【实验目的】

1. 了解Anaconda环境设置

2. 了解Python的基本数据分析方法和绘图方法。

【实验学时】

建议2学时

【实验环境配置】

1. Anaconda的基本配置和使用

1) 从清华镜像源（https://mirrors.tuna.tsinghua.edu.cn/）下载Anaconda 或 Miniconda，并配置系统环境(Python 3.11 x86-64 windows版本)。

2) 创建新的配置环境：conda create -n myics

3) 激活配置环境：conda activate myics

4) 退出配置环境：conda deactivate

2. Trae编辑器

3. VS Code 编辑器

4. PyCharm编辑器。

【实验原理】

1. Python的基本原理与使用：基于解释型、面向对象编程范式，通过简洁语法实现数据处理、脚本编写等，是数据分析与 AI 领域的基础工具。
学习链接：

Ø Python 基础教程 | 菜鸟教程

Ø 简介 - Python教程 - 廖雪峰的官方网站

Ø Welcome to Python.org

2. Markdown基本语法与使用：通过简单符号（如 `#`、`*`）实现文本格式排版，无需复杂操作即可生成结构化文档，常用于笔记、文档撰写与代码注释。

学习链接：

Ø Markdown 教程 | 菜鸟教程

Ø Markdown 基本语法 | Markdown 教程

Ø Markdown 语法速查表 | Markdown 官方教程

3. Print的基本语法与使用：Python 中 print() 是常用的输出函数，基本语法为：print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)

主要参数说明：

Ø *objects：需输出的对象（可多个，用逗号分隔）；

Ø sep：多个对象间的分隔符，默认空格；

Ø end：输出结束符，默认换行符 \n；

Ø file：输出目标，默认控制台，可指定文件对象。

【实验步骤】

1. 使用conda设置实验环境，运行jupyter lab。

2. 验证性实验内容：

1）数据的输入与输出

Print()函数：

[24]:
```
print("Hello Golang", "Hello PHP")          # 默认换行，以空格作为分隔符
print("Hello Java | ", end="")  # 此行输出后不换行
print("Hello Word", "Hello Python", sep="#")  # 以“#”号作为分隔符
```

```
Hello Golang Hello PHP
Hello Java | Hello Word#Hello Python
```

转义符：

[26]:
```
# 输出购物清教
print('\t\t\t购物清单')
print('商品名称\t',  end='')
print('购买数量\t',  end='')
print('商品单价\t', end='')
print('金额')

print('苹果\t', end='')
print('\t  1\t', end='')
print('\t  8\t', end='')
print('\t  8')

print('香蕉\t', end='')
print('\t  2\t', end='')
print('\t  4\t', end='')
print('\t  8')
```

```
                                购物清单
商品名称 购买数量 商品单价 金额
苹果                    1                    8                    8
香蕉                    2                    4                    8
```

Input函数

```
[30]: price = input('input the stock price of Apple: ')
      print(price)
      print(type(price))
```

```
input the stock price of Apple: 109
109
<class 'str'>
```

```
[31]: price1 = int(input('input the stock price of Apple: '))
      price2 = eval(input('input the stock price of Apple: '))
      print(price1, type(price1), sep='---')
      print(price2, type(price2), sep='---')
```

```
input the stock price of Apple: 109
input the stock price of Apple: 109
109---<class 'int'>
109---<class 'int'>
```

2）数据类型验证：

整型：

```
[1]: money = 1000
     print(money)
     print(type(money))
```

```
1000
<class 'int'>
```

```
[3]: a = 0b1010    # 二进制
     b = 0o257     #  八进制
     c = 0x1234    #  十六进制
     print(a,type(a))
     print(b,type(b))
     print(c,type(c))
```

```
10 <class 'int'>
175 <class 'int'>
4660 <class 'int'>
```

复数：

[5]:
```
var5 = 3 + 5j
var6 = complex(3.4e5, 7.8)
print(var5, type(var5))
print(var6, type(var6))
```

```
(3+5j) <class 'complex'>
(340000+7.8j) <class 'complex'>
```

布尔型：

[6]:
```
i_love_you = True
you_love_me = False
print(i_love_you, type(i_love_you))
print(you_love_me, type(you_love_me))
```

```
True <class 'bool'>
False <class 'bool'>
```

