项目 4 人民生活数据图形展示

一、课程基本信息

1. **课程名称**：Python 数据可视化
2. **授课章节**：项目四 人民生活数据图形展示
3. **授课对象**：[具体专业与年级]
4. **授课时间**：[X] 学时
5. **授课地点**：[机房名称]

二、教学目标

（一）知识目标

1. 深入理解 pyecharts 库的功能、特点及应用场景。
2. 熟悉 pyecharts 库中图表的类、初始配置项、全局配置项以及系列配置项的含义与用法。
3. 掌握不同类型图表（如柱形图、折线图、散点图、箱形图、矩形树图、漏斗图、层叠多图、并行多图、顺序多图、选项卡多图、时间线轮播多图、饼图、雷达图）在展示人民生活数据时的适用场景与数据要求。

（二）能力目标

1. 能够独立完成 pyecharts 库的安装与配置。
2. 熟练运用 pyecharts 库的相关功能，根据给定的居民收入与支出情况数据、居民主要食品消费量数据，准确选择合适的图表类型进行数据可视化展示。
3. 具备对图表进行个性化配置的能力，包括但不限于设置图表标题、坐标轴标签、颜色主题、数据标签等，以提升图表的可读性与美观性。
4. 能够将多图组合（如层叠多图、并行多图、顺序多图、选项卡多图、时间线轮播多图）的技巧应用于实际项目中，实现复杂数据关系的清晰呈现。

（三）素质目标

1. 培养学生的数据敏感度与数据分析思维，使其能够从数据中挖掘有价值的信息，并通过可视化手段清晰表达。
2. 提升学生的审美素养，使其在进行数据可视化时注重图表的整体布局、色彩搭配等美学元素，以增强可视化效果的吸引力。

3. 增强学生的团队协作意识，通过小组项目或讨论，促进学生之间的交流与合作，共同解决数据可视化过程中遇到的问题。
4. 培养学生的自主学习能力与问题解决能力，使其在面对新的可视化需求或技术难题时，能够主动查阅资料、探索解决方案。

三、教学重难点

（一）教学重点

1. pyecharts 库的安装与基础使用，包括快速绘图方法以及各类配置项的设置。
2. 不同类型图表（任务一和任务二中涉及的所有图表）的绘制方法与关键参数调整，以准确展示人民生活数据特征。
3. 多图组合的实现方式与应用场景，如何根据数据逻辑关系选择合适的多图展示形式。

（二）教学难点

1. 针对不同的人民生活数据特点，精准选择最恰当的图表类型进行可视化，避免因图表选择不当导致数据信息传达不准确。
2. 深入理解并灵活运用图表的全局配置项和系列配置项，实现复杂图表样式与交互效果的定制，满足实际项目需求。
3. 在多图组合时，确保各图表之间的数据关联性与逻辑一致性，以及整体布局的合理性与美观性。

四、教学方法

1. **讲授法**：讲解项目背景、学习目标、pyecharts 库的理论知识、图表类型的特点及适用场景等内容，使学生对课程有初步的系统认识。
2. **演示法**：在机房通过实际操作演示 pyecharts 库的安装过程、各类图表的绘制代码实现、配置项的调整效果等，让学生更直观地理解和掌握操作步骤。
3. **实践法**：安排学生在机房进行大量的实践练习，根据给定的人民生活数据完成各个任务中的图表绘制，通过实际操作加深对知识的理解与运用能力。
4. **小组讨论法**：组织学生进行小组讨论，针对任务中的数据处理、图表选择、多图组合等问题交流思路与解决方案，培养学生的团队协作与思维碰撞能力。
5. **案例分析法**：引入实际的人民生活数据可视化案例，分析其图表选择、设计思路、呈现效果等，引导学生学习优秀案例的经验，提升自身的可视化设计水平。

五、教学过程

（一）课程导入

1. 项目背景介绍

通过展示一些实际的人民生活数据报告或统计资料，如国家统计局发布的居民收入与消费数据报告，引出本项目的主题 —— 人民生活数据图形展示。

阐述在当今大数据时代，数据可视化对于理解和传达人民生活相关信息的重要性。例如，清晰直观的图表能够帮助政策制定者快速了解居民生活状况，为政策制定提供依据；也能让普通民众更容易理解复杂的数据，关注自身生活水平的变化。

