第 6 周 Python 代码组织

1. 创建 conda 环境

```yaml
! environment.yml
1   name: week06
2   channels:
3     - conda-forge
4   dependencies:
5     - python=3.12
6     - wat-inspector
```

```
(base) cherry@LAPTOP-QR2UKG4V MINGW64 ~/repo/week06 (main)
$ conda activate week06
(week06)
cherry@LAPTOP-QR2UKG4V MINGW64 ~/repo/week06 (main)
```

2. 创建一个 guessing_game.py 文件，运用 pdb 调试器理解其运行流程

```
 3
 4      def guessing_game():
 5          # 生成 1 到 100 之间的随机整数
 6  ->      secret_number = random.randint(1, 100)
 7          n = 0
 8
 9          print("欢迎来到猜数字游戏！我已经想好了一个 1 到 100 之间的数
10
11          while True:
```

调用函数，生成 1~100 之间的随机数

```
11  ->      while True:
12              n += 1
13              # 获取玩家输入
14              guess = input(
15                  f"(第 {n} 次尝试) 请输入你猜的数字 (输入整数，或者输
```

条件循环，条件就是 True

```
(Pdb) n
> c:\users\cherry\repo\week06\guessing_game.py(12)guessing_game()
-> n += 1
```

n=n+1

```
(Pdb) p n
1
(Pdb) n
(第 1 次尝试）请输入你猜的数字（输入整数，或者输入 q 回车退出）: 10
> c:\users\cherry\repo\week06\guessing_game.py(17)guessing_game()
-> guess = guess.strip()  # 去除多余空白字符
```

```
(Pdb) p guess
'10'
(Pdb) p type(guess)
<class 'str'>
```

- 猜的数字是字符串

```
(Pdb) p '  ab cd   '.strip()
'ab cd'
```

- 只去除两边的空格

```
 19  ->          if guess == "q":
 20                  break
 21
```

- 如果猜的数字 guess="q"，那么就会 break 即跳出 while 循环，反之则会忽略 break 继续往下运行

```
(Pdb) p guess == 'q'
True
(Pdb) n
> c:\users\cherry\repo\week06\guessing_game.py(46)guessing_game()
-> print("游戏结束，再见👋。")
```

- 运行 break 跳出循环

```
(Pdb) p guess
'60'
(Pdb) p int(guess)
60
```

- 把字符串转化为整数

```
22              try:
23  ->              guess = int(guess)
24              except ValueError:
25                  print("输入无效🙅，请输入一个整数。")
26                  continue
```

- 流程控制语句，如果输入的不是整数，就会输出“输入无效，请输入一个整数”，反之则继续

```
(Pdb) p guess
'67.8'
(Pdb) p int(guess)
*** ValueError: invalid literal for int() with base 10: '67.8'
(Pdb) n
ValueError: invalid literal for int() with base 10: '67.8'
> c:\users\cherry\repo\week06\guessing_game.py(23)guessing_game()
-> guess = int(guess)
(Pdb) n
> c:\users\cherry\repo\week06\guessing_game.py(24)guessing_game()
-> except ValueError:
(Pdb) n
> c:\users\cherry\repo\week06\guessing_game.py(25)guessing_game()
-> print("输入无效🙅，请输入一个整数。")
(Pdb) n
输入无效🙅，请输入一个整数。
```

- 如果输入的不是整数，则会输出以上结果

```
(Pdb) p n
2
(Pdb) n
> c:\users\cherry\repo\week06\guessing_game.py(15)guessing_game()
-> f"(第 {n} 次尝试) 请输入你猜的数字 (输入整数，或者输入 q 回车退出):
(Pdb)
> c:\users\cherry\repo\week06\guessing_game.py(14)guessing_game()
-> guess = input(
(Pdb)
(第 2 次尝试) 请输入你猜的数字 (输入整数，或者输入 q 回车退出):
```

- 接下来会进行 continue，即跳至下一轮循环，继续从 while 循环开始，进行第二次尝试

```
 36  ->          if guess < secret_number:
 37                  print("猜的数字太小了，再试试↗。")
 38                  continue
 39
 40              if guess > secret_number:
 41                  print("猜的数字太大了，再试试↘。")
(Pdb) n
> c:\users\cherry\repo\week06\guessing_game.py(40)guessing_game()
-> if guess > secret_number:
(Pdb) p guess < secret_number
False
```

```
(Pdb) p guess
67
(Pdb) p secret_number
20
(Pdb) p guess > secret_number
True
(Pdb) n
> c:\users\cherry\repo\week06\guessing_game.py(41)guessing_game()
-> print("猜的数字太大了，再试试↘。")
(Pdb) n
猜的数字太大了，再试试↘。
> c:\users\cherry\repo\week06\guessing_game.py(42)guessing_game()
-> continue
```

