Python可视化从入门到入院

1.Json文件

json是一种轻量级的数据交互格式。可以按照json指定的格式去组织和封装数据。本质上是一个带有特定格式的字符串。它是一种在各个编程语言中流通的数据格式，负责不同编程语言中的数据传递和交互。为了让不同的语言都能互相通用的传递数据，json是一种非常良好的中转数据格式。Python数据和Json数据的相互转化如下：

```
# 导入json模块
import json

# 准备符合格式json格式要求的python数据(准备列表，列表内每一个元素都是字典)
data = [{"name": "capper", "age": "27"}, {"name": "小鬼", "age": "25"}]

# 通过json.dumps(data)方法把python数据转化为json数据
json_str = json.dumps(data)
print(type(json_str)) # <class 'str'>
print(json_str)
# [{"name": "capper", "age": "27"}, {"name": "\u5c0f\u9b3c", "age": "25"}]
"""
中文在转化时涉及到编码问题，在dumps()内再传一个参数"ensure_ascii=False"即可解决
"""

# 通过json.loads(data)方法把json数据转化为python数据
s = '[{"name": "capper", "age": "27"}, {"name": "小鬼", "age": "25"}]'
l = json.loads(s)
print(type(l))
print(l)
```

2.Pyecharts库

在实现数据可视化时，需要使用第三方pyecharts库，官网如下：pyecharts.org（画廊gallery.pyecharts.org）。

3.制作Line折线图

在使用pyecharts前，需要导包，格式如下

```
# 制作基础折线图
# 导包，导入Line功能构建折线图对象和可配置选项
from pyecharts.charts import Line  # 用来导入折线图模块
```

```
from pyecharts.options import *  # 用来导入配置相关的模块

# 得到一个折线图对象
line = Line()
#  添加x轴数据
line.add_xaxis(["中国", "美国", "英国"])  # 添加x轴数据时，传入的是一个列表
# 添加y轴数据
line.add_yaxis("GDP", [30, 20, 10])  # 添加y轴数据，首先给出y轴的名称，在传入列表数据，要和x轴传入的数据对应
# 全局配置项的设置
line.set_global_opts(
    # 图表的标题
    title_opts=TitleOpts(title="GDP展示", pos_left="center", pos_bottom="1%"),
    # 图例
    legend_opts=LegendOpts(is_show=True),
    # 工具箱
    toolbox_opts=ToolboxOpts(is_show=True),
    # 视觉映射配置
    visualmap_opts=VisualMapOpts(is_show=True)
# 全局配置项还有很多，可以在官网配置 - 全局配置项 找到对应代码
)
# 生成图表
line.render()
# 生成的图表会保存在同根目录下，是一个html文件，如果没有给render()传参，文件默认名字是render.html，可以通过浏览器查看
```

大体上，方法有：

add_xaxis()添加x轴数据，传入一个列表

add_yaxis()添加y轴数据，传入y轴名称和一个列表

set_global_opts()设置全局配置，其中里面又有：

- title_opts=TitleOpts(title="xxx", pos_left="center", pos_bottom="1%")图表的标题
- legend_opts=LegendOpts(is_show=True)图例
- toolbox_opts=ToolboxOpts(is_show=True)工具箱
- visualmap_opts=VisualMapOpts(is_show=True)视觉映射配置