2. 学习目标阐述

详细解读本项目的知识目标、能力目标和素质目标，让学生明确通过本项目的学习，他们应该掌握哪些知识、具备哪些技能以及在个人素养方面能够得到怎样的提升。

强调学习目标与后续课程以及实际工作的紧密联系，激发学生的学习动力。

（二）技术准备

1. pyecharts 的介绍与安装

理论讲解：介绍 pyecharts 是一个用于生成 Echarts 图表的 Python 库，Echarts 是一款由百度开源的强大的可视化工具，pyecharts 使得 Python 开发者能够方便地利用 Echarts 的丰富功能进行数据可视化。讲解 pyecharts 的特点，如简单易用、支持多种图表类型、可生成交互式图表等。

安装演示：在机房电脑上，通过命令行工具演示如何使用 pip 命令安装 pyecharts 库。详细讲解安装过程中可能遇到的问题及解决方法，如网络连接问题、Python 环境配置问题等。安装完成后，通过简单的代码测试确保 pyecharts 库安装成功。

2. pyecharts 快速绘图

代码演示：通过一个简单的示例，如绘制一个基本的柱形图展示某地区居民的月平均收入，向学生演示 pyecharts 的快速绘图方法。代码如下（此处代码为示例框架，具体数据可根据实际情况设定）：

```
from pyecharts import options as opts
# 导入 Bar 类，绘制柱形图
from pyecharts.charts import Bar
# 导入 faker 模块，生成随机数据
from pyecharts.faker import Faker
# 创建一个 Bar 类的对象
bar = Bar(init_opts=opts.InitOpts(width='450px', height='300px'))
# 添加 X 轴数据
bar.add_xaxis(Faker.clothes)
```

```
# 添加 Y 轴数据
bar.add_yaxis("商家 A", Faker.values())
# bar.add_yaxis("商家 B", Faker.values())
# 将图表渲染到 Jupyter Notebook 中
bar.render_notebook()
```

原理讲解：逐行解释代码的含义，包括导入必要的模块、准备数据、创建图表对象、添加坐标轴数据以及渲染图表生成 HTML 文件等步骤。让学生了解 pyecharts 绘图的基本流程。

3. **图表的类、图表的初始配置项、图表的全局配置项、图表的系列配置项**

图表的类：介绍 pyecharts 中不同图表类型对应的类，如 Bar 类用于柱形图、Line 类用于折线图、Scatter 类用于散点图等。讲解如何根据不同的可视化需求选择合适的图表类。

图表的初始配置项：以柱形图为例，演示如何设置图表的初始配置项，如设置图表的宽度、高度、背景颜色等。代码如下：

```
from pyecharts.charts import Bar
from pyecharts import options as opts
from pyecharts.faker import Faker
bar = (
    Bar(
    # 初始化配置项
    init_opts=opts.InitOpts(
        # 设置画布宽度
        width='500px',
        # 设置画布高度
        height='300px',
        # 设置主题
        theme='white',
        # 设置背景色
        bg_color='rgba(145, 214, 243, 0.4)')
    )
    .add_xaxis(Faker.clothes)
    .add_yaxis("商家 A", Faker.values())
    .add_yaxis("商家 B", Faker.values())
)
bar.render_notebook()
```

图表的全局配置项：讲解全局配置项的作用，如设置图表的标题、副标题、坐标轴标签、坐标轴刻度、图例等。通过代码演示如何进行全局配置，如下：

```
from pyecharts import options as opts
from pyecharts.faker import Faker
bar = (
    Bar(init_opts=opts.InitOpts(
        width='500px',
        height='300px',
        theme='white')
    )
    .add_xaxis(Faker.clothes)
    .add_yaxis("商家 A", Faker.values())
    .add_yaxis("商家 B", Faker.values())
    .set_global_opts(
        title_opts=opts.TitleOpts(title="商家 A 和商家 B 商品销售额",
                                  subtitle='单位：万元',
                                  pos_left='left',
                                  pos_top='top',
                                  # 主标题样式
                                  title_textstyle_opts=opts.TextStyleOpts(
                                      color='red'),
                                  # 副标题样式
                                  subtitle_textstyle_opts=opts.TextStyleOpts(
                                      color='blue')),
        legend_opts=opts.LegendOpts(# 图例是否显示
                                    is_show=True,
                                    # 图例位置
                                    pos_left='60%',
                                    pos_bottom='90%',
                                    # 图例形状
                                    legend_icon='rect',
                                    # 图例布局
                                    orient='horizontal',
                                    # 图例间隔
```

```
                                    item_gap=20,
                                    # 图例文本样式
                                    textstyle_opts=opts.TextStyleOpts(
                                        color='blue')

    ))
)
bar.render_notebook()om pyecharts.charts import Bar
```

