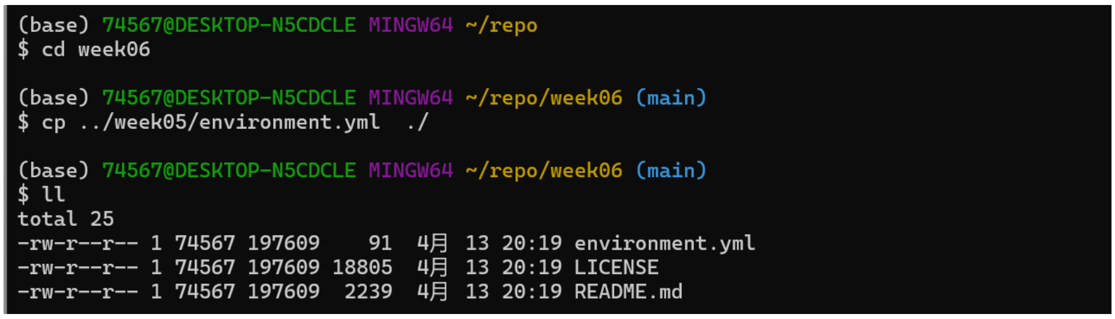
第六周学习报告

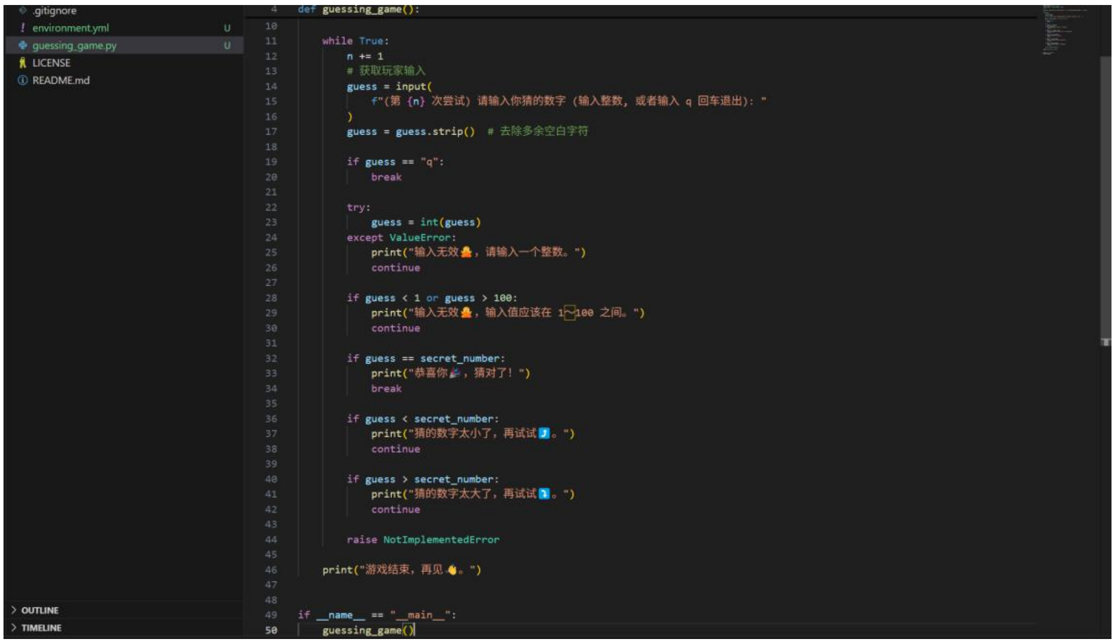
1. Fork 第 06 周打卡 仓库至你的名下，然后将你名下的这个仓库 Clone 到你的本地计算机。

2. 用 VS Code 打开项目目录，新建一个 environment.yml 文件，指定安装 Python 3.12，然后运行 conda env create 命令创建 Conda 环境。

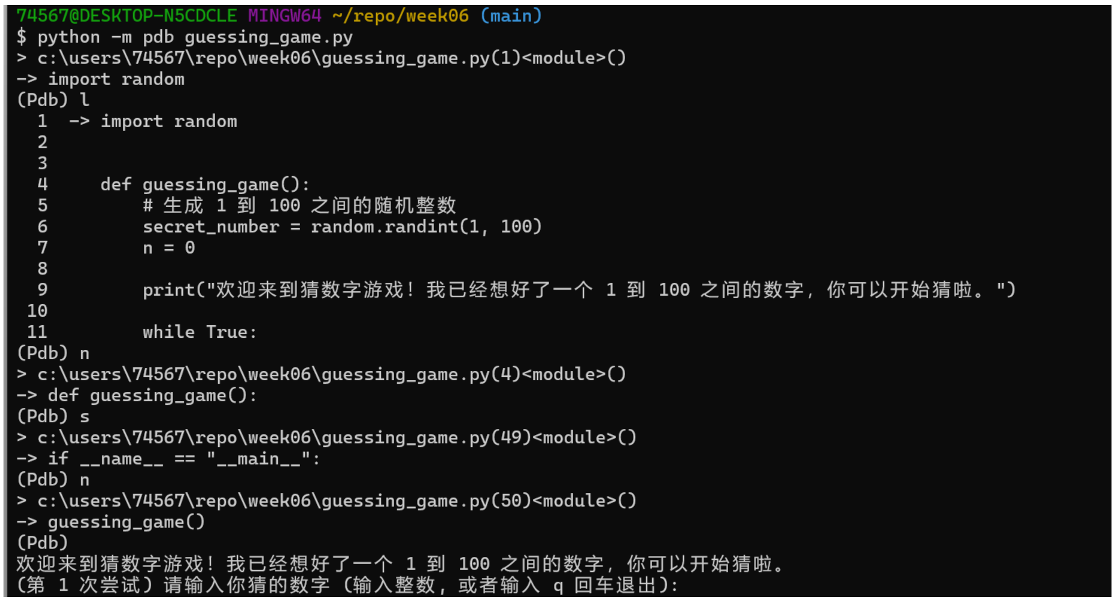


```
(base) 74567@DESKTOP-N5CDCLE MINGW64 ~/repo
$ cd week06

(base) 74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week06 (main)
$ cp ../week05/environment.yml  ./

(base) 74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week06 (main)
$ ll
total 25
-rw-r--r-- 1 74567 197609    91  4月  13 20:19 environment.yml
-rw-r--r-- 1 74567 197609 18805  4月  13 20:19 LICENSE
-rw-r--r-- 1 74567 197609  2239  4月  13 20:19 README.md
```

