

金融编程与计算

Week6——Python 代码组织（初级）

一、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

（一）fork

The screenshot shows the 'Fork' page on GitCode. The URL is <https://gitcode.com/cueb-fintech/week06/fork/create>. The page title is '首经金融科技实... / 第06周打卡'. Below the title, there are tabs for '代码', 'Issues', 'Pull Requests', '讨论', and '分析'. The 'Fork项目' section contains the following fields:

- 项目名称**: 第06周打卡 (Note: When forking a project, the system will default to using the same name as the original project, but you can also use a custom name.)
- 项目路径**: TTATLib / week06 (Note: When forking a project, the system will default to using the same path as the original project, but you can also use a custom path.)
- 项目描述**: 学生 Fork 此仓库并通过 PR 提交第 6 周学习报告

Below the 'Fork项目' section, there is a '选择要 Fork 的分支' (Select the branch to fork) dropdown menu with 'main' selected. A note states: '只复制 main 分支。你可以在 fork 的项目中创建你自己的分支版本，并通过 Pull Request 将你写的代码贡献给 cueb-fintech/week06 项目。' (Only the main branch is copied. You can create your own branch version in the forked project and contribute your code to the cueb-fintech/week06 project via Pull Request.)

At the bottom, there are two buttons: '取消' (Cancel) and '创建Fork项目' (Create Fork Project).

（二）Clone

The screenshot shows a 'Clone' dialog box with a close button (X) in the top right corner. The dialog has two tabs: 'HTTPS' and 'SSH'. The 'SSH' tab is selected. The dialog contains the following instructions and code snippets:

- # 使用 SSH 协议时，请在本地生成 SSH 公钥进行克隆、推送等操作**
- # 复制项目地址**
`git@gitcode.com:TTATLib/week06.git`
- # 生成 RSA 密钥**
`ssh-keygen -t rsa -b 2048 -C TTATLib@noreply.gitcode.com`
- # 查看 RSA 公钥，并配置到 SSH key 中**
`cat ~/.ssh/id_rsa.pub`
- # 克隆到本地**
`git clone git@gitcode.com:TTATLib/week06.git`

```

(base) Administrator@PC-20220115WOJX MINGW64 ~
$ cd repo

(base) Administrator@PC-20220115WOJX MINGW64 ~/repo
$ ls -l
total 28
drwxr-xr-x 1 Administrator 197121 0 3月 24 01:38 myproject/
drwxr-xr-x 1 Administrator 197121 0 3月 17 01:10 mywork/
drwxr-xr-x 1 Administrator 197121 0 3月 23 00:57 prj1/
-rw-r--r-- 1 Administrator 197121 0 3月 10 00:44 script1.py
drwxr-xr-x 1 Administrator 197121 0 3月 10 02:18 week01/
drwxr-xr-x 1 Administrator 197121 0 3月 17 01:17 week02/
drwxr-xr-x 1 Administrator 197121 0 3月 24 02:34 week03/
drwxr-xr-x 1 Administrator 197121 0 3月 30 21:54 week04/
drwxr-xr-x 1 Administrator 197121 0 4月 14 00:14 week05/
drwxr-xr-x 1 Administrator 197121 0 3月 19 21:03 work/

(base) Administrator@PC-20220115WOJX MINGW64 ~/repo
$ git clone git@gitcode.com:TTATLib/week06.git
Cloning into 'week06'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 5 (delta 0), reused 5 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (5/5), 8.45 KiB | 1.69 MiB/s, done.

```

二、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

（一）Week04 里面有了，直接 cp:

```

(base) Administrator@PC-20220115WOJX MINGW64 ~/repo/week06 (main)
$ ls -l ../myproject/
total 52114
-rw-r--r-- 1 Administrator 197121 87 3月 24 01:28 environment.yml
-rw-r--r-- 1 Administrator 197121 53362688 3月 24 01:53 EPA_SmartLocationDatabase_V3
-rw-r--r-- 1 Administrator 197121 713 3月 24 01:52 main.py

(base) Administrator@PC-20220115WOJX MINGW64 ~/repo/week06 (main)
$ cat ../myproject/environment.yml
name: myproject
channels:
  - conda-forge
dependencies:
  - python=3.12
  - pandas

(base) Administrator@PC-20220115WOJX MINGW64 ~/repo/week06 (main)
$ cp ../myproject/environment.yml
cp: missing destination file operand after '../myproject/environment.yml'
Try 'cp --help' for more information.

(base) Administrator@PC-20220115WOJX MINGW64 ~/repo/week06 (main)
$ cp ../myproject/environment.yml ./