图表的系列配置项：解释系列配置项用于设置每个数据系列的样式和属性，如柱形图中柱子的颜色、宽度，折线图中线的样式、标记点等。通过代码展示系列配置项的设置方法：

```
from pyecharts.charts import Bar
from pyecharts import options as opts
from pyecharts.faker import Faker
bar = (
    Bar(init_opts=opts.InitOpts(
        width='500px',
        height='300px',
        theme='white')
    )
    .add_xaxis(Faker.clothes)
    .add_yaxis("商家 A", Faker.values())
    .add_yaxis("商家 B", Faker.values())
    .set_global_opts(
        title_opts=opts.TitleOpts(title="商家 A 和商家 B 商品销售额",
                                  subtitle='单位：万元',
                                  pos_left='left',
                                  pos_top='top',
                                  title_textstyle_opts=opts.TextStyleOpts(
                                      color='red'),
                                  subtitle_textstyle_opts=opts.TextStyleOpts(
                                      color='blue')),
        legend_opts=opts.LegendOpts(is_show=True,
                                    pos_left='60%',
                                    pos_bottom='90%',
```

```
                                    legend_icon='rect',
                                    orient='horizontal',
                                    item_gap=20,
                                    textstyle_opts=opts.TextStyleOpts(
                                        color='blue')
    ))
    .set_series_opts(
        # 关闭标签
        label_opts=opts.LabelOpts(is_show=False),
        markpoint_opts=opts.MarkPointOpts(
            data=[
                # 根据最大值和最小值定位
                opts.MarkPointItem(type_="min", name="Y 轴最小", value_index=1),
                opts.MarkPointItem(type_="max", name="Y 轴最大", value_index=1)
            ]))
)
bar.render_notebook()
```

（三）任务一 居民收入与支出情况图形展示

1. 任务描述

给出具体的居民收入与支出情况数据集，包括不同年份或季度的居民收入、各项支出数据。

明确任务要求学生通过不同类型的图表对这些数据进行可视化展示，以清晰呈现居民收入与支出的变化趋势、各项支出的占比情况以及收入与支出之间的关系等信息。

2. 绘制柱形图

概念讲解：柱形图，又称柱状统计图，是一种以宽度相同的长方形的长短或高度表示数据大小的图表，X 轴表示分类，Y 轴表示数值。通过绘制柱形图，可以直观地对数据的大小并进行比较，可以明显地观察到各数据之间的差异。

pyecharts 中 Bar()类用来绘制柱形图，其绘制柱形图的基本程序结构如下所示。

```
bar = (
Bar(init_opts=opts.InitOpts())    # 设置初始化配置项
.add_xaxis()                    # 设置 X 轴数据
.add_yaxis()                    # 设置 Y 轴数据
```

```
.reversal_axis()                # 翻转 xY 轴数据
.set_global_opts()              # 设置全局配置项
.set_series_opts()              # 设置系列配置项
)
bar.render_notebook()
```

案例讲解：编写代码，绘制柱形图展示城镇居民人均可支配收入情况。

详细解释代码中每个部分的作用；

讲解模块的导入；

讲解数据的读取；

讲解如何为图表对象添加 x 轴和 y 轴数据；

讲解如何设置图表的标题；

讲解如何渲染图表。

学生实践：让学生根据给定的的城镇居民人均收入情况数据，绘制 2014-2023 年城镇居民人均可支配收入的柱形图，展示其在不同时间段的变化情况。教师在学生实践过程中巡视指导，解决学生遇到的问题。

3. **绘制折线图**

概念讲解：折线图是由许多点用直线连接形成的图表，用于显示数据在一个连续的时间间隔或者时间跨度上的变化，它的特点是反映事物随时间或有序类别而变化的趋势。在折线图中，类别数据沿 X 轴均匀分布，例如可以表示具有相同间隔的时间推移，而 Y 轴表示均与分布的数据大小。

通过折线图，可以清晰地观察到数据是递增还是递减、增减的速率、增减的规律等。因此，折线图常用来分析数据随时间的变化趋势，也可用来分析多组数据随时间变化的相互作用和相互影响。

pyecharts 中 Line()类用来绘制折线图，其绘制折线图的基本程序结构如下所示。

```
line = (
Line(init_opts=opts.InitOpts())        # 设置初始化配置项
    .add_xaxis()                    # 设置 X 轴数据
    .add_yaxis()                    # 设置 Y 轴数据
    .set_global_opts()              # 设置全局配置项
    .set_series_opts()              # 设置系列配置项
)
line.render_notebook()
```

