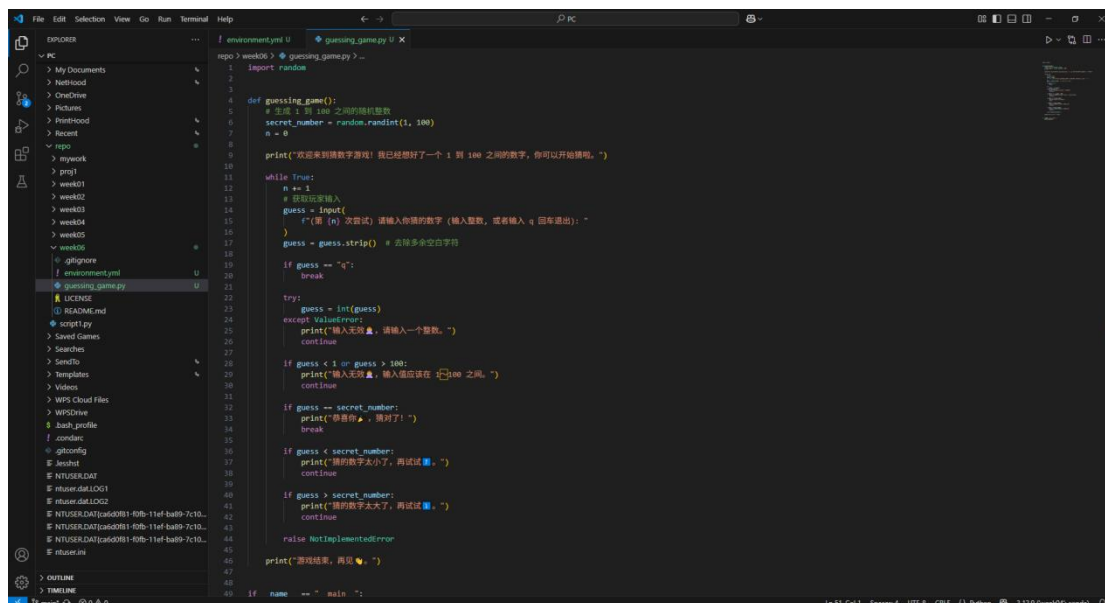
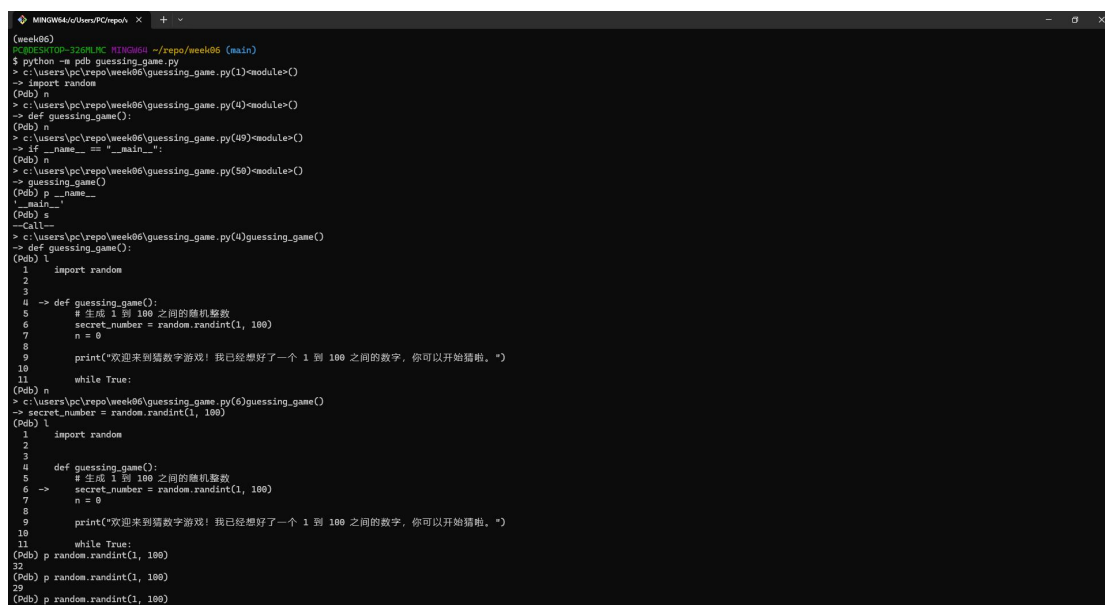