```

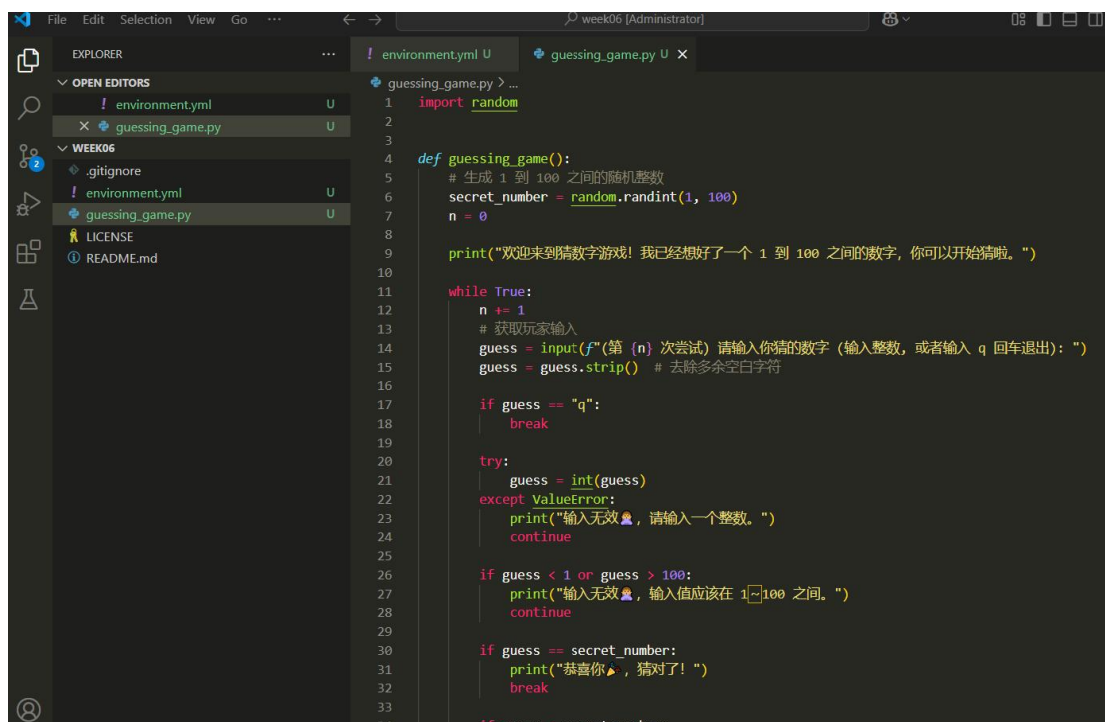
改名字，name 改为 week06

```
! environment.yml U X
! environment.yml
1  name: week06
2  channels:
3    - conda-forge
4  dependencies:
5    - python=3.12
6    - pandas
```

(二) 运行 `conda env create`，创建 conda 环境

Conda activate week06 激活环境

三、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：



```
File Edit Selection View Go ... week06 [Administrator]
EXPLORER
  OPEN EDITORS
    ! environment.yml U
    X guessing_game.py U
  WEEK06
    .gitignore
    ! environment.yml U
    guessing_game.py U
    LICENSE
    README.md
! environment.yml U X
guessing_game.py U X
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         # 获取玩家输入
14         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): ")
15         guess = guess.strip() # 去除多余空白字符
16
17         if guess == "q":
18             break
19
20         try:
21             guess = int(guess)
22         except ValueError:
23             print("输入无效, 请输入一个整数。")
24             continue
25
26         if guess < 1 or guess > 100:
27             print("输入无效, 输入值应该在 1-100 之间。")
28             continue
29
30         if guess == secret_number:
31             print("恭喜你, 猜对了!")
32             break
33
34         if guess < secret_number:
```

进入调试环境

```
(week06)
Administrator@PC-20220115W0JX MINGW64 ~/repo/week06 (main)
$ python -m pdb guessing_game.py
> c:\users\administrator\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，你可以来猜")
10
11     while True:
(Pdb) p random.randint(1, 100)
*** NameError: name 'random' is not defined. Did you forget to import 'random'
```