3. 创建一个 guessing_game.py 文件，复制粘贴以下代码，运用 pdb 调试器理解其运行流程：



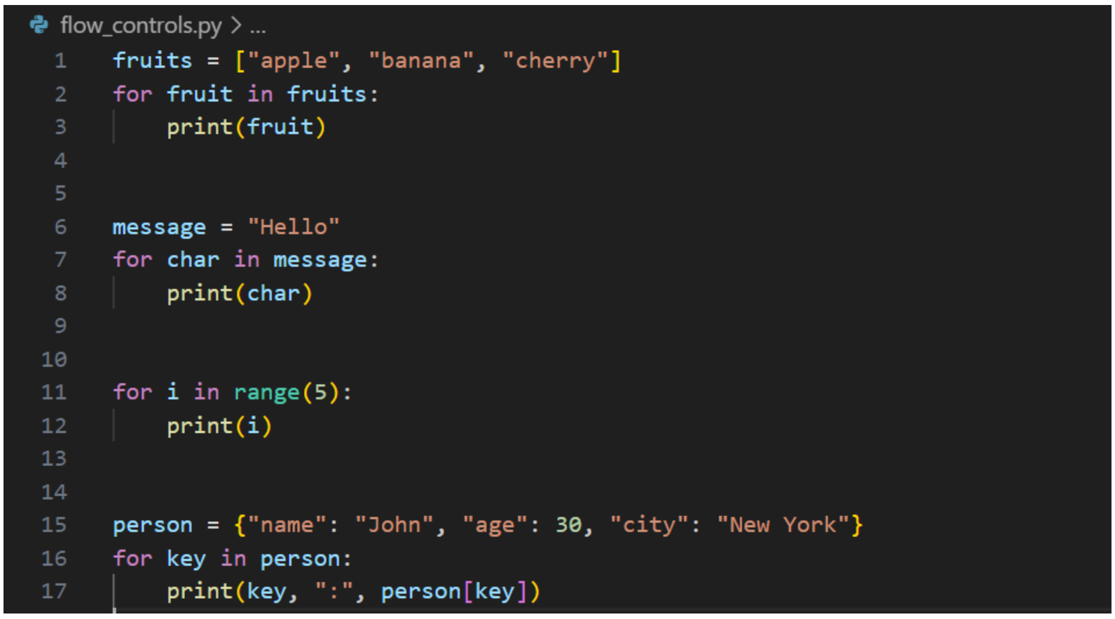
```python
def guessing_game():

    while True:
        n += 1
        # 获取玩家输入
        guess = input(
            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): "
        )
        guess = guess.strip()  # 去除多余空白字符

        if guess == "q":
            break

        try:
            guess = int(guess)
        except ValueError:
            print("输入无效🙅，请输入一个整数。")
            continue

        if guess < 1 or guess > 100:
            print("输入无效🙅，输入值应该在 1~100 之间。")
            continue

        if guess == secret_number:
            print("恭喜你🎉，猜对了！")
            break

        if guess < secret_number:
            print("猜的数字太小了，再试试⤴️。")
            continue

        if guess > secret_number:
            print("猜的数字太大了，再试试⤵️。")
            continue

        raise NotImplementedError

    print("游戏结束，再见👋。")


if __name__ == "__main__":
    guessing_game()
```



```
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week06 (main)
$ python -m pdb guessing_game.py
> c:\users\74567\repo\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\74567\repo\week06\guessing_game.py(4)<module>()
-> def guessing_game():
(Pdb) s
> c:\users\74567\repo\week06\guessing_game.py(49)<module>()
-> if __name__ == "__main__":
(Pdb) n
> c:\users\74567\repo\week06\guessing_game.py(50)<module>()
-> guessing_game()
(Pdb)
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）:
```

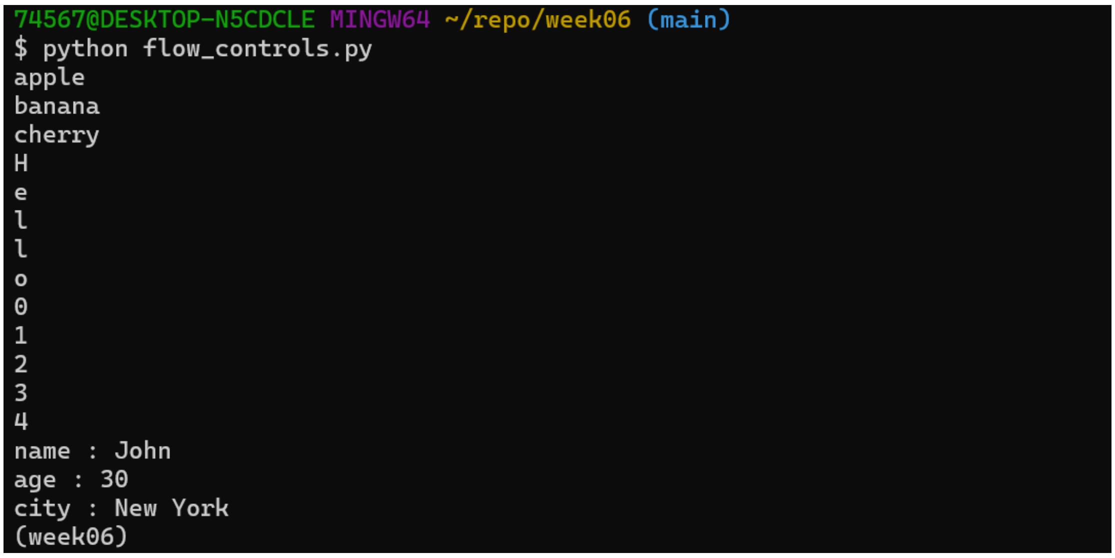
4. 创建一个 flow_controls.py 文件，让豆包（或 DeepSeek 等任何大模型）生成例子，尝试运行，体会理解以下 Python 流程控制语句：