- 运行 continue 会进行下一轮的 while 循环

```
$  python guessing_game.py
欢迎来到猜数字游戏！我已经想好了一个 1 到 100 之间的数字，你可以开始猜啦。
(第 1 次尝试) 请输入你猜的数字 (输入整数，或者输入 q 回车退出): 40
猜的数字太小了，再试试↗。
(第 2 次尝试) 请输入你猜的数字 (输入整数，或者输入 q 回车退出): 60
猜的数字太大了，再试试↘。
(第 3 次尝试) 请输入你猜的数字 (输入整数，或者输入 q 回车退出): 50
猜的数字太大了，再试试↘。
(第 4 次尝试) 请输入你猜的数字 (输入整数，或者输入 q 回车退出): 47
猜的数字太大了，再试试↘。
(第 5 次尝试) 请输入你猜的数字 (输入整数，或者输入 q 回车退出): 44
猜的数字太小了，再试试↗。
(第 6 次尝试) 请输入你猜的数字 (输入整数，或者输入 q 回车退出): 45
恭喜你🎉，猜对了！
游戏结束，再见👋。
```

- 正式运行
- 在循环判断时可以先判断最特殊的情况

3. 调用函数

1) Func1，没有形参，没有返回值

mylib.py > ...

```python
def func1():
    x = 50
    y = x**0.5 - 7
    print(y)
```

- 在 mylib 中定义函数，没有 return，返回值是 none

myjob.py

```python
import mylib  # noqa: F401

mylib.func1()
```

```
cherry@LAPTOP-QR2UKG4V MINGW64 ~/repo/week06 (main)
$ python myjob.py
0.0710678118654755
```

- 在 myjob 中调用函数

```python
y = mylib.func1()
print(y)
```

```
$ python myjob.py
0.0710678118654755
None
```

- 没有返回值，返回值是 none

```
6   try:
7       y = mylib.func1(0)
8   except TypeError as e:
9       print(e)
```

```
$ python myjob.py
0.0710678118654755
None
func1() takes 0 positional arguments but 1 was given
```

- 定义函数时没有形参，所以不能往里面传参数

2) Func2，没有形参，有返回值

```
7   def func2():
8       x = 120
9       y = x**0.5 - 7
10      print(y)
11      return y
```

```
11  y = mylib.func2()
12  print(y)
```

```
$ python myjob.py
0.0710678118654755
None
func1() takes 0 positional arguments but 1 was given
3.9544511501033224
3.9544511501033224
```

- 有返回值。一旦遇到 return 就会跳出这个函数。

3) Func3，只有一个位置形参

```
14  def func3(x):
15      y = x**0.5 - 7
16      return y
```

位置形参

- 位置形参：只是一个符号，没有具体的值，在调用的时候回往里面传具体的参数即实参。

```
14  y = mylib.func3(70)
15  print(y)
```

位置实参

```
func1() takes 0 positional arguments but 1 was given
3.9544511501033224
3.9544511501033224
1.3666002653407556
```

```
17    y = mylib.func3(x=55)
18    print(y)
```

```
1.3666002653407556
0.4161984870956663
```

- 命名实参

```
20    try:
21        y = mylib.func3()
22    except TypeError as e:
23        print(e)
```

```
0.4161984870956663
func3() missing 1 required positional argument: 'x'
```

- 不传实参

4) Func4，只有一个命名形参

```
19    def func4(x=55):
20        y = x**0.5 - 7
21        return y
```

命名形参

```
30    y = mylib.func4(50)
31    print(y)
32
33    y = mylib.func4(x=49)
34    print(y)
```

位置实参

命名实参

```
0.07106781186547524
0.0
```

```
36    y = mylib.func4()
37    print(y)
```

```
0.0
0.4161984870956663
```

- 不传参数，默认值

5) Func5，多个位置形参和命名形参

```python
# 定义函数，其中 base_score 和 bonus 是位置形参，special_reward 是命名形参
def calculate_score(base_score, bonus, special_reward=False):
    total_score = base_score + bonus
    if special_reward:
        total_score += 10
    return total_score
```

```python
score1 = mylib.calculate_score(80, 15, False)
print(score1)
```

```
0.4161984870956639
95
```

