法将图表渲染到 Jupyter Notebook 中。如果使用 pycharm 绘制图表，则使用 `render()` 方法将图表保存到 HTML 文件中。

例 3.2 使用 `pyecharts` 快速绘制柱状图并保存到例 3.3.html 文件中，示例代码如下：

```
from pyecharts.charts import Bar
from pyecharts import options as opts
bar = Bar()
bar.add_xaxis(['衬衣', '毛衣', '领带', '裤子', '风衣', '高跟鞋', '袜子'])
bar.add_yaxis("101 号店铺", [114, 55, 27, 101, 125, 27, 105])
bar.set_global_opts(title_opts=opts.TitleOpts(title='某商场销售情况'))
bar.render('例 3.3.html')
```

以上示例代码首先从 `pyecharts.charts` 模块中导入表示柱形图的类 `Bar`，接着创建一个柱状图 `Bar`（`Bar` 首字母要大写）对象并赋值给变量 `bar`；然后分别调用 `add_xaxis()` 和 `add_yaxis()` 方法为柱状图添加 x 轴和 y 轴数据；接着设置图表的标题等配置项；最后调用 `bar.render()` 方法在当前目录生成例 3.3.html 文件。在浏览器中打开例 3.3.html 文件，效果如图 3.3 所示。

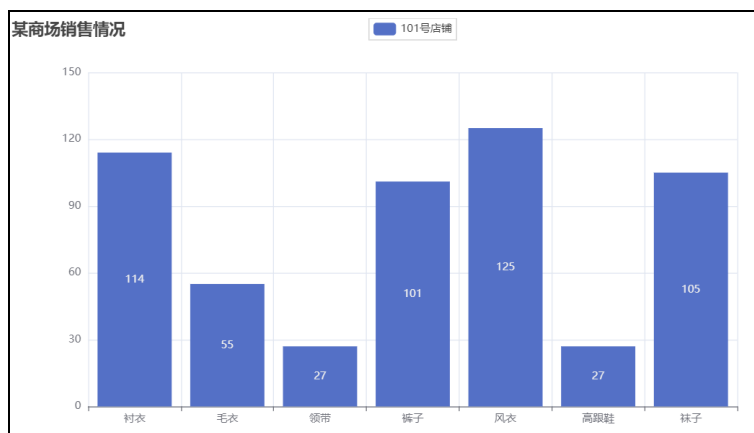


图 3.3 快速绘制柱状图

`Pyecharts` 在 v1 版本中增加了链式调用的功能。链式调用是指简化同一对象多次访问属性或调用方法的编码方式，以避免多次重复使用同一个对象变量，使代码变得简洁、易懂。

例 3.3 将例 3.2 的示例代码改为链式调用的方法，改后的代码如下：

```
from pyecharts.charts import Bar
from pyecharts import options as opts

bar = (
    Bar()
    .add_xaxis(['衬衣', '毛衣', '领带', '裤子', '风衣', '高跟鞋', '袜子'])
    .add_yaxis("101 号店铺", [114, 55, 27, 101, 125, 27, 105])
    .set_global_opts(title_opts=opts.TitleOpts(title="某商场销售情况"))
)

bar.render('例 3.3.html')
```

最终生成的例 3.3.html 文件中的柱状图与例 3.3.html 文件中的柱状图是一样的。

2 绘制交互式图形

【知识准备】

3.4 认识 pyecharts 图表类

在 pyecharts 中,Base 类是所有图表的基类,支持绘制 30 余种丰富的 Echarts 图表。pyecharts 针对每种图表均提供了相应的类,并将这些图表类封装到 pyecharts.charts 模块中,任务 1 中绘制柱状图的是 Bar 类。pyecharts.charts 模块常用的图表类如表 3.1 所示。



表 3.1 pyecharts.charts 模块的常用图表类

类	说明	
Bar	柱形图/条形图	直角坐标系图表
Line	折线图	
Scatter	散点图	
EffectScatter	带有涟漪特效动画的散点图	
HeatMap	热力图	
Boxplot	箱形图	
Line3D	3D 折线图	3D 图表
Bar3D	3D 柱形图	
Scatter3D	3D 散点图	
Surface3D	3D 曲面图	
Map	统计地图	地理坐标系图表
Tree	树状图	树型图表
Pie	饼图	其他图表
Funnel	漏斗图	
Gauge	仪表盘	
Sankey	桑基图	
Radar	雷达图	

WordCloud	词云图	
-----------	-----	--

表 3.1 中的图表类都可以使用与类同名的构造方法创建对象，Bar 类的构造方法的语法格式如下：

```
Bar(init_opts=opts.InitOpts())
```

其中 init_opts 参数表示初始化配置项，该参数需要接收一个 InitOpis 类的对象，通过构建的 Ini0pis 类的对象为图表指定一些通用的属性，例如背景颜色、画布大小等。任务 1 创建了指定画布大小的 Bar 类的对象，具体代码如下：

```
Bar(opts.InitOpts(width='600px',height='300px'))
```

3.5 认识 pyecharts 系列配置项



pyecharts 是一个用于创建交互式图表的 Python 库。在使用 pyecharts 时，可以通过配置项来定制图表的外观和行为。pyecharts.options 模块中包含众多关于定制图表组件及样式的配置项。按照配置内容的不同，配置项可以分为全局配置项和系列配置项。

3.5.1 全局配置项

全局配置项是一些针对图表通用属性的配置项，主要包括初始化属性、标题组件、图例组件、工具箱组件、视觉映射组件、提示框组件、数据区域缩放组件，其中每个配置项都对应一个类。Pyecharts 的全局配置项如表 3.2 所示。

表 3.2 pyecharts 全局配置项

类	说明
InitOpts	初始化配置项
AnimationOpts	ECharts 画图动画配置项
ToolBoxFeatureOpis	工具箱工具配置项
ToolboxOpts	工具箱配置项
BrushOprrs	区域选择组件配置项
TitleOpis	标题配置项
DataZoomOpts	数据区域缩放配置项
LegendOpts	图例配置项
VisualMapOpts	视觉映射配置项
TooltipOpts	提示框配置项
AxisLineOpts	坐标轴轴脊配置项
AxisTickOpts	坐标轴刻度配置项
AxisPointerOpts	坐标轴指示器配置项
AxisOpts	坐标轴配置项
SingleAxisOpts	单轴配置项
GraphicGroup	原生图形元素组件

表 3.2 中每个类都可以通过与之同名的构造方法创建实例，任务 1 中创建的 InitOpts

类的对象。每个类的构造方法的参数各有不同，由于篇幅有限，大家可以自行阅读 `pyecharts` 官方文档，此处不再赘述。

若 `pyecharts` 需要为图表设置全局配置项（`InitOpts` 除外），则需要将全局配置项传入 `set_global_opts()` 方法，语法格式如下：

```
set_global_opts(self,title_opts=opts.TitleOpts(),legend_opts=opts.LegendOpts(),tooltip_opts=None,toolbox_opts=None,brush_opts=None,xaxis_opts=None,yaxis_opts=None,visualmap_opts=None,datazoom_opts=None,graphic_opts=None,axispointer_opts=None)
```

该方法各参数的含义如表 3.2 所示。

下面介绍初始化配置项、标题配置项、图例配置项和坐标轴配置项的使用格式和各参数含义。

1. 初始化配置项

初始化配置项是在初始化对象中进行配置的，可以设置画布的长与宽、网页标题、图表主题、背景色等。初始配置项是通过 `options` 模块中的 `InitOpts` 类实现的，可以使用 `init_opts` 作为参数传递。`InitOpts` 类的使用格式如下：

```
class InitOpts(width='900px',height='500px',chart_id=None,renderer=RenderType.CANVAS,page_title='Awesome-pyecharts',theme='white',bg_color=None,js_host="",animation_opts=AnimationOpts())
```

各参数的含义如表 3.3 所示。

表 3.3 标题配置项参数

参数	说明
width	接收 str，表示图表画布宽度。默认为 900px
height	接收 str，表示图表画布高度。默认为 500px
chart_id	接收 str，表示图表 ID，图表唯一标识，可用于在多个图表合并时进行图表之间的区分。默认为 None
renderer	接收 str，表示渲染风格，可选 canvas 或 svg。默认为 canvas
page_title	接收 str，表示网页标题。默认为 Awesome-pyecharts
theme	接收 str，表示图表主题。默认为 white
bg_color	接收 str，表示图表背景颜色。默认为 None

1. 标题配置项

标题配置项是通过 `options` 模块中的 `TitleOpts` 类实现的，可以使用 `title_opts` 作为参数传递给 `set_global_opts()` 方法。`TitleOpts` 类的基本使用格式如下：

```
class TitleOpts(title=None,title_link=None,title_target=None,subtitle=None,subtitle_link=None,subtitle_target=None,pos_left=None,pos_right=None,pos_top=None,pos_bottom=None,padding=5,item_gap=10,title_textstyle_opts=None,subtitle_textstyle_opts=None)
```

各参数的含义如表 3.4 所示。

表 3.4 标题配置项参数

参数	说明
title	接收 str，表示主标题文本，支持使用\n换行。默认为 None

title_link	接收 str，表示主标题跳转 URL 链接。默认为 None
title_target	接收 str，表示主标题跳转链接方式，可选 self、blank，self 表示当前窗口打开，blank 表示新窗口打开。默认为 blank
subtitle	接收 str，表示副标题文本，支持使用\n 换行。默认为 None
subtitle_link	接收 str，表示副标题跳转 URL 链接。默认为 None
subtitle_target	接收 str，表示副标题跳转链接方式。默认为 blank
item_gap	接收 numeric，表示主副标题之间的间距。默认为 10
title_textstyle_opts	表示主标题字体样式配置项
subtitle_textstyle_opts	表示副标题字体样式配置项

