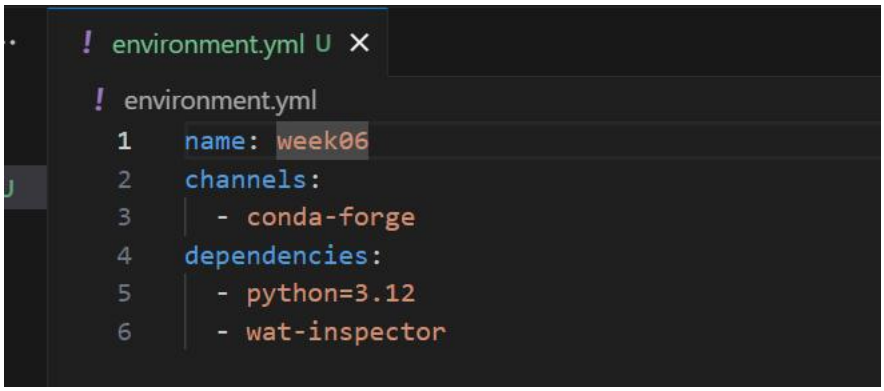
复制 environment.yml 文件到 week06

```
(base) yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repoo/week06 (main)
$ cp ../environment.yml ./
cp: cannot stat '../environment.yml': No such file or directory

(base) yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repoo/week06 (main)
$ cp ../week04/environment.yml ./

(base) yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repoo/week06 (main)
$ ls -l
total 25
-rw-r--r-- 1 yangzihan 197121    91  4月 21 10:39 environment.yml
-rw-r--r-- 1 yangzihan 197121 18805  4月 21 10:34 LICENSE
-rw-r--r-- 1 yangzihan 197121  2239  4月 21 10:34 README.md
```

修改名称



创建 conda 环境

```
(base) yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repoo/week06 (main)
$ conda env list
# conda environments:
#
base                  *  D:\Anaconda
myproject                D:\Anaconda\envs\myproject
prj1                     D:\Anaconda\envs\prj1
prj2                     D:\Anaconda\envs\prj2
python=3.12              D:\Anaconda\envs\python=3.12
week04                   D:\Anaconda\envs\week04
week05                   D:\Anaconda\envs\week05
week06                   D:\Anaconda\envs\week06
```

删除 conda 指令

```
(base) yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repoo/week06 (main)
$ conda env remove -n prj2
```

激活 conda 环境

```
(base) yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repoo/week06 (main)
$ conda activate week06
(week06)
```

创建一个 guessing_game.py 文件，复制粘贴以下代码，运用 pdb 调试器

```
yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repoo/week06 (main)
$ python -m pdb  guessing_game.py
> c:\users\yangzihan\repoo\week06\guessing_game.py(1)<module>()
-> import random
(Pdb) l
  1  -> import random
  2
  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 之间的数字，你可以开始猜啦。")
 10
 11         while True:
(Pdb) n
> c:\users\yangzihan\repoo\week06\guessing_game.py(4)<module>()
-> def guessing_game():
(Pdb) l
  1     import random
  2
  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 之间的数字，你可以开始猜啦。")
```

```
 22             try:
(Pdb) l
 23                 guess = int(guess)
 24             except ValueError:
 25                 print("输入无效🙅，请输入一个整数。")
 26                 continue
 27
 28             if guess < 1 or guess > 100:
 29                 print("输入无效🙅，输入值应该在 1~100 之间。")
 30                 continue
 31
 32             if guess == secret_number:
 33                 print("恭喜你🎉，猜对了！")
(Pdb) l
 34                 break
 35
 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↘。")
 42                 continue
 43
 44             raise NotImplementedError
(Pdb) l
 45
 46         print("游戏结束，再见👋。")
 47
```

```
yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repoo/week06 (main)
$ 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）: 45
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）: 55
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）: 60
猜的数字太小了，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）: 70
猜的数字太小了，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）: 80
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）: 85
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）: 75
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）: 76
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）: 77
猜的数字太小了，再试试⤴。
（第 10 次尝试）请输入你猜的数字（输入整数，或者输入 q 回车退出）: 78
猜的数字太小了，再试试⤴。
（第 11 次尝试）请输入你猜的数字（输入整数，或者输入 q 回车退出）: 79
恭喜你🎉，猜对了！
游戏结束，再见👋。
(week06)
```

