把一段 Python 写对、读懂并清楚输出

第 2.1-2.4 节｜代码格式、标识符、变量、输入输出与实训

Python 零基础培训

90 分钟课堂

今天只解决一件事：让数据顺利走完整条链

让解释器读懂代码，让名称准确指向值，让输入能计算，让输出可检查。

- 读懂注释、缩进与换行表达的层级
- 写出合法、清楚、稳定的变量名
- 识别常见类型，理解 `input()` 的字符串结果
- 完成购物小票与植树证书两个输出任务

“能运行”不代表“算对了”

```
price = "19.9"
quantity = 3
print(price * quantity)
```

先预测再运行

结果会重复字符串三次。

问题不在乘号，而在 price 的类型。

`float(price) * quantity`

注释负责解释口径，代码负责执行

```
# 退单商品不计入净销售数量
net_quantity = sales_quantity - return_quantity
```

- 单行注释以 # 开始；# 后留一个空格
- 行尾注释与代码之间至少留出清楚间隔
- 三引号常用于函数、类、模块的说明文档

好注释回答“为什么这样算”，不机械复述代码。

缩进不是装饰，它决定“这一行属于谁”

同一代码块对齐

```
if quantity > 0:
    print(" 有销量")
    print(" 继续计算")
```

层级不一致

```
if quantity > 0:
    print(" 有销量")
  print(" 继续计算")
```

默认使用 **4 个空格**；不要混用 Tab 与空格。

长条件用括号换行，结构会自己说话

```
is_target = (
    sales_quantity >= 100
    and unit_price < 200
    and return_quantity <= 10
)
```

- 小括号、中括号、大括号内部可以自然换行
- 同层条件对齐，能快速发现漏项和分组错误
- 反斜杠换行容易受行尾空格影响，初学阶段少用

变量名先过三关：合法、可读、一致

名称	判断	原因
`sales_quantity`	**推荐**	小写下划线，业务含义明确
`2price`	**非法**	不能以数字开头
`class`	**非法**	Python 关键字
`Sales / sales`	注意	都合法，但区分大小写

变量与函数常用 `snake_case`；常量约定使用全大写。

关键字由 Python 保留，运行时可以直接检查

```
import keyword

print(keyword.iskeyword("class"))
print(keyword.kwlist)
```

- 关键字参与语法，不能重新当作变量名
- class、if、for、import 都是关键字
- 关键字集合会随 Python 版本变化，查询比死记数量更可靠

变量是名称对当前值的引用

```
quantity = 125
quantity = 130
quantity_text = "130"
```

- = 建立赋值关系；== 比较两个值
- 后一次赋值改变名称当前指向的值
- Python 允许动态改变类型，但业务变量应保持含义稳定

六类数据，先会辨认再深入操作

类型	示例	业务含义
int	125	数量、件数
float	199.0	价格、比例
str	" 双肩包"	商品名、店铺名
bool	True	条件是否成立
list	[19.9, 29.9]	一组有序数据
dict	{"price": 199}	带字段名的记录

用 type(value) 检查类型；它不会自动转换原值。

input() 返回文本：计算前必须转换

```
price_text = input(" 单价: ")
qty_text = input(" 数量: ")

price = float(price_text)
quantity = int(qty_text)
total = price * quantity
```

输入 → 转换 → 计算

`"19.9" * 3` 会重复文本。
`19.9 * 3` 才进行数值乘法。
看到数字外观，不等于已经是数字类型。

print() 让结果既能展示，也能检查

```
print(product, quantity, unit_price, sep=" | ")
print(f" 销售额：{quantity * unit_price:.2f} 元")
print("quantity =", repr(quantity),
      "type =", type(quantity).__name__)
```

- sep 控制多个对象之间的分隔
- end 控制结尾；默认换行
- f-string 负责清晰展示；repr + type 负责诊断

实训一：购物小票 = 数据 + 计算 + 固定输出

```
def print_receipt(product, quantity, unit_price):
    total = quantity * unit_price
    print("=" * 24)
    print(f" 商品：{product}")
    print(f" 数量：{quantity}")
    print(f" 单价：{unit_price:.2f} 元")
    print(f" 合计：{total:.2f} 元")
    return total
```

返回总额，后续才能继续汇总；输出格式不改变原始数值。

实训二：证书模板应把变化数据留成参数

```
def print_certificate(nickname, plant_name, cert_id):
    print(" 植 树 证 书")
    print(f" 申请人：{nickname}")
    print(f" 植物：  {plant_name}")
    print(f" 编号：  {cert_id}")

print_certificate(" 小数同学", " 梭梭树",
                  "TREE-2026-0722")
```

固定结构保持一致，**变化字段**通过参数传入。

报错时按固定顺序排查，速度会越来越快

现象	常见原因	第一检查点
缩进错误	层级不一致	同一代码块是否对齐
语法错误	非法名称、漏引号	当前行及上一行
名称错误	名称未定义或拼错	赋值是否先执行
类型错误	类型不兼容	`repr()` 与 `type()`
重复文本	数字仍是字符串	是否调用 `int/float`

离场任务：独立走完一次数据链

输入两个数字文本，转换后计算销售额，再输出带标签的两位小数结果。

- 变量名表达商品、数量、单价与总额
- 至少一条注释解释业务口径
- 代码可从上到下一次运行
- 能口头说明 = 与 == 的差别

格式让代码可执行，命名让代码可读，类型让数据可计算，输出让结果可验证。

下一步：打开 课后练习.py，只修改 TODO。