2. 图例配置项

图例配置项是通过 options 模块中的 LegendOpts 类实现的，可以使用 legend_opts 作为参数传递给 set_global_opts()方法。LegendOpts 类的基本使用格式如下。

```
class LegendOpts(type_=None, selected_mode=None, is_show=True, pos_left=None, pos_right=None, pos_top=None, pos_bottom=None, orient=None, align=None, padding=5, item_gap=10, item_width=25, item_height=14, inactive_color=None, textstyle_opts=None, legend_icon=None)
```

各参数的含义如表 3.5 所示。

表 3.5 标题配置项参数

参数	说明
type_	接收 str，表示图例的类型。可选 plain、scroll，plain 表示普通图例，scroll 表示可滚动翻页的图例。默认为 None
is_show	接收 bool，表示是否显示图例组件，默认为 True
orient	接收 str，表示图例列表的布局朝向，可选 horizontal、vertical。默认为 None
item_gap	接收 int，表示图例每项之间的间隔。默认为 10
inactive_color	接收 str，表示图例关闭时的颜色。默认为#ccc
pos_left	接收 str、numeric，表示图例组件离容器左侧的距离。默认为 None
pos_right	接收 str、numeric，表示图例组件离容器右侧的距离。默认为 None
pos_top	接收 str、numeric，表示图例组件离容器上侧的距离。默认为 None
pos_bottom	接收 str、numeric，表示图例组件离容器下侧的距离。默认为 None

3. 坐标轴配置项

坐标轴配置项是通过 options 模块中的 AxisOpts 类实现的，可以使用 xaxis_opts 或 yaxis_opts 作为参数传递给 set_global_opts()方法。AxisOpts 类的基本使用格式如下。

```
Class AxisOpts(type_=None,name=None,is_show=True,is_scale=False,is_inverse=False,name_location='end', name_gap=15,name_rotate=None,interval=None,grid_index=0,position=None,offset=0, split_number=5,boundary_gap=None,min_=None,max_=None,min_interval=0,max_interval=None,axisline_opts=None ,axistick_opts=None,axislabel_opts=None,axispointer_opts=None,name_textstyle_opts=None,splitarea_opts=None,splitline_opts= SplitLineOpts(), minor_tick_opts=None, minor_split_line_opts=None)
```

各参数的含义如表 3.6 所示。

表 3.6 标题配置项参数

参数	说明
type_	接收 str，表示坐标轴类型。可选 value、category、time、log，value 表示数值轴，适用于连续数据；category 表示类目轴，适用于离散的类目数据；time 表示时间轴，适用于连续的时序数据；log 表示对数轴，适用于对数数据。默认为 None
name	接收 str，表示坐标轴名称。默认为 None
is_show	接收 bool，表示是否显示 X 坐标轴。默认为 True
is_inverse	接收 bool，表示是否反向坐标轴。默认为 False
name_gap	接收 numeric，表示坐标轴名称与轴线之间的距离。默认为 15
name_rotate	接收 numeric，表示坐标轴名字旋转角度值。默认为 None
position	接收 str，表示 X 轴的位置，可选 top、bottom，top 表示在上侧，bottom 表示在下侧。默认为 None
split_number	接收 numeric，表示坐标轴的分割段数。默认为 5
min_	接收 str、numeric，表示坐标轴刻度最小值。默认为 None
max_	接收 str、numeric，表示坐标轴刻度最大值。默认为 None

3.5.2 系列配置项

系列配置项是一些针对图表特定元素属性的配置项，包括文本样式、标签、线条样式、标记样式、填充样式等，其中每个配置项都对应一个类。pyecharts 的系列配置项如表 3.7 所示。

表 3.7 pyecharts 系列配置项

类	说明
IemStyleOpts	图元样式配置项
TextStyleOpts	文本样式配置项
LabelOpts	标签配置项
LineStyleOpts	线条样式配置项
SplitLineOpts	分割线配置项
MarkPointOpts	标记点配置项
MarkLineOpts	标记线配置项
MarkAreaOpts	标记区域配置项
EffectOpts	涟漪特效配置项
AreaStyleOpts	区域填充样式配置项
SplitAreaOpts	分隔区域配置项
GridOpts	直角坐标系网格配置项

以上每个类都可以通过与之同名的构造方法创建实例。例如，创建一个标签配置项的代码如下：

```
label_opts=opts.LabelOpts(is_show=True,position='right',color='gray',font_size=14,rotate=10)
```

其中，LabelOpts()方法的参数 is_show 设为 True，表示显示标签；参数 position 设

为'right'，表示标注于图形右侧；参数 color 设为'gray'，表示标签文本的颜色为灰色；参数 font_size 设为 14，说明标签文本的字体大小为 14 号；参数 rotate 设为 10，说明标签逆时针旋转 10 度。

若 pyecharts 需要为图表设置系列配置项，则需要将系列配置项传入 add()或 add_xx(方法（直角坐标系图表一般使用 add_yaxis()方法）中。例如，隐藏任务 1 柱形图的注释文本，改后的代码如下：

```
bar.add_yaxis("101 店铺",[114,55,27,101,125,27,105],label_opts=opts.LabelOpts(is_show=False))
```

下面介绍文字样式配置项、标签配置项、线样式配置项和标记点配置项的使用格式和各参数含义。

1. 文字样式配置项

文字样式配置项是通过 options 模块中的 TextStyleOpts 类实现的，可以使用 text_style_opts 作为参数传递给 set_series_opts()方法。TextStyleOpts 类的基本使用格式如下。

```
class TextStyleOpts(color=None, font_style=None,font_weight=None, font_family=None, font_size=None, align=None,vertical_align=None,line_height=None,background_color=None,border_color=None,border_width=None, border_radius=None,padding=None,shadow_color=None,shadow_blur=None, width=None, height=None, rich=None)
```

各参数的含义如表 3.8 所示。

表 3.8 文字样式配置项参数

参数	说明
color	接收 str，表示文字颜色。默认为 None
font_style	接收 str，表示文字字体风格，可选 normal、italic、oblique。默认为 None
font_weight	接收 str，表示主标题字体的粗细，可选 normal、bold、bolder、lighter。默认为 None
font_family	接收 str，表示文字的字体系列。默认为 None
font_size	接收 numeric，表示文字的字体大小。默认为 None
align	接收 str，表示文字水平对齐方式。默认为 None
vertical_align	接收 str，表示文字垂直对齐方式。默认为 None
line_height	接收 str，表示行高。默认为 None
background_color	接收 str，表示文字块背景色。默认为 None
border_color	接收 str，表示文字块边框颜色。默认为 None
border_width	接收 numeric，表示文字块边框宽度。默认为 None

3. 标签配置项

标签配置项是通过 options 模块中的 LabelOpts 类实现的，可以使用 label_opts 作为参数传递给 set_series_opts()方法。LabelOpts 类的基本使用格式如下。

```
class LabelOpts(is_show=True, position='top', color=None, distance=None, font_size=12, font_style=None, font_weight=None, font_family=None, rotate=None, margin=8, interval=None, horizontal_align=None, vertical_align=None, formatter=None, rich=None)
```

各参数的含义如表 3.9 所示。

表 3.9 标签配置项参数

参数	说明
color	接收 str，表示文字颜色。默认为 None
font_style	接收 str，表示文字字体风格，可选 normal、italic、oblique。默认为 None
font_weight	接收 str，表示主标题字体的粗细，可选 normal、bold、bolder、lighter。默认为 None
font_family	接收 str，表示文字的字体系列。默认为 None
font_size	接收 numeric，表示文字的字体大小。默认为 None
align	接收 str，表示文字水平对齐方式。默认为 None
vertical_align	接收 str，表示文字垂直对齐方式。默认为 None
line_height	接收 str，表示行高。默认为 None
background_color	接收 str，表示文字块背景色。默认为 None
border_color	接收 str，表示文字块边框颜色。默认为 None
border_width	接收 numeric，表示文字块边框宽度。默认为 None

3. 线样式配置项

线样式配置项是通过 options 模块中的 LineStyleOpts 类实现的，可以使用 line_style_opts 作为参数传递给 set_series_opts()方法。LineStyleOpts 类的基本使用格式如下。

```
class LineStyleOpts(is_show=True, width=1, opacity=1, curve=0, type_='solid', color=None)
```

各参数的含义如表 3.10 所示。

表 3.10 线样式配置项参数

参数	说明
is_show	接收 bool，表示是否显示线。默认为 True
width	接收 numeric，表示线的宽度。默认为 1
opacity	接收 numeric，表示图形透明度，支持从 0 到 1 的数字。默认为 1
curve	接收 numeric，表示线的弯曲度，0 表示完全不弯曲。默认为 0
type_	接收 str，表示线的类型，常用 solid、dashed、dotted。默认为 solid
color	接收 str，表示线的颜色。默认为 None

4. 标记点配置项

标记点配置项是通过 options 模块中的 MarkPointOpts 类实现的，可以使用 markpoint_opts 作为参数传递给 set_series_opts()方法。MarkPointOpts 类的基本使用格式如下。

```
class MarkPointOpts(data=None, symbol=None, symbol_size=None, label_opts=options.LabelOpts(position='inside', color='#fff'))
```