for 迭代循环 (iteration loop)

```python
# for语句
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)

fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    fruit += ", ok"
    print(fruit)  # 遍历列表

message = "Hello"
for char in message:
    print(char)  # 遍历字符串

for i in range(5):
    print(i)  # 结合range函数

student = {"name": "Alice", "age": 20, "grade": "A"}
for key, value in student.items():
    print(f"{key}: {value}")  # 遍历字典

student = {"name": "Alice", "age": 20, "grade": "A"}
for value in student.values():
    print(value)  # 遍历字典
```

```
yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repoo/week06 (main)
$ python flow_controls.py
apple
banana
cherry
apple, ok
banana, ok
cherry, ok
H
e
l
l
o
0
1
2
3
4
name: Alice
age: 20
grade: A
Alice
20
A
0
1
2
3
4
5
4
3
```

while 条件循环 (conditional loop)

```python
# while语句
count = 0
while count < 5:
    print(count)
    count = count + 1  # 简单计数

numbers = [1, 2, 3, 4, 5]
while numbers:
    print(numbers.pop())  # 从列表中移除元素

valid_input = False
while not valid_input:
    user_input = input("请输入一个大于 10 的数字: ")
    try:
        num = int(user_input)
        if num > 10:
            print("输入有效。")
            valid_input = True
        else:
            print("输入的数字必须大于 10, 请重新输入。")
    except ValueError:
        print("输入无效，请输入一个有效的数字。")  # 用户输入验证

```

```
0
1
2
3
4
5
4
3
2
1
请输入一个大于 10 的数字： 11
输入有效。
```

- break 打断跳出循环

```python
# break语句
fruits = ["apple", "banana", "cherry", "date"]
for fruit in fruits:
    if fruit == "cherry":
        break
    print(fruit)  # 在 for 循环中使用 break

count = 0
while count < 10:
    if count == 5:
        break
    print(count)
    count = count + 1  # 在 while 循环中使用 break

for i in range(3):
    for j in range(3):
        if i + j == 2:
            break
        print(f"({i}, {j})")  # 嵌套循环中使用 break
```

(variable) i: int

```
apple
banana
0
1
2
3
4
(0, 0)
(0, 1)
(1, 0)
```

- continue 跳至下一轮循环

```python
# continue语句
numbers = [1, 2, 3, 4, 5]
for num in numbers:
    if num == 3:
        continue
    print(num)  # 在 for 循环中使用 continue

count = 0
while count < 5:
    count = count + 1
    if count == 2:
        continue
    print(count)  # 在 while 循环中使用 continue

count = 0
while count < 5:
    count = count + 1
    if count == 2:
        continue
    print(count)  # 嵌套循环中使用 continue
```

```
1
2
4
5
1
3
4
5
1
3
4
5
```

- for...else 循环未被打断的处理

```python
# for...else语句
numbers = [1, 3, 5, 7, 9]
for num in numbers:
    if num % 2 == 0:
        print(f"找到了偶数: {num}")
        break
else:
    print("列表中没有偶数。")  # 查找列表中的偶数

text = "Hello, World!"
target = "z"
for char in text:
    if char == target:
        print(f"找到了字符 '{target}'。")
        break
else:
    print(f"字符串中没有字符 '{target}'。")  # 检查字符串中是否包含特定字符

student_scores = {"Alice": 85, "Bob": 90, "Charlie": 78}
search_name = "David"
for name in stu    (variable) search_name: Literal['David']
    if name == search_name:
        print(f"找到了 {search_name}，分数是 {student_scores[search_name]}。")
        break
else:
    print(f"没有找到 {search_name} 的记录。")  # 遍历字典的键
```

```
列表中没有偶数。
字符串中没有字符 'z'。
没有找到 David 的记录。
```

- if 条件分支

```python
# if语句
age = 18
if age >= 18:
    print("你已经成年了。")  # 简单的 if 语句

num = 10
if num % 2 == 0:
    print(f"{num} 是偶数。")
else:
    print(f"{num} 是奇数。")  # if-else 语句

score = 85
if score >= 90:
    print("成绩为 A。")
elif score >= 80:
    print("成绩为 B。")
elif score >= 70:
    print("成绩为 C。")
elif score >= 60:
    print("成绩为 D。")
else:
    print("成绩为 F。")  # if-elif-else 语句

is_member = True
money = 200
product_price = 150
if is_member:
    if money >= product_price:
        print("你是会员，且余额足够，可以购买该商品。")
    else:
        print("你是会员，但余额不足，无法购买该商品。")
else:
```

```
你已经成年了。
10 是偶数。
成绩为 B。
你是会员，且余额足够，可以购买该商品。
```

- if...elif[...elif] 多重条件分支

```python
# if...elif语句
month = 7
if 3 <= month <= 5:
    print("当前季节是春季")
elif 6 <= month <= 8:
    print("当前季节是夏季")
elif 9 <= month <= 11:
    print("当前季节是秋季")
elif month == 12 or 1 <= month <= 2:
    print("当前季节是冬季")
else:
    print("输入的月份无效")
```

```
当前季节是夏季
```

if...else 未满足条件的处理

```python
# if...else语句
number = 10
if number % 2 == 0:
    print(f"{number} 是偶数。")
else:
    print(f"{number} 是奇数。")