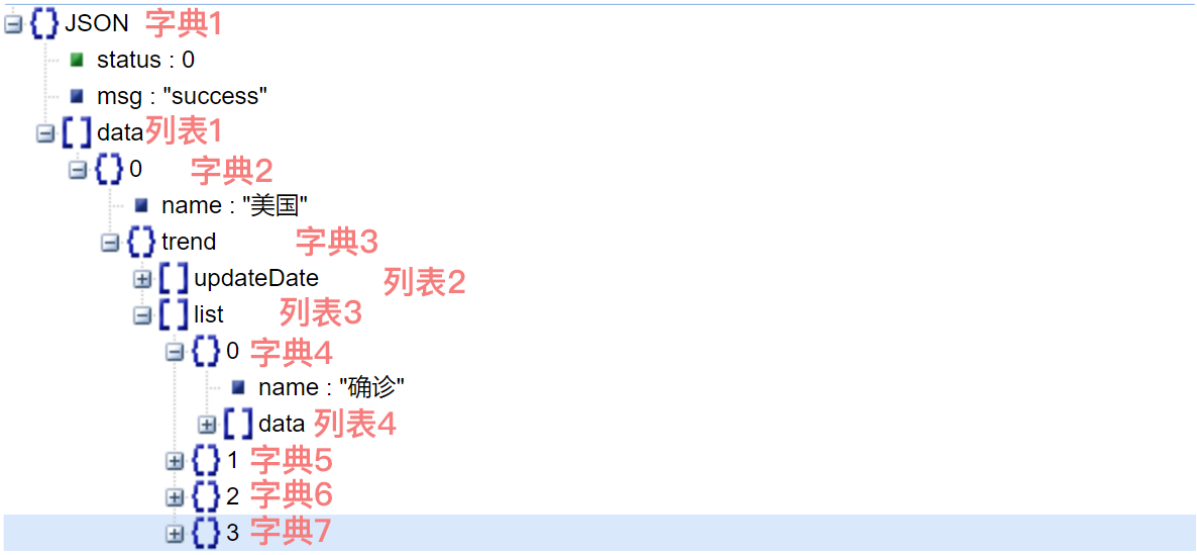
render()生成图表

接下来进行实际案例分析

现在针对D盘下的美国.txt、日本.txt，印度.txt，需要通过Python实现对他们2020年新冠确诊人数的可视化。

打开txt文件后，可以发现它们的身体部分都是json文件，但头部和尾部都是一些无关代码(json应该以‘{’ 开头，‘}’ 结尾)，而且层级关系比较混乱，不利于我们提取数据，所以还需要先格式化json文件。打开网址https://www.bejson.com/jsonviewernew/，把json文件的中间部分粘贴，点击左上角格式化，在屏幕中间就可观察到json视图，把各项展开，就可以看到清楚的层级关系。

以美国为例，现来分析一下这个json文件。



展开json后，可以看到字典1中有三项，分别是：

键status对应的值0，键msg对应的值"success"，键data对应的值列表1（数组）

展开列表1，里面有1个元素（标号0，相当于只有一个0索引）字典2

展开字典2，里面有2个元素，分别是键name对应的值"美国"，键trend对应的值字典3

展开字典3，里面有2个元素，分别是键updateDate对应的值列表2，键list对应的值列表3

展开列表2，里面存放着和时间有关的数据，由于它是列表，可以通过索引找到对应关系

展开列表3，里面有4个元素，根据索引顺序分别是字典4（0索引）、字典5（1索引）、字典6（2索引）、字典7（3索引）

展开字典4，里面有2个元素，分别是键name对应的键"确诊"和键data对应的值列表4

展开列表4，里面存放着和时间对应的确诊人数，相同地，可以根据索引来确定对应关系

剩下的字典5、6、7和字典4大同小异，不再说明

不难看出，我们真正需要获取的，其实就是列表2和列表4的数据，接下来开始代码的实现：

```
# 导入相关包
import json
from pyecharts.charts import Line
from pyecharts.options import *

# 处理数据

f_us = open("D:/美国.txt", "r", encoding="UTF-8")

# 读取美国的全部内容
us_data = f_us.read()
```

```
# 去掉不合json规范的开头(应该以花括号开头和结尾)
us_data = us_data.replace("jsonp_1629344292311_69436(", "")

# 去掉不合json规范的结尾(使用序列的切片)
us_data = us_data[:-2]

# json转python字典 (使用到了json包)
us_dict = json.loads(us_data)

# 获取trend key 通过key和索引来定位(由于时间和人数都在这个字典当中，索性先把字典提取出来)
us_trend_data = us_dict['data'][0]['trend']

# 获取日期数据，用于x轴(由于只需要2020年的数据，所以日期取到12.31即可，对应的索引是0-313，由于切片包头不包尾，所以要取到314)
us_x_data = us_trend_data['updateDate'][:314]

# 获取确认数据，用于y轴，取到2020年(相同地，数据取到到313索引即可)
us_y_data = us_trend_data['list'][0]['data'][:314]
```