各参数的含义如表 3.11 所示。

表 3.11 标记点配置项参数

参数	说明
----	----

data	接收 Sequence 对象，表示标记点数据。默认为 None
symbol	接收 str，表示标记的图形，提供的标记类型包括 circle、rect、roundrect、triangle、diamond、pin、arrow、None。默认为 None
symbol_size	接收 numeric，表示标记的大小，可以设置成单一的数字，如 10；也可以使用数组分开表示宽和高，例如，[20, 10]表示标记宽为 20，高为 10。默认为 None
label_opts	表示标签配置项

3.6 渲染图表



Pycharts 图表基类 Base 主要提供了两个渲染图表的方法：render() 和 render_notebook()。具体介绍如下。

1. render()方法

render()方法用于将图表渲染到 HTML 文件，默认为位于程序根目录的 render.html 文件。render()方法的语法格式如下：

```
render(self,path="render.html",template_name="simple_chart.html",env=None,**kwargs)
```

以上方法中的参数 path 表示生成文件的路径，默认为 render.html；template_name 表示模板的路径。Render()方法会返回 HTML 文件的路径字符串。

3.render_notebook()方法

render_notebook()方法用于将图表渲染到 Jupyter Notebook 工具中，它无须接收任何参数。例如，任务 1 中渲染图表到 Jupyter Notebook 中的代码如下：

```
bar.render_notebook()
```

3.7 绘制柱状图



在 pyecharts 库中，可使用 Bar 类绘制条形图或柱形图。Bar 类的基本使用格式如下。

```
class Bar(init_opts=opts.InitOpts()).add_xaxis(xaxis_data).add_yaxis(series_name,y_axis, is_selected
=True,xaxis_index=None,yaxis_index=None,is_legend_hover_link=True,color=None,is_show_background
=False,background_style=None,stack=None,bar_width=None,bar_max_width=None,bar_min_width=None,
bar_min_height=0,category_gap='20%',gap='30%', is_large=False, large_threshold=400, dimensions=None,
series_layout_by='column',dataset_index=0, is_clip=True, z_level=0, z=2, label_opts=opts.LabelOpts(),
markpoint_opts=None,markline_opts=None,tooltip_opts=None,itemstyle_opts=None,encode=None)
.set_series_opts().set_global_opts()
```

各参数的含义如表 3.12 所示。

表 3.12 柱状图 Bar 类参数

参数	说明
init_opts=opts.InitOpts()	设置初始配置项，参考 3.5.1 小节
add_xaxis()	添加 X 轴数据项
xaxis_data	接收 Sequence，表示 X 轴数据项。无默认值

add_yaxis()	添加 Y 轴数据项
series_name	系列名称，用于 tooltip 的显示，legend 的图例筛选。无默认值
y_axis	接收 numeric、opts.BarItem、dict 型序列数据，表示系列数据。无默认值
is_selected	否选中图例。默认为 True
xaxis_index	使用的 x 轴的 index，在单个图表实例中存在多个 x 轴的时候有用。默认为 None
yaxis_index	使用的 y 轴的 index，在单个图表实例中存在多个 y 轴的时候有用。默认为 None
is_legend_hover_link	否启用图例在 hover 时的联动高亮。默认为 True
color	系列 label 颜色。默认为 None
is_show_background	是否显示柱条的背景色。默认为 False
stack	数据堆叠，同类目轴上系列配置相同的 stack 值可以堆叠放置。默认为 None
bar_width	柱条的宽度，不设置时为自适应。可以是绝对值或百分数，如 40、60%。在同一坐标系上，此属性会被多个 bar 系列共享。此属性应设置于此坐标系中最后一个 bar 系列上才会生效，并且是对此坐标系中所有 bar 系列生效。默认为 None
bar_max_width	柱条的最大宽度。默认为 None
bar_min_width	柱条的最小宽度。在直角坐标系中，默认为 1。否则，默认为 null
bar_min_height	柱条最小高度，可用于防止某数据项的值过小而影响交互。默认为 0
category_gap	同一系列的柱间距离。默认为 20%
set_series_opts()	设置系列配置项，参考 3.5.1 小节
set_global_opts()	设置全局配置项，参考 3.5.1 小节

例 3.4 绘制柱状图，展示 1~6 月 101 店铺连衣裙、雪纺衬衫、潮流 T 恤的销售情况。
示例代码如下：

```
from pyecharts.charts import Bar
from pyecharts import options as opts
bar = (
    Bar(init_opts=opts.InitOpts(theme=ThemeType.LIGHT))
    .add_xaxis(["一月", "二月", "三月", "四月", "五月", "六月"])
    .add_yaxis("连衣裙", [19, 32, 69, 42, 87, 128])
    .add_yaxis("雪纺衬衫", [51, 22, 48, 68, 95, 115])
    .add_yaxis("潮流 T 恤", [32, 22, 57, 42, 145, 125])
    # 标记数据的最大值、最小值和平均值
    .set_series_opts(label_opts=opts.LabelOpts(is_show=False),
        markpoint_opts=opts.MarkPointOpts(
            data=[opts.MarkPointItem(type_="max", name="最大值"),
                opts.MarkPointItem(type_="min", name="最小值"),
                opts.MarkPointItem(type_="average", name="平均值"),]
        ),
        markline_opts=opts.MarkLineOpts(
```

```

data=[opts.MarkLineItem(type_="average",name="平均值")]
)
)
.set_global_opts(title_opts=opts.TitleOpts(title="1~6 月销售情况", subtitle="101 店铺"))
)
bar.render("例 3.4.html")

```

运行程序，效果如图 3.9 所示。

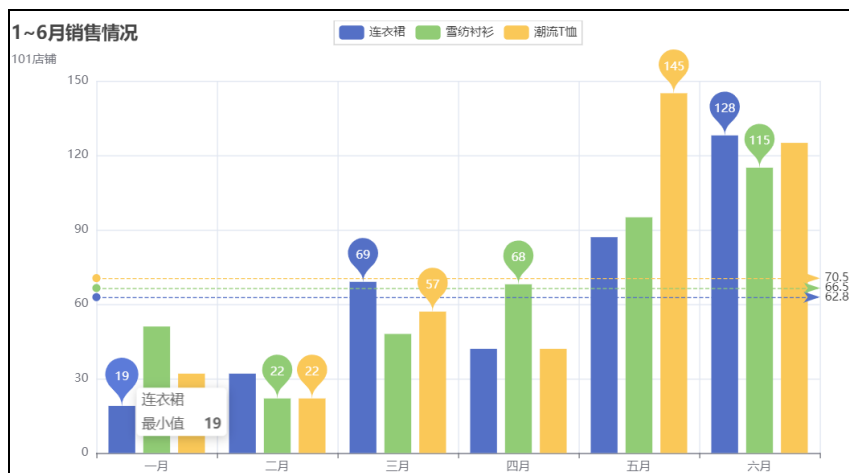


图 3.9 101 店铺销售情况柱状图

例 3.5 绘制堆积柱状图，展示 1~6 月 101、102、103、104、105 五家店铺连衣裙的销售情况。示例代码如下：

```

from pyecharts.charts import Bar
from pyecharts import options as opts
x_data = ["一月", "二月", "三月", "四月", "五月", "六月"]
y_data1 = [19, 32, 69, 42, 87, 128] # 101 店铺的数据
y_data2 = [30, 40, 50, 68, 95, 115] # 102 店铺的数据
bar = (Bar(init_opts=opts.InitOpts())
.add_xaxis(x_data).add_yaxis("101 店铺", y_data1).add_yaxis("102 店铺", y_data2)
.add_yaxis("103 店铺", [21, 26, 30, 36, 44, 90])
.add_yaxis("104 店铺", [12, 19, 21, 27, 34, 40]).add_yaxis("105 店铺", [31, 35, 38, 49, 87, 119])
.set_series_opts(stack="stack")
.set_global_opts(title_opts=opts.TitleOpts(title="1~6 月销售情况", subtitle="五家店铺连衣裙销售情况")))
bar.render("例 3.5.html")

```

运行程序，效果如图 3.10 所示。

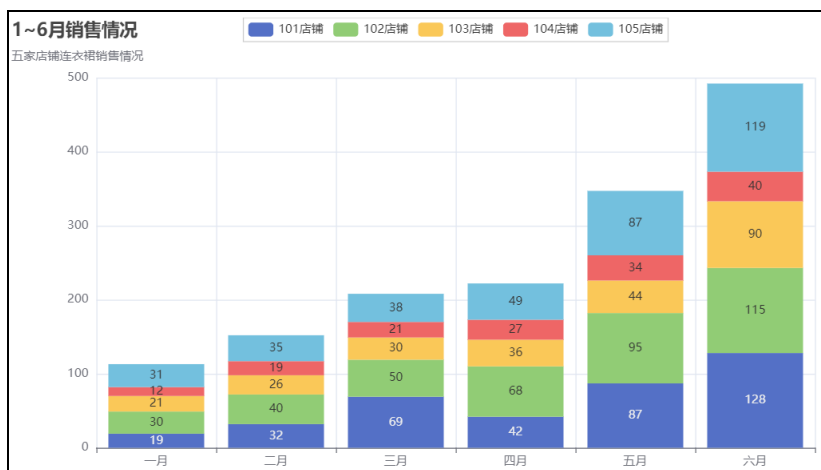


图 3.10 五家店铺销售情况堆积柱状图

例 3.6 绘制条形图，展示 6 月份 101、102、103 三家店铺连衣裙、雪纺衬衫和潮流 T 恤的销售情况。示例代码如下：

```
from pyecharts.charts import Bar
from pyecharts import options as opts

bar = Bar()
bar.add_xaxis(["101 店铺", "102 店铺", "103 店铺"]) # 添加 x 轴数据
# 添加 y 轴数据 添加 label_opts 使数据显示在右侧
bar.add_yaxis("连衣裙", [128, 115, 119], label_opts=LabelOpts(position="right"))
bar.add_yaxis("雪纺衬衫", [21, 27, 42], label_opts=LabelOpts(position="right"))
bar.add_yaxis("潮流 T 恤", [35, 47, 86], label_opts=LabelOpts(position="right"))
bar.reversal_axis() # 反转 x 和 y 轴
bar.render("例 3.6.html")
```