案例讲解：编写代码，绘制折线图展示城镇居民、农村居民和全国居民人均可支配收入情况。

详细解释代码中每个部分的作用；

讲解模块的导入；

讲解数据的读取；

讲解如何为图表对象添加 x 轴和 y 轴数据；

讲解如何设置图表的标题；

讲解如何渲染图表。

学生实践：让学生根据给定的城镇居民人均收入情况、农村居民人均收入情况和全国居民人均收入情况数据，绘制 2014-2023 年城镇居民人均可支配收入、农村居民人均可支配收入和全国居民人均收入情况变化趋势的折线图，展示其在不同时间段的变化情况。教师在学生实践过程中巡视指导，解决学生遇到的问题。引导学生对比折线图与柱形图在展示数据变化趋势上的特点。

4. **绘制散点图**

概念讲解：散点图是指在回归分析中，是因变量随自变量而变化的大致趋势图，据此可以选择合适的函数对数据点进行拟合。散点图通常用于比较跨类别的聚合数据。通过观察散点图上数据点的分布情况可以推断出变量间的相关性。如果变量之间不存在相互关系，在散点图上就会表现为随机分布的离散的点，如果存在某种相关性，大部分的数据点就会相对密集并以某种趋势呈现。

散点图通常用于显示和比较数值，散点图中包含的数据越多，比较的效果就越好。散点图的优点是可以直观表现出影响因素和预测对象之间的总体关系趋势；能通过直观醒目的图形方式，反映变量间的形态变化关系情况，以便于来模拟变量之间的关系。

pyecharts 中 Scatter()类用来绘制散点图，其绘制散点图的基本程序结构如下所示。

```
scatter = (
Scatter(init_opts=opts.InitOpts())     # 设置初始化配置项
    .add_xaxis()                       # 设置 X 轴数据
    .add_yaxis()                       # 设置 Y 轴数据
    .set_global_opts()                 # 设置全局配置项
    .set_series_opts()                 # 设置系列配置项
)
scatter.render_notebook()
```

案例讲解：编写代码，绘制散点图展示城镇居民人均可支配收入与服务性消费支出关系。

详细解释代码中每个部分的作用；

讲解模块的导入；

讲解数据的读取；

讲解如何为图表对象添加 x 轴和 y 轴数据；

讲解如何设置图表的标题；

讲解如何渲染图表。

学生实践：让学生根据给定的居民人均收入与支出情况数据，绘制 2014-2023 年城镇居民人均可支配收入与服务性消费支出关系的散点图，展示其在不同时间段的变化情况。教师在学生实践过程中巡视指导，解决学生遇到的问题。引导学生思考如何从散点图中解读数据之间的相关性。

5. **绘制箱形图**

概念讲解：箱形图又称箱线图，是一种用作显示一组数据分散情况资料的统计图，因图形如箱子，且在上下四分位数之外常有线条像胡须延伸出去而得名。它能显示出一组数据的最大值、最小值、中位数、及上下四分位数。

箱形图主要用于反映原始数据分布的特征，适用于进行多组数据分布特征的比较，多用于数值统计，能提供有关数据位置和分散情况的关键信息，尤其在比较不同的母体数据时更可表现其差异。

pyecharts 中 Boxplot()类用来绘制箱形图，其绘制箱形图的基本程序结构如下所示。

```
boxplot = (Boxplot(init_opts=opts.InitOpts()))    # 设置初始化配置项
boxplot.add_xaxis()                               # 设置 X 轴数据
boxplot.add_yaxis()                               # 设置 X 轴数据
boxplot.set_global_opts()                          # 设置全局配置项
boxplot.set_series_opts()                          # 设置系列配置项
boxplot.render_notebook()
```

案例讲解：编写代码，绘制箱形图展示城镇居民、农村居民和全国居民人均可支配收入情况。

详细解释代码中每个部分的作用；

讲解模块的导入；

讲解数据的读取；

讲解如何为图表对象添加 x 轴和 y 轴数据；

讲解如何设置图表的标题；

讲解如何渲染图表。

学生实践：让学生根据给定的居民人均收入情况数据，绘制 2014-2023 年城镇居民、农村居民和全国居民人均可支配收入的箱形图，展示居民收入的分布情况，分析数据的分布特征。

