

# Python 代码组织

——语句 (statement)、模块 (module) 和 软件包 (package) 等

## 一、猜数游戏

### 1. 准备环境并复制粘贴游戏代码

```
WEEK06
├── .gitignore
├── environment.yml
├── guessing_game.py
├── LICENSE
└── README.md

guessing_game.py > ...
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(
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         )
17         guess = guess.strip() # 去除多余空白字符
```

### 2. 先尝试运用 AI 理解这个代码

#### 1. 导入模块

```
python
import random
```

- 代码开始时导入了 `random` 模块，这个模块提供了生成随机数的功能，我们将在程序中用它来生成一个随机的“秘密数字”。

#### 2. 定义 `guessing_game()` 函数

```
python
def guessing_game():
```

- 这里定义了一个函数 `guessing_game`，它包含了猜数字游戏的整个流程。

#### 3. 生成随机数

```
python
secret_number = random.randint(1, 100)
```

- 使用 `random.randint(1, 100)` 生成一个 1 到 100 之间的随机整数，作为玩家需要猜的“秘密数字”。这个数字在游戏过程中是 ↓ 的，玩家每次猜的结果都要与它比较。

#### 7. 记录猜测次数

```
python
n += 1
```

- 每次进入循环，玩家的猜测次数 `n` 会增加 1。

#### 8. 获取玩家输入

```
python
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()
```

- 提示玩家输入他们的猜测，并用 `guess.strip()` 去除输入中的多余空格。这个 `input()` 函数会等待用户输入。

#### 4. 初始化计数器

```
python
n = 0
```

- 初始化 `n` 为 0，它用来记录玩家猜测数字的次数。

#### 5. 打印游戏欢迎信息

```
python
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。")
```

- 打印出游戏的欢迎信息，告知玩家游戏的规则。

#### 6. 进入循环：游戏的核心逻辑

```
python
while True:
```

- 游戏的核心逻辑通过 `while True:` 无限循环进行，直到玩家猜对数字或选择退出。

#### 9. 退出条件

```
python
if guess == "q":
    break
```

- 如果玩家输入 "q"，游戏就会退出。这里的 `break` 语句会跳出 `while` 循环，结束游戏。

#### 10. 处理无效输入（无法转换为整数）

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

- 使用 `try-except` 来尝试将玩家的输入转换为整数。如果输入无法转换成整数（比如输入了字母），程序会捕获 `ValueError` 错误，并提示玩家输入无效，重新开始下一次循环。

## 11. 检查输入范围

```
python
复制代码

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

- 检查玩家输入的数字是否在 1 到 100 的有效范围内。如果不在范围内，提示玩家输入无效，并继续下一次循环。

## 12. 判断玩家的猜测

- 这个部分通过 `if-else` 语句来判断玩家的猜测与秘密数字的关系。

### a. 猜对了

```
python
复制代码

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

- 如果玩家的猜测与 `secret_number` 相等，打印“恭喜你猜对了”并跳出循环结束游戏。

### b. 猜小了

```
python
复制代码

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

- 如果玩家猜的数字小于 `secret_number`，提示玩家猜的数字太小，继续下一轮猜测。

### c. 猜大了

```
python
复制代码

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

- 如果玩家猜的数字大于 `secret_number`，提示玩家猜的数字太大，继续下一轮猜测。

## 13. 未处理的异常

```
python
复制代码

raise NotImplementedError
```

- 这里的 `raise NotImplementedError` 其实永远不会被触发，因为所有的情况都已经被上面的 `if-else` 判断处理过了。这个语句是为了代码结构的完整性，或者是未来如果添加新功能时的占位符。

## 14. 游戏结束

```
python
复制代码

print("游戏结束，再见 🙋。")
```

- 游戏结束后，打印再见信息。

## 15. 主程序入口

```
python
复制代码

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

- `if __name__ == "__main__":`：这一行确保只有当脚本被直接运行时，才会调用 `guessing_game()` 函数。如果这个脚本被作为模块导入，`guessing_game()` 就不会自动执行。

## 总结流程：

- 生成一个 1 到 100 的随机数字 `secret_number`。
- 初始化猜测次数 `n`。
- 打印欢迎信息，提示玩家输入猜测。
- 玩家输入数字或退出。
  - 如果输入无效（不是数字或超出范围），程序会提示并要求重新输入。
  - 如果猜对数字，游戏结束，打印“恭喜”信息。
  - 如果猜小了或猜大了，提示玩家继续猜测。
- 游戏结束时打印“再见”信息。

## 3. 启用 pdb 理解、验证上述流程

```
(Pdb) l
 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
(Pdb) p secret_number
40
```