运行程序，效果如图 3.11 所示。

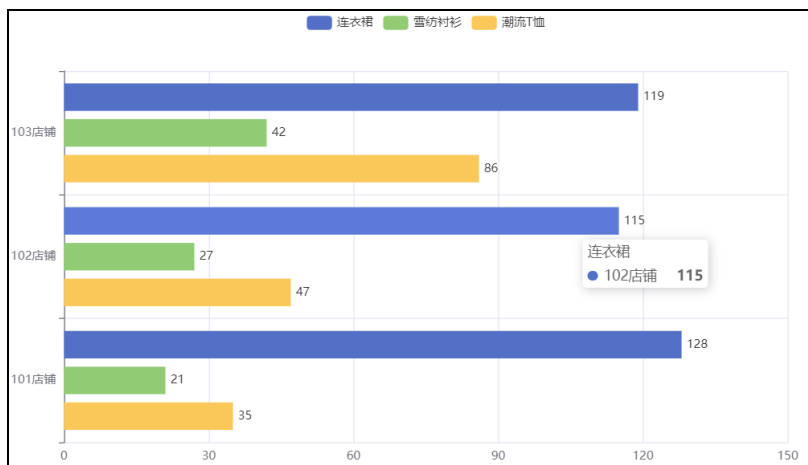


图 3.11 三家店铺销售情况条形图

3.8 绘制折线图

在 pyecharts 库中，可使用 Line 类绘制折线图。Line 类的基本使用格式如下。



```
class Line(init_opts=opts.InitOpts()).add_xaxis(xaxis_data)
    .add_yaxis(series_name,y_axis,is_selected=True,is_connect_nones=False,xaxis_index=None,yaxis_index=
None,color=None,is_symbol_show=True,symbol=None,symbol_size=4,stack=None,is_smooth=False,is_clip=Tr
ue,is_step=False,is_hover_animation=True,z_level=0,z=0,markpoint_opts=None,markline_opts=None,
tooltip_opts=None,label_opts=opts.LabelOpts(),linestyle_opts=opts.LineStyleOpts(),areastyle_opts=
opts.AreaStyleOpts(),itemstyle_opts=None)
.set_series_opts()
```

各参数的含义如表 3.13 所示。

表 3.13 折线图 Line 类参数

参数	说明
add_xaxis()	添加 X 轴数据项
xaxis_data	X 轴数据项。无默认值
add_yaxis()	添加 Y 轴数据项
series_name	系列名称，用于 tooltip 的显示，legend 的图例筛选。无默认值
y_axis	系列数据。无默认值
is_selected	是否选中图例。默认为 True
is_connect_nones	是否连接空数据。当含有空数据时，使用 None 填充。默认使 False
xaxis_index	使用的 x 轴的 index，在单个图表实例中存在多个 x 轴的时候有用。默认为 None
yaxis_index	使用的 y 轴的 index，在单个图表实例中存在多个 y 轴的时候有用。默认为 None
color	系列 label 颜色。默认为 None
is_symbol_show	是否显示 symbol。如果为 false，那么只有在 tooltip hover 的时候显示。默认为 True
symbol	标记的图形，可选标记类型包括 circle、rect、roundrect、triangle、diamond、pin、arrow、None。默认为 None
symbol_size	标记的大小，可以设置成单一的数字，如 10；也可以用数组分开表示宽和高，例如，[20, 10]表示标记宽为 20，高为 10。默认为 4
stack	数据堆叠，同类目轴上系列配置相同的 stack 值可以堆叠放置。默认为 None
is_smooth	是否平滑曲线。默认为 Flase
is_clip	是否裁剪超出坐标系部分的图形。默认为 True
is_step	是否显示成阶梯图。默认为 False

例 3.7 绘制折线图，展示 1~6 月 101 店铺连衣裙、雪纺衬衫、潮流 T 恤的销售情况。示例代码如下。

```
from pyecharts.charts import Line
from pyecharts import options as opts
```

```

line = (
    Line().add_axis(["Jan", "Feb", "Mar", "Apr", "May", "Jun"])
    # 设置平滑线、标记图形、标记大小
    .add_yaxis("连衣裙", [19, 30, 69, 40, 81, 128], is_smooth=True, symbol='diamond', symbol_size=15)
    .add_yaxis("雪纺衬衫", [51, 35, 39, 68, 95, 115], symbol='triangle', symbol_size=15)
    .add_yaxis("潮流 T 恤", [27, 18, 57, 50, 145, 105], symbol='arrow', symbol_size=15)
    .set_global_opts(title_opts=opts.TitleOpts(title="101 店铺 1~6 月销售趋势"),
        yaxis_opts=opts.AxisOpts(name="销售数量（件）",
            name_location="center", name_gap=40))
)
line.render("例 3.7.html")

```

运行程序，效果如图 3.12 所示。

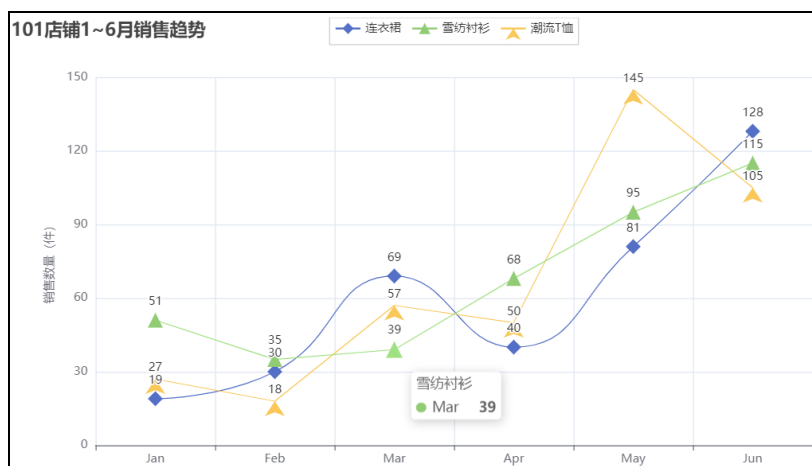


图 3.12 101 店铺销售情况折线图

例 3.8 绘制堆积面积图，展示 1~6 月 101 店铺连衣裙、雪纺衬衫、潮流 T 恤的销售情况。示例代码如下。

面积图又称区域图，强调数量随时间而变化的程度，也可用于引起人们对总值趋势的注意。Line 类绘制面积图，主要是在 add_yaxis 参数中配置区域填充样式配置项，即 options.AreaStyleOpts(opacity=0, color=None)，其中 opacity 参数为图形透明度，支持从 0 到 1 的数字，为 0 时不绘制该图形，color 参数为填充的颜色。

```

from pyecharts import options as opts
from pyecharts.charts import Line

line = (
    Line().add_axis(["Jan", "Feb", "Mar", "Apr", "May", "Jun"])
    .add_yaxis("连衣裙", [19, 30, 69, 40, 81, 128], is_smooth=True,
        areastyle_opts=opts.AreaStyleOpts(opacity=0.5))
    .add_yaxis("雪纺衬衫", [51, 35, 39, 68, 95, 115], is_smooth=True,
        areastyle_opts=opts.AreaStyleOpts(opacity=0.5))
)

```

```

.add_yaxis("潮流 T 恤", [27, 18, 57, 50, 145, 105],is_smooth=True,
          areastyle_opts=opts.AreaStyleOpts(opacity=0.5))
.set_global_opts(title_opts=opts.TitleOpts(title="101 店铺 1~6 月销售趋势"),
                 yaxis_opts=opts.AxisOpts(name="销售数量（件）",
                                           name_location="center",name_gap=40))
)
line.render("例 3.8.html")

```

运行程序，效果如图 3.13 所示。

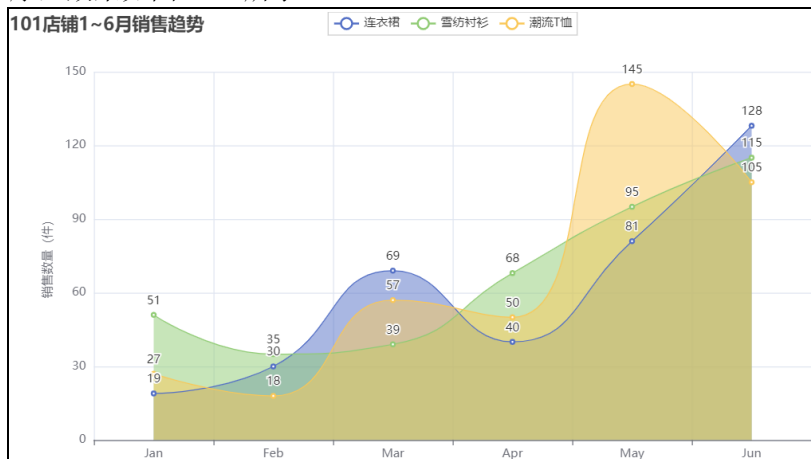


图 3.13 101 店铺销售情况堆积面积图

3.9 绘制饼图

在 pyecharts 库中，可使用 Pie 类绘制折线图。Pie 类的基本使用格式如下。

```

class Pie(init_opts=opts.InitOpts()).add(series_name, data_pair, color=None, radius=None,
center=None, rosetype=None, is_clockwise=True, label_opts=opts.LabelOpts(), tooltip_opts=None,
itemstyle_opts=None, encode=None).set_series_opts().set_global_opts()