教师在学生实践过程中巡视指导，解决学生遇到的问题。引导学生思考如何从散点图中解读数据之间的相关性。

6. **绘制矩形树图**

概念讲解：矩形树图是属于树图的一种，采用矩形表示层次结构里的节点，父子节点之间的层次关系用矩形之间的相互嵌套隐喻来表达。矩形树图适合展现具有层级关系的数据，能够直观体现同级之间的比较。

pyecharts 中 TreeMap()类用来绘制矩形树图，其绘制矩形树图的基本程序结构如下所示。

```
treemap = (
TreeMap(init_opts=opts.InitOpts())    # 设置初始化配置项
    .add()                          # 设置系列名称等参数
    .set_global_opts()                # 设置全局配置项
    .set_series_opts()                # 设置系列配置项
)
treemap.render_notebook()
```

案例讲解：编写代码，绘制矩形树图展示城镇居民和农村居民在食品烟酒、衣着、居住、交通通信等领域的人均支出情况。

详细解释代码中每个部分的作用；

讲解模块的导入；

讲解数据的读取；

讲解如何设置图表的标题；

讲解如何渲染图表。

学生实践：让学生根据给定的城镇居民人均支出情况和农村居民人均支出情况数据，绘制2023 年城镇居民和农村居民在食品烟酒、衣着、居住、交通通信、教育文化娱乐和医疗保健方面的人均支出情况矩形树图，展示居民各项支出在总支出中的占比结构以及各项支出的细分情况，以支出项目的层级关系。学生根据实际数据，调整矩形树图的层级和数据，展示自己感兴趣的居民收入或支出的结构细分情况。教师在学生实践过程中巡视指导，解决学生遇到的问题。

7. **绘制漏斗图**

概念讲解：漏斗图是一种形如漏斗状图形，用来展示事件的环节，整个图形由多个梯形从上而下叠加而成。图形一般由竖形条状或横行条状拼接而成，按照一定的顺序组成阶段层级，每一层都用于表示不同的阶段，从而呈现出这些阶段之间的某项要素或指标递减的趋势。

pyecharts 中 Funnel()类用来绘制漏斗图，其绘制漏斗图的基本程序结构如下所示。

```
funnel = (
Funnel(init_opts=opts.InitOpts())       # 设置初始化配置项
    .add()                             # 设置系列名称等参数
    .set_global_opts()                  # 设置全局配置项
    .set_series_opts()                  # 设置系列配置项
)
funnel.render_notebook()
```

案例讲解：编写代码，绘制漏斗图展示全国居民按收入五等份分组的人均可支配收入情况。

详细解释代码中每个部分的作用；

讲解模块的导入；

讲解数据的读取；

讲解如何设置图表的标题；

讲解如何渲染图表。

学生实践：让学生根据给定的全国居民按收入五等份分组的收入情况数据，绘制 2023 年全国居民按收入五等份分组的人均可支配收入漏斗图。学生根据自己对数据的理解，设计类似的漏斗图场景，展示居民收入与支出相关的流程转化数据。教师在学生实践过程中巡视指导，解决学生遇到的问题。

8. **绘制层叠多图**

概念讲解：pyecharts 中 overlap()方法可以将多个图表叠加在一个画布中显示，以柱形图和折线图叠加为例，其绘制层叠多图的基本程序结构如下所示。

```
bar = (
Bar(init_opts=opts.InitOpts())        # 设置初始化配置项
    .add_xaxis()                     # 设置 X 轴
    .add_yaxis()                     # 设置 Y 轴
    .extend_axis()                   # 扩展 X/Y 轴
    .set_global_opts()                # 设置全局配置项
    .set_series_opts()                # 设置系列配置项
)
line = Line().add_xaxis().add_yaxis()    # 设置折线图参数
bar.overlap(line)                      # 叠加柱形图和折线图
bar.render_notebook()
```

案例讲解：编写代码，绘制展示农村居民人均可支配收入柱形图、农村居民人均可支配收入比上年增长折线图叠加的层叠多图。

详细解释代码中每个部分的作用；

讲解模块的导入；

讲解数据的读取；

讲解如何为图表对象添加 x 轴和 y 轴数据；

讲解如何设置图表的标题；

讲解如何渲染图表。

学生实践：让学生根据给定的农村居民人均收入情况数据，绘制 2014-2023 年农村居民人均可支配收入柱形图、农村居民人均可支配收入比上年增长折线图叠加在一起的层叠多图。学生根据自己对数据的理解，设计类似的层叠多图场景，展示居民收入与支出相关的数据。教师在学生实践过程中巡视指导，解决学生遇到的问题。