```
(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): 50
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): 80
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): 90
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): 95
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): 98
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): 99
恭喜你 🎉，猜对了！
游戏结束，再见 🙋。
--Return--
> c:\users\71970\repo\week06\guessing_game.py(50)<module>()->None
-> guessing_game()
(Pdb)
```

## 二、理解 Python 流程控制语句

### 1.for 迭代循环

```
print("for 迭代循环 (iteration loop)举例: ")
# 1. 遍历列表
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    fruit += " is good"
    print(fruit)

# 2. 遍历字符串
for char in "hello":
    print(char) # 输出每个字符

# 3. 结合 range() 的固定次数循环
for i in range(3):
    print(i) # 0, 1, 2

# 4. 遍历字典的键值对
person = {"name": "Alice", "age": 25}
for key in person.keys():
    print(key)

person = {"name": "Alice", "age": 25}
for value in person.values():
    print(value)

person = {"name": "Alice", "age": 25}
for key, value in person.items():
    print(f"{key}: {value}") # f-string 的真正价值在于动态嵌入变量或表达式
```

### 2.while 条件循环、continue、break、else

```
print("while 条件循环 (conditional loop)举例: ")
# 1. 基本用法: 条件为真时循环
count = 0
while count < 3:
    print(f"Count: {count}") # 输出 0, 1, 2
    count += 1

# 2. 带 break 的循环 (打断跳出循环)
while True:
    user_input = input("输入 'exit' 退出: ")
    if user_input == "exit":
        break # 中断循环
    print(f"你输入了: {user_input}")

# 3. 带 continue 的循环 (跳至下一轮循环)
num = 0
while num < 5:
    num += 1
    if num == 3:
        continue # 跳过 3
    print(num) # 输出 1, 2, 4, 5

# 4. 结合 else (循环正常结束时执行)
n = 0
while n < 3:
    print(n) # 输出 0, 1, 2
    n += 1
else:
    print("循环完成! ")
```

### 3.if 条件分支

```
# if 条件分支
day = "周三"
if day == "周一":
    print("上班")
elif day == "周二":
    print("继续上班")
elif day == "周三":
    print("摸鱼") # 输出: 摸鱼
elif day == "周四":
    print("快周末了")
else:
    print("休息日")
```

### 4.异常处理

```
# 异常处理
# try...except捕捉异常的处理
try:
    num = int(input("请输入数字: ")) # 可能出错的地方
    result = 10 / num # 可能出错的地方
    print(f"结果是: {result}")
except ValueError:
    print("错误: 请输入有效数字!")
except ZeroDivisionError:
    print("错误: 不能除以0!")
else:
    print("计算成功!") # 没有错误时执行
finally:
    print("程序结束") # 无论是否有错都会执行

# raise主动抛出异常
age = int(input("请输入年龄: "))
if age < 0:
    raise ValueError("年龄不能小于0!")
elif age < 18:
    print("未成年")
else:
    print("成年")
```

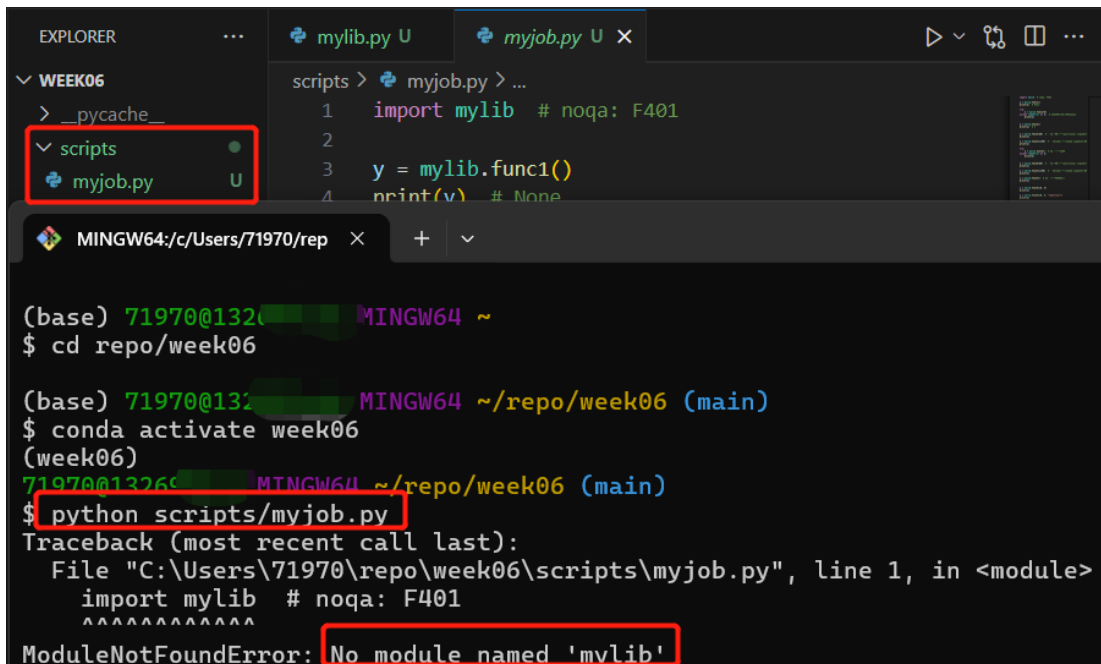
### 三、在模块里定义函数、脚本里调用函数（先 import）