for 迭代循环（iteration loop）



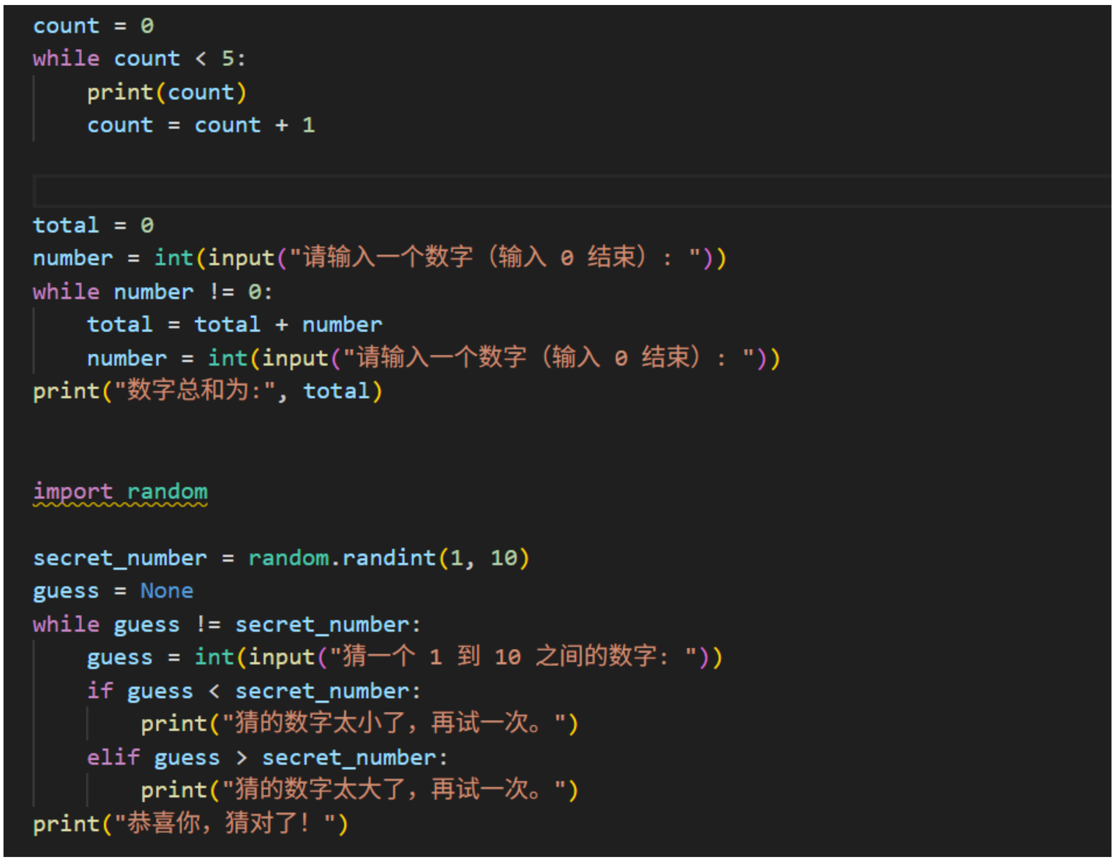
flow_controls.py > ...

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


message = "Hello"
for char in message:
    print(char)


for i in range(5):
    print(i)


person = {"name": "John", "age": 30, "city": "New York"}
for key in person:
    print(key, ":", person[key])
```

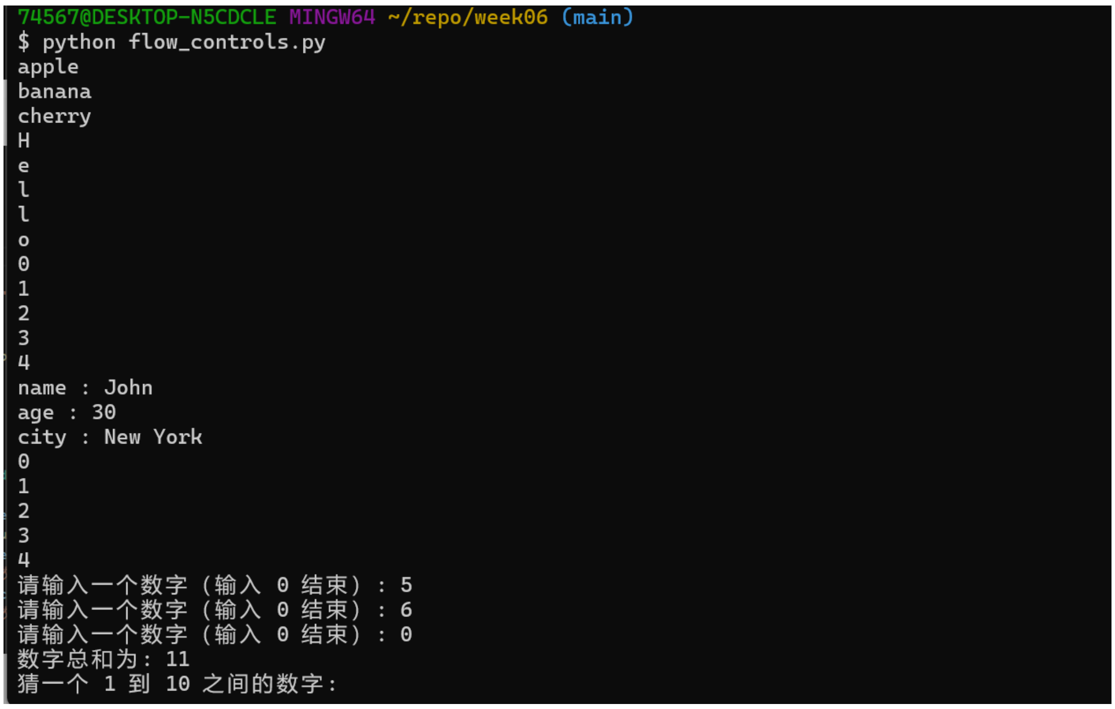


```
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week06 (main)
$ python flow_controls.py
apple
banana
cherry
H
e
l
l
o
0
1
2
3
4
name : John
age : 30
city : New York
(week06)
```

while 条件循环 (conditional loop)



```python
count = 0
while count < 5:
    print(count)
    count = count + 1


total = 0
number = int(input("请输入一个数字（输入 0 结束）: "))
while number != 0:
    total = total + number
    number = int(input("请输入一个数字（输入 0 结束）: "))
print("数字总和为:", total)


import random

secret_number = random.randint(1, 10)
guess = None
while guess != secret_number:
    guess = int(input("猜一个 1 到 10 之间的数字: "))
    if guess < secret_number:
        print("猜的数字太小了，再试一次。")
    elif guess > secret_number:
        print("猜的数字太大了，再试一次。")
print("恭喜你，猜对了！")
```