1. 创建一个 `guessing_game.py` 文件，输入猜数字游戏代码，运用 `pdb` 调试器理解其运行流程。



```
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:
12         n += 1
13         # 获取用户输入
14         guess = input(
15             f"（第 {n} 次尝试）请输入你猜的数字（输入整数，或者输入 q 回车退出）：")
16         guess = guess.strip() # 去掉多余空白字符
17
18         if guess == "q":
19             break
20
21         try:
22             guess = int(guess)
23         except ValueError:
24             print("输入无效，请输入一个整数。")
25             continue
26
27         if guess < 1 or guess > 100:
28             print("输入无效，输入应该在 1-100 之间。")
29             continue
30
31         if guess == secret_number:
32             print("恭喜你，猜对了！")
33             break
34
35         if guess < secret_number:
36             print("猜的数字太小了，再试试。")
37             continue
38
39         if guess > secret_number:
40             print("猜的数字太大了，再试试。")
41             continue
42
43         raise NotImplementedError
44
45     print("游戏结束，再见！")
46
47
48 if __name__ == "__main__":
```



```
(week06)
PC@DESKTOP-326HLC MINGW64 ~/repo/week06 (main)
$ python -m pdb guessing_game.py
> c:\users\pc\repo\week06\guessing_game.py(1)<module>()
-> import random
(Pdb) n
> c:\users\pc\repo\week06\guessing_game.py(4)<module>()
-> def guessing_game():
(Pdb) n
> c:\users\pc\repo\week06\guessing_game.py(49)<module>()
-> if __name__ == "__main__":
(Pdb) n
> c:\users\pc\repo\week06\guessing_game.py(50)<module>()
-> guessing_game()
(Pdb) p __name__
'__main__'
(Pdb) s
-> Call
> c:\users\pc\repo\week06\guessing_game.py(4)guessing_game()
-> def guessing_game():
(Pdb) l
1 import random
2
3
4
5 -> def guessing_game():
6     # 生成 1 到 100 之间的随机整数
7     secret_number = random.randint(1, 100)
8     n = 0
9
10    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。")
11
12    while True:
13        n += 1
14        # 获取用户输入
15        guess = input(
16            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）：")
17        guess = guess.strip() # 去掉多余空白字符
18
19        if guess == "q":
20            break
21
22        try:
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，输入应该在 1-100 之间。")
30            continue
31
32        if guess == secret_number:
33            print("恭喜你，猜对了！")
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
45
46    print("游戏结束，再见！")
47
48 if __name__ == "__main__":
```

```
MINGW64/c/Users/PC/repo/h x + v
(Pdb) n
> c:\users\pc\repo\week06\guessing_game.py(7)guessing_game()
-> n = 0
(Pdb) l
  2
  3
  4
  5 def guessing_game():
  6     # 生成 1 到 100 之间的随机整数
  7     secret_number = random.randint(1, 100)
  8     n = 0
  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:
 12         n += 1
 13
 14 (Pdb) p secret_number
25
(Pdb) n
> c:\users\pc\repo\week06\guessing_game.py(9)guessing_game()
-> 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。")
(Pdb) l
  4
  5 def guessing_game():
  6     # 生成 1 到 100 之间的随机整数
  7     secret_number = random.randint(1, 100)
  8     n = 0
  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:
 12         n += 1
 13         # 获取玩家输入
 14         guess = input(
(Pdb) p n
0
(Pdb) n
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。
> c:\users\pc\repo\week06\guessing_game.py(11)guessing_game()
-> while True:
(Pdb) l
  6
  7     secret_number = random.randint(1, 100)
  8     n = 0
  9
 10     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。")
 11
 12     while True:
 13         n += 1
 14         # 获取玩家输入
 15         guess = input(
 16             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) n
```

```
MINGW64/c/Users/PC/repo/h x + v
(Pdb) n
> c:\users\pc\repo\week06\guessing_game.py(14)guessing_game()
-> guess = input(
(Pdb) l
  9
 10     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。")
 11
 12     while True:
 13         n += 1
 14         # 获取玩家输入
 15         guess = input(
 16             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）: "
 17         )
 18         guess = guess.strip() # 去除多余空白字符
 19
 20         if guess == "q":
(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）: 35
> c:\users\pc\repo\week06\guessing_game.py(17)guessing_game()
-> guess = guess.strip() # 去除多余空白字符
(Pdb) l
 12
 13         n += 1
 14         # 获取玩家输入
 15         guess = input(
 16             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）: "
 17         )
 18         guess = guess.strip() # 去除多余空白字符
 19
 20         if guess == "q":
 21             break
 22
 23     try:
(Pdb) p guess
'35'
```

```
MINGW64/c/Users/PC/repo/h x + v
(Pdb) p type(guess)
<class 'str'>
(Pdb) p " abc".strip()
'abc'
(Pdb) p "abc ".strip()
'abc'
(Pdb) p " abc ".strip()
'abc'
(Pdb) p " ab cd ".strip()
'ab cd'
(Pdb) import wat
(Pdb) wat / str.strip

value: <method 'strip' of 'str' objects>
type: method_descriptor
signature: def strip(self, chars=None, /)
"""
Return a copy of the string with leading and trailing whitespace removed.

If chars is given and not None, remove characters in chars instead.
"""
```

```
MINGW64~/Users/PC/repo/  + -
(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): q
> c:\users\pc\repo\week06\guessing_game.py(17)guessing_game()
-> guess = guess.strip() # 去除多余空白字符
(Pdb) n
> c:\users\pc\repo\week06\guessing_game.py(19)guessing_game()
-> if guess == "q":
(Pdb) l
14         guess = input(
15             f"({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): "
16         )
17         guess = guess.strip() # 去除多余空白字符
18     -> if guess == "q":
19         break
20     -> if guess == "q":
21         break
22     try:
23         guess = int(guess)
24     except ValueError:
(Pdb) p guess
'q'
(Pdb) n
> c:\users\pc\repo\week06\guessing_game.py(20)guessing_game()
-> break
(Pdb) l
15         f"({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): "
16     )
17     guess = guess.strip() # 去除多余空白字符
18 -> if guess == "q":
19     break
20 -> if guess == "q":
21     break
22     try:
23         guess = int(guess)
24     except ValueError:
25         print("输入无效, 请输入一个整数。")
(Pdb) p guess == "q"
True
(Pdb) n
> c:\users\pc\repo\week06\guessing_game.py(46)guessing_game()
-> print("游戏结束, 再见! 🎉")
```

```
(week06)
PC@DESKTOP-326MLMC MINGW64 ~/repo/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): a
输入无效 🎲, 请输入一个整数。
(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): bb
输入无效 🎲, 请输入一个整数。
(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): 30
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): 20
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): 10
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): 5
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): 4
恭喜你 🎉, 猜对了!
游戏结束, 再见! 🎉
```