```
mylib.py U X
mylib.py > ...
1 def func1(): # 没有形参, 没有返回值
2     x = 50
3     y = x**0.5 - 7
4     print(y) # 将内容输出到控制台, 仅供查看, 不影响程序逻辑
5
6
7 def func2(): # 没有形参, 有返回值
8     x = 81
9     y = x**0.5 - 7
10    print(y)
11    return y # 结束函数的执行, 并将值返回给调用者, 这个值可以被其他代码
12
13
14 def func3(x): # 有一个位置形参 (positional parameter)
15     y = x**0.5 - 7
16     return y
17
18
19 def func4(x=49): # 有一个命名形参 (named parameter)
20     y = x**0.5 - 7
21     return y
22
23
24 # 接受多个位置形参和命名形参, 注意位置参数必须排在命名参数之前:
25 def func5(a, b, operation="add"):
26     if operation == "add":
27         return a + b
28     elif operation == "subtract":
29         return a - b
30     else:
31         return None
32
33
34 # / 限定其左侧的形参只接受位置实参
35 def func6(a, /, b, operation="add"):
36     if operation == "add":
37         return a + b
38     elif operation == "subtract":
39         return a - b
40     else:
41         return None
42
43
myjob.py U X
myjob.py > ...
1 import mylib # noqa: F401
2
3 y = mylib.func1()
4 print(y) # None
5
6 try:
7     y = mylib.func1(0)
8 except TypeError as e: # 不报错, 返回错误信息
9     print(e)
10
11 y = mylib.func2()
12 print(y) # 2
13
14 y = mylib.func3(60) # 传入位置实参(positional argument)调用
15 print(y)
16
17 y = mylib.func3(x=60) # 传入命名实参(named argument)调用
18 print(y)
19
20 try:
21     y = mylib.func3() # 不传实参报错
22 except TypeError as e:
23     print(e)
24
25 y = mylib.func4(60) # 传入位置实参(positional argument)调用
26 print(y)
27
28 y = mylib.func4(x=60) # 传入命名实参(named argument)调用
29 print(y)
30
31 y = mylib.func4() # 不传实参取默认值
32 print(y)
33
34 y = mylib.func5(10, 6)
35 print(y)
36
37 y = mylib.func5(10, 6, "subtract")
38 print(y)
39
40
41
42
```

```
mylib.py U X
mylib.py > ...
45 def func7(a, b, *, operation="add"):
46     elif operation == "subtract":
47         return a - b
48     else:
49         return None
50
51
52 # 在形参名称前使用 * 允许传入任意数量的位置实参(打包成元组)
53 def func8(*numbers):
54     total = 0
55     for num in numbers:
56         total = total + num
57     return total
58
59
60 # 形参名称前使用 ** 允许传入任意数量的命名实参 (打包为字典)
61 def func9(**user):
62     for key, value in user.items():
63         print(f"{key}:{value}")
64
65
66 # 在调用时使用 * 将可选对象 (如元组或列表) 自动解包, 按位置实参传入
67 def func10(a, b, name="Unknown"):
68     print(f"a = {a}, b = {b}, name = {name}")
69
70
71 # 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
72 def func11(x, name="Unknown", age=0):
73     print(f"x = {x}, name = {name}, age = {age}")
74
75
76 # 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)
77 def func12(name: str, age: int = 18, is_student: bool = True) -> str:
78     """
79     根据用户信息生成欢迎消息。
80     Examples:
81     >>> func12("Alice", 20, False)
82     'Hello Alice, age 20, student: False'
83     """
84     return f"Hello {name}, age {age}, student: {is_student}"
85
86
87
myjob.py U X
myjob.py > ...
25 print(e)
26
27 y = mylib.func4(60) # 传入位置实参(positional argument)调用
28 print(y)
29
30 y = mylib.func4(x=60) # 传入命名实参(named argument)调用
31 print(y)
32
33 y = mylib.func4() # 不传实参取默认值
34 print(y)
35
36 y = mylib.func5(10, 6)
37 print(y)
38
39 y = mylib.func5(10, 6, "subtract")
40 print(y)
41
42 try:
43     y = mylib.func6(a=10, b=6)
44 except TypeError as e:
45     print(e)
46
47 try:
48     y = mylib.func7(10, 6, "subtract")
49 except TypeError as e:
50     print(e)
51
52 print(mylib.func8(4, 8, 8))
53
54 mylib.func9(name="Jac", city="LA")
55
56 args = (10, 20) # 元组
57 mylib.func10(*args, name="Bob")
58
59 kwargs = {"name": "Bob", "age": 30}
60 mylib.func11(20, **kwargs)
61
62 print(mylib.func12("Mia", 23))
63
64
65
66
```

#### 四、软件包的配置、构建与安装

1. vscode 新建文件夹 scripts，把 myjob.py 脚本移进去，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



The screenshot shows the VS Code interface with a file explorer on the left showing a folder named 'scripts' containing 'myjob.py'. The main editor shows the content of 'myjob.py', which includes an import statement for 'mylib'. Below the editor, a terminal window shows the command 'python scripts/myjob.py' being executed, resulting in a 'ModuleNotFoundError: No module named 'mylib'' error.

