### Gauge 仪表盘

1、**Gauge——**基本示例

from pyecharts.globals import CurrentConfig

CurrentConfig.ONLINE\_HOST = "http://127.0.0.1:8001/assets/"

from pyecharts import options as opts

from pyecharts.charts import Gauge, Page

def gauge\_base() -> Gauge:

c = (

Gauge()

.add("", [("完成率", 66.6)])

.set\_global\_opts(title\_opts=opts.TitleOpts(title="Gauge-基本示例"))

)

return c

gauge\_base().render\_notebook()



2、**Gauge——**不同颜色

def gauge\_color() -> Gauge:

c = (

Gauge()

.add(

"业务指标",

[("完成率", 55.5)],

axisline\_opts=opts.AxisLineOpts(

linestyle\_opts=opts.LineStyleOpts(

color=[(0.3, "#67e0e3"), (0.7, "#37a2da"), (1, "#fd666d")], width=30

)

),

)

.set\_global\_opts(

title\_opts=opts.TitleOpts(title="Gauge-不同颜色"),

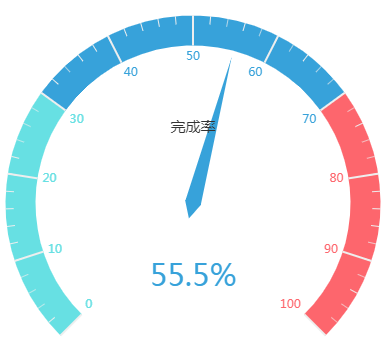
legend\_opts=opts.LegendOpts(is\_show=False),

)

)

return c

gauge\_color().render\_notebook()



**3、Gauge-**分割段数-Label

def gauge\_splitnum\_label() -> Gauge:

c = (

Gauge()

.add(

"业务指标",

[("完成率", 55.5)],

split\_number=5,

axisline\_opts=opts.AxisLineOpts(

linestyle\_opts=opts.LineStyleOpts(

color=[(0.3, "#67e0e3"), (0.7, "#37a2da"), (1, "#fd666d")], width=30

)

),

label\_opts=opts.LabelOpts(formatter="{value}"),

)

.set\_global\_opts(

title\_opts=opts.TitleOpts(title="Gauge-分割段数-Label"),

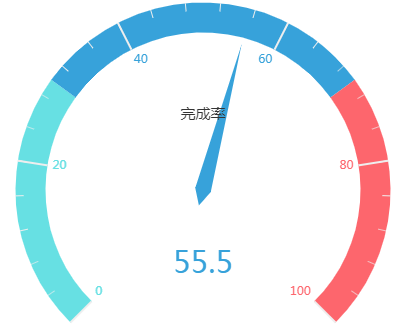
legend\_opts=opts.LegendOpts(is\_show=False),

)

)

return c

gauge\_splitnum\_label().render\_notebook()



### [Graph：关系图](https://pyecharts.org/" \l "/zh-cn/basic_charts?id=graph%ef%bc%9a%e5%85%b3%e7%b3%bb%e5%9b%be)

**1、Graph-基本示例**

import json

import os

from pyecharts import options as opts

from pyecharts.charts import Graph, Page

def graph\_base() -> Graph:

nodes = [

{"name": "结点1", "symbolSize": 10},

{"name": "结点2", "symbolSize": 20},

{"name": "结点3", "symbolSize": 30},

{"name": "结点4", "symbolSize": 40},

{"name": "结点5", "symbolSize": 50},

{"name": "结点6", "symbolSize": 40},

{"name": "结点7", "symbolSize": 30},

{"name": "结点8", "symbolSize": 20},

]

links = []

for i in nodes:

for j in nodes:

links.append({"source": i.get("name"), "target": j.get("name")})

c = (

Graph()