2. 创建一个 flow_controls.py 文件, 理解 Python 流程控制语句。

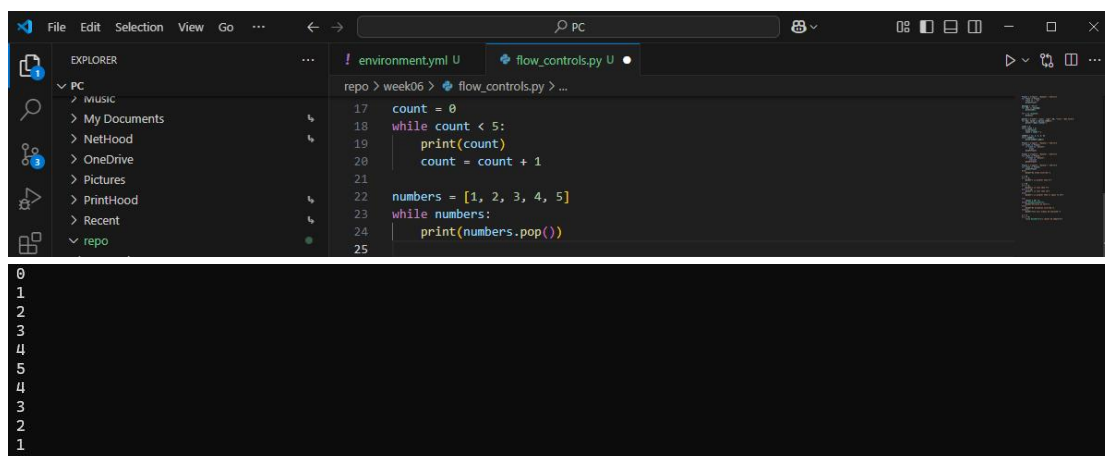
(1) for 迭代循环, for 循环用于遍历可迭代对象 (像列表、元组、字符串等) 中的每个元素。

```
File Edit Selection View Go ...  PC
EXPLORER
PC
  MUSIC
  My Documents
  NetHood
  OneDrive
  Pictures
  PrintHood
  Recent
  repo
  mywork
  proj1
  week01
  week02
  week03
  week04

environment.yml U  flow_controls.py U X
repo > week06 > flow_controls.py > ...
1  fruits = ["apple", "banana", "cherry"]
2  for fruit in fruits:
3      fruit += ", ok"
4      print(fruit)
5
6  message = "Hello"
7  for char in message:
8      print(char)
9
10 for i in range(5):
11     print(i)
12
13 person = {"name": "john", "age": 30, "city": "New York"}
14 for key, value in person.items():
15     print(f"{key}:{value}")
16
```

```
(week06)
PC@DESKTOP-326MLMC MINGW64 ~/repo/week06 (main)
$ python flow_controls.py
apple, ok
banana, ok
cherry, ok
H
e
l
l
o
0
1
2
3
4
name:john
age:30
city:New York
```

(2) while 条件循环，while 循环会在指定条件为真时持续执行代码。



```
environment.yml U  flow_controls.py U
repo > week06 > flow_controls.py > ...
17 count = 0
18 while count < 5:
19     print(count)
20     count = count + 1
21
22 numbers = [1, 2, 3, 4, 5]
23 while numbers:
24     print(numbers.pop())
25
```

```
0
1
2
3
4
5
4
3
2
1
```

(3) break 打断跳出循环，break 语句用于在循环中立即终止循环并跳出。

(4) continue 跳至下一轮循环，continue 语句用于跳过当前循环的剩余代码，直接进入下一轮循环。

(5) for...else 循环未被打断的处理，for 循环结束后，如果没有被 break 语句打断，就会执行 else 中的代码。

(6) if 条件分支，if 语句用于根据条件判断是否执行代码。

(7) if...elif [...elif] 多重条件分支，elif 语句用于在 if 条件不满足时检查其他条件。

(8) if...else 未满足条件的处理，else 语句用于在 if 条件不满足时执行代码。

(9) try...except [...except...else...finally] 捕捉异常的处理，try 用于包裹可能会抛出异常的代码，except 用于捕获并处理异常，else 在没有异常发生时执行，finally 无论是否有异常都会执行。