9. 绘制并行多图

概念讲解：在进行数据分析的过程中，为了从多个角度进行描述，需要将多个图表绘制在同一个区域，pyecharts 中的 Grid()类可以用来绘制并行多图，可以采用上下布局和左右布局的方式绘图。

Grid()类中有一个 add()方法，使用该方法可以为组合图表添加配置项，add()方法的语法格式如下：

```
add(chart, grid_opts, grid_index=0, is_control_axis_index=False)
```

add()方法的常用参数说明如表 1 所示。

表 1　add()常用参数

参数名称	说明
chart	图表实例
grid_opts	直角坐标系网格配置项
grid_index	直角坐标系网格索引
is_control_axis_index	是否由自己控制 Axis 索引

案例讲解：编写代码，绘制城镇居民人均可支配工资性收入、经营净收入、财产净收入、转移净收入柱形图和饼图组成的并行多图。

详细解释代码中每个部分的作用；

讲解模块的导入；

讲解数据的读取；

讲解如何为图表对象添加 x 轴和 y 轴数据；

讲解如何设置图表的标题；

讲解如何渲染图表。

学生实践：让学生根据给定的城镇居民人均收入情况数据，绘制 2023 年城镇居民人均可支配工资性收入、经营净收入、财产净收入、转移净收入柱形图和饼图组成的并行多图。学生根据自己对数据的理解，设计类似的并行多图场景，展示居民收入与支出相关的数据。教师在学生实践过程中巡视指导，解决学生遇到的问题。

10. 绘制顺序多图

概念讲解：如果需要对不同的数据进行展示，可以将多个图表按照顺序绘制在同一个区域，pyecharts 中 Page()类可以绘制顺序多图。

Page()类的语法格式如下：

```
Page(page_title="Awesome-pyecharts", js_host="", interval=1,
layout=PageLayoutOpts())
```

Page()类的常用参数说明如表 2 所示。

表 2 Page()类常用参数

参数名称	说明
page_title	HTML 标题
js_host	远程 HOST，默认为 "https://assets.pyecharts.org/assets/"
interval	每个图例之间的间隔
layout	布局配置项

案例讲解：编写代码，绘制城镇居民和农村居民人均收入情况数据绘制柱形图、城镇居民和农村居民人均支出情况数据绘制折线图，并将柱形图和折线图按照顺序组合在一起。解释顺序多图是按照一定顺序依次展示多个图表，通常用于展示数据在不同阶段的变化或不同维度的信息。

详细解释代码中每个部分的作用；

讲解模块的导入；

讲解数据的读取；

讲解如何为图表对象添加 x 轴和 y 轴数据；

讲解如何设置图表的标题；

讲解代码中 Page 类的使用方法，它用于将多个图表按添加的顺序进行组合并渲染成一个 HTML 页面。

学生实践：让学生根据给定的城镇居民和农村居民人均收入情况数据绘制柱形图，根据 2014-2023 年城镇居民和农村居民人均支出情况数据绘制折线图，并将柱形图和折线图按照顺序组合在一起的顺序多图。学生根据自己对数据的理解，设计类似的顺序多图场景，鼓励学生思考如何通过顺序展示突出居民收入与支出相关的数据变化。教师在学生实践过程中巡视指导，解决学生遇到的问题。

11. 绘制选项卡多图

概念讲解：在进行数据可视化时，如果不方便在同一个区域一次展示多个图表，可以使用 pyecharts 中 Tab()类绘制选项卡多图，通过点击选项卡来切换到不同的图表进行分析。

Tab()类中有一个 add()方法，使用该方法可以为组合图表添加图表，add()方法的语法格式如下：

add(chart, tab_name)
add()方法的常用参数说明如表 3 所示。

表 3　add()常用参数

参数名称	说明
chart	任意图表类型
tab_name	标签名称

案例讲解：编写代码，绘制城镇居民和农村居民人均可支配收入柱形图，人均可支配收入比上年增长折线图，并以选项卡多图的形式组合在一起。讲解选项卡多图是在一个页面中通过选项卡的形式切换显示不同的图表，适用于展示多种类型但相互关联的数据，避免页面过于拥挤。

详细解释代码中每个部分的作用；

讲解模块的导入；

讲解数据的读取；

讲解如何为图表对象添加 x 轴和 y 轴数据；

讲解如何设置图表的标题；

详细解释 Tab 类的使用，通过 add 方法将不同的图表添加到不同的选项卡中，并为每个选项卡设置名称；

讲解如何渲染图表。

学生实践：让学生根据给定的城镇居民和农村居民人均收入情况数据，绘制 2014-2023 年城镇居民和农村居民人均可支配收入柱形图，人均可支配收入比上年增长折线图，并以选项卡多图的形式组合在一起。学生根据自己对数据的理解，设计类似的选项卡多图场景，提醒学生注意选项卡名称的设置要清晰准确，方便用户理解图表内容。教师在学生实践过程中巡视指导，解决学生遇到的问题。