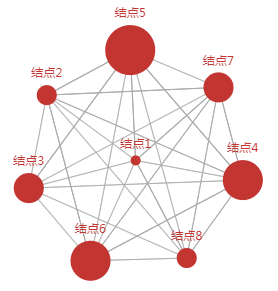
.add("", nodes, links, repulsion=8000)

.set\_global\_opts(title\_opts=opts.TitleOpts(title="Graph-基本示例"))

)

return c

graph\_base().render\_notebook()



**2、Graph-GraphNode-GraphLink**

def graph\_with\_opts() -> Graph:

nodes = [

opts.GraphNode(name="结点1", symbol\_size=10),

opts.GraphNode(name="结点2", symbol\_size=20),

opts.GraphNode(name="结点3", symbol\_size=30),

opts.GraphNode(name="结点4", symbol\_size=40),

opts.GraphNode(name="结点5", symbol\_size=50),

]

links = [

opts.GraphLink(source="结点1", target="结点2"),

opts.GraphLink(source="结点2", target="结点3"),

opts.GraphLink(source="结点3", target="结点4"),

opts.GraphLink(source="结点4", target="结点5"),

opts.GraphLink(source="结点5", target="结点1"),

]

c = (

Graph()

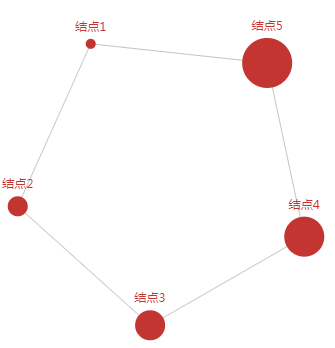
.add("", nodes, links, repulsion=4000)

.set\_global\_opts(title\_opts=opts.TitleOpts(title="Graph-GraphNode-GraphLink"))

)

return c

graph\_with\_opts().render\_notebook()



**3、Graph-微博转发关系图**

def graph\_weibo() -> Graph:

with open(os.path.join("fixtures", "weibo.json"), "r", encoding="utf-8") as f:

j = json.load(f)

nodes, links, categories, cont, mid, userl = j

c = (

Graph()

.add(

"",

nodes,

links,

categories,

repulsion=50,

linestyle\_opts=opts.LineStyleOpts(curve=0.2),

label\_opts=opts.LabelOpts(is\_show=False),

)

.set\_global\_opts(

legend\_opts=opts.LegendOpts(is\_show=False),

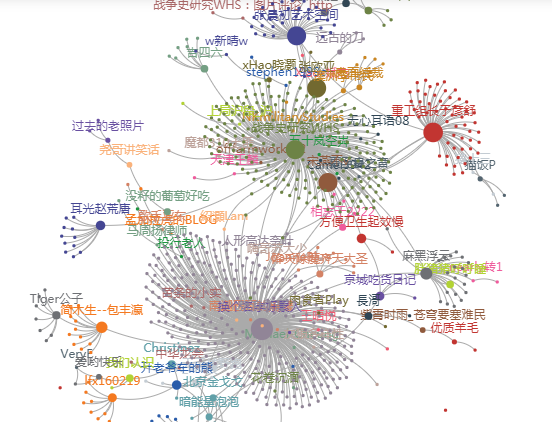
title\_opts=opts.TitleOpts(title="Graph-微博转发关系图"),

)

)

return c

graph\_weibo().render\_notebook()



**4、Graph-les-miserables**

def graph\_les\_miserables():

with open(

os.path.join("fixtures", "les-miserables.json"), "r", encoding="utf-8"

) as f:

j = json.load(f)

nodes = j["nodes"]

links = j["links"]

categories = j["categories"]

c = (

Graph(init\_opts=opts.InitOpts(width="1000px", height="600px"))

.add(

"",

nodes=nodes,

links=links,

categories=categories,

layout="circular",

is\_rotate\_label=True,

linestyle\_opts=opts.LineStyleOpts(color="source", curve=0.3),

label\_opts=opts.LabelOpts(position="right"),

)

