Lab 10 · Part 4 抽象与组织 AI Level 1

类（二）：继承、特殊方法与模块

为 Rover 增加新角色，让图形、金额和牌组支持统一操作，再将图形报告拆成三个真实文件。

截止：小作业 4 · 12 月 8 日 周二 23:59

从课堂到实验

课堂上的探险队已经能通过继承扩展角色，通过特殊方法支持 Python 操作。本次实验先用四个小任务练习这些机制，最后复用图形代码，做一个可以独立运行的**图形面积报告程序**。

下载实验骨架与公开测试 ZIP · 打开 HW4 提交页

使用 Python 3.13 和标准库。下载包中的前四个目录各对应一道题；在相应目录编辑 `submission.py`，运行 `python -m pytest -q test_public.py`。如果尚未安装 pytest，使用 `python -m pip install pytest`。模块工程有独立的三文件目录和打包脚本。

部分	任务	自动分	人工分
A	图形、Rover 子类、Money、Deck	16	8
B	三文件图形报告	8	4
C	环境观察，选做	0	0

第10周合计 **24 自动分＋12 人工分**。HW4 覆盖第10–12周，截止以提交页和课程校历为准。

AI 使用：Level 1

可以通过聊天询问思路、解释报错、调试或修改局部代码。请自己完成整体实现，不让 AI 生成整题完整答案，不使用 coding agent 自动编辑和运行工程。**在 HW4 末尾填写一次 AI 使用说明**，说明会进入评测报告；各题和文字卡无需重复填写。未使用 AI 也请如实说明。最终仍只需提交评测报告。

各代码题独立提交，保留骨架中提供的辅助函数或基类，不依赖课堂上已经运行的代码。补全方法的类型提示，`self` 无需注解。评测调用接口，不使用 `input()`。

A 段 · 类与特殊方法

10-1 图形家族

目标：让同一个 `total_area()` 处理不同图形。

```
class Shape:
    def area(self) -> float: ...
    def name(self) -> str: ...
class Circle(Shape):
    def __init__(self, r: float) -> None: ...
    def area(self) -> float: ...
class Rectangle(Shape):
    def __init__(self, w: float, h: float) -> None: ...
    def area(self) -> float: ...
class Triangle(Shape):
    def __init__(self, a: float, b: float, c: float) -> None: ...
    def area(self) -> float: ...
def total_area(shapes: list[Shape]) -> float: ...
```

骨架提供 `positive_dimensions(*values)` 和 `triangle_area(a, b, c)`：前者检查有限正数，后者检查三角形并计算面积，非法时抛 `ValueError`。直接调用即可，无需自己推导公式。圆面积用 `math.pi`。

- `Shape.area()` 抛 `NotImplementedError`；`name()` 返回实际类名，例如 `Circle`。三个图形继承 `Shape`，各自保存尺寸并重写 `area()`。

- 构造时调用提供的辅助函数拒绝非法尺寸，包括不满足严格三角不等式的三角形。
- `total_area()` 逐个调用对象的 `area()`，空列表返回 `0`，不修改列表或对象。以后新增其他 `Shape` 子类也应能直接参与计算，无需按类名分支。

```
items = [Circle(1), Rectangle(2, 3), Triangle(3, 4, 5)]
# [item.name() for item in items] -> ["Circle", "Rectangle", "Triangle"]
# total_area(items) -> math.pi + 12
```

入口 `w10-shapes`，4 自动分。测试维度为适中大小的实数；不要求为其他对象类型提供自动转换。

10-2 Rover 的新角色

目标：复用基类初始化和移动行为，为调查员增加采样记录。

骨架已完整提供 `Rover(name, x=0, y=0)`：每个实例有 `name`、`x`、`y` 和独立的 `bag` 列表；`move(dx, dy)` 改变坐标，`collect(item)` 向 `bag` 追加物品，两者返回 `None`。`report()` 返回 `"Ada @ (3, 4)"` 这样的字符串。