(10) raise 语句用于主动抛出异常，

```
repo > week06 > flow_controls.py U
26 fruits = ["apple", "banana", "cherry"]
27 for fruit in fruits:
28     if fruit == "banana":
29         break
30     print(fruit)
31
32 fruits = ["apple", "banana", "cherry"]
33 for fruit in fruits:
34     if fruit == "banana":
35         continue
36     print(fruit)
37
38 fruits = ["apple", "banana", "cherry"]
39 for fruit in fruits:
40     print(fruit)
41 else:
42     print("No break occurred.")
43
44 x = 10
45 if x > 5:
46     print("x is greater than 5")
47
48 x = 10
49 if x < 5:
50     print("x is less than 5")
51 elif x < 15:
52     print("x is less than 15")
53 else:
54     print("x is greater than or equal to 15")
55
56 try:
57     result = 10 / 0
58 except ZeroDivisionError:
59     print("Division by zero!")
60 else:
61     print("No exception occurred.")
62 finally:
63     print("This will always be executed.")
64
65 x = -1
66 if x < 0:
67     raise ValueError("x cannot be negative")
68
```

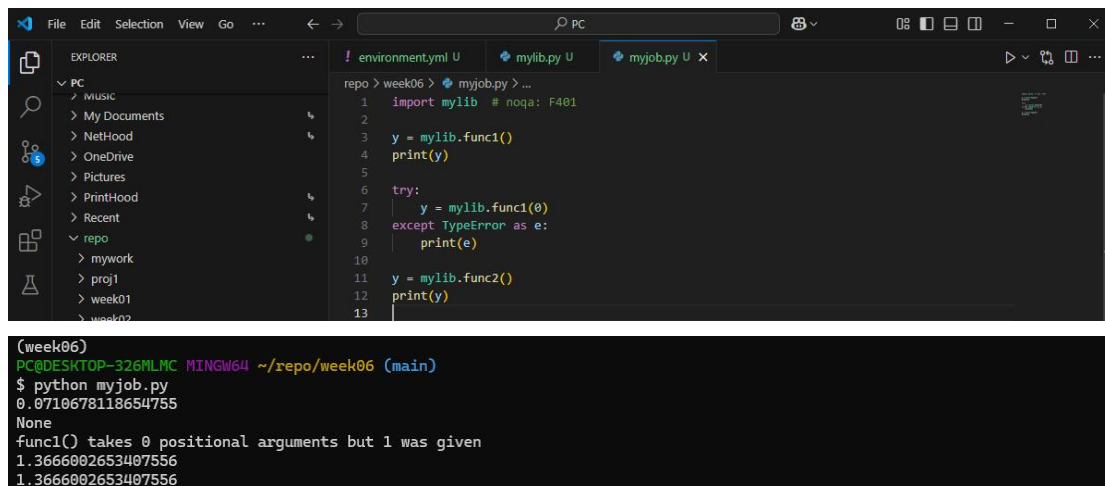
```
apple
apple
cherry
apple
banana
cherry
No break occurred.
x is greater than 5
x is less than 15
Division by zero!
This will always be executed.
Traceback (most recent call last):
  File "C:\Users\PC\repo\week06\flow_controls.py", line 67, in <module>
    raise ValueError("x cannot be negative")
ValueError: x cannot be negative
```

3. 创建一个 mylib.py 模块(module)，在里面定义各类函数，再创建一个 myjob.py 脚本 (script)，从 mylib.py 导入已定义函数并调用。

(1) 函数 func1，没有形参，没有返回值

(2) 函数 func2，没有形参，有返回值

```
environment.yml U mylib.py U myjob.py U
repo > week06 > mylib.py U
1 def func1():
2     x = 50
3     y = x**0.5 - 7
4     print(y)
5
6
7 def func2():
8     x = 70
9     y = x**0.5 - 7
10    print(y)
11    return y
12
```



The screenshot shows the Visual Studio Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders like 'music', 'My Documents', 'Nethood', 'OneDrive', 'Pictures', 'PrintHood', 'Recent', 'repo', 'mywork', 'proj1', 'week01', and 'week02'. The code editor shows a file named 'myjob.py' with the following code:

```
1 import mylib # noqa: F401
2
3 y = mylib.func1()
4 print(y)
5
6 try:
7     y = mylib.func1(0)
8 except TypeError as e:
9     print(e)
10
11 y = mylib.func2()
12 print(y)
13
```

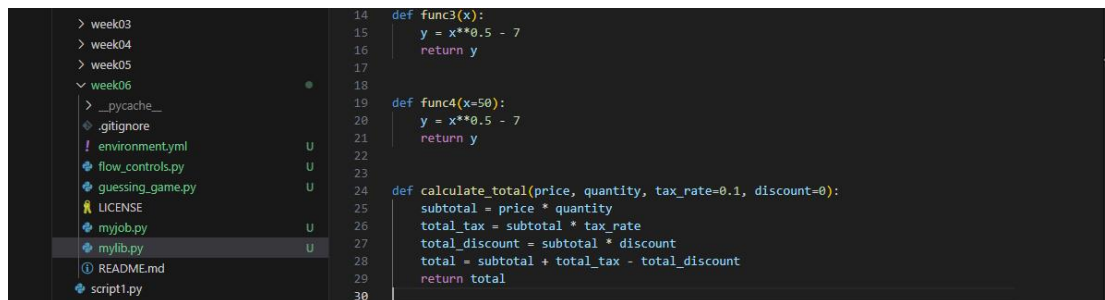
The terminal output shows the command 'python myjob.py' being executed, resulting in the following output:

```
(week06)
PC@DESKTOP-326MLMC MINGW64 ~/repo/week06 (main)
$ python myjob.py
0.0710678118654755
None
func1() takes 0 positional arguments but 1 was given
1.3666002653407556
1.3666002653407556
```