```

```
10 是偶数。
```

try...except[...except...else...finally] 捕捉异常的处理

```python
# try...except...else...finally语句
try:
    num1 = int(input("请输入第一个整数: "))
    num2 = int(input("请输入第二个整数: "))
    result = num1 / num2
except ValueError:
    print("输入错误: 请输入有效的整数。")
except ZeroDivisionError:
    print("错误: 除数不能为零。")
else:
    print(f"除法运算结果是: {result}")
finally:
    print("无论是否发生异常，此代码块都会执行。")

```

raise 主动抛出异常

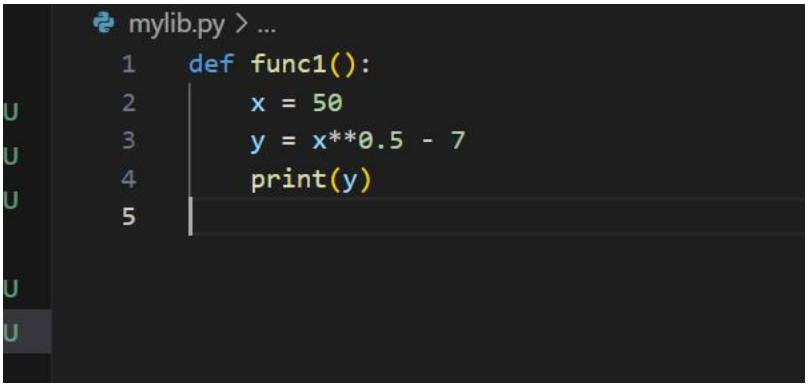
```python
# raise语句
def divide_numbers(a, b):
    if b == 0:
        raise ZeroDivisionError("除数不能为零。")
    return a / b


try:
    result = divide_numbers(10, 0)
    print(result)
except ZeroDivisionError as e:
    print(f"捕获到异常: {e}")

```

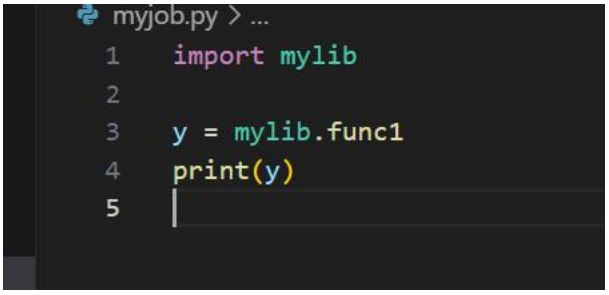
```
错误： 除数不能为零。
请输入第一个整数： 3
请输入第二个整数： 4
除法运算结果是： 0.75
无论是否发生异常，此代码块都会执行。
捕获到异常： 除数不能为零。
(week06)
```

- 创建一个 mylib.py 模块 (module)，在里面定义以下函数，再创建一个 myjob.py 脚本 (script)，从 mylib.py 导入函数并尝试调用：



```
mylib.py > ...
1  def func1():
2      x = 50
3      y = x**0.5 - 7
4      print(y)
5
```

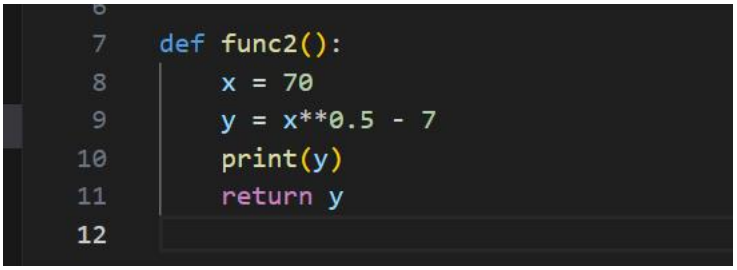
- 定义函数 func1，没有形参，没有返回值



```
myjob.py > ...
1  import mylib
2
3  y = mylib.func1
4  print(y)
5
```

```
yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repoo/week06 (main)
$ python myjob.py
<function func1 at 0x000001821198AFC0>
(week06)
```

- 定义函数 func2，没有形参，有返回值



```
7  def func2():
8      x = 70
9      y = x**0.5 - 7
10     print(y)
11     return y
12
```

- 定义函数 func3，只有一个 位置形参 (positional parameter)，先尝试传入 位置实参 (positional argument) 调用，再尝试传入 命名实参 (named argument) 调用，再尝试不传实参 (会报错)

```
13
14  def func3():
15      y = x**0.5 - 7
16      return y
17
```

```
6   y = mylib.func3(45)
7   print(y)  # 位置实参
8
9   y = mylib.func3(x=47)
10  print(y)  # 命名实参
11
```