```
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week06 (main)
$ python flow_controls.py
apple
banana
cherry
H
e
l
l
o
0
1
2
3
4
name : John
age : 30
city : New York
0
1
2
3
4
请输入一个数字（输入 0 结束）: 5
请输入一个数字（输入 0 结束）: 6
请输入一个数字（输入 0 结束）: 0
数字总和为: 11
猜一个 1 到 10 之间的数字:
```

break 打断跳出循环

continue 跳至下一轮循环

for...else 循环未被打断的处理

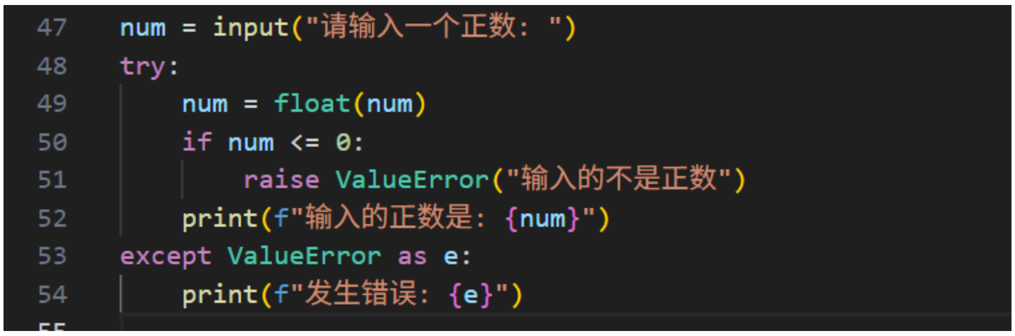
if 条件分支

if...elif[...elif] 多重条件分支

if...else 未满足条件的处理

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

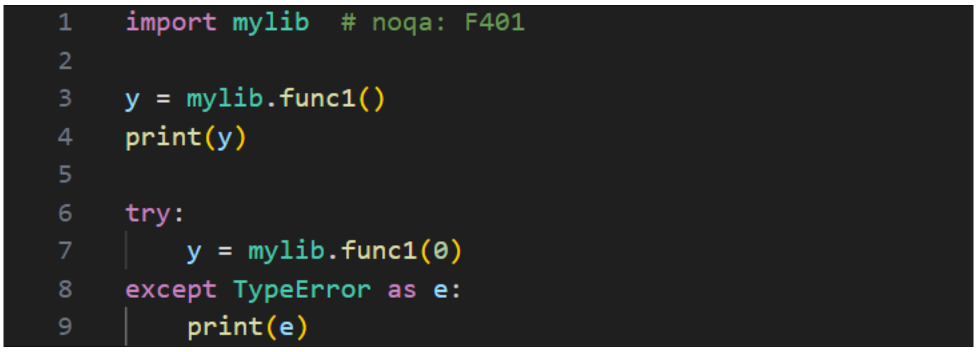
raise 主动抛出异常



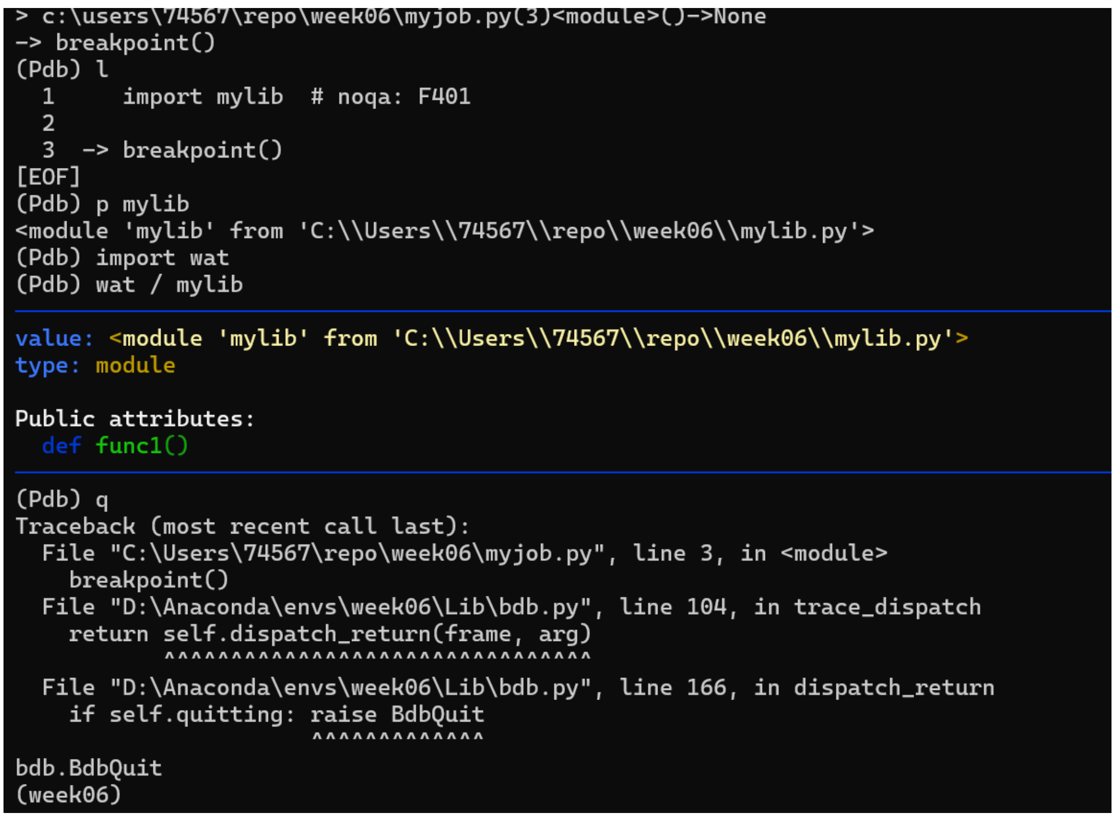
```python
num = input("请输入一个正数: ")
try:
    num = float(num)
    if num <= 0:
        raise ValueError("输入的不是正数")
    print(f"输入的正数是: {num}")
except ValueError as e:
    print(f"发生错误: {e}")
```

5.　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：

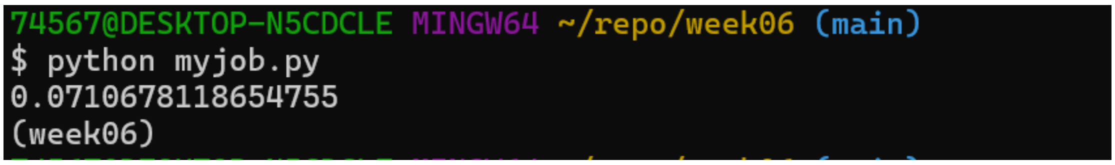
定义函数 func1，没有形参，没有返回值



```python
import mylib  # noqa: F401

y = mylib.func1()
print(y)