至此，关于美国在2020年各时间的确诊人数变化已经提取成两份列表，接下来以相同的方式提取日本和印度的相关数据：

```
# 处理日本数据
f_jp = open("D:/日本.txt", "r", encoding="UTF-8")
jp_data = f_jp.read()
jp_data = jp_data.replace("jsonp_1629350871167_29498(", "")
jp_data = jp_data[:-2]
jp_dict = json.loads(jp_data)
jp_trend_data = jp_dict['data'][0]['trend']
jp_x_data = jp_trend_data['updateDate'][:314]
jp_y_data = jp_trend_data['list'][0]['data'][:314]

# 处理印度数据
f_in = open("D:/印度.txt", "r", encoding="UTF-8")
in_data = f_in.read()
in_data = in_data.replace("jsonp_1629350745930_63180(", "")
in_data = in_data[:-2]
in_dict = json.loads(in_data)
in_trend_data = in_dict['data'][0]['trend']
in_x_data = in_trend_data['updateDate'][:314]
in_y_data = in_trend_data['list'][0]['data'][:314]
```

数据获取完毕后，就可以准备生成图表了。

```
# 构建折线图对象(需要导入pyecharts.chart包里的Line模块)
line = Line()

# 添加x轴数据(由于x轴代表时间，所以添加一次即可)
line.add_xaxis(us_x_data)
```

```
# 添加y轴数据(分别添加美国、日本、印度的数据，先起名、后传入列表，为了最后生成的折线图比较清晰，将显示标签改为否)
line.add_yaxis("美国确诊人数", us_y_data, label_opts=LabelOpts(is_show=False))
line.add_yaxis("日本确诊人数", jp_y_data, label_opts=LabelOpts(is_show=False))
line.add_yaxis("印度确诊人数", in_y_data, label_opts=LabelOpts(is_show=False))

# 设置全局选项
line.set_global_opts(
    # 标题设置 标题-居中-向下
    title_opts=TitleOpts(title="2020年美日印确诊人数折线图", pos_left="center", pos_bottom="1%")
)

# 生成图表(可以在括号内传入该html名称)
line.render("2020年美日印三国新冠确诊人数.html")
# 关闭文件对象
f_us.close()
f_jp.close()
f_in.close()
```

4.制作Map地图

和制作折线图的流程差不多，这里拿中国地图大概演示一下：

导包的时候模块名选择“Map”，同时导入options模块方便后期对配置的设置。由于地图对于视觉化要求比较高，所以要打开视觉显示器，并添加颜色分段，也可以手动指定分段，视觉颜色的代码可以查看网址 https://tool.oschina.net/commons?type=3

```
from pyecharts.charts import Map
from pyecharts.options import *

# 准备地图对象
map = Map()

# 准备数据(数据要求是列表，每个元素都是一个元组，元组中输入地名和对应信息)
data = [
    ("北京市", 99),
    ("上海市", 63),
    ("广东省", 94),
    ("浙江省", 86),
    ("湖南省", 72)
]

# 添加数据(分别将标题，数据列表和地图类型传入)
map.add("测试地图", data, "china")
```