(3) 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。

(4) 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。

(5) 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。



The screenshot shows the Visual Studio Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders like 'week03', 'week04', 'week05', 'week06', and files like 'myjob.py', 'mylib.py', 'README.md', and 'script1.py'. The code editor shows a file named 'mylib.py' with the following code:

```
14 def func3(x):
15     y = x**0.5 - 7
16     return y
17
18
19 def func4(x=50):
20     y = x**0.5 - 7
21     return y
22
23
24 def calculate_total(price, quantity, tax_rate=0.1, discount=0):
25     subtotal = price * quantity
26     total_tax = subtotal * tax_rate
27     total_discount = subtotal * discount
28     total = subtotal + total_tax - total_discount
29     return total
30
```

```

14 y = mylib.func3(45)
15 print(y)
16
17 y = mylib.func3(x=47)
18 print(y)
19
20 try:
21     y = mylib.func3()
22 except TypeError as e:
23     print(e)
24
25 try:
26     mylib.func3(y=47)
27 except TypeError as e:
28     print(e)
29
30 y = mylib.func4(48)
31 print(y)
32
33 y = mylib.func4(x=49)
34 print(y)
35
36 y = mylib.func4()
37 print(y)
38
39 print(mylib.calculate_total(10, 5))
40 print(mylib.calculate_total(price=20, quantity=3, tax_rate=0.15, discount=0.1))
41 print(mylib.calculate_total(10, 5, tax_rate=0.2, discount=0.05))

```

```

-0.2917960675006306
-0.1443453995989561
func3() missing 1 required positional argument: 'x'
func3() got an unexpected keyword argument 'y'
-0.07179676972449123
0.0
0.0710678118654755
55.0
63.0
57.5

```

(6) 函数 func6，在形参列表中使用 / 来限定只接受位置实参的形参

(7) 函数 func7，在形参列表中使用 * 来限定只接受命名实参的形参

```

32 def func6(price, quantity, /, tax_rate=0.1, discount=0):
33     subtotal = price * quantity
34     total_tax = subtotal * tax_rate
35     total_discount = subtotal * discount
36     total = subtotal + total_tax - total_discount
37     return total
38
39
40 def func7(price, quantity, tax_rate=0.1, *, discount=0):
41     subtotal = price * quantity
42     total_tax = subtotal * tax_rate
43     total_discount = subtotal * discount
44     total = subtotal + total_tax - total_discount
45     return total
46

```

```

42 try:
43     print(mylib.func6(price=20, quantity=3))
44 except TypeError as e:
45     print(e)
46
47 try:
48     print(mylib.func7(10, 5, 0.1, 0))
49 except TypeError as e:
50     print(e)
51

```

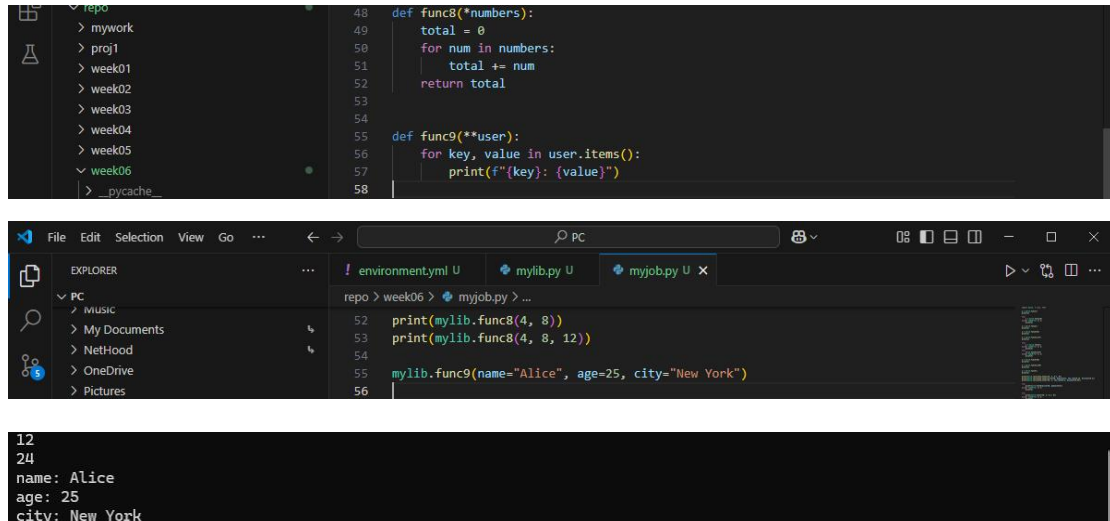
```

func6() got some positional-only arguments passed as keyword arguments: 'price, quantity'
func7() takes from 2 to 3 positional arguments but 4 were given