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

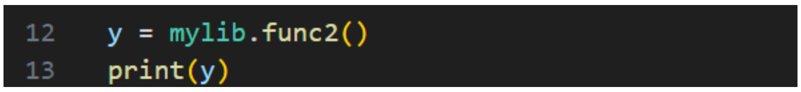


```
> c:\users\74567\repo\week06\myjob.py(3)<module>()->None
-> breakpoint()
(Pdb) l
  1     import mylib  # noqa: F401
  2
  3  -> breakpoint()
[EOF]
(Pdb) p mylib
<module 'mylib' from 'C:\\Users\\74567\\repo\\week06\\mylib.py'>
(Pdb) import wat
(Pdb) wat / mylib

value: <module 'mylib' from 'C:\\Users\\74567\\repo\\week06\\mylib.py'>
type: module

Public attributes:
  def func1()

(Pdb) q
Traceback (most recent call last):
  File "C:\Users\74567\repo\week06\myjob.py", line 3, in <module>
    breakpoint()
  File "D:\Anaconda\envs\week06\Lib\bdb.py", line 104, in trace_dispatch
    return self.dispatch_return(frame, arg)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "D:\Anaconda\envs\week06\Lib\bdb.py", line 166, in dispatch_return
    if self.quitting: raise BdbQuit
                      ^^^^^^^^^^^^^
bdb.BdbQuit
(week06)
```

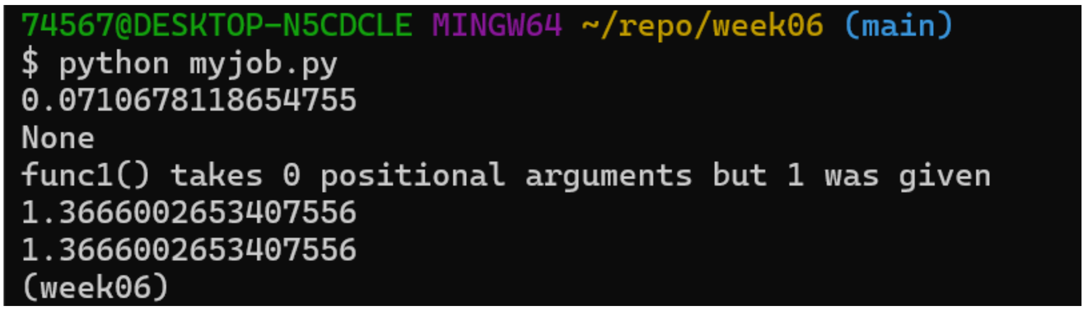


```
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week06 (main)
$ python myjob.py
0.0710678118654755
(week06)
```

定义函数 func2，没有形参，有返回值

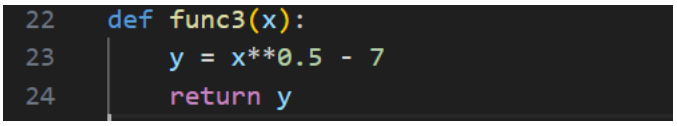


```python
y = mylib.func2()
print(y)
```

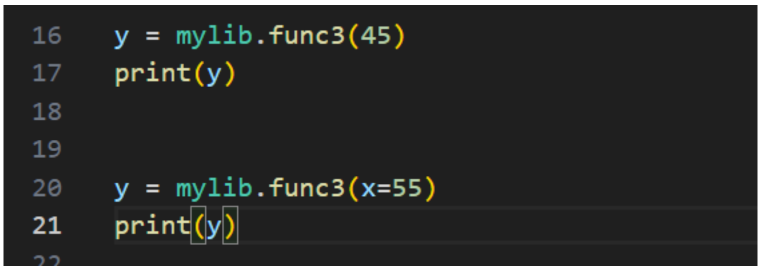


```
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week06 (main)
$ python myjob.py
0.07106781186547555
None
func1() takes 0 positional arguments but 1 was given
1.3666002653407556
1.3666002653407556
(week06)
```

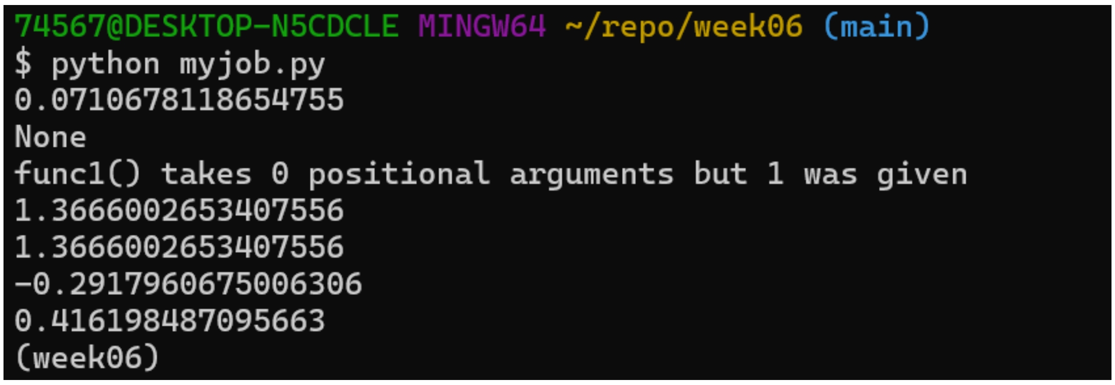
定义函数 func3，只有一个**位置形参**（positional parameter），先尝试传入**位置实参**（positional argument）调用，再尝试传入**命名实参**（named argument）调用，再尝试不传实参（会报错）