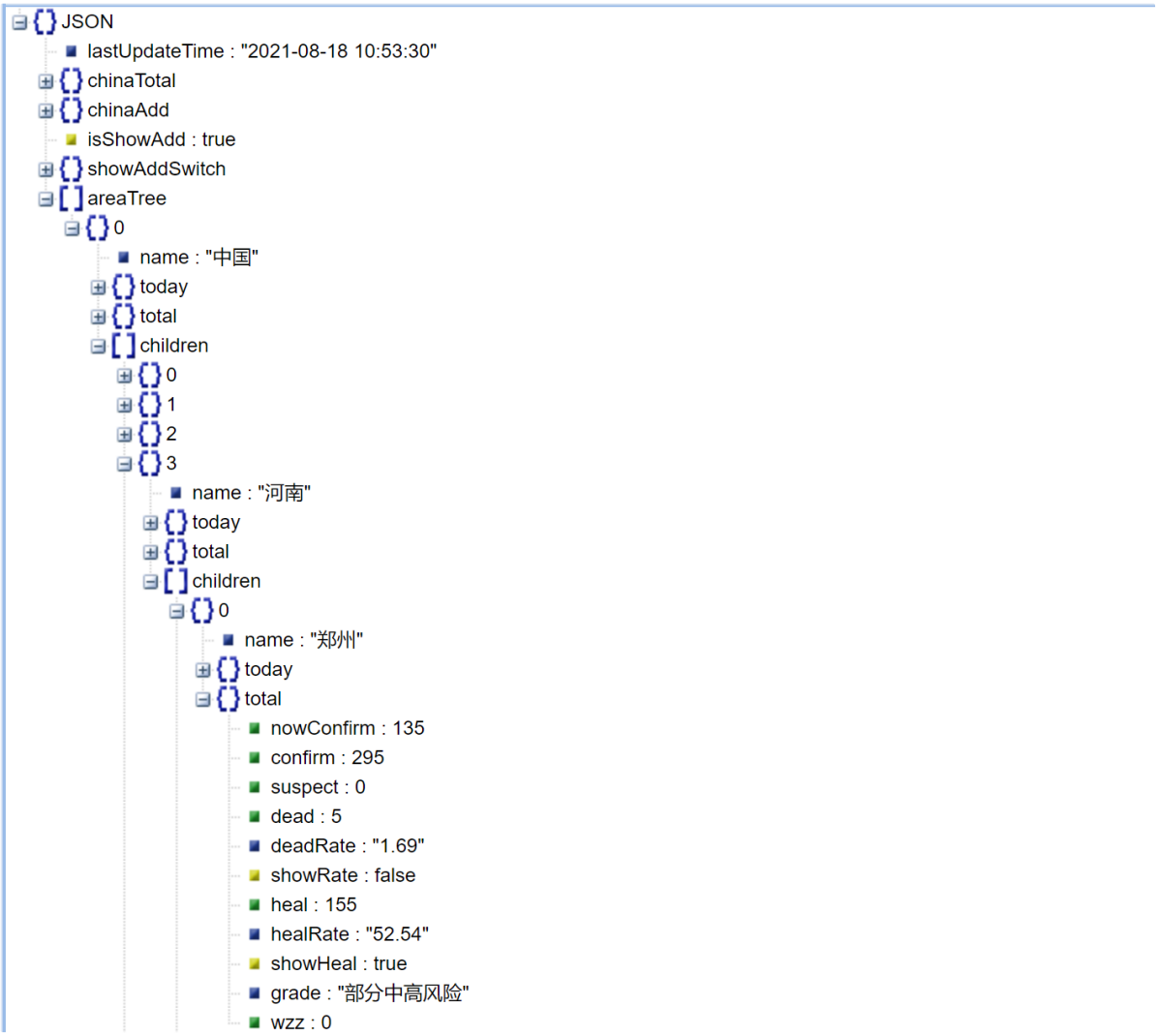
```
# 设置全局选项
map.set_global_opts(
    visualmap_opts=VisualMapOpts(
        # 添加视觉显示器
        is_show=True,
        # 添加颜色分段
        is_piecewise=True,
        # 手动指定分段
        pieces=[
            # 这是一个嵌套字典的列表，字典中包含每段的最小、最大值，标签以及该段颜色
            {"min": 1, "max": 50, "label": "1-50", "color": "#CCFFFF"},
            {"min": 51, "max": 80, "label": "51-80", "color": "#FF6666"},
            {"min": 81, "max": 100, "label": "81-100", "color": "#990033"}
        ]
    )
)

# 绘图
map.render()
```

接下来进行实际案例分析

现在针对D盘下的疫情.txt，需要通过Python实现对河南省各区疫情的地图可视化。

打开txt文件，可以发现里面的内容就是标准的json文件，无需我们做修改。直接打开网址对该json进行格式化，可以看到该文件包含了中国各地的所有疫情数据，而我们所需要的只是河南省的数据。