出现问题：解决如下

在 (Pdb) 提示符下输入 n (next 的缩写)，让程序执行完 import random 这行代码，之后再进行与 random.randint 相关的调试操作，就不会报此错误了。

```
(Pdb) n
> c:\users\administrator\repo\week06\guessing_game.py(4)<module>()
-> def guessing_game():
(Pdb) p random.randint(1, 100)
85
```

每调用一次，就会返回一个 (1, 100) 之间的随机整数

```
(Pdb) p random.randint(1, 100)
*** NameError: name 'random' is not defined
(Pdb) n
> c:\users\administrator\repo\week06\guessing_game.py(4)<module>()
-> def guessing_game():
(Pdb) p random.randint(1, 100)
85
(Pdb) p random.randint(1, 100)
10
(Pdb)
49
(Pdb) p random.randint(1, 100)
57
(Pdb) p random.randint(1, 100)
16
(Pdb) p random.randint(1, 100)
39
```

```

(Pdb) n
> c:\users\administrator\repo\week06\guessing_game.py(47)<module>()
-> if __name__ == "__main__":
(Pdb) n
> c:\users\administrator\repo\week06\guessing_game.py(48)<module>()
-> guessing_game()
(Pdb) l
43
44         print("游戏结束，再见 🙋。")
45
46
47     if __name__ == "__main__":
48 ->         guessing_game()
[EOF]

```

退出调试模式，直接运行

```

(Pdb) q
(week06)
Administrator@PC-20220115W0JX 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）：

```

再次进入调试

```

(Pdb) n
> c:\users\administrator\repo\week06\guessing_game.py(4)<module>()
-> def guessing_game():
(Pdb)
> c:\users\administrator\repo\week06\guessing_game.py(47)<module>()
-> if __name__ == "__main__":
(Pdb)
> c:\users\administrator\repo\week06\guessing_game.py(48)<module>()
-> guessing_game()
(Pdb) s
--Call--
> c:\users\administrator\repo\week06\guessing_game.py(4)guessing_game()
-> def guessing_game():

```

注：guessing_game()要输入 s 进去，进入函数内部

一直回车、回车、回车，然后 p secret_number:

```
(Pdb) n
> c:\users\administrator\repo\week06\guessing_game.py(6)guessing_game()
-> secret_number = random.randint(1, 100)
(Pdb)
> c:\users\administrator\repo\week06\guessing_game.py(7)guessing_game()
-> n = 0
(Pdb)
> c:\users\administrator\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)
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\administrator\repo\week06\guessing_game.py(11)guessing_game()
-> while True:
(Pdb)
> c:\users\administrator\repo\week06\guessing_game.py(12)guessing_game()
-> n += 1
(Pdb)
> c:\users\administrator\repo\week06\guessing_game.py(14)guessing_game()
-> guess = input(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): ")
(Pdb) p n
1
(Pdb) p secret_number
32
```

```
(Pdb) n
> c:\users\administrator\repo\week06\guessing_game.py(21)guessing_game()
-> guess = int(guess)
(Pdb) l
16
17             if guess == "q":
18                 break
19
20             try:
21 ->                 guess = int(guess)
22             except ValueError:
23                 print("输入无效 🙅, 请输入一个整数。")
24                 continue
25
26             if guess < 1 or guess > 100:
(Pdb) p guess
'50'
(Pdb) p int(guess)
50
```

注：if 是条件判断，except 是 try 的分支，不运行的话就直接跳到下

一行，如果 用户输入的不是数字，就会报错，如下图：

```
(Pdb) n
> c:\users\administrator\repo\week06\guessing_game.py(21)guessing_game()
-> guess = int(guess)
(Pdb) l
16
17             if guess == "q":
18                 break
19
20             try:
21 ->                 guess = int(guess)
22             except ValueError:
23                 print("输入无效 🙅, 请输入一个整数。")
24                 continue
25
26             if guess < 1 or guess > 100:
(Pdb) p int(guess)
*** ValueError: invalid literal for int() with base 10: '85a'
(Pdb)
```

```

(Pdb) n
> c:\users\administrator\repo\week06\guessing_game.py(26)guessing_game()
-> if guess < 1 or guess > 100:
(Pdb) l
21             guess = int(guess)
22         except ValueError:
23             print("输入无效 🙅，请输入一个整数。")
24             continue
25
26 ->         if guess < 1 or guess > 100:
27             print("输入无效 🙅，输入值应该在 1~100 之间。")
28             continue
29
30         if guess == secret_number:
31             print("恭喜你 🎉，猜对了！")
(Pdb) p guess
88
(Pdb) p guess < 1
False