12. 绘制时间线轮播多图

概念讲解：在进行数据分析时，有时需要展示不同时间段的数据变化情况，如果将所有数据绘制在一张图表中会使图表展示不清晰，如果在同一个页面绘制多个图表会浪费大量空间，pyecharts 中的 Timeline()类用来绘制时间线轮播多图，可以让数据随着时间轴动态变化，通过点击时间线上的不同时间来显示不同的图表。

Timeline()类中有一个 add()方法，使用该方法可以为时间线轮播多图添加图表和时间点，add()方法的语法格式如下：

add(chart, time_point)
add()方法的常用参数说明如表 4 所示。

表 4　add()常用参数

参数名称	说明
chart	图表实例
time_point	时间点

案例讲解：编写代码，绘制城镇居民和农村居民收入情况柱形图，并以时间线轮播多图的形式呈现。强调时间线轮播多图主要用于展示数据随时间的动态变化，能够直观地呈现数据的发展趋势和变化过程。

详细解释代码中每个部分的作用；

讲解模块的导入；

讲解数据的读取；

讲解如何为图表对象添加 x 轴和 y 轴数据；

讲解如何设置图表的标题；

解释代码中 Timeline 类的使用，通过循环创建不同时间点的图表对象，并使用 add 方法将其添加到时间线中；

讲解如何渲染图表。

学生实践：让学生根据给定的城镇居民和农村居民按收入五等份分组的收入情况数据，绘制 2019-2023 年的收入情况柱形图，并以时间线轮播多图的形式呈现，展示居民收入随时间的增长情况。学生根据自己对数据的理解，设计类似的时间线轮播多图场景，引导学生思考如何优化时间线的显示效果，如设置合适的播放速度、添加数据标签等。教师在学生实践过程中巡视指导，解决学生遇到的问题。

（四）任务二 居民主要食品消费量图形展示

1. 任务描述

从国家统计局公布的全国居民主要食品消费量数据可以看出，随着国家经济的快速发展，居民的主要食品消费量呈现稳定发展的趋势，说明随着生活水平的提升，居民更加注重饮食均衡和饮食健康。要求学生通过饼图展示消费结构占比，通过雷达图对比不同食品的营养均衡性。

2. 绘制饼图

概念讲解：饼图是一个划分为几个扇形的圆形统计图表，每个扇形的弧长（以及圆心角和面积）大小，表示该种类占总体的比例，且这些扇形合在一起刚好是一个完全的圆形。饼图由于采用圆形的特点，展示效果更直观，更易于展示占比情况。因此，饼图适用于想要突出表示某个部分在整体中所占比例，而且各不同分类间的占比差异明显的场景。

pyecharts 中 Pie()类用来绘制饼图，其绘制饼图的基本程序结构如下所示。

```
pie = (
Pie(init_opts=opts.InitOpts())     # 设置初始化配置项
    .add()                      # 设置数据
    .set_colors()                 # 设置颜色
    .set_global_opts()             # 设置全局配置项
    .set_series_opts()             # 设置系列配置项
)
pie.render_notebook()
```

案例讲解：编写代码，绘制全国居民主要食品消费量的饼图。展示一个已绘制好的居民主要食品消费量饼图案例，引导学生分析各类食品的占比情况，以及饼图在展示比例数据方面的优势。

详细解释代码中每个部分的作用；

讲解模块的导入；

讲解数据的读取；

讲解使用 add 方法添加数据，数据格式为元组列表，每个元组包含食品名称和消费量

讲解如何设置图表的标题；

讲解如何渲染图表。

学生实践：让学生根据给定的全国居民主要食品消费量数据，绘制 2022 年全国居民主要食品消费量的饼图。学生根据自己对数据的理解，设计类似的饼图场景，并根据实际情况调整图表的颜色、标签位置等属性，突出消费量最大的食品类别，使图表更加美观和清晰。教师在学生实践过程中巡视指导，解决学生遇到的问题。