展开视图，最底层的列表展开后，就是我们需要的河南省各地区疫情人数，而我们需要的数据就是里面的total字典里面的confirm。

分析完后，开始代码的实现。

```
import json
from pyecharts.charts import Map
from pyecharts.options import *

# 读取文件
f = open("D:/疫情.txt", "r", encoding="UTF-8")
data = f.read()
# 关闭文件
f.close()
# 获取河南省数据
# 将json数据转化为字典
data_dict = json.loads(data)
# 取到河南省数据(对应上面的json视图找出各省数据)
cities_data = data_dict['areaTree'][0]['children'][3]['children']
# 准备数据为元组并放入list
# 创建一个空列表占位
data_list = []
# 使用for循环将得到的城市名与其数据进行匹配成元组，再放入列表当中
for city_data in cities_data:
```

```
    # 由于新版的pyecharts的Map不允许我们使用地区缩写，所以还需要将每一个地区补上完整名字
    city_name = city_data['name'] + '市'
    city_confirm = city_data['total']['confirm']
    # 将名字与数据嵌套，然后传入列表中
    data_list.append((city_name, city_confirm))

# 构建地图
map = Map()
# 分别将副标题，数据列表和地图类型传入
map.add("河南省疫情分布", data_list, "河南")

# 设置全局选项
map.set_global_opts(
    # 设置主标题
    title_opts=TitleOpts(title="河南省疫情地图",pos_left="center",pos_bottom="1%"),
    visualmap_opts=VisualMapOpts(
        is_show=True,
        is_piecewise=True,
        pieces=[
            # 可以先生成一次图表，找到人数的最大值后，根据最大值再划分视觉颜色段
            {"min": 1, "max": 99, "label": "1~99人", "color": "#CCFFFF"},
            {"min": 100, "max": 199, "label": "100~199人", "color": "#FFF99"},
            {"min": 200, "max": 299, "label": "200~299人", "color": "#FF9966"},
            {"min": 300, "max": 399, "label": "300~399人", "color": "#FFF6666"},
            {"min": 400, "max": 499, "label": "400~499人", "color": "#CC3333"},
            {"min": 500, "label": "500+人", "color": "#990033"},
        ]
    )
)
# 绘图
map.render("河南省疫情地图.html")
```

5.制作柱状图

导包时是选择pyecharts.charts里面的Bar模块，如果要制作时间线，需要在同一个包里导入Timeline模块。修改全局设置需要的包仍是pyecharts.options。在后期修改柱状图的主题时，用到的是pyecharts.globals包下的ThemeType模块。

现在制作一个带有时间线的柱状图，分别展示中美日三国在2021、2022年的GDP值。

```
from pyecharts.charts import Bar, Timeline
from pyecharts.options import *
from pyecharts.globals import ThemeType
```

```
# 构建柱状图对象(一个存放2021年的数据，一个存放2022年的数据)
bar1 = Bar()
bar2 = Bar()
# 添加x轴数据(和折线图一样，将x轴列表传进去)
bar1.add_xaxis(["中国", "美国", "日本"])
# 添加y轴数据(从左到右分别是柱状图名称，数据列表和标签放在右边)
bar1.add_yaxis("GDP", [24, 20, 10], label_opts=LabelOpts(position="right"))
# 反转xy轴
bar1.reversal_axis()
# 使用相同的方法制作2022年的数据
bar2.add_xaxis(["中国", "美国", "日本"])
# 添加y轴数据 把数值标签移动到右侧
bar2.add_yaxis("GDP", [30, 22, 17], label_opts=LabelOpts(position="right"))
# 反转xy轴
bar2.reversal_axis()

# 创建时间线对象并设置主题颜色
timeline = Timeline({'theme': ThemeType.DARK})
# timeline对象添加bar柱状图
timeline.add(bar1, "2021年GDP")
timeline.add(bar2, "2022年GDP")

# 设置自动播放
timeline.add_schema(
    # 是否自动播放
    is_auto_play=True,
    # 是否循环播放
    is_loop_play=True,
    # 自动播放的间隔，单位毫秒
    play_interval=1000,
    # 自动播放时显示时间线
    is_timeline_show=True
)
# 通过时间线绘图(如果制作的是无时间线的单个柱状图，直接用bar.render()即可)
timeline.render("基础柱状图-时间线.html")
```

其中，Bar常用的方法和Line相似，不过我们还使用了反转xy轴的方法：

```
add_xaxis()添加x轴数据，传入一个列表

add_yaxis()添加y轴数据，传入y轴名称和一个列表

set_global_opts()设置全局配置

reversal_axis()反转xy轴，可以不传参
```

而Timeline和Bar都在pyecharts.charts包下，并且导入pyecharts.globals包的ThemeType模块后，它在创建对象时就可指定它的主题颜色：

```
timeline = Timeline({'theme': ThemeType.xx})
```

timeline的常用方法有

```
add()传入一个柱状图对象以及该该柱状图对应的时间字符串

add_schema()设置自动播放，其中里面又有：

•   is_auto_play=True是否自动播放

•   is_loop_play=True是否循环播放

•   play_interval=1000自动播放的间隔(单位毫秒)

•   is_timeline_show=True自动播放时显示时间线

render()绘制时间线，可以传入html名称
```