```

各参数的含义如表 3.14 所示。

表 3.14 饼图 Pie 类参数

参数	说明
series_name	系列名称，用于 tooltip 的显示，legend 的图例筛选。无默认值
data_pair	系列数据项，格式为[(key 1, value1), (key2, value2)]。无默认值
color	系列 label 颜色。默认为 None
radius	饼图的半径，数组的第一项是内半径，第二项是外半径。默认为 None
center	饼图的中心（圆心）坐标，数组的第一项是横坐标，第二项是纵坐标，默认设置成百分比。当设置成百分比时第一项是相对于容器宽度，第二项是相对于容器高度。默认为 None
rosetype	是否展示成南丁格尔图，通过半径区分数据大小，有 radius 和 area 两种模式。radius 表示扇区圆心角展现数据的百分比，半径展现数据的大小，area 表示所有扇区圆心角相同，仅通过半径展现数据大小。默认为 None



is_clockwise	饼图的扇区是否是顺时针排布。默认值是 True
--------------	-------------------------

例 3.9 绘制饼图，展示某公众号的粉丝来源，粉丝来源包含搜索引擎、朋友分享、视频号引流、软文推广、小程序引流五大类。示例代码如下。

```
from pyecharts.charts import Pie
from pyecharts import options as opts
pie = Pie()
pie.add("公众号粉丝来源", [('搜索引擎', 1048), ('朋友分享', 735), ('视频号引流', 580), ('软文推广', 484),
                             ('小程序引流', 300)])
pie.set_global_opts(title_opts=opts.TitleOpts(title="粉丝来源"),
                    legend_opts=opts.LegendOpts(orient="vertical", pos_top="15%", pos_left="2%"))
pie.render("例 3.9.html")
```

运行程序，效果如图 3.14 所示。

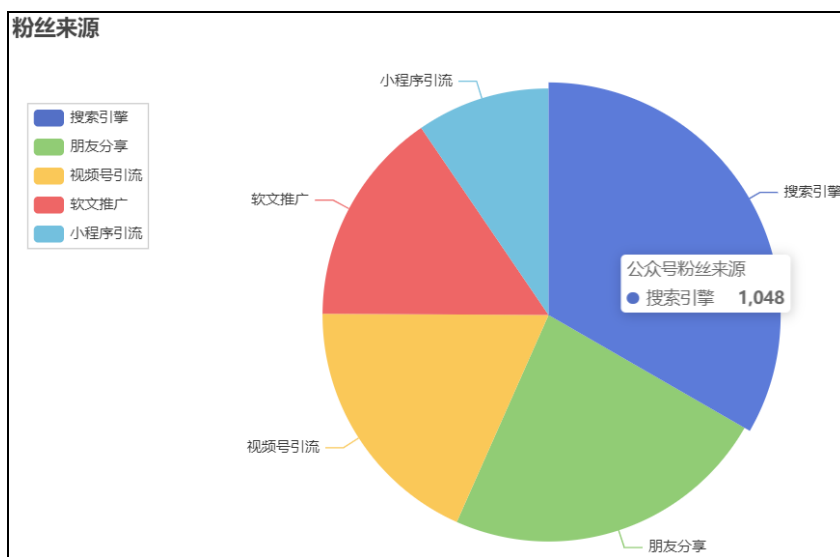


图 3.14 某公众号粉丝来源饼图

例 3.10 绘制环形图，展示某公众号的粉丝来源，粉丝来源包含搜索引擎、朋友分享、视频号引流、软文推广、小程序引流五大类。示例代码如下。

环形图（Circular Sector Graph）与饼图类似，但又有区别。环形图中间有一个空洞，每个样本用一个环来表示，样本中的每一部分数据用环中的一段表示。可以通过 add 函数中增加 radius 参数设置绘制环形图。

```
from pyecharts.charts import Pie
from pyecharts import options as opts
# 数据
labels = ["搜索引擎", "朋友分享", "视频号引流", "软文推广", "小程序引流"]
data = [1048, 735, 580, 484, 300]
ring = (Pie().add("", [list(z) for z in zip(labels, data)], radius=["40%", "75%"],) # 设置内半径和外半径
        .set_global_opts(title_opts=opts.TitleOpts(title="公众号粉丝来源"))
```

```

        .set_series_opts(label_opts=opts.LabelOpts(is_show=True)) # 不显示标签
    )
    ring.render("例 3.10.html")

```

运行程序，效果如图 3.15 所示。

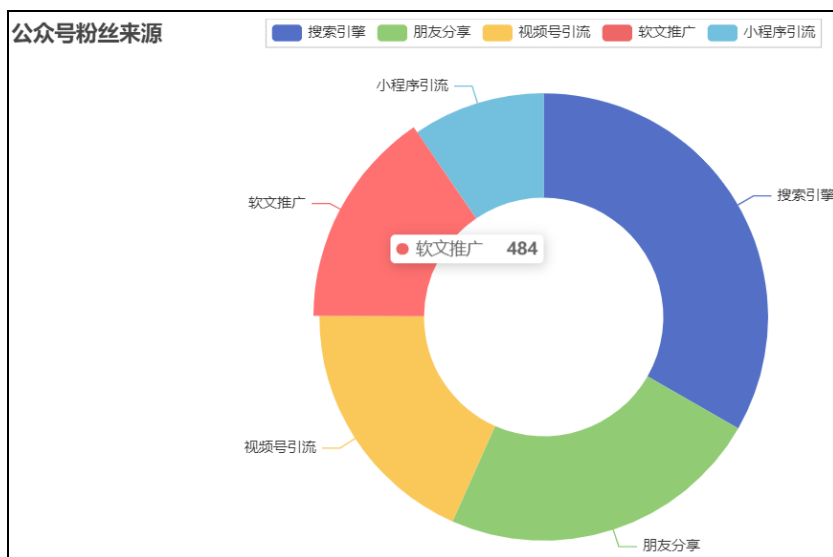


图 3.15 某公众号粉丝来源环形图

例 3.11 绘制玫瑰图，展示某公众号的粉丝来源，粉丝来源包含搜索引擎、朋友分享、视频号引流、软文推广、小程序引流五大类。示例代码如下。

玫瑰图（Rose Graph）又称为极面积图，使用圆弧的半径长短表示数据量。读者可以通过 Pie 类绘制玫瑰图，只需要在 add 函数中增加 rosetype 参数设置即可完成玫瑰图的绘制。

虽然玫瑰图反映的比例关系与饼图、环形饼图是一致的，但通过扇区圆心角展现数据的百分比的直观显示，可以一目了然的看出各组成部分所占的比例关系。

```

from pyecharts.charts import Pie
from pyecharts import options as opts
label = ["搜索引擎","朋友分享","视频号引流","软文推广","小程序引流"]
values = [1048,735,580,484,300]
pie=(Pie().add("",[list(z) for z in zip(label, values)],radius=["30%", "75%"],center=["50%", "50%"],
        rosetype="radius",label_opts=opts.LabelOpts(is_show=False),)
    .set_global_opts(title_opts=opts.TitleOpts(title="公众号粉丝来源"))
    .set_series_opts(label_opts=opts.LabelOpts(formatter="{b}:{c} {d}%")) # {d}%为百分比
)
pie.render("例 3.11.html")

```

运行程序，效果如图 3.16 所示。

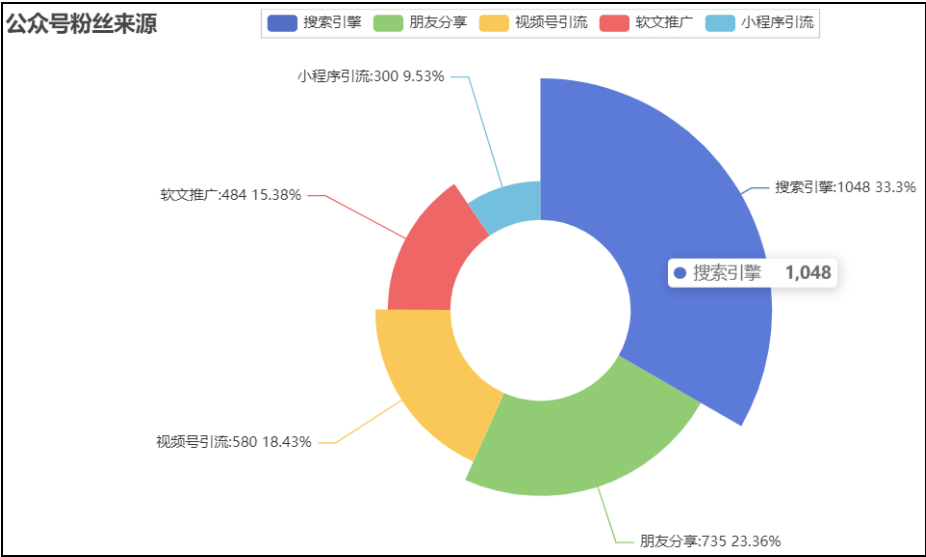


图 3.16 某公众号粉丝来源玫瑰图

3 绘制高级图形

【知识准备】

3.10 绘制散点图

在 pyecharts 库中，可使用 Scatter 类绘制散点图。Scatter 类的基本使用格式如下。



```
class Scatter(init_opts=opts.InitOpts()).add_xaxis(xaxis_data)
.add_yaxis(series_name,y_axis,is_selected=True,xaxis_index=None,yaxis_index=None,color=None,symbol
=None,symbol_size=10,symbol_rotate=None,label_opts=opts.LabelOpts(position='right'),markpoint_opts=None,
markline_opts=None,markarea_opts=None,tooltip_opts=None,itemstyle_opts=None,encode=None)
.set_series_opts().set_global_opts()
```