- 定义函数 func4，只有一个 命名形参 (named parameter)，先传入 位置实参 调用，再传入 命名实参 调用，再尝试不传实参 (取默认值)

```
18  def func4(x=50):
19      y = x**0.5 - 7
20      return y
```

```
12  y = mylib.func4(47)
13  print(y)  # 位置实参
14
15  y = mylib.func4(x=49)
16  print(y)  # 命名实参
17
18  y = mylib.func4()
19  print(y)  # 不传实参
```

- 定义函数 func5，接受多个位置形参和命名形参，尝试以位置/命名各种不同方式传入实参，注意位置参数必须排在命名参数之前

```
22  def caculate(a,b,operation='add'):
23      if operation == 'add':
24          return a + b
25      elif operation == 'subtract':
26          return a - b
27      else :
28          return None
```

```
print(mylib.caculate(5,10,"add"))
print(mylib.caculate(operation="add",b=5,a=10))
print(mylib.caculate(b=5,a=10))
print(mylib.caculate(5,8,operation="subtract"))
```

- 定义函数 func6，在形参列表中使用 / 来限定只接受位置实参的形参

```
def func6(a, /, b, operation="add"):
    if operation == "add":
        return a + b
    elif operation == "subtract":
        return a - b
    else:
        return None
```

- 定义函数 func7，在形参列表中使用 * 来限定只接受命名实参的形参

```
def func7(a, /, b, *, operation="add"):
    if operation == "add":
        return a + b
    elif operation == "subtract":
        return a - b
    else:
        return None
```

- 定义函数 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)
- 定义函数 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)
- 定义函数 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

```python
def func8(*numbers):
    total = 0
    for num in numbers:
        total = total + num
    return total


def func9(**user):
    for key, value in user.items():
        print(f"{key}: {value}")


def func10(arg1, arg2, named_arg="default"):
    print(f"位置实参  arg1): {arg1}")
    print(f"位置实参  arg2): {arg2}")
    print(f"命名实参  named_arg): {named_arg}")

```

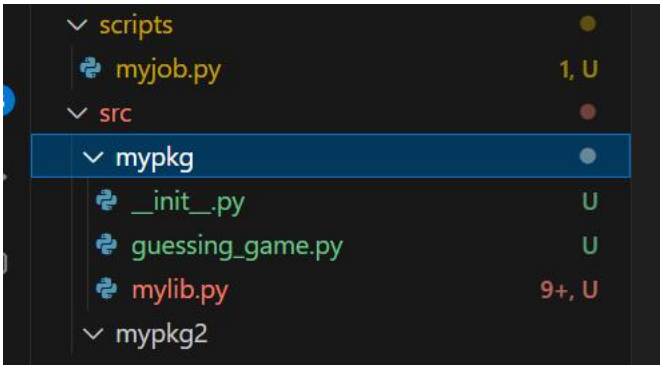
- 定义函数 func11，接受一个位置形参，两个命名形参，尝试在调用时使用 ** 将映射对象 (如字典) 自动解包，按命名实参传入
- 定义函数 func12，给函数添加 内嵌文档 (docstring)，给形参和返回值添加 类型注解 (type annotation)，提高函数签名的可读性

```python
def func11(arg1, arg2):
    print(f"arg1 的值是: {arg1}")
    print(f"arg2 的值是: {arg2}")


def func12(arg1: str, arg2: int, named_arg: str = "default") -> None:
    "多个参数的调用例子"
    print(f"位置实参  arg1): {arg1}")
    print(f"位置实参  arg2): {arg2}")
    print(f"命名实参  named_arg): {named_arg}")

```

- 把 myjob.py 脚本移动至 scripts/myjob.py，再次尝试运行，会发现 import mylib 失败，这是由于 mylib 并没有打包成 软件包 (package) 安装
- 将 mylib.py 模块移动至 src/mypkg/mylib.py，创建 src/mypkg/__init__.py 文件，准备好软件包的源代码



- 创建 pyproject.toml 配置文件，按照 文档 填写基本的软件包信息
- 在 pyproject.toml 配置文件里，按照 文档 填写软件包的 构建 (build) 配置

pyproject.toml

```toml
[project]
name = "mypackage"
version = "2025.4.15"
dependencies = [
  "openpyxl",
]
authors = [
  {name = "haha", email = "3249196473@qq.com"},
]
description = "测试用的软件包"

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

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

[tool.hatch.build.targets.wheel]
packages = [
"src/mypkg",
]
```

- 使用 pip install -e . 以本地可编辑模式把当前软件包安装进当前 Conda 环境
- 修改 environment.yml 文件，使得 conda env create 自动安装本地可编辑软件包

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