


水球图

```
from pyecharts import options as opts # type: ignore
from pyecharts.charts import Liquid

c = (
    Liquid()
    .add("lq", [0.6, 0.7]) # 0.6 表示数字 0.7表示水的淹没范围
    .set_global_opts(title_opts=opts.TitleOpts(title="Liquid-基本示例"))
    .render_notebook()
)

c
```



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

c = (
    Liquid()
    .add("lq", [0.3, 0.7], is_outline_show=False, shape='roundRect')
    .set_global_opts(title_opts=opts.TitleOpts(title="Liquid-Shape-arrow"))
    .render_notebook()
)

c
```



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

c = (
    Liquid()
    .add("lq", [0.6, 0.6], is_outline_show=False)
    .set_global_opts(title_opts=opts.TitleOpts(title="Liquid-无边框"))
    .render_notebook()
)

c
```



```
# 漏斗图
```

```
from pyecharts import options as opts
from pyecharts.charts import Funnel
from pyecharts.faker import Faker

c = (
    Funnel()
    .add("商品", [list(z) for z in zip(Faker.choose(), Faker.values())])
    .set_global_opts(title_opts=opts.TitleOpts(title="Funnel-基本示例"))
    .render_notebook()
)

c
```



```
# Funnel - Funnel_label_inside
```

```
from pyecharts import options as opts
from pyecharts.charts import Funnel
from pyecharts.faker import Faker

c = (
    Funnel()
    .add(
        "商品",
        [list(z) for z in zip(Faker.choose(), Faker.values())],
        label_opts=opts.LabelOpts(position="inside"),
    )
    .set_global_opts(title_opts=opts.TitleOpts(title="Funnel-Label (inside)"))
    .render_notebook()
)

c
```



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

c = (
    Gauge()
    .add("ddd", [("完成率", 6.6)],
        end_angle = -45,
        detail_label_opts = opts.GaugeDetailOpts(
            formatter="{value}%",
            border_color = 'red',
            border_width = 1,
            font_size = 30,
            offset_center= [0, "40%"]
        ),
    )
    .set_global_opts(title_opts=opts.TitleOpts(title="Gauge-基本示例"))
    .render_notebook()
)
```

c



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

,

```

c = (
    Gauge()
    .add(
        "业务指标",
        [("完成率", 55.5)],
        split_number= 5, # 分割段数
        axisline_opts=opts.AxisLineOpts(
            linestyle_opts=opts.LineStyleOpts(
                color=[(0.3, "#67e0e3"), (0.7, "#37a2da"), (1, "#fd666d")], width=30
            )
        ),
        detail_label_opts = opts.GaugeDetailOpts(
            formatter="{value}%",
            border_color = 'red',
            border_width = 1,
            font_size = 30,
            offset_center= [0, "40%"]
        )
    )
    .set_global_opts(
        title_opts=opts.TitleOpts(title="Gauge-不同颜色"),
        legend_opts=opts.LegendOpts(is_show=False),
    )
    .render_notebook()
)
c

```



```
# 日历图
```

```
import datetime
import random
```

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

```
begin = datetime.date(2017, 1, 1)
end = datetime.date(2017, 12, 31)
data = [
    [str(begin + datetime.timedelta(days=i)), random.randint(1000, 25000)]
    for i in range((end - begin).days + 1)]
```

```
c = (
    Calendar()
    .add("", data, calendar_opts=opts.CalendarOpts(range_="2017"))
    .set_global_opts(
        title_opts=opts.TitleOpts(title="Calendar-2017年微信步数情况"),
        visualmap_opts=opts.VisualMapOpts(
            max_=20000,
            min_=500,
            orient="horizontal",
            is_piecewise=True, # # 是否为分段型
            pos_top="230px",
            pos_left="100px",
        ),
    )
    .render_notebook()
)
```