```


(8) 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。

(9) 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。



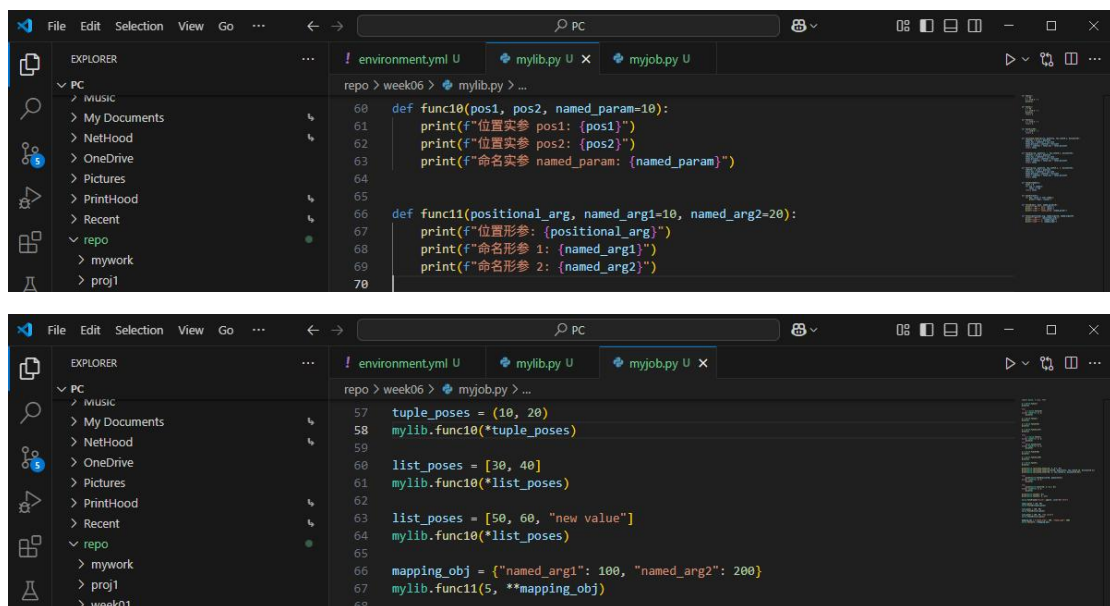
```
48 def func8(*numbers):
49     total = 0
50     for num in numbers:
51         total += num
52     return total
53
54
55 def func9(**user):
56     for key, value in user.items():
57         print(f"{key}: {value}")
58
```

```
52 print(mylib.func8(4, 8))
53 print(mylib.func8(4, 8, 12))
54
55 mylib.func9(name="Alice", age=25, city="New York")
56
```

```
12
24
name: Alice
age: 25
city: New York
```

(10) 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 * 将可迭代对象自动解包, 按位置实参传入。

(11) 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 ** 将映射对象自动解包, 按命名实参传入。



```
60 def func10(pos1, pos2, named_param=10):
61     print(f"位置实参 pos1: {pos1}")
62     print(f"位置实参 pos2: {pos2}")
63     print(f"命名实参 named_param: {named_param}")
64
65
66 def func11(positional_arg, named_arg1=10, named_arg2=20):
67     print(f"位置形参: {positional_arg}")
68     print(f"命名形参 1: {named_arg1}")
69     print(f"命名形参 2: {named_arg2}")
70
```

```
57 tuple_poses = (10, 20)
58 mylib.func10(*tuple_poses)
59
60 list_poses = [30, 40]
61 mylib.func10(*list_poses)
62
63 list_poses = [50, 60, "new value"]
64 mylib.func10(*list_poses)
65
66 mapping_obj = {"named_arg1": 100, "named_arg2": 200}
67 mylib.func11(5, **mapping_obj)
68
```

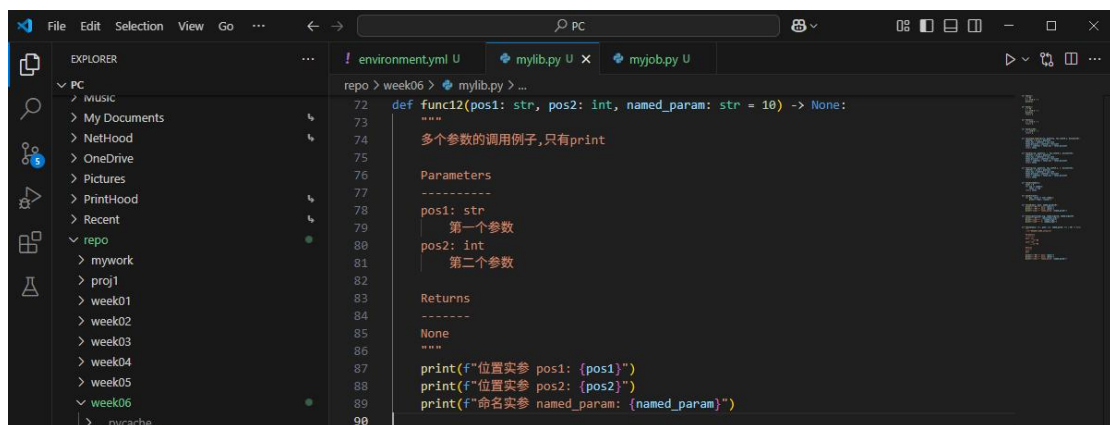


```

位置实参 pos1: 10
位置实参 pos2: 20
命名实参 named_param: 10
位置实参 pos1: 30
位置实参 pos2: 40
命名实参 named_param: 10
位置实参 pos1: 50
位置实参 pos2: 60
命名实参 named_param: new value
位置形参: 5
命名形参 1: 100
命名形参 2: 200