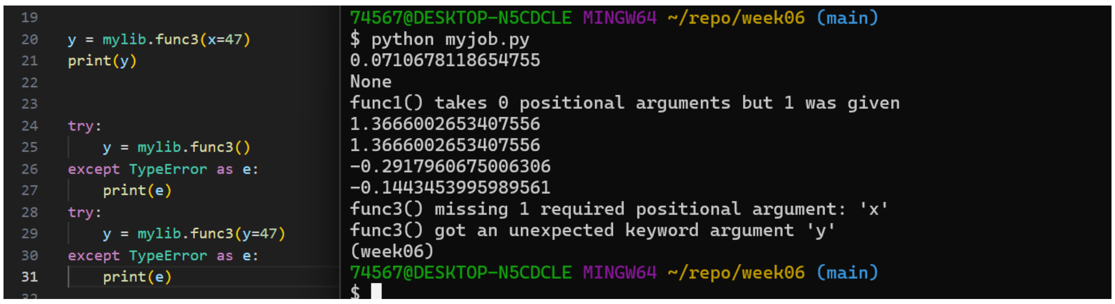
```python
def func3(x):
    y = x**0.5 - 7
    return y
```



```python
y = mylib.func3(45)
print(y)


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



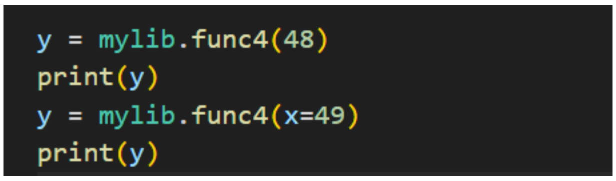
```
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week06 (main)
$ python myjob.py
0.07106781186547555
None
func1() takes 0 positional arguments but 1 was given
1.3666002653407556
1.3666002653407556
-0.2917960675006306
0.416198487095663
(week06)
```



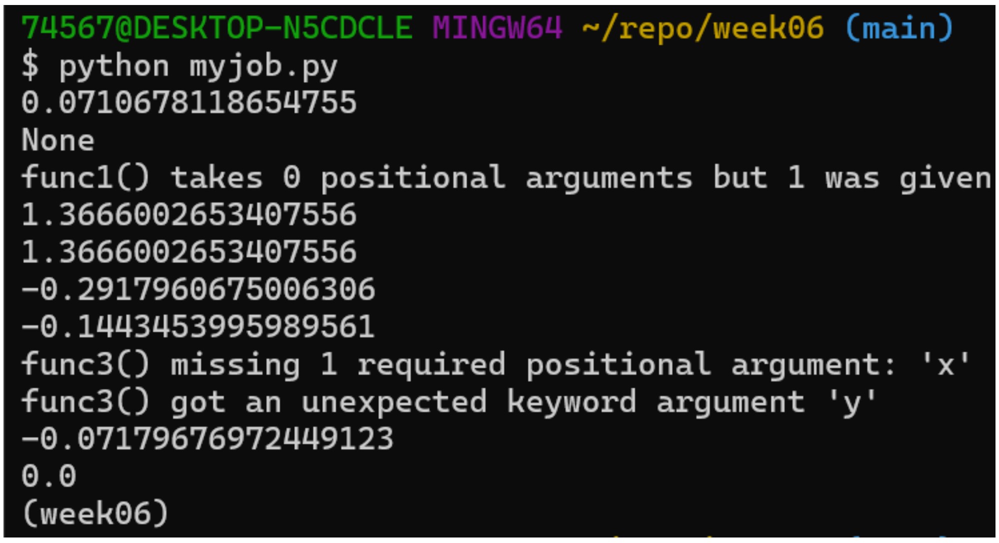
```python

y = mylib.func3(x=47)
print(y)


try:
    y = mylib.func3()
except TypeError as e:
    print(e)
try:
    y = mylib.func3(y=47)
except TypeError as e:
    print(e)
```

```
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week06 (main)
$ python myjob.py
0.07106781186547555
None
func1() takes 0 positional arguments but 1 was given
1.3666002653407556
1.3666002653407556
-0.2917960675006306
-0.1443453995989561
func3() missing 1 required positional argument: 'x'
func3() got an unexpected keyword argument 'y'
(week06)
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week06 (main)
$
```

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）

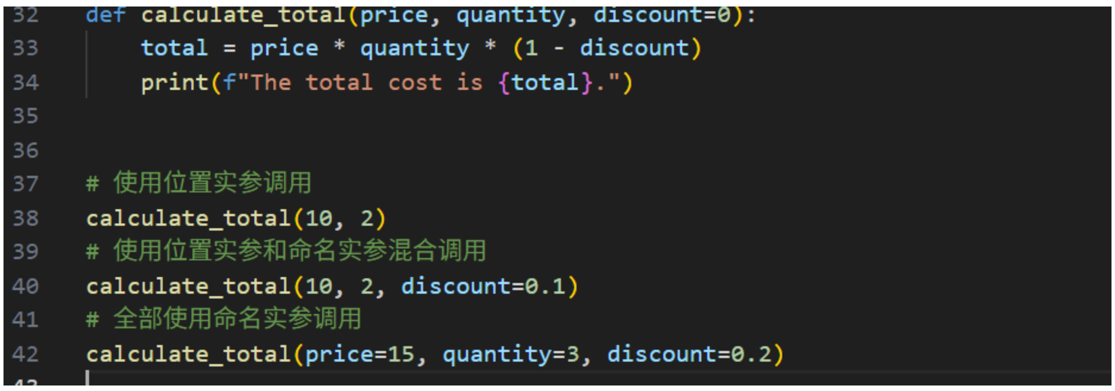


```python
y = mylib.func4(48)
print(y)
y = mylib.func4(x=49)
print(y)
```



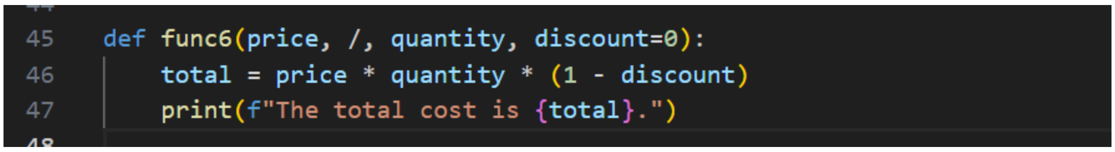
```
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week06 (main)
$ python myjob.py
0.0710678118654755
None
func1() takes 0 positional arguments but 1 was given
1.3666002653407556
1.3666002653407556
-0.2917960675006306
-0.1443453995989561
func3() missing 1 required positional argument: 'x'
func3() got an unexpected keyword argument 'y'
-0.0717967697244912З
0.0
(week06)
```

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



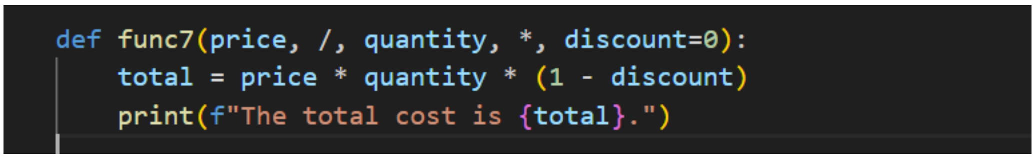
```python
def calculate_total(price, quantity, discount=0):
    total = price * quantity * (1 - discount)
    print(f"The total cost is {total}.")