```
print(data)
```

```
c
```

```
[[['2017-01-01', 9225], ['2017-01-02', 17428], ['2017-01-03', 14104], ['2017-01-04', 4744], ['2017-01-05', 12125], ['2017-01-06', 14104], ['2017-01-07', 17428], ['2017-01-08', 14104], ['2017-01-09', 17428], ['2017-01-10', 14104], ['2017-01-11', 17428], ['2017-01-12', 14104], ['2017-01-13', 17428], ['2017-01-14', 14104], ['2017-01-15', 17428], ['2017-01-16', 14104], ['2017-01-17', 17428], ['2017-01-18', 14104], ['2017-01-19', 17428], ['2017-01-20', 14104], ['2017-01-21', 17428], ['2017-01-22', 14104], ['2017-01-23', 17428], ['2017-01-24', 14104], ['2017-01-25', 17428], ['2017-01-26', 14104], ['2017-01-27', 17428], ['2017-01-28', 14104], ['2017-01-29', 17428], ['2017-01-30', 14104], ['2017-01-31', 17428], ['2017-02-01', 14104], ['2017-02-02', 17428], ['2017-02-03', 14104], ['2017-02-04', 17428], ['2017-02-05', 14104], ['2017-02-06', 17428], ['2017-02-07', 14104], ['2017-02-08', 17428], ['2017-02-09', 14104], ['2017-02-10', 17428], ['2017-02-11', 14104], ['2017-02-12', 17428], ['2017-02-13', 14104], ['2017-02-14', 17428], ['2017-02-15', 14104], ['2017-02-16', 17428], ['2017-02-17', 14104], ['2017-02-18', 17428], ['2017-02-19', 14104], ['2017-02-20', 17428], ['2017-02-21', 14104], ['2017-02-22', 17428], ['2017-02-23', 14104], ['2017-02-24', 17428], ['2017-02-25', 14104], ['2017-02-26', 17428], ['2017-02-27', 14104], ['2017-02-28', 17428], ['2017-03-01', 14104], ['2017-03-02', 17428], ['2017-03-03', 14104], ['2017-03-04', 17428], ['2017-03-05', 14104], ['2017-03-06', 17428], ['2017-03-07', 14104], ['2017-03-08', 17428], ['2017-03-09', 14104], ['2017-03-10', 17428], ['2017-03-11', 14104], ['2017-03-12', 17428], ['2017-03-13', 14104], ['2017-03-14', 17428], ['2017-03-15', 14104], ['2017-03-16', 17428], ['2017-03-17', 14104], ['2017-03-18', 17428], ['2017-03-19', 14104], ['2017-03-20', 17428], ['2017-03-21', 14104], ['2017-03-22', 17428], ['2017-03-23', 14104], ['2017-03-24', 17428], ['2017-03-25', 14104], ['2017-03-26', 17428], ['2017-03-27', 14104], ['2017-03-28', 17428], ['2017-03-29', 14104], ['2017-03-30', 17428], ['2017-03-31', 14104], ['2017-04-01', 17428], ['2017-04-02', 14104], ['2017-04-03', 17428], ['2017-04-04', 14104], ['2017-04-05', 17428], ['2017-04-06', 14104], ['2017-04-07', 17428], ['2017-04-08', 14104], ['2017-04-09', 17428], ['2017-04-10', 14104], ['2017-04-11', 17428], ['2017-04-12', 14104], ['2017-04-13', 17428], ['2017-04-14', 14104], ['2017-04-15', 17428], ['2017-04-16', 14104], ['2017-04-17', 17428], ['2017-04-18', 14104], ['2017-04-19', 17428], ['2017-04-20', 14104], ['2017-04-21', 17428], ['2017-04-22', 14104], ['2017-04-23', 17428], ['2017-04-24', 14104], ['2017-04-25', 17428], ['2017-04-26', 14104], ['2017-04-27', 17428], ['2017-04-28', 14104], ['2017-04-29', 17428], ['2017-04-30', 14104], ['2017-05-01', 17428], ['2017-05-02', 14104], ['2017-05-03', 17428], ['2017-05-04', 14104], ['2017-05-05', 17428], ['2017-05-06', 14104], ['2017-05-07', 17428], ['2017-05-08', 14104], ['2017-05-09', 17428], ['2017-05-10', 14104], ['2017-05-11', 17428], ['2017-05-12', 14104], ['2017-05-13', 17428], ['2017-05-14', 14104], ['2017-05-15', 17428], ['2017-05-16', 14104], ['2017-05-17', 17428], ['2017-05-18', 14104], ['2017-05-19', 17428], ['2017-05-20', 14104], ['2017-05-21', 17428], ['2017-05-22', 14104], ['2017-05-23', 17428], ['2017-05-24', 14104], ['2017-05-25', 17428], ['2017-05-26', 14104], ['2017-05-27', 17428], ['2017-05-28', 14104], ['2017-05-29', 17428], ['2017-05-30', 14104], ['2017-05-31', 17428], ['2017-06-01', 14104], ['2017-06-02', 17428], ['2017-06-03', 14104], ['2017-06-04', 17428], ['2017-06-05', 14104], ['2017-06-06', 17428], ['2017-06-07', 14104], ['2017-06-08', 17428], ['2017-06-09', 14104], ['2017-06-10', 17428], ['2017-06-11', 14104], ['2017-06-12', 17428], ['2017-06-13', 14104], ['2017-06-14', 17428], ['2017-06-15', 14104], ['2017-06-16', 17428], ['2017-06-17', 14104], ['2017-06-18', 17428], ['2017-06-19', 14104], ['2017-06-20', 17428], ['2017-06-21', 14104], ['2017-06-22', 17428], ['2017-06-23', 14104], ['2017-06-24', 17428], ['2017-06-25', 14104], ['2017-06-26', 17428], ['2017-06-27', 14104], ['2017-06-28', 17428], ['2017-06-29', 14104], ['2017-06-30', 17428], ['2017-07-01', 