```

字符串会报错 '88'

退出调试，开始玩：

```

(week06)
Administrator@PC-20220115WOJX 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): 54
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): 70
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): 80
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): 75
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): 78
猜的数字太小了，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): 79
恭喜你 🎉，猜对了！
游戏结束，再见 🙋。

(week06)
Administrator@PC-20220115WOJX 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): 85
猜的数字太大了，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): 65
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): 75
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): 70
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): 68
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): 67
猜的数字太大了，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): 66
恭喜你 🎉，猜对了！
游戏结束，再见 🙋。

```

四、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)
- `while` 条件循环 (conditional loop)
- `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

(一) `for` 迭代循环 (iteration loop)

让豆包生成例子

1. 遍历列表

```
flow_controls.py > ...
1  fruits = ["apple", "banana", "cherry"]
2  for fruit in fruits:
3      print(fruit)
4  # 输出:
5  # apple
6  # banana
7  # cherry
```



```
(week06)
Administrator@PC-20220115WOJX MINGW64 ~/repo/week06
$ python flow_controls.py
apple
banana
cherry
```

```
flow_controls.py > ...
1  fruits = ["apple", "banana", "cherry"]
2  for fruit in fruits:
3      fruit += ", ok"
4      print(fruit)
5
6  message = "hello"
7  # 输出:
8  # apple
9  # banana
10 # cherry
```

```
(week06)
Administrator@PC-20220115WOJX MINGW64 ~/repo/week06 (r
$ python flow_controls.py
apple, ok
banana, ok
cherry, ok
```

把字符都取出来

```
message = "hello"
for char in message:
    print(char)
```

h
e
l
l
o

2. 遍历字典（键值对）

```
person = {"name": "Alice", "age": 25, "city": "New York"}
for key, value in person.items():
    print(f"{key}: {value}")
```

输出:

name: Alice

age: 25

city: New York

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

（二）while 条件循环 (conditional loop)

1. 简单循环计数

```
count = 1
while count <= 5:
    print(count)
    count += 1
# 输出: 1 2 3 4 5
```

2. 无限循环 + break 退出

```
while True:
    user_input = input("输入内容 (输入 'q' 退出) : ")
    if user_input == 'q':
        print("退出循环")
        break
    print(f"你输入了: {user_input}")
```

```
输入内容 (输入 'q' 退出) :
```

(三) raise 主动抛出异常

1. 抛出内置异常（以 ValueError 为例）

```
def check_positive(num):  
    if num <= 0:  
        raise ValueError("数字必须是正数")  
    return num  
  
try:  
    result = check_positive(-5)  
except ValueError as e:  
    print(f"捕获异常: {e}") # 输出: 捕获异常: 数字必须是正数
```

```
退出循环  
请输入一个整数: 10  
你输入的整数是: 10  
捕获异常: 数字必须是正数
```

(四) python 用 try 和 raise 配合流程控制

```
try:  
    # 获取用户输入并转换为整数  
    num = int(input("请输入 1 到 100 之间的整数: "))  
    # 检查输入是否符合范围, 不符合则抛出异常  
    if num < 1 or num > 100:  
        raise ValueError("输入的数字必须在 1 到 100 之间")  
    print(f"你输入的数字是 {num}, 符合要求!")  
except ValueError as ve:  
    # 捕获并处理值错误 (输入范围不正确)  
    print(f"值错误: {ve}")  
except Exception as e:  
    # 捕获其他意外异常 (如输入非数字)  
    print(f"发生其他错误: {e}")
```

```
捕获异常: 数字必须是正数  
请输入 1 到 100 之间的整数: 85  
你输入的数字是 85, 符合要求!
```

五、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

(一) 定义函数 `func1`，没有形参，没有返回值

有返回值就是写 `return`

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

myjob.py
1 import mylib # noqa
2
3 breakpoint()
```

```
myjob.py
1 import mylib # noqa: F401
2
3 mylib.func1()
```

```
Administrator@PC-2022
$ python myjob.py
0.0710678118654755
```

添加返回值

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

$ python myjob.py
0.0710678118654755
None
```

(二) 定义函数 `func2`，没有形参，有返回值

```
def func2():
    x = 70
    y = x**0.5 - 7
    print(y)
    return y

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