```

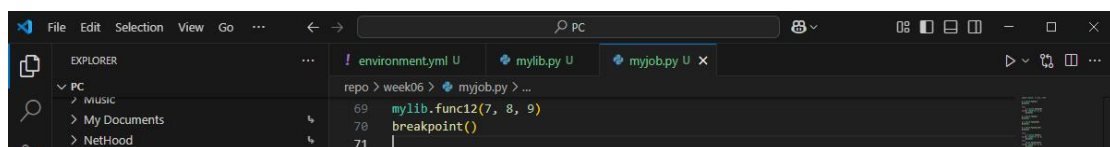
(12) 函数 func12，给函数添加内嵌文档（docstring），给形参和返回值添加类型注解（type annotation），提高函数签名的可读性。



```

File Edit Selection View Go ...
PC
EXPLORER
  PC
    MUSIC
    My Documents
    NetHood
    OneDrive
    Pictures
    PrintHood
    Recent
    repo
      mywork
      proj1
      week01
      week02
      week03
      week04
      week05
      week06
        __pycache__
mylib.py
myjob.py
repo > week06 > mylib.py > ...
72 def func12(pos1: str, pos2: int, named_param: str = 10) -> None:
73     """
74     多个参数的调用例子,只有print
75
76     Parameters
77     -----
78     pos1: str
79         第一个参数
80     pos2: int
81         第二个参数
82
83     Returns
84     -----
85     None
86     """
87     print(f"位置实参 pos1: {pos1}")
88     print(f"位置实参 pos2: {pos2}")
89     print(f"命名实参 named_param: {named_param}")
90

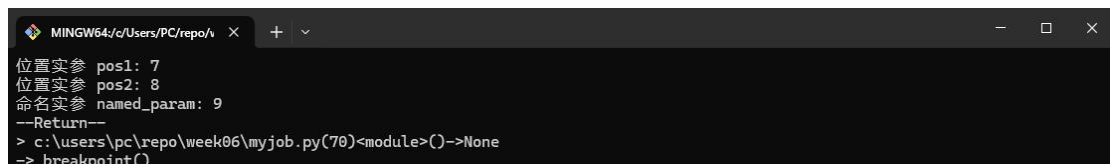
```



```

File Edit Selection View Go ...
PC
EXPLORER
  PC
    MUSIC
    My Documents
    NetHood
mylib.py
myjob.py
repo > week06 > myjob.py > ...
69 mylib.func12(7, 8, 9)
70 breakpoint()
71

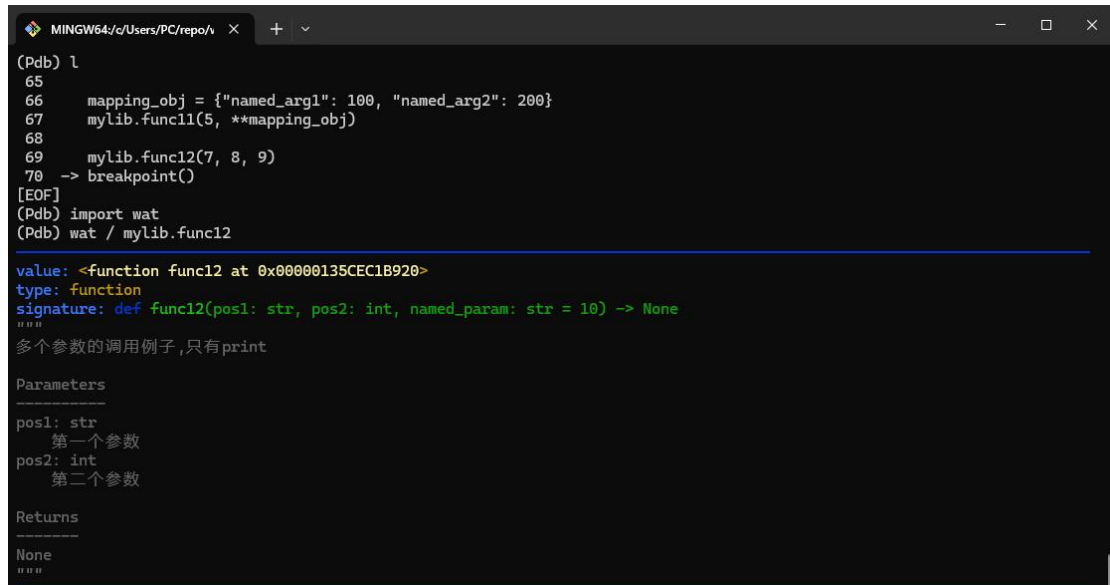
```



```

MINGW64/c/Users/PC/repo/
位置实参 pos1: 7
位置实参 pos2: 8
命名实参 named_param: 9
--Return--
> c:\users\pc\repo\week06\myjob.py(70)<module>()->None
-> breakpoint()

```



```

MINGW64/c/Users/PC/repo/
(Pdb) l
65
66     mapping_obj = {"named_arg1": 100, "named_arg2": 200}
67     mylib.func11(5, **mapping_obj)
68
69     mylib.func12(7, 8, 9)
70     -> breakpoint()
[EOF]
(Pdb) import wat
(Pdb) wat / mylib.func12

value: <function func12 at 0x00000135CEC1B920>
type: function
signature: def func12(pos1: str, pos2: int, named_param: str = 10) -> None
"""
多个参数的调用例子,只有print

Parameters
-----
pos1: str
    第一个参数
pos2: int
    第二个参数

Returns
-----
None
"""