14104], ['2017-07-02', 17428], ['2017-07-03', 14104], ['2017-07-04', 17428], ['2017-07-05', 14104], ['2017-07-06', 17428], ['2017-07-07', 14104], ['2017-07-08', 17428], ['2017-07-09', 14104], ['2017-07-10', 17428], ['2017-07-11', 14104], ['2017-07-12', 17428], ['2017-07-13', 14104], ['2017-07-14', 17428], ['2017-07-15', 14104], ['2017-07-16', 17428], ['2017-07-17', 14104], ['2017-07-18', 17428], ['2017-07-19', 14104], ['2017-07-20', 17428], ['2017-07-21', 14104], ['2017-07-22', 17428], ['2017-07-23', 14104], ['2017-07-24', 17428], ['2017-07-25', 14104], ['2017-07-26', 17428], ['2017-07-27', 14104], ['2017-07-28', 17428], ['2017-07-29', 14104], ['2017-07-30', 17428], ['2017-07-31', 14104], ['2017-08-01', 17428], ['2017-08-02', 14104], ['2017-08-03', 17428], ['2017-08-04', 14104], ['2017-08-05', 17428], ['2017-08-06', 14104], ['2017-08-07', 17428], ['2017-08-08', 14104], ['2017-08-09', 17428], ['2017-08-10', 14104], ['2017-08-11', 17428], ['2017-08-12', 14104], ['2017-08-13', 17428], ['2017-08-14', 14104], ['2017-08-15', 17428], ['2017-08-16', 14104], ['2017-08-17', 17428], ['2017-08-18', 14104], ['2017-08-19', 17428], ['2017-08-20', 14104], ['2017-08-21', 17428], ['2017-08-22', 14104], ['2017-08-23', 17428], ['2017-08-24', 14104], ['2017-08-25', 17428], ['2017-08-26', 14104], ['2017-08-27', 17428], ['2017-08-28', 14104], ['2017-08-29', 17428], ['2017-08-30', 14104], ['2017-08-31', 17428], ['2017-09-01', 14104], ['2017-09-02', 17428], ['2017-09-03', 14104], ['2017-09-04', 17428], ['2017-09-05', 14104], ['2017-09-06', 17428], ['2017-09-07', 14104], ['2017-09-08', 17428], ['2017-09-09', 14104], ['2017-09-10', 17428], ['2017-09-11', 14104], ['2017-09-12', 17428], ['2017-09-13', 14104], ['2017-09-14', 17428], ['2017-09-15', 14104], ['2017-09-16', 17428], ['2017-09-17', 14104], ['2017-09-18', 17428], ['2017-09-19', 14104], ['2017-09-20', 17428], ['2017-09-21', 14104], ['2017-09-22', 17428], ['2017-09-23', 14104], ['2017-09-24', 17428], ['2017-09-25', 14104], ['2017-09-26', 17428], ['2017-09-27', 14104], ['2017-09-28', 17428], ['2017-09-29', 14104], ['2017-09-30', 17428], ['2017-10-01', 14104], ['2017-10-02', 17428], ['2017-10-03', 14104], ['2017-10-04', 17428], ['2017-10-05', 14104], ['2017-10-06', 17428], ['2017-10-07', 14104], ['2017-10-08', 17428], ['2017-10-09', 14104], ['2017-10-10', 17428], ['2017-10-11', 14104], ['2017-10-12', 17428], ['2017-10-13', 14104], ['2017-10-14', 17428], ['2017-10-15', 14104], ['2017-10-16', 17428], ['2017-10-17', 14104], ['2017-10-18', 17428], ['2017-10-19', 14104], ['2017-10-20', 17428], ['2017-10-21', 14104], ['2017-10-22', 17428], ['2017-10-23', 14104], ['2017-10-24', 17428], ['2017-10-25', 14104], ['2017-10-26', 17428], ['2017-10-27', 14104], ['2017-10-28', 17428], ['2017-10-29', 14104], ['2017-10-30', 17428], ['2017-10-31', 14104], ['2017-11-01', 17428], ['2017-11-02', 14104], ['2017-11-03', 17428], ['2017-11-04', 14104], ['2017-11-05', 17428], ['2017-11-06', 14104], ['2017-11-07', 17428], ['2017-11-08', 14104], ['2017-11-09', 17428], ['2017-11-10', 14104], ['2017-11-11', 17428], ['2017-11-12', 14104], ['2017-11-13', 17428], ['2017-11-14', 14104], ['2017-11-15', 17428], ['2017-11-16', 14104], ['2017-11-17', 17428], ['2017-11-18', 14104], ['2017-11-19', 17428], ['2017-11-20', 14104], ['2017-11-21', 17428], ['2017-11-22', 14104], ['2017-11-23', 17428], ['2017-11-24', 14104], ['2017-11-25', 17428], ['2017-11-26', 14104], ['2017-11-27', 17428], ['2017-11-28', 14104], ['2017-11-29', 17428], ['2017-11-30', 14104], ['2017-12-01', 17428], ['2017-12-02', 14104], ['2017-12-03', 17428], ['2017-12-04', 14104], ['2017-12-05', 17428], ['2017-12-06', 14104], ['2017-12-07', 17428], ['2017-12-08', 14104], ['2017-12-09', 17428], ['2017-12-10', 14104], ['2017-12-11', 17428], ['2017-12-12', 14104], ['2017-12-13', 17428], ['2017-12-14', 14104], ['2017-12-15', 17428], ['2017-12-16', 14104], ['2017-12-17', 17428], ['2017-12-18', 14104], ['2017-12-19', 17428], ['2017-12-20', 14104], ['2017-12-21', 17428], ['2017-12-22', 14104], ['2017-12-23', 17428], ['2017-12-24', 14104], ['2017-12-25', 17428], ['2017-12-26', 14104], ['2017-12-27', 17428], ['2017-12-28', 14104], ['2017-12-29', 17428], ['2017-12-30', 14104], ['2017-12-31', 17428]]]
```