```
None
1.3666002653407556
1.3666002653407556
(1.3666002653407556)
```

为什么有两个？

Function 内部也有一个 print，print 完了之后有个返回值

(三) 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)

注：位置形参：def func3(x) () 里面有 x

定义函数时，叫形参 (parameter)，形式上的参数，类似占位符

调用函数时，给函数赋值，叫实参 (argument)

位置实参：给个值

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

1. 不传实参会报错

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

```
TypeError: func3() missing 1 required positional argument: 'x'
(1.3666002653407556)
```

2. 传实参

```
y = mylib.func3(85)
print(y)
```

```
1.3666002653407556
1.3666002653407556
2.219544457292887
(2.219544457292887)
```


(四) 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)

命名形参:

```
def func4(x=50):
```

```
y = mylib.func4(48)
print(y)
```

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

```
-0.07179676972449123
0.0
```

(五) 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

```
# 定义一个函数，包含位置形参和命名形参
def calculate_area(length, width=10):
    """
    计算矩形的面积
    :param length: 矩形的长，位置形参
    :param width: 矩形的宽，命名形参，默认值为 10
    :return: 矩形的面积
    """
    return length * width

# 使用位置实参调用函数
area1 = calculate_area(5)
print(f"使用位置实参，面积为: {area1}")

# 使用位置实参和命名实参混合调用函数
area2 = calculate_area(8, width=12)
print(f"使用位置实参和命名实参混合，面积为: {area2}")

# 只使用命名实参调用函数
area3 = calculate_area(length=6, width=15)
print(f"只使用命名实参，面积为: {area3}")
```

```
print(mylib.calculate_area(10))
```

```
100
```

(六) 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(length, /, width=10):  
    """  
    计算矩形的面积  
    :param length: 矩形的长, 位置形参  
    :param width: 矩形的宽, 命名形参, 默认值为 10  
    :return: 矩形的面积  
    """  
    return length * width  
  
# 使用位置实参调用函数  
area1 = calculate_area(5)  
print(f"使用位置实参, 面积为: {area1}")  
  
# 使用位置实参和命名实参混合调用函数  
area2 = calculate_area(8, width=12)  
print(f"使用位置实参和命名实参混合, 面积为: {area2}")  
  
# 只使用命名实参调用函数  
area3 = calculate_area(length=6, width=15)  
print(f"只使用命名实参, 面积为: {area3}")
```

```
print(mylib.func6(10))
```

```
0.0  
100  
100
```

(八) 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)

() 里面可以给任意数量的位置实参：

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

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

12

什么都不给就是 0

(九) 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)

```
def func9(**user_info):  
    return user_info  
user = func9(name="Alice", age=25, city="New York")  
print(user)
```

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

```
{'name': 'Alice', 'age': 25, 'city': 'New York'}
```

(十) 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

```
def func10(a, b, c=10):
    print(f"位置形参 a = {a}, 位置形参 b = {b}, 命名形参 c = {c}")

# 元组解包
data_tuple = (20, 30)
func10(*data_tuple) # 输出: a=20, b=30, c=10 (c 使用默认值)

# 列表解包 + 显式命名形参
data_list = [5, 15]
func10(*data_list, c=25) # 输出: a=5, b=15, c=25 (c 被显式赋值)
```

```
data_tuple = (20, 30)
mylib.func10(*data_tuple)
data_list = [5, 15]
mylib.func10(*data_list)
```

```
位置形参 a = 20, 位置形参 b = 30, 命名形参 c = 10
位置形参 a = 5, 位置形参 b = 15, 命名形参 c = 10
```

(十一) 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 func11 函数, 接受三个命名形参
def func11(param1=0, param2=1, param3=2):
    print(f"param1 的值是: {param1}")
    print(f"param2 的值是: {param2}")
    print(f"param3 的值是: {param3}")
# 创建一个字典作为映射对象
data_dict = {
    "param1": 10,
    "param2": 20,
    "param3": 30
}
# 使用 ** 解包字典并将其作为命名实参传入函数
func11(**data_dict)
```

```

# 定义 func11 函数, 接受三个命名形参
def func11(param1=0, param2=1, param3=2):
    print(f"param1 的值是: {param1}")
    print(f"param2 的值是: {param2}")
    print(f"param3 的值是: {param3}")