- 使用位置实参

```python
score1 = mylib.calculate_score(bonus=15, base_score=80, special_reward=False)
print(score1)
```

```
95
95
```

- 使用命名实参，可打乱顺序

```python
score4 = mylib.calculate_score(90, 16, special_reward=True)
print(score4)
```

```
95
95
95
116
```

- 位置实参和命名实参混合使用
- 在定义函数时位置参数应该排在命名参数前面

6) Func6，在形参列表中使用 / 来限定只接受位置实参的形参

```python
def func6(base_score, /, bonus, special_reward=False):
    total_score = base_score + bonus
    if special_reward:
        total_score += 10
    return total_score
```

形参列表

在形参列表中加入“/”后，“/”左边只能按照位置形参传参数

7) Func7，在形参列表中使用 * 来限定只接受命名实参的形参

```python
def func7(base_score, /, bonus, *, special_reward=False):
    total_score = base_score + bonus
    if special_reward:
        total_score += 10
    return total_score
```

在形参列表中加入“*”后，“*”右边只能按照命名的方式传参数

8) Func8，在位置形参的最后，在形参名称前使用 * 允许传入任意数量的位置实参(被打包为元组)

```
46  def func8(*numbers):
47      """计算任意数量数字的总和"""
48      total = 0
49      for num in numbers:
50          total += num
51      return total
```

```
63  print(mylib.func8(3, 4, 5))
```

```
12
```

- 可以接收任意数量的位置实参

```
(Pdb) p numbers
(3, 4, 5, 20)
(Pdb) p type(numbers)
<class 'tuple'>
```

- 实参被打包成元组

9) Func9，在命名形参的最后，在形参名称前使用 ** 允许传入任意数量的命名实参(被打包为字典)

```
54  def func9(**info):
55      """打印关于一个人的信息"""
56      for key, value in info.items():
57          print(f"{key}: {value}")
```

```
name: Alice
age: 25
city: New York
```

- 可以接受任意的命名实参，输出键值对

```
58              print(f"{key}: {value}")
[EOF]
(Pdb) p info
{'name': 'Alice', 'age': 25, 'city': 'New York'}
```

- 被打包为字典

10) func10，接受两个位置形参，一个命名形参，尝试在调用时使用 * 将可迭代对象(如元组或列表)自动解包，按位置实参传入

```
60  def func10(arg1, arg2, named_arg=10):
61      return arg1 + arg2 + named_arg
```

```
69  args_tuple = (20, 30)
70  result = mylib.func10(*args_tuple)
71  print(result)
72
73  args_list = [30, 40]
74  print(mylib.func10(*args_list))
```

(variable) args_list: list[int]

```
60
80
```

11) Func11，接受一个命名形参，两个命名形参，尝试在调用时使用 ** 将映射对象(如字典) 自动解包，按命名实参传入

```
64  # 定义 func11 函数，接受三个命名形参
65  def func11(param1=1, param2=2, param3=3):
66      return param1 + param2 + param3
```

```
80  params_dict = {"param1": 10, "param2": 20, "param3": 30}
81  result = mylib.func11(**params_dict)
82  print(result)
```

```
60
```

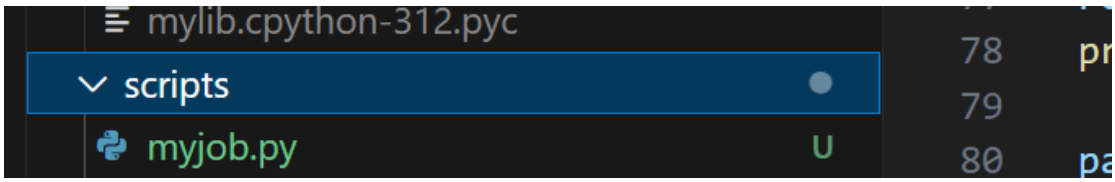
12) Func12，给函数添加 内嵌文档 (docstring)，给形参和返回值添加类型注解 (type annotation)，提高函数签名的可读性

```
69  def func12(arg1: int, arg2: int, named_arg: int = 10) -> None:
70      "多个参数调用的例子"
71      return arg1 + arg2 + named_arg
```

```
signature: def func12(arg1: int, arg2: int, named_arg: int = 10) -> None
"""多个参数调用的例子"""
```