```
# 饼图
```

```
from pyecharts import options as opts
from pyecharts.charts import Pie
from pyecharts.faker import Faker
```

```
c = (
    Pie()
    .add("", [list(z) for z in zip(Faker.choose(), Faker.values())])
    .set_global_opts(title_opts=opts.TitleOpts(title="Pie-基本示例"))
    .set_series_opts(label_opts=opts.LabelOpts(formatter="{b}: {c}"))
    .render_notebook()
)
```

```
print([list(z) for z in zip(Faker.choose(), Faker.values())])
```

```
c
```

```
[['衬衫', 118], ['毛衣', 109], ['领带', 70], ['裤子', 76], ['风衣', 81], ['高跟鞋', 80], ['袜
```




```
# 设置饼图颜色
```

```
from pyecharts import options as opts
from pyecharts.charts import Pie
from pyecharts.faker import Faker
```

```
c = (
    Pie()
    .add("", [list(z) for z in zip(Faker.choose(), Faker.values())])
    .set_colors(["blue", "green", "yellow", "red", "pink", "orange", "purple"])
    .set_global_opts(title_opts=opts.TitleOpts(title="Pie-设置颜色"))
    .set_series_opts(label_opts=opts.LabelOpts(formatter="{b}: {c}"))
    .render_notebook()
)
```