参数的含义如表 3.15 所示。

表 3.15 散点图 Scatter 类参数

参数	说明
symbol	标记的图形，可选的标记类型包括 cirde、rect、roundrect、triangle、diamond、pin、arrow、None。默认为 None
symbol_size	标记的大小，可以设置成单一的数字，如 10；也可以用数组分开表示宽和高，例如，[20, 10]表示标记宽为 20，高为 10。默认为 10
symbol_rotate	标记的旋转角度。默认为 None

例 3.12 绘制散点图，展示不同气温下的销售额情况，示例代码如下。

```
from pyecharts.charts import Scatter
from pyecharts import options as opts
```

```

scatter_chart = Scatter(init_opts=opts.InitOpts(width="70%", height="400px")) # 设置图形大小
scatter_chart.set_global_opts(
    title_opts=opts.TitleOpts(title="不同气温下的销售额分析"),
    axis_opts=opts.AxisOpts(splitline_opts=opts.SplitLineOpts(is_show=True)),
    yaxis_opts=opts.AxisOpts(splitline_opts=opts.SplitLineOpts(is_show=True),
        name="销售额（万元）", name_location="center", name_gap=40))
x_data = ["5°", "10°", "15°", "20°", "25°", "27°", "30°"] #X 轴温度
y_data = [3.5, 5.3, 7.8, 10.9, 13.5, 20.1, 40.7] #Y 轴是销售额（万元）
scatter_chart.add_xaxis(x_data)
scatter_chart.add_yaxis("销售额", y_data,
    symbol_size=8, # 设置点的大小
    symbol='triangle', # 设置点的形状为三角形
    color='red' # 设置点的颜色为红色
)
scatter_chart.render("例 3.13.html")

```

运行程序，效果如图 3.21 所示。

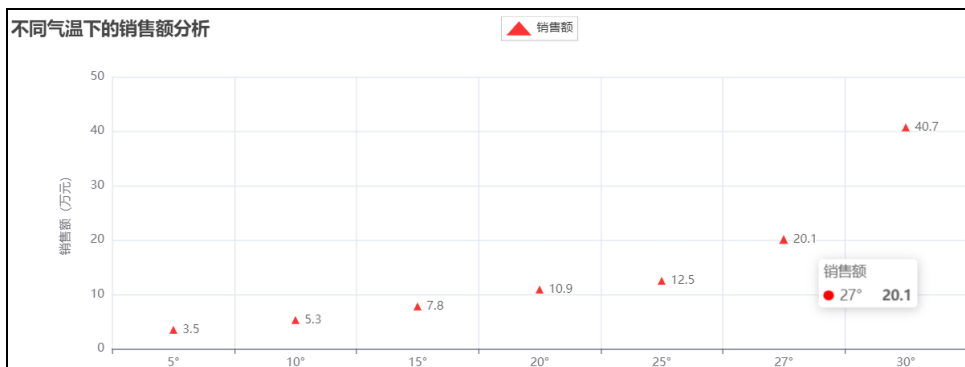


图 3.21 不同气温下的销售额分析散点图

例 3.13 绘制带涟漪的散点图，展示不同气温下的销售额情况，示例代码如下。

```

from pyecharts import options as opts
from pyecharts.charts import EffectScatter
scatter_chart = EffectScatter(init_opts=opts.InitOpts(width="70%", height="400px")) # 设置图形大小
scatter_chart.set_global_opts(
    title_opts=opts.TitleOpts(title="不同气温下的销售额分析", pos_bottom="bottom", pos_left="center"),
    axis_opts=opts.AxisOpts(splitline_opts=opts.SplitLineOpts(is_show=True)),
    yaxis_opts=opts.AxisOpts(splitline_opts=opts.SplitLineOpts(is_show=True),
        name="销售额（万元）", name_location="center", name_gap=40)
)
x_data = ["5°", "10°", "15°", "20°", "25°", "27°", "30°"]
y_data = [3.5, 5.3, 7.8, 10.9, 13.5, 20.1, 40.7]
scatter_chart.add_xaxis(x_data)

```

```
scatter_chart.add_y_axis("销售额", y_data, symbol="pin",
                        label_opts=opts.LabelOpts(position="top", distance=15))
scatter_chart.render("例 3.13.html")
```

运行程序，效果如图 3.22 所示。

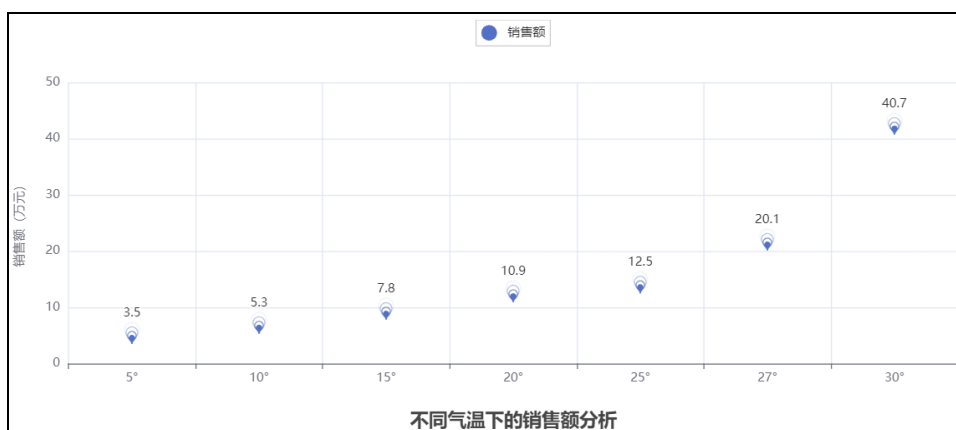


图 3.22 带涟漪的散点图

3.11 绘制箱线图

箱线图（Box-plot）又称为盒须图、盒式图或箱型图，是一种用作显示一组数据分散情况资料的统计图。因形状如箱子而得名，它主要用于反映原始数据分布的特征，还可以进行多组数据分布特征的比较。箱线图的绘制方法是：先找出一组数据的上边缘、下边缘、中位数和两个四分位数；然后，连接两个四分位数画出箱体；再将上边缘和下边缘与箱体相连接，中位数在箱体中间，如图 3.23 所示。

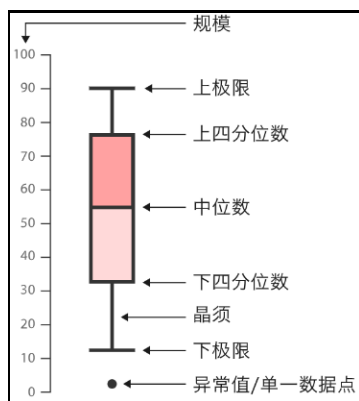


图 3.23 箱线图

在 pyecharts 库中，可使用 Boxplot 类绘制箱线图。Boxplot 类的基本使用格式如下。

```
class Boxplot(init_opts=opts.InitOpts()).add_axis(xaxis_data)
.add_y_axis(series_name, y_axis, is_selected=True, xaxis_index=None, yaxis_index=None, label_opts=opts.LabelOpts(), markpoint_opts=opts.MarkPointOpts(), markline_opts=opts.MarkLineOpts(), tooltip_opts=None, itemstyle_opts=None).set_series_opts().set_global_opts()
```


参数的含义，参考其他直角坐标系类图表说明。

例 3.14 绘制箱线图，展示大一新生 1 班，2 班，3 班，4 班的数学考试成绩，示例代码如下。

```
from pyecharts import options as opts
from pyecharts.charts import Boxplot
data = [ [68,99,46,77,94,40,79,20,88,89,76,92,95], # 1 班数据
         [79,88,35,57,78,69,78,99,75,46,88,87,89], # 2 班数据
         [91,82,63,86,77,78,32,96,80,86,64,67,96], # 3 班数据
         [72,82,45,100,67,89,90,90,89,69,79,91,92]] # 4 班数据
x_axis=["1 班", "2 班","3 班","4 班"]
boxplot =(Boxplot().add_xaxis(x_axis)
            .add_yaxis("数学成绩", Boxplot.prepare_data(data))
            .set_global_opts(title_opts=opts.TitleOpts(title="大一新生四个班数学成绩"),
                              toolbox_opts=opts.ToolboxOpts(is_show=True,feature={"saveAsImage": {}}),
                              tooltip_opts=opts.TooltipOpts(is_show=True,trigger="item",formatter="{a}<br/>{b} : {c}")),
            .set_series_opts(markpoint_opts=opts.MarkPointOpts(data=[{"type": "max", "name": "最大值"},
                              {"type": "min", "name": "最小值"}]),
                              markline_opts=opts.MarkLineOpts(data=[{"type": "average", "name": "平均值"}])))
boxplot.render("例 3.14.html")
```