3. 绘制雷达图

概念讲解：雷达图又称蜘蛛网图或极坐标图，将多个维度的数据量映射到起始于同一个圆心的坐标轴上，结束于圆周边缘，然后将同一组的点使用线连接起来。

利用雷达图，可以直观地展现多维数据集，查看哪些变量具有相似的值、变量之间是否有异常值。雷达图也可用于查看哪些变量在数据集内得分较高或较低，因此非常适合显示性能。

pyecharts 中 Radar()类用来绘制雷达图，其绘制雷达图的基本程序结构如下所示。

```
radar = (
Radar(init_opts=opts.InitOpts())     # 设置初始化配置项
    .add_schema()                # 设置雷达图参数
    .add()                       # 设置系列名称等参数
    .set_global_opts()             # 设置全局配置项
    .set_series_opts()             # 设置系列配置项
)
radar.render_notebook()
```

案例讲解：编写代码，绘制全国居民人均蔬菜及食用菌、肉类、蛋类、奶类和干鲜瓜果类消费量的雷达图。解释雷达图适用于展示多个维度数据的综合情况，在居民主要食品消费量分析中，可以将不同食品的消费量作为不同维度，通过雷达图直观地看到各类食品消费量在整体中的相对位置和综合水平。

详细解释代码中每个部分的作用；

讲解模块的导入；

讲解数据的读取；

讲解使用 add_schema 方法将指标体系添加到雷达图中；

讲解使用 add 方法添加数据，数据格式为一个列表，每个元素对应一个数据点在各个指标上的值；

讲解通过 set_global_opts 设置图表标题；

讲解如何渲染图表。

学生实践：让学生根据给定的 2014-2023 年全国居民主要食品消费量数据，绘制 2020-2023 年全国居民人均蔬菜及食用菌、肉类、蛋类、奶类和干鲜瓜果类消费量的雷达图。学生根据自己对数据的理解，设计类似的雷达图场景，引导学生观察雷达图的形状，分析各类食品消费量在多个维度上的分布特点，以及如何通过雷达图进行不同食品消费量的综合比较。教师在学生实践过程中巡视指导，解决学生遇到的问题。

六、知识拓展

1. 其他可视化库介绍

简要介绍除 pyecharts 之外的其他常用数据可视化库，如 matplotlib、seaborn 等。

对比这些库与 pyecharts 在功能、适用场景、语法特点等方面的异同。例如，matplotlib 功能强大，基础绘图能力出色，但语法相对复杂；seaborn 在统计图表绘制方面有独特优势，图形风格更加美观；而 pyecharts 专注于 Web 端可视化，交互性强。

2. 数据可视化的最佳实践

讲解数据可视化的基本原则，如简洁性（避免图表过于复杂，突出关键信息）、准确性（数据准确无误，图表解读符合数据实际情况）、美观性（合理选择颜色、字体、布局等，提升图表的视觉效果）。

展示一些优秀的数据可视化案例，分析它们是如何遵循这些原则，以及如何通过巧妙的设计和布局更好地传达数据信息。同时，展示一些反面案例，让学生分析其中存在的问题，加深对最佳实践的理解。

3. 动态图表的交互设计

针对时间线轮播多图等具有交互功能的图表，介绍交互设计的重要性和一些常用的交互方式。例如，添加鼠标悬停提示，让用户在鼠标移动到图表元素上时显示详细的数据信息；设置图表的缩放功能，方便用户查看局部数据；添加动画效果，增强图表的动态展示效果，但要注意动画不能过于复杂，以免干扰用户对数据的理解。

七、项目小结

1. 知识总结

回顾整个项目中涉及的知识点，包括 pyecharts 的安装与使用、各种图表（柱形图、折线图、散点图、箱形图、矩形树图、漏斗图、层叠多图、并行多图、顺序多图、选项卡多图、时间线轮播多图、饼图、雷达图）的绘制方法和适用场景。

强调不同图表在展示人民生活数据时的优势和局限性，以及如何根据数据特点和展示需求选择合适的图表类型。

2. 能力提升

总结学生在项目实践过程中所提升的能力，如数据处理能力（数据收集、整理和清洗）、编程能力（使用 pyecharts 编写绘图代码）、图形设计能力（优化图表的颜色、标签、布局等）和数据分析能力（通过图表解读数据背后的信息）。

鼓励学生在今后的学习和工作中，继续运用所学的数据可视化技能，将复杂的数据以直观、清晰的方式呈现出来。

3. 素质培养

强调在项目实施过程中培养的严谨、细致的工作态度和团队协作精神。在数据处理和图表绘制过程中，任何一个小的错误都可能导致结果的偏差，因此要养成认真核对数据和代码的习惯。同时，鼓励学生在小组合作中相互学习、交流经验，共同解决问题，提高团队协作能力。