# 使用位置实参调用
calculate_total(10, 2)
# 使用位置实参和命名实参混合调用
calculate_total(10, 2, discount=0.1)
# 全部使用命名实参调用
calculate_total(price=15, quantity=3, discount=0.2)
```

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



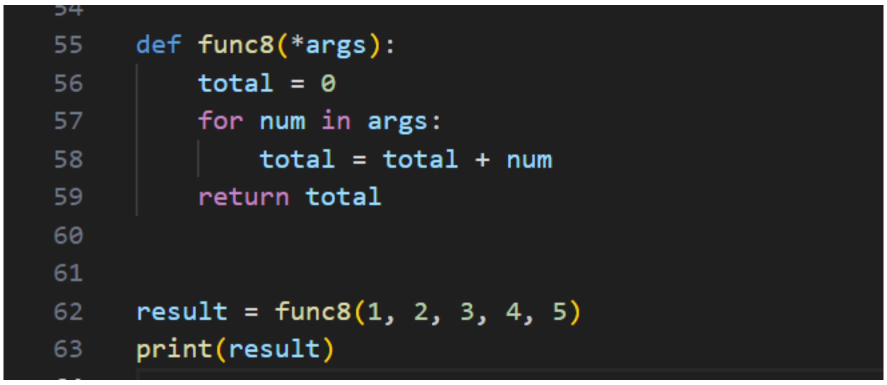
```python
def func6(price, /, quantity, discount=0):
    total = price * quantity * (1 - discount)
    print(f"The total cost is {total}.")
```

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

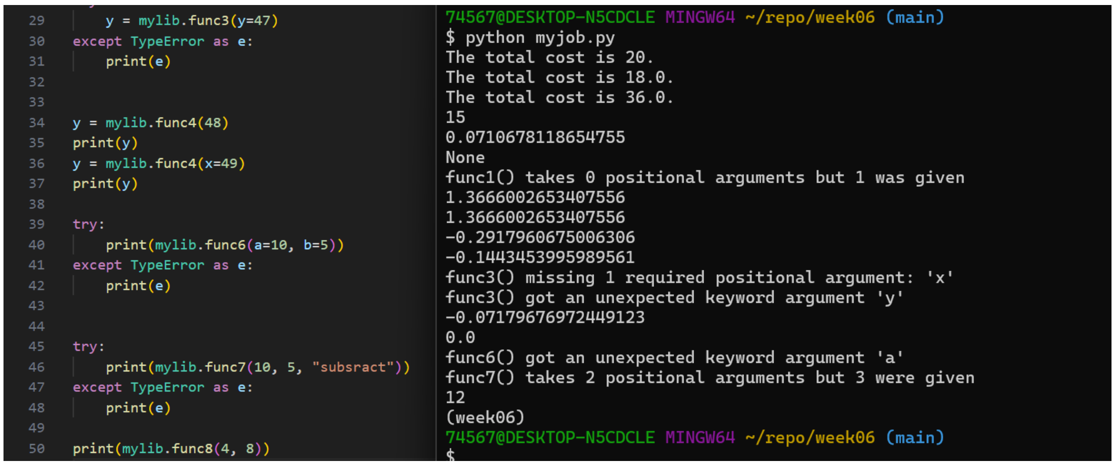


```python
def func7(price, /, quantity, *, discount=0):
    total = price * quantity * (1 - discount)
    print(f"The total cost is {total}.")
```

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）



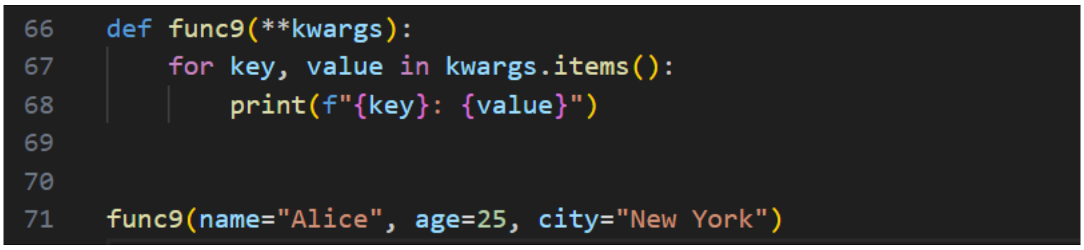
```python
def func8(*args):
    total = 0
    for num in args:
        total = total + num
    return total


result = func8(1, 2, 3, 4, 5)
print(result)
```



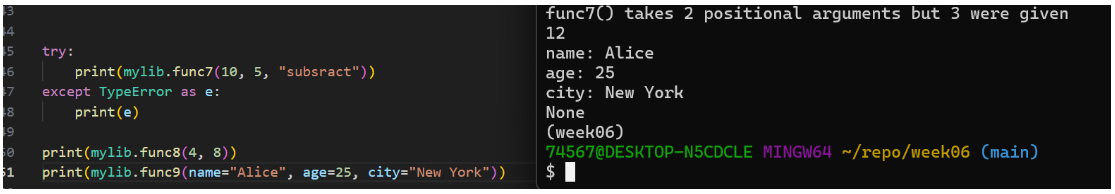
```python
    y = mylib.func3(y=47)
except TypeError as e:
    print(e)


y = mylib.func4(48)
print(y)
y = mylib.func4(x=49)
print(y)

try:
    print(mylib.func6(a=10, b=5))
except TypeError as e:
    print(e)


try:
    print(mylib.func7(10, 5, "subsract"))
except TypeError as e:
    print(e)

print(mylib.func8(4, 8))
```

```
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week06 (main)
$ python myjob.py
The total cost is 20.
The total cost is 18.0.
The total cost is 36.0.
15
0.0710678118654755
None
func1() takes 0 positional arguments but 1 was given
1.3666002653407556
1.3666002653407556
-0.2917960675006306
-0.1443453995989561
func3() missing 1 required positional argument: 'x'
func3() got an unexpected keyword argument 'y'
-0.0717967697244912З
0.0
func6() got an unexpected keyword argument 'a'
func7() takes 2 positional arguments but 3 were given
12
(week06)
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week06 (main)
$
```

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）



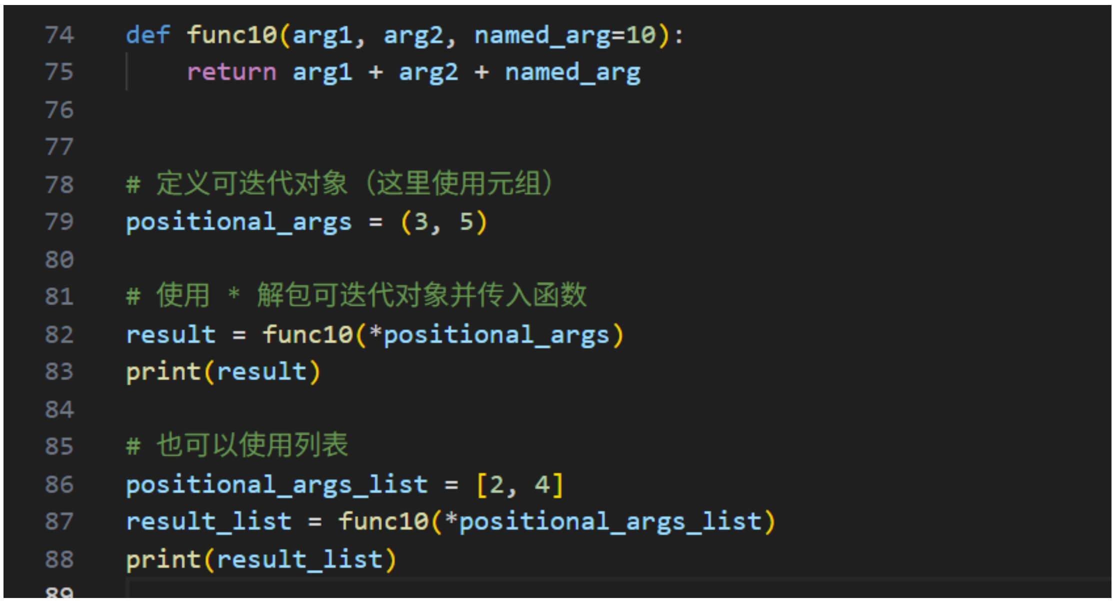
```python
def func9(**kwargs):
    for key, value in kwargs.items():
        print(f"{key}: {value}")