```
c
```



```
from pyecharts import options as opts
from pyecharts.charts import Pie
from pyecharts.faker import Faker

c = (
    Pie()
    .add(
        "",
        [list(z) for z in zip(Faker.choose(), Faker.values())],
        radius=["40%", "75%"],
    )
    .set_global_opts(
        title_opts=opts.TitleOpts(title="Pie-Radius"),
        legend_opts=opts.LegendOpts(orient="vertical", pos_top="15%", pos_left="2%"),
    )
    .set_series_opts(label_opts=opts.LabelOpts(formatter="{b}: {c}"))
    .render_notebook()
)

c
```




```

# 定制饼图
import pyecharts.options as opts
from pyecharts.charts import Pie

x_data = ["直接访问", "邮件营销", "联盟广告", "视频广告", "搜索引擎"]
y_data = [335, 310, 274, 235, 400]
data_pair = [list(z) for z in zip(x_data, y_data)]
data_pair.sort(key=lambda x: x[1]) # 根据 列表的第二个索引进行排序

c = (
    Pie(init_opts=opts.InitOpts(width="800px", height="400px", bg_color="#2c343c"))
    .add(
        series_name="访问来源",
        data_pair=data_pair,
        rosetype="radius",
        radius="55%",
        center=["50%", "50%"],
    )
    .set_global_opts(
        title_opts=opts.TitleOpts(
            title="Customized Pie",
            pos_left="center",
            pos_top="20",
            title_textstyle_opts=opts.TextStyleOpts(color="yellow"), # #fff
        ),
        legend_opts=opts.LegendOpts(is_show=False),
    )
    .set_series_opts(
        tooltip_opts=opts.TooltipOpts(
            trigger="item", formatter="{a} <br/> {b}: {c} ({d}%)" # {a}: 系列名。# {b}: 数据名。# {c}: 数据值。
        ),
        label_opts=opts.LabelOpts(color="white"), # rgba(255, 255, 255, 0.3) grey
    )
    .render_notebook()
)
print(data_pair)

```

c

```

[['视频广告', 235], ['联盟广告', 274], ['邮件营销', 310], ['直接访问', 335], ['搜索引擎', 400]]

```



```

import pycharts.options as opts
from pycharts.charts import Pie

x_data = ["直接访问", "邮件营销", "联盟广告", "视频广告", "搜索引擎"]
y_data = [335, 310, 234, 135, 1548]

c = (
    Pie(init_opts=opts.InitOpts(width="800px", height="400px"))
    .add(
        series_name="访问来源",
        data_pair=[list(z) for z in zip(x_data, y_data)],
        radius=["50%", "70%"], # 内半径和外半径
        label_opts=opts.LabelOpts(is_show=False, position="center"),
    )
    .set_global_opts(legend_opts=opts.LegendOpts(pos_left="left", orient="vertical"))
    .set_series_opts(
        tooltip_opts=opts.TooltipOpts(
            trigger="item", formatter="{a} <br/> {b}: {c} ({d}%)"
        ),
        # label_opts=opts.LabelOpts(formatter="{b}: {c}")
    )
    .render_notebook()
)
c

```



```

# Pie_rosetype 玫瑰图

from pyecharts import options as opts
from pyecharts.charts import Pie
from pyecharts.faker import Faker

v = Faker.choose()
c = (
    Pie()
    .add(
        "",
        [list(z) for z in zip(v, Faker.values())],
        radius=["30%", "75%"],
        center=["25%", "50%"],
        rosetype="radius",
        label_opts=opts.LabelOpts(is_show=False),
    )
    .add(
        "",
        [list(z) for z in zip(v, Faker.values())],
        radius=["30%", "75%"],
        center=["75%", "50%"],
        rosetype="area",
    )
    .set_global_opts(title_opts=opts.TitleOpts(title="Pie-玫瑰图示例"))
    .render_notebook()
)

c

```



```

# Effectscatter - Effectscatter_base
from pyecharts import options as opts
from pyecharts.charts import EffectScatter
from pyecharts.faker import Faker

c = (
    EffectScatter()
    .add_xaxis(Faker.choose())
    .add_yaxis("", Faker.values())
    .set_global_opts(title_opts=opts.TitleOpts(title="EffectScatter-基本示例"))
    .render_notebook()
)