运行程序，效果如图 3.24 所示。

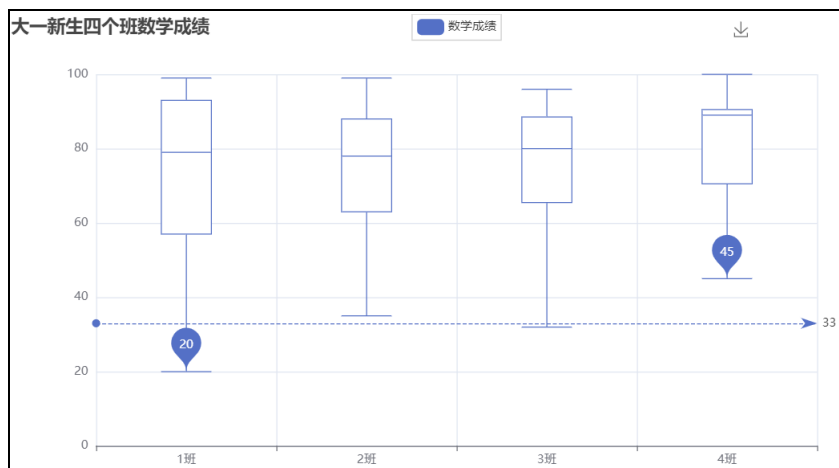


图 3.24 数学考试成绩分析箱线图

3.12 绘制雷达图

在 pyecharts 库中，可使用 Radar 类绘制散点图。Radar 类的基本使用格式如下。

```
class Radar(init_opts=opts.InitOpts())
```



```
.add_schema(schema,shape,center,radius,start_angle,textstyle_opts,splitline_opt,splitarea_opt,axisline_opt,axisTick_opt,axislabel_opt,radiusaxis_optsangleaxis_optspolar_opts)

.add(series_namedata,is_selected,symbol,color,label_opts,linestyle_opts,areastyle_opts,tooltip_opts,emphasis_opts).set_series_opts().set_global_opts()
```

参数的含义如表 3.16 所示。

表 3.16 雷达图 Radar 类参数

参数	说明
schema	雷达指示器配置项列表
shape	雷达图绘制类型，可选 'polygon' 和 'circle'
center	雷达的中心（圆心）坐标，数组的第一项是横坐标，第二项是纵坐标。支持设置成百分比，设置成百分比时第一项是相对于容器宽度，第二项是相对于容器高度。
radius	雷达的半径。可以是：（1）number：直接指定外半径值。（2）string：例如，'20%'，外半径为可视区尺寸的 20% 长度。（3）Array.<number string>：数组的第一项是内半径，第二项是外半径。每一项遵从上述 number，string 的描述。
start_angle	坐标系起始角度，也就是第一个指示器轴的角度。
splitline_opt	分割线配置项
splitarea_opt	分隔区域配置项
axisline_opt	坐标轴轴线配置项
axisTick_opt	坐标轴刻度相关设置
axislabel_opt	坐标轴刻度标签的相关设置
radiusaxis_opts	极坐标系的径向轴
angleaxis_opts	极坐标系的角度轴
polar_opts	极坐标系配置
data	系列数据项
symbol	标记类型包括 'circle', 'rect', 'roundRect', 'triangle','diamond', 'pin', 'arrow', 'none'

例 3.15 绘制雷达图，展示三年级 1 班，2 班，3 班的语文、数学、英语、物理、化学、历史考试成绩的雷达图，示例代码如下。

```
from pyecharts import options as opts
from pyecharts.charts import Radar
data_1 = [[95, 96, 85, 63, 91, 72]]
data_2 = [[75,81,66,85,88,89]]
data_3 = [[86,76,96,93,67,78]]
ra=(Radar(init_opts=opts.InitOpts())
.add_schema(center=["35%", "50%"],
schema=[opts.RadarIndicatorItem(name="语文", max_=100),
opts.RadarIndicatorItem(name="数学", max_=100),
opts.RadarIndicatorItem(name="英语", max_=100),
```

```

        opts.RadarIndicatorItem(name="物理", max_=100),
        opts.RadarIndicatorItem(name="化学", max_=100),
        opts.RadarIndicatorItem(name="历史", max_=100)],
        splitarea_opt=opts.SplitAreaOpts(is_show=True, areastyle_opts=opts.AreaStyleOpts(opacity=1)),
        textstyle_opts=opts.TextStyleOpts(color="#000"),)
    .add(series_name="1 班",data=data_1,linestyle_opts=opts.LineStyleOpts(color="#CD0000"),)
    .add(series_name="2 班",data=data_2,linestyle_opts=opts.LineStyleOpts(color="#5CACEE"),)
    .add(series_name="3 班",data=data_3,linestyle_opts=opts.LineStyleOpts(color="#0000CD"),)
    .set_series_opts(label_opts=opts.LabelOpts(is_show=False))
    .set_global_opts(title_opts=opts.TitleOpts(title="三年级模拟考试成绩分析"),)
)
ra.render("例 3.15.html")

```

运行程序，效果如图 3.25 所示。

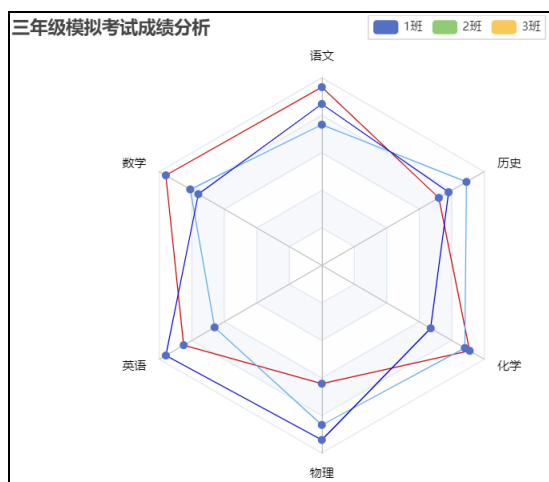


图 3.25 模拟考试成绩分析雷达图

3.14 绘制仪表盘

在 pyecharts 库中，可使用 Gauge 类绘制热力图。Gauge 类的基本使用格式如下。

```

class Gauge(init_opts=opts.InitOpts())
    .add(series_name,data_pair,is_selected,min_,max_,split_number,center,radius,start_angle,end_angle,is_cloc
k_wise,title_label_opts,detail_label_opts,progress,pointer,anchor,tooltip_opts,axisline_opts,axisstick_opts,axislab
el_opts,itemstyle_opts,emphasis_opts)).set_series_opts().set_global_opts()

```

参数的含义，如表 3.18 所示。

表 3.18 仪表盘 Gauge 类参数

参数	说明
data_pair	系列数据项，格式为 [(key1, value1), (key2, value2)]
min_, max_	最小，最大的数据值

split_number	仪表盘平均分割段数
center	仪表盘的圆心（圆心）坐标，数组的第一项是横坐标，第二项是纵坐标。支持设置成百分比，设置百分比时第一项是相对于容器宽度，第二项是相对于容器高度。
start_angle	仪表盘起始角度。圆心 正右侧为 0 度，正上方为 90 度，正左侧为 180 度。
end_angle	仪表盘结束角度。
is_clock_wise	仪表盘刻度是否是顺时针增长
title_label_opts	轮盘内标题文本项标签配置项
detail_label_opts	轮盘内数据项标签配置项
progress	仪表盘进度配置项
pointer	仪表盘指针配置项
anchor	仪表盘表盘中指针的固定点

例 3.17 绘制仪表盘，展示截止到 2024 年 5 月公司销售指标完成率，示例代码如下。

```
from pyecharts import options as opts
from pyecharts.charts import Gauge
ga=(Gauge().add(series_name="业务指标", data_pair=[["完成率",55.5]])
    .set_global_opts(legend_opts=opts.LegendOpts(is_show=False),
        tooltip_opts=opts.TooltipOpts(is_show=True,formatter="{a} <br/>{b} : {c}%"),)
    .set_series_opts(axisline_opts=opts.AxisLineOpts(linestyle_opts=opts.LineStyleOpts(
        color=[[0.3, "#67e0e3"], [0.7, "#37a2da"], [1, "#fd666d"]], width=10))))
ga.render("例 3.17.html")
```

运行程序，效果如图 3.27 所示。

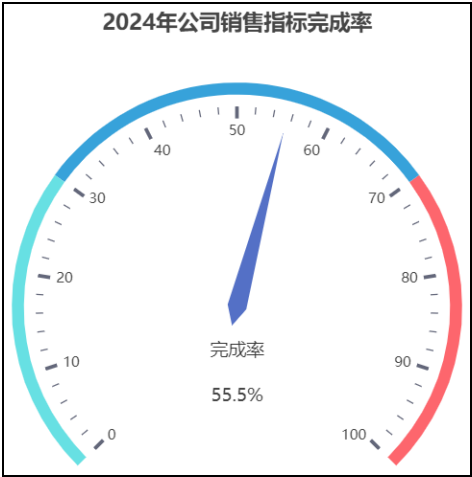


图 3.27 公司销售指标完成率仪表盘

3.15 绘制漏斗图

在 pyecharts 库中，可使用 Funnel 类绘制漏斗图。Funnel 类的基本使用格式如下。

```
class Funnel(init_opts=opts.InitOpts())
```

```
.add(series_name,data_pair,is_selected=True,color=None,sort_='descending',gap=0,label_opts=opts.LabelOpts(),tooltip_opts=None,itemstyle_opts=None).set_series_opts().set_global_opts()
```

参数的含义，如表 3.19 所示。

表 3.19 漏斗图 Funnel 类参数

参数	说明
add()	添加数据
data_pair	接收 Sequence，表示数据项，格式为[(key1, value1), (key2, value2)]。无默认值
sort_	接收 str，表示数据排序，可以取 ascending、descending、None（按 data 顺序）。默认为 descending
gap	接收 numeric，表示数据图形间距。默认为 0