func9(name="Alice", age=25, city="New York")
```



```python

try:
    print(mylib.func7(10, 5, "subsract"))
except TypeError as e:
    print(e)

print(mylib.func8(4, 8))
print(mylib.func9(name="Alice", age=25, city="New York"))
```

```
func7() takes 2 positional arguments but 3 were given
12
name: Alice
age: 25
city: New York
None
(week06)
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week06 (main)
$ 
```

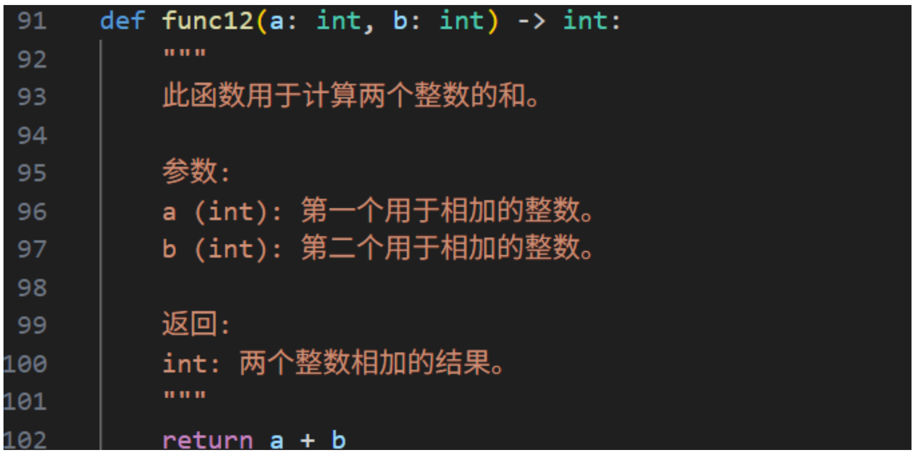
定义函数 func10，接受两个位置形参，一个命名形参，尝试在调用时使用 * 将可迭代对象（如元组或列表）自动解包，按位置实参传入



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


# 定义可迭代对象（这里使用元组）
positional_args = (3, 5)

# 使用 * 解包可迭代对象并传入函数
result = func10(*positional_args)
print(result)

# 也可以使用列表
positional_args_list = [2, 4]
result_list = func10(*positional_args_list)
print(result_list)
```

定义函数 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

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



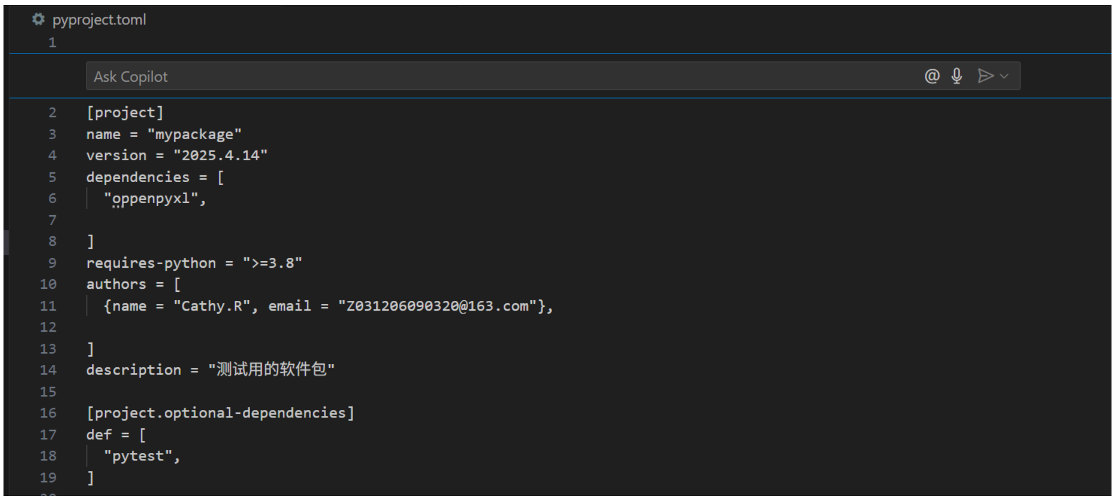
```python
def func12(a: int, b: int) -> int:
    """
    此函数用于计算两个整数的和。

    参数:
    a (int): 第一个用于相加的整数。
    b (int): 第二个用于相加的整数。

    返回:
    int: 两个整数相加的结果。
    """
    return a + b
```

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

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

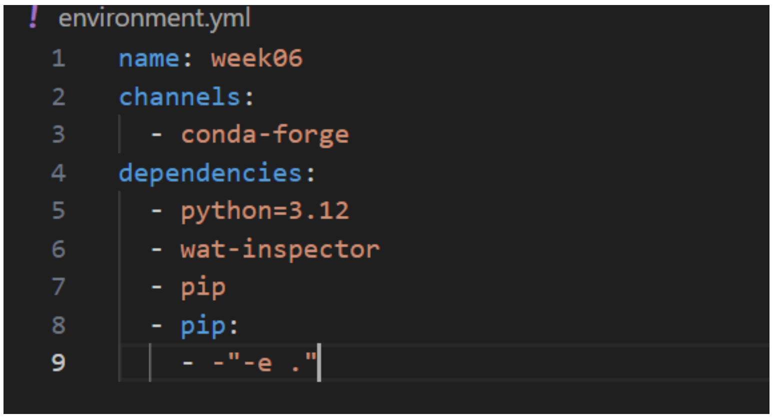


```toml

[project]
name = "mypackage"
version = "2025.4.14"
dependencies = [
  "oppenpyxl",

]
requires-python = ">=3.8"
authors = [
  {name = "Cathy.R", email = "Z031206090320@163.com"},

]
description = "测试用的软件包"

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

在 pyproject.toml 配置文件里，按照 文档 填写软件包的 **构建** (build) 配置

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



environment.yml

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