c

```



```
from pyecharts import options as opts
from pyecharts.charts import EffectScatter
from pyecharts.faker import Faker

c = (
    EffectScatter()
    .add_xaxis(Faker.choose())
    .add_yaxis("", Faker.values(), symbol='arrow') # circle line arrow
    .set_global_opts(title_opts=opts.TitleOpts(title="EffectScatter-不同Symbol"))
    .render_notebook()
)

c
```



```

from pyecharts import options as opts
from pyecharts.charts import EffectScatter
from pyecharts.faker import Faker

c = (
    EffectScatter()
    .add_xaxis(Faker.choose())
    .add_yaxis(" ", Faker.values())
    .set_global_opts(
        title_opts=opts.TitleOpts(title="EffectScatter-显示分割线"),
        xaxis_opts=opts.AxisOpts(splitline_opts=opts.SplitLineOpts(is_show=True)),
        yaxis_opts=opts.AxisOpts(splitline_opts=opts.SplitLineOpts(is_show=True)),
    )
    .render_notebook()
)

c

```



箱型图

```

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

v1 = [
    [850, 740, 900, 1070, 930, 850, 950, 980, 980, 880, 1000, 980],
    [960, 940, 960, 940, 880, 800, 850, 880, 900, 840, 830, 790],
]

v2 = [
    [890, 810, 810, 820, 800, 770, 760, 740, 750, 760, 910, 920],
    [890, 840, 780, 810, 760, 810, 790, 810, 820, 850, 870, 870],
]

c = Boxplot()
c.add_xaxis(["expr1", "expr2"])
c.add_yaxis("A", c.prepare_data(v1))
c.add_yaxis("B", c.prepare_data(v2))
c.set_global_opts(title_opts=opts.TitleOpts(title="BoxPlot-基本示例"))
c.render_notebook()
# c.prepare_data(v1)

```



```
import random

from pyecharts import options as opts
from pyecharts.charts import HeatMap
from pyecharts.faker import Faker

value = [[i, j, random.randint(0, 50)] for i in range(24) for j in range(7)]
c = (
    HeatMap()
    .add_xaxis(Faker.clock)
    .add_yaxis("series0", Faker.week, value)
    .set_global_opts(
        title_opts=opts.TitleOpts(title="HeatMap-基本示例"),
        visualmap_opts=opts.VisualMapOpts(),
    )
    .render_notebook()
)

c
```



```

# 象形柱状图
from pyecharts import options as opts
from pyecharts.charts import PictorialBar

location = ["山西", "四川", "西藏", "北京", "上海", "内蒙古", "云南", "黑龙江", "广东", "福建"]
values = [13, 42, 67, 81, 86, 94, 166, 220, 249, 262]

c = (
    PictorialBar()
    .add_xaxis(location)
    .add_yaxis(
        "",
        values,
        label_opts=opts.LabelOpts(is_show=False),
        symbol_size=18,
        symbol_repeat=True,
        is_symbol_clip=True,
    )
    .reversal_axis()
    .set_global_opts(
        title_opts=opts.TitleOpts(title="PictorialBar-各省份人口数量（虚假数据）"),
        xaxis_opts=opts.AxisOpts(is_show=False),
        yaxis_opts=opts.AxisOpts(
            axistick_opts=opts.AxisTickOpts(is_show=False),
            axisline_opts=opts.AxisLineOpts(
                linestyle_opts=opts.LineStyleOpts(opacity=0)
            ),
        ),
    ),
)
c.render_notebook()

```



```
# 散点图 scatter

from pyecharts import options as opts
from pyecharts.charts import Scatter
from pyecharts.faker import Faker

c = (
    Scatter()
    .add_xaxis(Faker.choose())
    .add_yaxis("商家A", Faker.values())
    .set_global_opts(
        title_opts=opts.TitleOpts(title="Scatter-显示分割线"),
        xaxis_opts=opts.AxisOpts(splitline_opts=opts.SplitLineOpts(is_show=True)),
        yaxis_opts=opts.AxisOpts(splitline_opts=opts.SplitLineOpts(is_show=True)),
    )
    .render_notebook()
)

c
```



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

c = (
    Scatter()
    .add_xaxis(Faker.choose())
    .add_yaxis("商家A", Faker.values())
    .set_global_opts(
        title_opts=opts.TitleOpts(title="Scatter-VisualMap(Color)"),
        visualmap_opts=opts.VisualMapOpts(max_=150),
    )
    .render_notebook()
)

c
```