保留基类，补全子类的三个方法：

```
class SurveyRover(Rover):
    def __init__(self, name: str, x: int = 0, y: int = 0) -> None: ...
    def record(self, sample: str) -> None: ...
    def report(self) -> str: ...
```

1. `__init__()` 用 `super().__init__()` 建立基类状态，再建立每个实例独立的 `samples` 列表。
2. `record(sample)` 向 `samples` 追加一条字符串记录，允许重复，返回 `None`。采样记录与背包物品分开。
3. `report()` 调用 `super().report()`，在其后加上 `" | samples: N"`，`N` 为记录条数。继承 `move()` 和 `collect()`，无需重写。

```
r = SurveyRover("Ada", 3, 4)
r.move(2, -1)
r.collect("apple")
r.record("rock")
# r.report() -> "Ada @ (5, 3) | samples: 1"
# r.bag -> ["apple"]; r.samples -> ["rock"]
```

输入保证名字、物品和样本为字符串，坐标和位移为整数，无需添加地图边界或其他游戏规则。公开测试会临时替换基类方法，检查子类确实复用了基类初始化和报告，而非复制其实现。

入口 `w10-rover`，4 自动分。

10-3 Money

目标：让金额对象支持显示、加减和比较。

骨架提供 `to_cents(yuan)`，将元转换为整数分并完成校验、舍入。例如 `1.005` 得到 `101`，`-1.005` 得到 `-101`；接受有限的 `int/float` 且绝对值不超过 `10**9`，否则抛 `ValueError`。本题直接使用此函数，无需实现 Decimal 舍入。

```
class Money:
    def __init__(self, yuan: float) -> None: ...
    def __repr__(self) -> str: ...
    def __add__(self, other: "Money") -> "Money": ...
    def __sub__(self, other: "Money") -> "Money": ...
    def __eq__(self, other: object) -> bool: ...
    def __lt__(self, other: "Money") -> bool: ...
```

- `__init__()` 将 `to_cents(yuan)` 存为 `self._cents`。`repr()` 固定两位小数，例如 `Money(12.50)`、`Money(-1.01)`、`Money(0.00)`。
- 加减返回**新 Money 对象**，原操作数不变。结果仍保存整数分，允许超过构造入参的范围；可先创建 `Money(0)` 再设置结果的 `_cents`，避免经过浮点转换。

- 相等与排序按分比较。四个二元特殊方法收到非 Money 对象时均返回 `NotImplemented`，交给 Python 处理后续协议。

```
# Money(0.1) + Money(0.2) == Money(0.3) -> True
# repr(Money(10.5) - Money(0.25)) -> "Money(10.25)"
# Money(1) == object() -> False
# Money(1) + 1 -> TypeError
```

入口 `w10-money`，4 自动分。

10-4 Deck

目标：让自定义牌组像容器一样使用。

骨架已提供 `__init__()` 和 `shuffle()`，保留即可。初始牌组为52张独立牌面，点数 `A/2/…/10/J/Q/K`，按花色 `C/D/H/S` 排列，存放在每个实例自己的 `self._cards` 列表中。

```
class Deck:
    def __init__(self) -> None: ...        # 已提供
    def __len__(self) -> int: ...
    def __getitem__(self, i: int) -> str: ...
    def __contains__(self, card: str) -> bool: ...
    def shuffle(self) -> None: ...         # 已提供
    def deal(self, n: int) -> list[str]: ...
```

- `len(deck)` 返回剩余张数；下标支持正负整数，越界抛 `IndexError`。`card in deck` 返回是否仍有这张牌。
- `deal(n)` 从**开头**取走 `n` 张，返回独立的新列表。`n` 须为非负整数且不超过余量，非法时抛 `ValueError`，牌组保持原样；`deal(0)` 返回空列表。
- 两个牌组互不影响。`__getitem__` 在越界时抛异常后，Python 也能通过序列协议执行 `list(deck)`；无需另写 `__iter__()`。

```
# deck = Deck(): len(deck) -> 52; deck[0] -> "AC"; deck[-1] -> "KS"
# deck.deal(2) -> ["AC", "2C"]; len(deck) -> 50; "AC" in deck -> False
```