.set\_global\_opts(

title\_opts=opts.TitleOpts(title="Graph-Les Miserables"),

legend\_opts=opts.LegendOpts(

orient="vertical", pos\_left="2%", pos\_top="20%"

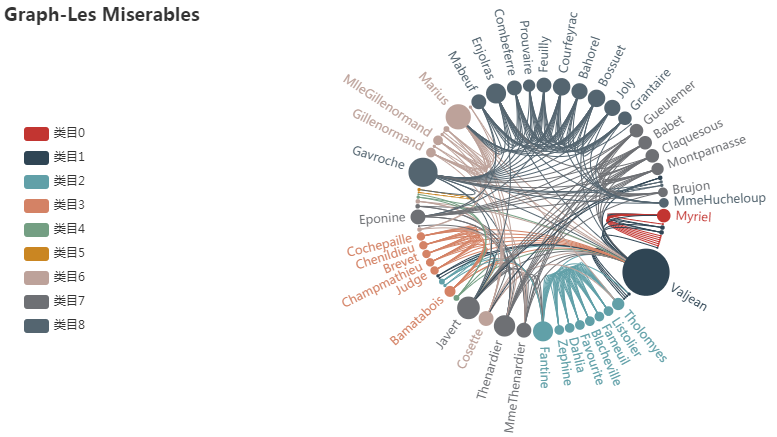
),

)

)

return c

graph\_les\_miserables().render\_notebook()



**5、Graph-NPM Dependencies**

def graph\_npm\_dependencies() -> Graph:

with open(os.path.join("fixtures", "npmdepgraph.json"), "r", encoding="utf-8") as f:

j = json.load(f)

nodes = [

{

"x": node["x"],

"y": node["y"],

"id": node["id"],

"name": node["label"],

"symbolSize": node["size"],

"itemStyle": {"normal": {"color": node["color"]}},

}

for node in j["nodes"]

]

edges = [

{"source": edge["sourceID"], "target": edge["targetID"]} for edge in j["edges"]

]

c = (

Graph(init\_opts=opts.InitOpts(width="1000px", height="600px"))

.add(

"",

nodes=nodes,

links=edges,

layout="none",

label\_opts=opts.LabelOpts(is\_show=False),

linestyle\_opts=opts.LineStyleOpts(width=0.5, curve=0.3, opacity=0.7),

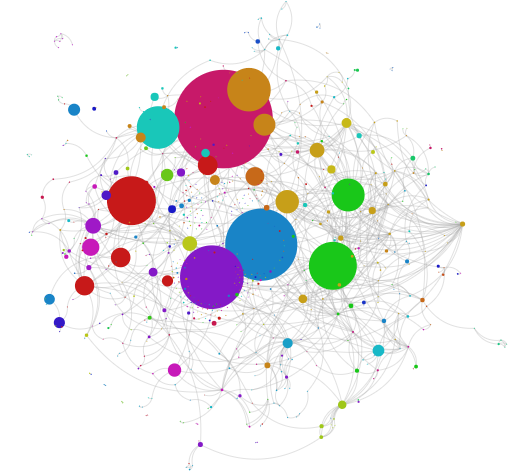
)

.set\_global\_opts(title\_opts=opts.TitleOpts(title="Graph-NPM Dependencies"))

)

return c

graph\_npm\_dependencies().render\_notebook()



## [Pie：饼图](https://pyecharts.org/" \l "/zh-cn/basic_charts?id=pie%ef%bc%9a%e9%a5%bc%e5%9b%be)

**1、Pie-基本示例**

from example.commons import Faker

from pyecharts import options as opts

from pyecharts.charts import Page, Pie

def pie\_base() -> Pie:

c = (

Pie()

.add("", [list(z) for z in zip(Faker.choose(), Faker.values())])