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

c = (
    Scatter()
    .add_xaxis(Faker.choose())
    .add_yaxis("商家A", Faker.values())
    .add_yaxis("商家B", Faker.values())
    .set_global_opts(
        title_opts=opts.TitleOpts(title="Scatter-VisualMap(Size)"),
        visualmap_opts=opts.VisualMapOpts(type_="size", max_=150, min_=20),
    )
    .render_notebook()
)

c
```



```
# 层叠多图
```

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

```
x_data = ["1月", "2月", "3月", "4月", "5月", "6月",
          "7月", "8月", "9月", "10月", "11月", "12月"]
```

```
bar = (
    Bar(init_opts=opts.InitOpts(width="800px", height="400px"))
    .add_xaxis(xaxis_data=x_data)
    .add_yaxis(
        series_name="蒸发量",
        y_axis=[
            2.0,
            4.9,
            7.0,
            23.2,
            25.6,
            76.7,
            135.6,
            162.2,
            32.6,
            20.0,
            6.4,
            3.3,
        ],
        label_opts=opts.LabelOpts(is_show=False),
    )
    .add_yaxis(
        series_name="降水量",
        y_axis=[
            2.6,
            5.9,
            9.0,
            26.4,
            28.7,
            70.7,
            175.6,
            182.2,
            48.7,
            18.8,
            6.0,
            2.3,
        ],
        label_opts=opts.LabelOpts(is_show=False),
    )
    .extend_axis(
        yaxis=opts.AxisOpts(
            name="温度",
            type_="value",
            min_=0,
            max_=25,
            interval=5,
            axislabel_opts=opts.LabelOpts(formatter="{value} °C"),
        )
    )
    .set_global_opts(
        # 提示框设置
        tooltip_opts=opts.TooltipOpts(
            is_show=True, trigger="axis", axis_pointer_type="cross"
        ),
        xaxis_opts=opts.AxisOpts(
            type_="category",
            axispointer_opts=opts.AxisPointerOpts(
                is_show=True, type_="shadow"
            ),
        ),
        yaxis_opts=opts.AxisOpts(
            name="水量",
            type_="value",
            min_=0,
            max_=250,
            interval=50,
            axislabel_opts=opts.LabelOpts(formatter="{value} ml"),
        ),
    )
)
```

```
# 折线图
```

```
line = (
```

```

Line()
.add_xaxis(xaxis_data=x_data)
.add_yaxis(
    series_name="平均温度",
    yaxis_index=1, # 使用的 y 轴的 index
    y_axis=[2.0, 2.2, 3.3, 4.5, 6.3, 10.2,
            20.3, 23.4, 23.0, 16.5, 12.0, 6.2],
    label_opts=opts.LabelOpts(is_show=True),
)
)

bar.overlap(line).render_notebook()

```



```

# 地理图    geography

from pyecharts import options as opts
from pyecharts.charts import Geo
from pyecharts.faker import Faker

c = (
    Geo()
    .add_schema(maptype="china") # pyecharts.datasets.map_filenames.json
    .add("geo", [list(z) for z in zip(Faker.provinces, Faker.values())]) # ['长春市', '吉林市']
    .set_series_opts(label_opts=opts.LabelOpts(is_show=False))
    .set_global_opts(
        visualmap_opts=opts.VisualMapOpts(), title_opts=opts.TitleOpts(title="Geo-基本示例")
    )
    .render_notebook()
)

c

```

```
from pyecharts import options as opts
from pyecharts.charts import Geo
from pyecharts.faker import Faker

c = (
    Geo()
    .add_schema(maptype="china")
    .add(
        "geo",
        [list(z) for z in zip(Faker.provinces, Faker.values())],
        type_='effectScatter', # heatmap
    )
    .set_series_opts(label_opts=opts.LabelOpts(is_show=False))
    .set_global_opts(title_opts=opts.TitleOpts(title="Geo-EffectScatter"))
    .render_notebook()
)

c
```