```

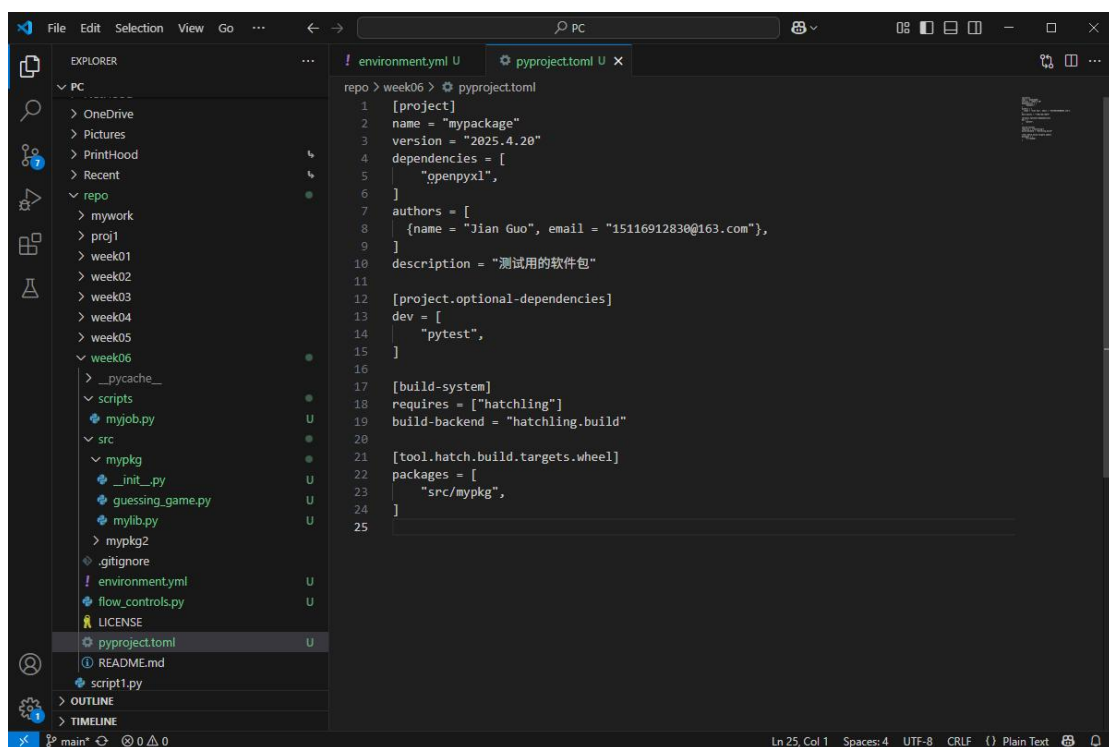
4. 把 mylib.py 模块转变为软件包（package）安装进当前的 Conda 环境来使用。

（1）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。

（2）创建 pyproject.toml 配置文件并构建（build）相应配置。

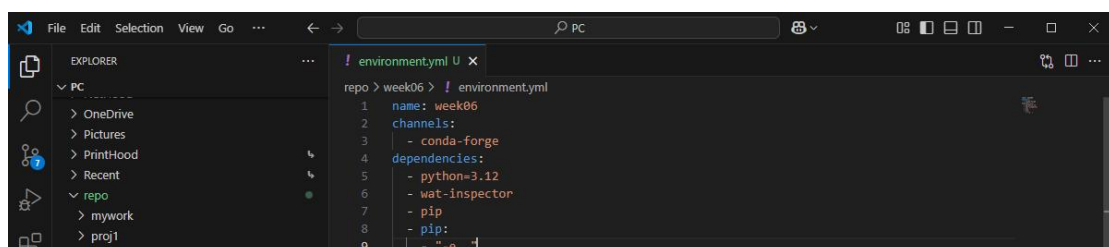
（3）使用 `pip install -e .` 以本地可编辑模式把当前软件包安装进当前 Conda 环境。

（4）修改 environment.yml 文件，使得 `conda env create` 自动安装本地可编辑软件包。



The screenshot shows the VS Code interface with the `pyproject.toml` file open. The Explorer panel on the left shows the project structure, including the `src/mypkg` directory. The main editor displays the following TOML configuration:

```
1 [project]
2 name = "mypackage"
3 version = "2025.4.20"
4 dependencies = [
5     "openpyxl",
6 ]
7 authors = [
8     {name = "Jian Guo", email = "15116912830@163.com"},
9 ]
10 description = "测试用的软件包"
11
12 [project.optional-dependencies]
13 dev = [
14     "pytest",
15 ]
16
17 [build-system]
18 requires = ["hatchling"]
19 build-backend = "hatchling.build"
20
21 [tool.hatch.build.targets.wheel]
22 packages = [
23     "src/mypkg",
24 ]
25
```



The screenshot shows the VS Code interface with the `environment.yml` file open. The Explorer panel on the left shows the project structure. The main editor displays the following YAML configuration:

```
1 name: week06
2 channels:
3   - conda-forge
4 dependencies:
5   - python=3.12
6   - wat-inspector
7   - pip
8   - pip:
9     - -e .
```