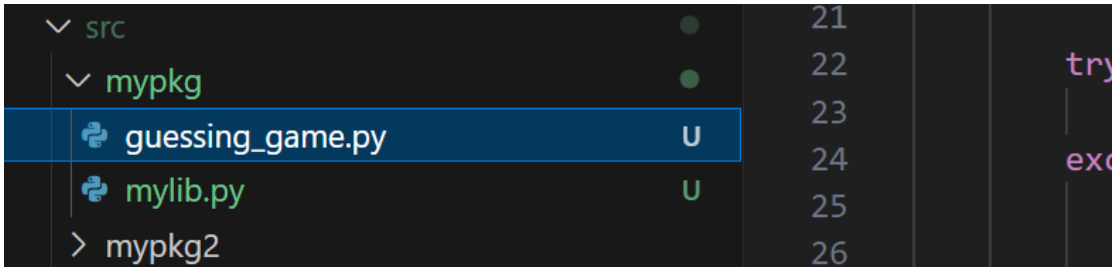
- 类型注解，但没什么约束力（传参数的时候可以类型不匹配）

4. 把 mylib 模块转变为软件包 (package) 安装进当前的 Conda 环境来使用




```
$ python scripts/myjob.py
Traceback (most recent call last):
  File "C:\Users\cherry\repo\week06\scripts\myjob.py", line 1, in <module>
    import mylib  # noqa: F401
    ^^^^^^^^^^^^
ModuleNotFoundError: No module named 'mylib'
```

- 将脚本移至新建的 scripts 文件夹后无法运行



- 将脚本移至 src/mypkg

- 有了这个文件之后 python 就会知道这是个软件包

```toml
[project]
name = "mypackage"
version = "2025.4.17"
dependencies = [
  "openpyxl",
]
authors = [
  {name = "cherry", email = "3155496417@qq.com"},
]
description = "测试用的软件包"

[project.optional-dependencies]
dev = [
  "pytest",
]
```

- 至少要有名称、版本、作者、软件包描述这些信息。

```toml
[build-system]
requires = ["hatchling"]
build-backend = "hatchling.build"
```

- 构建配置

environment.yml

```yaml
name: week06
channels:
  - conda-forge
dependencies:
  - python=3.12
  - wat-inspector
  - pip
  - pip:
    - "-e ."
```

- 重装 week06 环境，安装 pip，并通过 pip 把当前文件夹的软件包按照可编辑的形式安装进来

```
$ conda list
# packages in environment at D:\Anaconda\envs\week06:
#
# Name                    Version                   Build  Channel
bzip2                     1.0.8                h2466b09_7    conda-forge
ca-certificates           2025.1.31            h56e8100_0    conda-forge
et-xmlfile                2.0.0                    pypi_0    pypi
libexpat                  2.7.0                he0c23c2_0    conda-forge
libffi                    3.4.6                h537db12_1    conda-forge
liblzma                   5.8.1                h2466b09_0    conda-forge
libsqlite                 3.49.1               h67fdade_2    conda-forge
libzlib                   1.3.1                h2466b09_2    conda-forge
mypackage                 2025.4.17                pypi_0    pypi
```

- 软件包成功安装进来

```
>>> mypkg.mylib.func1()
0.0710678118654755
>>> import mypkg.guessing_game
>>> mypkg.guessing_game.guessing_game()
欢迎来到猜数字游戏！我已经想好了一个 1 到 100 之间的数字，你可以开始猜啦
(第 1 次尝试) 请输入你猜的数字 (输入整数，或者输入 q 回车退出): 40
猜的数字太小了，再试试↗。
(第 2 次尝试) 请输入你猜的数字 (输入整数，或者输入 q 回车退出): 60
猜的数字太小了，再试试↗。
(第 3 次尝试) 请输入你猜的数字 (输入整数，或者输入 q 回车退出): 80
猜的数字太大了，再试试↘。
(第 4 次尝试) 请输入你猜的数字 (输入整数，或者输入 q 回车退出): 75
猜的数字太大了，再试试↘。
(第 5 次尝试) 请输入你猜的数字 (输入整数，或者输入 q 回车退出): 69
猜的数字太大了，再试试↘。
(第 6 次尝试) 请输入你猜的数字 (输入整数，或者输入 q 回车退出): 64
猜的数字太大了，再试试↘。
(第 7 次尝试) 请输入你猜的数字 (输入整数，或者输入 q 回车退出): 62
猜的数字太小了，再试试↗。
(第 8 次尝试) 请输入你猜的数字 (输入整数，或者输入 q 回车退出): 61
猜的数字太小了，再试试↗。
(第 9 次尝试) 请输入你猜的数字 (输入整数，或者输入 q 回车退出): 63
恭喜你🎉，猜对了！
游戏结束，再见👋。
>>>
```

- 运行