例 3.18 绘制漏斗图，展示某淘宝店铺的订单转化率统计数据，网购环节包括五步，浏览商品（2000 人），加入购物车（900 人），生成订单（400 人），支付订单（320 人），完成交易（280 人）。示例代码如下：

```
from pyecharts import options as opts
from pyecharts.charts import Funnel
funnel = Funnel()
data = [("浏览商品", 2000),("加入购物车", 900),("生成订单", 400),("支付订单", 320),("完成交易", 280)]
funnel.add("购物流程", data)
funnel.set_global_opts(title_opts=opts.TitleOpts(title="购物订单转化率"))
funnel.render("例 3.18.html")
```

运行程序，效果如图 3.28 所示。

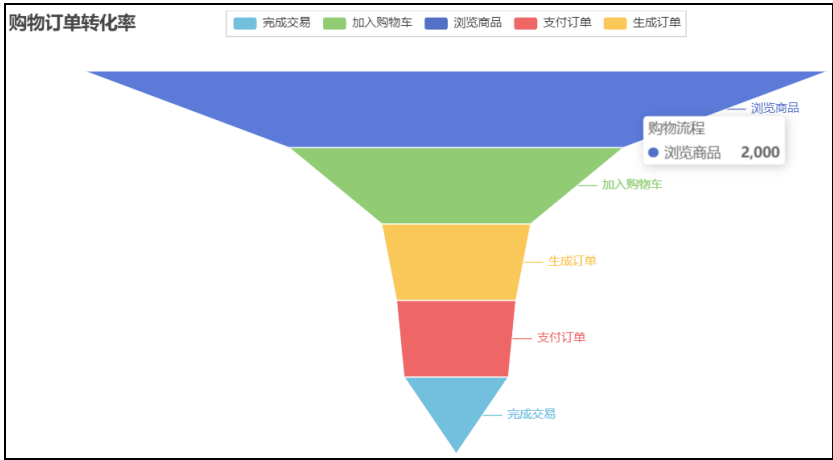


图 3.28 购物流程漏斗图

3.16 绘制地图

在 pyecharts 库中，可使用 Map 类绘制热力图。Map 类的基本使用格式如下。

```
class Map(init_opts=opts.InitOpts())
```

```
.add(series_name,data_pair,maptype,is_selected,is_roam,center,aspect_scale,bounding_coords,min_scale_limit,max_scale_limit,name_property,selected_mode,zoom,name_map,symbol,map_value_calculation,is_map_symbol_show,z_level,z,pos_left,pos_top,pos_right,pos_bottom,geo_index,series_layout_by,dataset_index,layout_center,layout_size,label_opts,tooltip_opts,itemstyle_opts,emphasis_label_opts,emphasis_itemstyle_opts).add(series_name,data_pair,is_selected,min_,max_,split_number,center,radius,start_angle,end_angle,is_clock_wise,title_label_opts,detail_label_opts,progress,pointer,anchor,tooltip_opts,axisline_opts,axistick_opts,axislabel_opts,itemstyle_opts,emphasis_opts)).set_series_opts().set_global_opts()
```

参数的含义，如表 3.20 所示。

表 3.20 地图 Map 类参数

参数	说明
data_pair	数据项（坐标点名称，坐标点值）
maptype	地图类型
is_roam	是否开启鼠标缩放和平移漫游
center	当前视角的中心点，用经纬度表示
aspect_scale	参数用于 scale 地图的长宽比
bounding_coords	二维数组，定义定位的左上角以及右下角分别所对应的经纬度
min_scale_limit	最小的缩放值
max_scale_limit	最大的缩放值
name_property	默认是 'name'，针对 GeoJSON 要素的自定义属性名称，作为主键用于关联数据点和 GeoJSON 地理要素。
selected_mode	选中模式，表示是否支持多个选中，默认关闭，支持布尔值和字符串。字符串取值可选'single'表示单选，或者'multiple'表示多选。
name_map	自定义地区的名称映射
map_value_calculation	多个拥有相同地图类型的系列会使用同一个地图展现。这个配置项就是用于配置统计的方式，目前有：'sum' 取和。'average' 取平均值。'max' 取最大值。'min' 取最小值。
geo_index	默认情况下，map 会自己生成内部专用的 geo 组件。
layout_center	pyecharts 暂时没有提供 left/top/right/bottom 的配置。Layout_center 和 layout_size 提供了除了 left/right/top/bottom/width/height 之外的布局手段。可以通过 layout_center 属性定义地图中心在屏幕中的位置，
layout_size	地图的大小，支持相对于屏幕宽高的百分比或者绝对的像素大小

例 3.19 绘制地图，展示邢台地区某网站某天的点击量，示例代码如下。

```
from pyecharts.charts import Map
from pyecharts import options as opts
data_map=[['桥西区',120],['桥东区',106],['邢台县',84],['南和县',76],['任县',66],['内丘县',69],
          ['临城县',61],['隆尧县',50],['柏乡县',56],['宁晋县',53],['巨鹿县',49],['平乡县',40],
          ['新河县',38],['广宗县',27],['威县',30],['临西县',46],['清河县',28],['沙河市',41],['南宫市',60]]
```

```
map_demo=(Map().add('网站某天点击量',data_map,'邢台')
        .set_global_opts(title_opts=opts.TitleOpts(title="邢台市"),visualmap_opts=opts.VisualMapOpts()))
map_demo.render('例 3.19.html')
```

运行程序，效果如图 3.29 所示。

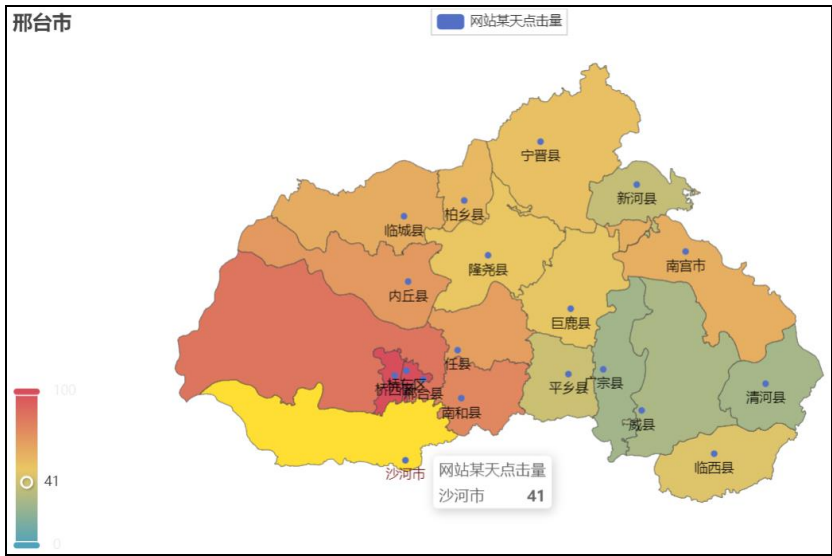


图 3.29 某网站某天的点击量地图

3.17 绘制词云图

在 pyecharts 库中，可使用 WordCloud 类绘制词云图。WordCloud 类的基本使用格式如下。

```
class WordCloud(init_opts=opts.InitOpts())
    .add(series_name,data_pair,shape='circle',mask_image=None,word_gap=20,word_size_range=None,rotate_
step=45,pos_left=None,pos_top=None,pos_right=None,pos_bottom=None,width=None,height=None,is_draw_o
ut_of_bound=False,tooltip_opts=None,textstyle_opts=None,emphasis_shadow_blur=None,emphasis_shadow_co
lor=None).set_series_opts().set_global_opts()
```

参数的含义，如表 3.21 所示。

表 3.21 词云图 WordCloud 类参数

参数	说明
data_pair	系列数据项，形如[(word1, count1), (word2, count2)]。无默认值
shape	词云图轮廓，可选 circle、cardioid、diamond、triangle-forward、triangle、pentagon。默认是 circle
mask_image	自定义的图片（目前支持 jpg、jpeg、png、ico 的格式）。默认为 None
word_gap	单词间隔。默认为 20
word_size_range	单词字体大小范围。默认为 None
rotate_step	旋转单词角度。默认为 45

pos_left	接收 str，表示距离左侧的距离。默认为 None
pos_top	接收 str，表示距离顶部的距离。默认为 None
pos_right	接收 str，表示距离右侧的距离。默认为 None
pos_bottom	接收 str，表示距离底部的距离。默认为 None
is_draw_out_of_bound	接收 bool，表示是否允许词云图的数据展示在画布范围之外。默认为 False

例 3.20 绘制词云图，展示有关旅游的词语和诗句，示例代码如下：

```
from pyecharts.charts import WordCloud
from pyecharts import options as opts
words = [("如诗如画", 43), ("沾衣欲湿杏花雨，吹面不寒杨柳风", 15), ("山清水秀", 438),
          ("云山雾罩", 957), ("黄河远上白云间，一片孤城万仞山", 927),
          ("不识庐山真面目，只缘身在此山中", 908),
          ("飞流直下三千尺，疑是银河落九天", 693),
          ("羌笛何须怨杨柳，春风不度玉门关", 611),
          ("芳原绿野恣行事，春入遥山碧四围", 512),
          ("湖光潋滟柳条青，古塔明珠景色亭。", 20),
          ("秋高气爽", 16), ("跋山涉川", 10), ("景色如画", 789), ("曲径通幽", 316),
          ("梨花风起正清明，游子寻春半出城", 303),
          ("名胜古迹", 196), ("望梅止渴", 93), ("极目远眺", 47), ("峰峦叠翠", 90),
          ("日出江花红胜火，春来江水绿如蓝", 32)]
wd = WordCloud()
wd.add("游园", words, word_size_range=[12, 55])
wd.set_global_opts(title_opts=opts.TitleOpts(title=""))
wd.render("例 3.20.html")
```

运行程序，效果如图 3.30 所示。



图 3.30 词云图