洗牌调用骨架中的 `random.shuffle(self._cards)`，原地改变顺序并返回 `None`。测试会用一个已知排列替换随机函数，核对提供的委托行为；不考随机算法，不要求每次洗牌都改变顺序。

入口 `w10-deck`，4 自动分。

A 机制与验证记录

入口 `w10-a-notes`，只回答两问，各4人工分：

1. 结合自己的 SurveyRover，说明 `super().__init__()` 建立哪些属性，`report()` 如何扩展基类行为；给出两个实例的操作及实际结果，检查列表是否共享。
2. 从 Money 或 Deck 选一个表达式，解释对应的特殊方法；给出一次非正常输入或状态变化的预期、实际结果，说明原对象是否变化。

每问写清机制和一个小例子即可，无需逐题复述接口。

B 段 · 三文件图形报告

10-5 拆分与导入

目标：复用第一题，让一个真正的多文件程序生成报告。

下载包的 `module-project/` 包含三个业务文件和工具脚本：

```
shapes.py           # 复用第一题完成的图形定义和 total_area
report.py           # 将图形对象整理为报告文本
main.py             # 创建示例，打印报告
pack_submission.py  # 已提供，生成平台提交文件
```

补全后，在该目录运行 `python main.py`，输出应为：

```
Rectangle: 6.00
Triangle: 6.00
Total: 12.00
```

具体接口：

- `shapes.py`：放入第一题完成的代码，不依赖 `submission.py`。
- `report.py`：实现 `build_report(items: list[Shape]) -> list[str]`。按原顺序输出每个对象的 `name()` 和 `area()`，面积保留两位小数；最后一行用从 shapes 导入的 `total_area(items)` 生成合计。空列表返回 `["Total: 0.00"]`。只返回文本，不打印、不修改输入，新增 Shape 子类也应能使用。
- `main.py`：实现 `run_demo() -> None`。通过 shapes 中的类创建 `Rectangle(2, 3)` 和 `Triangle(3, 4, 5)`，调用 report 中的 `build_report()`，逐行打印其结果。用 `if __name__ == "__main__":` 保护入口。

导入三个模块均不应输出或启动演示。 可以使用 `import module` 或 `from module import name`，依赖方向为 `main → report → shapes`，main 同时使用 shapes 中的构造器。公开测试会临时替换报告函数，检查主程序真实调用了模块接口；仅打印固定答案无法通过。

提交步骤：

```
python main.py
python -c "import shapes, report, main"
python pack_submission.py
python -m pytest -q test_public.py
```

第二条命令应无输出。打包脚本生成 `submission.py`，其中的 `project_files() -> dict[str, str]` 返回三个文件名及完整源码；将**生成文件的内容**粘贴到 `w10-module-project` 提交。每次修改工程后重新打包，无需手动将源码塞进字符串。单独下载打包脚本。

评测只使用这三个文件，工程仅依赖标准库，不读本机文件、不使用网络、`input()` 或图形窗口。`project_files()` 只返回源码，不运行工程。

入口 `w10-module-project`，8 自动分。文字入口 `w10-b-notes` 两问各2人工分：三文件职责及新增图形是否要改报告；直接运行与导入的实际命令、输出及原因。

C 段 · 环境观察（选做）

在模块工程目录运行：

```
python -c "import sys; print(sys.executable)"
python -c "import shapes; print(shapes.__file__)"
```

观察当前解释器和实际导入文件的路径。愿意记录时填写 `w10-c-tools`，也可留空，不影响完成与评分。

四、个人作业2启动

本周同时启动个人作业2：素数计数五法。它与 HW4 分开提交，截止以个人作业提交页和课程校历为准。先了解任务，算法细节在 L11 继续讲解。

▶ 查看个人作业2的完整要求（提交页也会显示）

提交自查

- 四道类练习和模块工程分别提交，两个文字卡填写简短机制说明和实际结果。
- 模块工程修改后重新打包，直接运行有报告，导入无输出。
- HW4 末尾统一填写 AI 使用说明，下载评测报告提交。