字符串：

[7]:
```
var7 = 'Hello Python'
var8 = "Hello Python"
print(var7, type(var7))
print(var7, type(var7))
```

```
Hello Python <class 'str'>
Hello Python <class 'str'>
```

[8]:
```
var9 = "China's National Day"
var10 = 'He said "Hello" to me'
print(var9)
print(var10)
```

```
China's National Day
He said "Hello" to me
```

[11]:
```
var11 = """这是第一行
这是第二行
这是第三行"""
print(var11)
```

```
这是第一行
这是第二行
这是第三行
```

思考：python如何进行数据类型的转换

在实验中，我们通过输入输出和数据类型验证的操作，发现Python里数据类型转换其实挺简单的，主要靠几个内置函数就能搞定。

比如，用 `input()` 输入的东西默认都是字符串，但很多时候我们需要整数或小数来计算，这时候就用 `int()` 或 `float()` 转一下。

像 `age = int(input("输入年龄: "))` ，就把**字符串转成整数**了。

反过来，**数字想变成字符串**方便打印，就用 `str()` ，比如 `print("结果是: " + str(3.14))` 。

布尔型转换也常用，像 `bool(0)` 会变成 `False` ，而 `bool("abc")` 就是 `True` 。

复数的话，可以用 `complex("2+3j")` 直接从字符串转。不过要注意，转换时得小心格式，比如 `int("12.3")` 会报错，得先转成 `float` 再处理。

实验中做类型验证时，这些转换特别实用，能帮我们快速检查数据是否符合要求。总之，多试试这几个函数，结合 `try-except` 防报错，用起来就很顺手了。

3. 综合练习：

1）购物结算

根据上表的内容，设置相关变量，完成消费金额总计，用python的print语言输出相应内容。

购买物品	单价	个数	金额
T恤	￥245	2	￥490
网球鞋	￥570	1	￥570
网球拍	￥320	1	￥320
折扣	-	-	8折
消费总金额	-	-	￥1104.0
实际交费	-	-	￥1500
找钱	-	-	￥396.0

```
print('\t\t\t购物清单')
print('商品名称\t', end='')
print('购买数量\t', end='')
print('商品单价\t', end='')
print('金额')

print('T恤\t', end='')
print('\t 2\t', end='')
print('\t ￥245\t', end='')
print('\t ￥490')

print('网球鞋\t', end='')
print('\t 1\t', end='')
print('\t ￥570\t', end='')
```

```
print('\t ￥570')

print('网球拍\t', end='')
print('\t 1\t', end='')
print('\t ￥320\t', end='')
print('\t ￥320')

print('折扣\t', end='')
print('\t -\t', end='')
print('\t -\t', end='')
print('\t 8折')

print('消费总金额\t', end='')
print('\t -\t', end='')
print('\t -\t', end='')
print('\t ￥1104.0')

print('实际交费\t', end='')
print('\t -\t', end='')
print('\t -\t', end='')
print('\t ￥1500')

print('找钱\t', end='')
print('\t -\t', end='')
print('\t -\t', end='')
print('\t ￥396.0')
```

我最开始是写的这个 按照上面的一个转义符的例子 但是我看对齐不了（每一列不能完全对齐）我就重新写了一个

```
# 使用固定宽度格式化输出
print('\t\t\t购物清单')
print(f'{"商品名称":<10}\t{"购买数量":<8}\t{"商品单价":<8}\t{"金额":<8}')

print(f'{"T恤":<10}\t{2:<8}\t{"￥245":<8}\t{"￥490":<8}')
print(f'{"网球鞋":<10}\t{1:<8}\t{"￥570":<8}\t{"￥570":<8}')
print(f'{"网球拍":<10}\t{1:<8}\t{"￥320":<8}\t{"￥320":<8}')
print(f'{"折扣":<10}\t{"-":<8}\t{"-":<8}\t{"8折":<8}')
print(f'{"消费总金额":<10}\t{"-":<8}\t{"-":<8}\t{"￥1104.0":<8}')
print(f'{"实际交费":<10}\t{"-":<8}\t{"-":<8}\t{"￥1500":<8}')
print(f'{"找钱":<10}\t{"-":<8}\t{"-":<8}\t{"￥396.0":<8}')
```