接下来进行实际案例分析

现在针对D盘下的1960-2019全球GDP数据.csv，需要通过Python按照时间线实现对这些数据的可视化。

选择打开方式为记事本，看到当前文件编码模式为ansi，这是windows系统默认编码格式，跟随操作系统的语言版本来，由于我的操作系统是中文，默认中文编码叫GB2312，所以这次读取文件时的编码格式不再是UTF-8。

而且该文件的第一行数据是年份、国家和GDP，我们不需要这个数据，所以要把它删除。后续是每一行都详细的时间，国家和GDP，所以这次文件读取方式可以选择readlines()。

```
from pyecharts.charts import Bar, Timeline
from pyecharts.options import *
from pyecharts.globals import ThemeType

# 读取数据
f = open("D:/1960-2019全球GDP数据.csv", "r", encoding="GB2312")
data_lines = f.readlines()
# 关闭文件
f.close()
# 删除第一条数据
data_lines.pop(0)
# 将数据转换为字典储存，格式为：
# { 年份：[[国家,gdp],[国家,gdp]....], 年份：[[国家,gdp],[国家,gdp]....]...}
# 先定义一个字典对象
data_dict = {}
# 使用for循环将每一行数据中的年份，国家，GDP分别读取出来
for line in data_lines:
```

```
    # 一旦使用分割符切割后，每行的三个元素就会按序存入列表当中，所以需要通过索引把它取出来储存
    year = int(line.split(",")[0])  # 年份(由于切割得到的是字符串，还需要把他转换成整型方便后面的比较)
    country = line.split(",")[1]  # 国家
    gdp = float(line.split(",")[2])  # GPD(把科学计数法强制转换成浮点数)
    # 先判断字典中有没有指定的key，没有时首先将key添加到列表(通过捕获异常的方式)
    try:
        data_dict[year].append([country, gdp])
    except KeyError:
        data_dict[year] = []  # 确定年份所对应的value，它是一个列表
        data_dict[year].append([country, gdp])
# 创建时间线对象
timeline = Timeline({"theme": ThemeType.LIGHT})
# 排序年份(由于字典是无序的，无法确定key是按照年份顺序排列，首先需要通过获取全部key，再将key排序)
sorted_year_list = sorted(data_dict.keys())
for year in sorted_year_list:
    # 因为最终每一年只取GDP前8的数据，所以在循环时要先给各国家GDP排序
    # 由于年份在字典中是作为key存在的，而value就是由国家和GDP嵌套的列表，现在对嵌套列表进行排序，并打开反转，由高到低排序
    data_dict[year].sort(key=lambda element: element[1], reverse=True)
    # 取出本年份前8名国家(使用列表的切片)
    year_data = data_dict[year][0:8]
    # 由于时间轴x、y都是列表，为了方便添加数据，先创建一个空列表
    x_data = []
    y_data = []
    # 嵌套一层for循环，将本年份前8名国家的数据分别存入xy轴
    for country_gdp in year_data:
        x_data.append(country_gdp[0])  # x轴添加国家
        y_data.append(country_gdp[1] / 100000000)  # y轴添加GDP数据,注意单位是亿
    # 构建柱状图对象
    bar = Bar()
    # 将数据反转，此次反转是将GDP高的数据显示在坐标轴上方
    x_data.reverse()
    y_data.reverse()
    bar.add_xaxis(x_data)
    bar.add_yaxis("GDP(亿)", y_data, label_opts=LabelOpts(position="right"))  # 给y轴命名，并让标签在右侧显示
    # 反转xy轴
    bar.reversal_axis()
    # 设置每一年图表标题
    bar.set_global_opts(
        title_opts=TitleOpts(title=f"{year}年全球前八GDP数据")
    )
    # 把bar添加到时间线，并给当前时间命名 由于命名只接受字符串，还需要将年份类型转换
```

```
    timeline.add(bar, str(year))

# 设置时间线自动播放
timeline.add_schema(
    play_interval=500,
    is_timeline_show=True,
    is_auto_play=True,
    is_loop_play=False
)

# 绘图
timeline.render("1960~2019全球GDP前八国家.html")
```