# 创建一个字典作为映射对象
data_dict = {
    "param1": 10,
    "param2": 20,
    "param3": 30
}

print("即将调用 func11 函数, 传递的参数如下:")
for key, value in data_dict.items():
    print(f"{key}: {value}")
print("开始调用 func11 函数...")
func11(**data_dict)
print("func11 函数调用结束。")

```

```

即将调用 func11 函数, 传递的参数如下:
param1: 10
param2: 20
param3: 30
开始调用 func11 函数...
param1 的值是: 10
param2 的值是: 20
param3 的值是: 30
func11 函数调用结束。

```

(十二) 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


```
def func12(a: int, b: int, multiplier: float = 1.0) -> float:
    """
    计算两个整数的和与倍数因子的乘积。

    参数:
        a (int): 参与运算的第一个整数。
        b (int): 参与运算的第二个整数。
        multiplier (float, 可选): 倍数因子, 默认值为 1.0。用于放大 `(a + b)` 的结果。

    返回:
        float: `(a + b) * multiplier` 的计算结果, 返回值为浮点数。
    """
    return (a + b) * multiplier
```

```
# 调用 mylib 中的 func12 函数
result = mylib.func12(a=5, b=3, multiplier=2.0)
print(f"调用结果: {result}")
```

调用结果: 16.0

六、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

(一) 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

(二) 创建 pyproject.toml 配置文件, 按照 文档 填写基本的软件包信息

```
> __pycache__
▼ scripts
  📄 myjob.py
▼ src\mypkg
  📄 guessing_game.py
  📄 mylib.py
📄 .gitignore
! environment.yml
📄 flow_controls.py
📄 LICENSE
📄 README.md
```

```
⚙️ pyproject.toml
1  [project]
2  name = "mypackage"
3  version = "2025.4.14"
4  dependencies = [
5  |     "oppenpyx1",
6  | ]
7  authors = [
8  |     {name = "Rui Zhu", email = "2816232420@qq.com"},
9  | ]
10 description = "测试用的软件包"
11
12 [project.optional-dependencies]
13 dev = [
14 |     "pytest",
15 | ]
```

在终端运行：

```
(base) Administrator@PC-20220115W0JX MINGW64 ~/repo/week06/src (main)
$ python
Python 3.12.7 | packaged by Anaconda, Inc. | (main, Oct 4 2024, 13:17:27)
Type "help", "copyright", "credits" or "license" for more information.
>>> import mypkg
使用位置实参，面积为：50
使用位置实参和命名实参混合，面积为：96
只使用命名实参，面积为：90
使用位置实参，面积为：50
使用位置实参和命名实参混合，面积为：96
{'name': 'Alice', 'age': 25, 'city': 'New York'}
位置形参 a = 20, 位置形参 b = 30, 命名形参 c = 10
位置形参 a = 5, 位置形参 b = 15, 命名形参 c = 25
param1 的值是：10
param2 的值是：20
param3 的值是：30
>>> mypkg.mylib.func1()
0.0710678118654755

>>> import mypkg.guessing_game
>>> mypkg.guessing_game
<module 'mypkg.guessing_game' from 'C:\\Users\\Administrator\\repo\\week06\\src\\mypkg\\guessing_game.py'>
>>> 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):
```

(四) 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
1  name: week06
2  channels:
3    - conda-forge
4  dependencies:
5    - python=3.12
6    - wat-inspector
7    - pip:
8      - "-e ."
```

```
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
Everything found within the environment (D:\anaconda\envs\week06), including
y non-conda files, will be deleted. Do you wish to continue?
(y/[n])? y
```

```
(base) Administrator@PC-20220115W0JX MINGW64 ~/repo/week06 (main)
$ ls -l
total 30
drwxr-xr-x 1 Administrator 197121      0  4月 15 13:57 __pycache__/
-rw-r--r-- 1 Administrator 197121    115  4月 15 20:20 environment.yml
-rw-r--r-- 1 Administrator 197121   1791  4月 14 19:22 flow_controls.py
-rw-r--r-- 1 Administrator 197121  18805  4月 14 16:17 LICENSE
-rw-r--r-- 1 Administrator 197121    399  4月 15 19:40 pyproject.toml
-rw-r--r-- 1 Administrator 197121   2239  4月 14 16:17 README.md
drwxr-xr-x 1 Administrator 197121      0  4月 15 14:15 scripts/
drwxr-xr-x 1 Administrator 197121      0  4月 15 14:20 src/
```

```
(base) Administrator@PC-20220115W0JX MINGW64 ~/repo/week06 (main)
$ conda env create
```