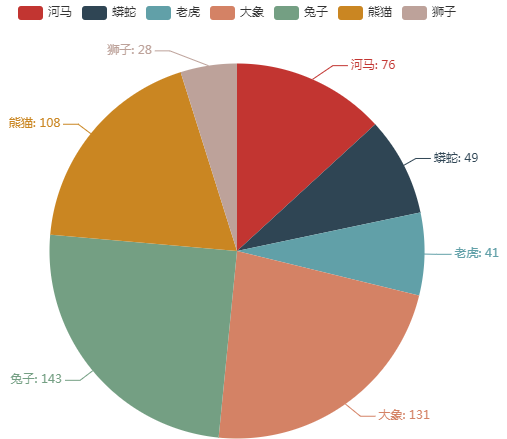
.set\_global\_opts(title\_opts=opts.TitleOpts(title="Pie-基本示例"))

.set\_series\_opts(label\_opts=opts.LabelOpts(formatter="{b}: {c}"))

)

return c

pie\_base().render\_notebook()



2、**Pie-设置颜色**

def pie\_set\_colors() -> Pie:

c = (

Pie()

.add("", [list(z) for z in zip(Faker.choose(), Faker.values())])

.set\_colors(["blue", "green", "yellow", "red", "pink", "orange", "purple"])

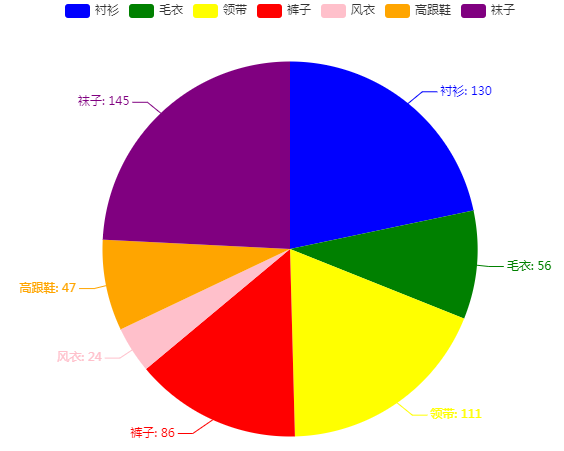
.set\_global\_opts(title\_opts=opts.TitleOpts(title="Pie-设置颜色"))

.set\_series\_opts(label\_opts=opts.LabelOpts(formatter="{b}: {c}"))

)

return c

pie\_set\_colors().render\_notebook()



3、**Pie-调整位置**

def pie\_position() -> Pie:

c = (

Pie()

.add(

"",

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

center=["35%", "50%"],

)

.set\_global\_opts(

title\_opts=opts.TitleOpts(title="Pie-调整位置"),

legend\_opts=opts.LegendOpts(pos\_left="15%"),

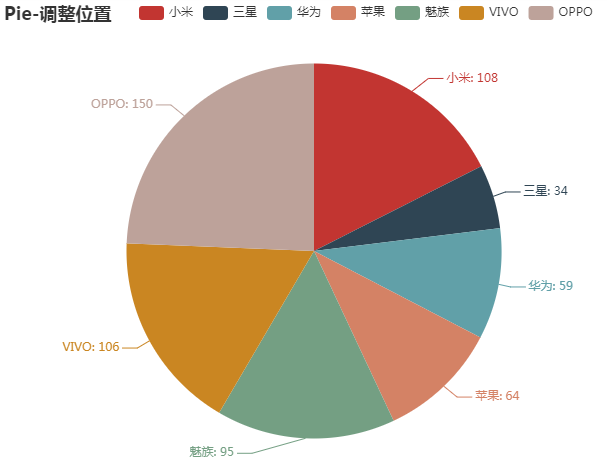
)

.set\_series\_opts(label\_opts=opts.LabelOpts(formatter="{b}: {c}"))

)

return c

pie\_position().render\_notebook()



4、**Pie-Radius**

def pie\_radius() -> Pie:

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}"))

)

return c

pie\_radius().render\_notebook()