```
EXPLORER
WEEK06
  > __pycache__
  > scripts
    myjob.py
scripts > myjob.py > ...
1 import mylib # noqa: F401
2
3 y = mylib.func1()
4 print(y) # None

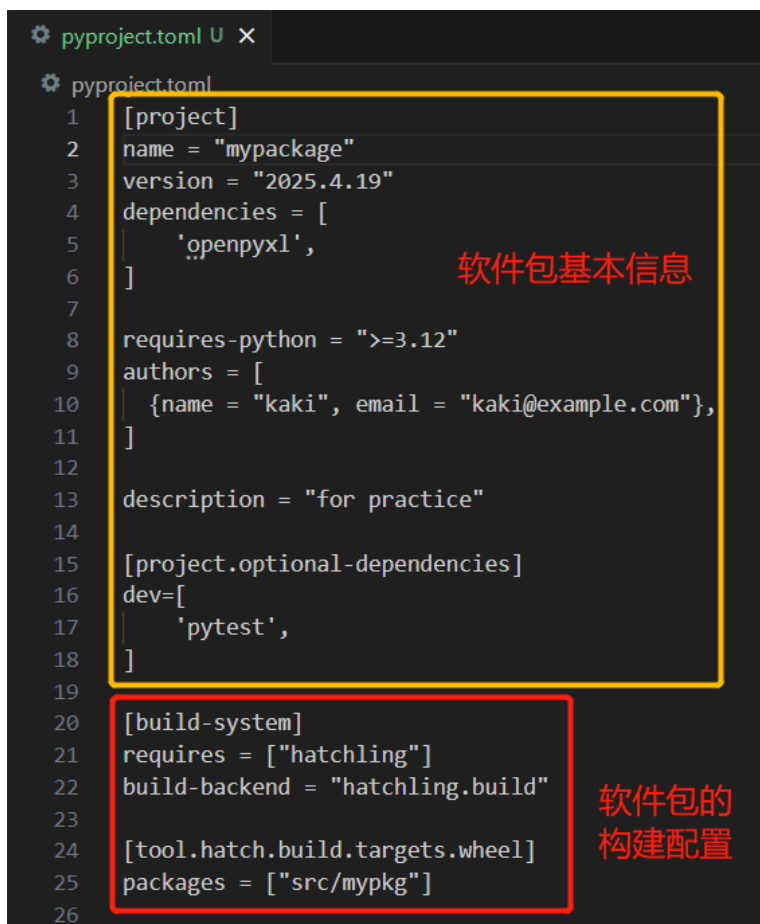
MINGW64:/c/Users/71970/rep
(base) 71970@132 MINGW64 ~
$ cd repo/week06

(base) 71970@132 MINGW64 ~/repo/week06 (main)
$ conda activate week06
(week06)
71970@132 MINGW64 ~/repo/week06 (main)
$ python scripts/myjob.py
Traceback (most recent call last):
  File "C:\Users\71970\repo\week06\scripts\myjob.py", line 1, in <module>
    import mylib # noqa: F401
    ^^^^^^^^^^^
ModuleNotFoundError: No module named 'mylib'
```

2. vscode 新建文件夹 src，其下再建文件夹 mypkg，将 mylib.py 模块移进去；创建 \_\_init\_\_.py 文件（注意是双下划线 dunder），准备好软件包的源代码

3. 项目目录下创建 pyproject.toml 配置文件

按照 python 官方文档填写基本的软件包信息和软件包的构建 (build) 配置



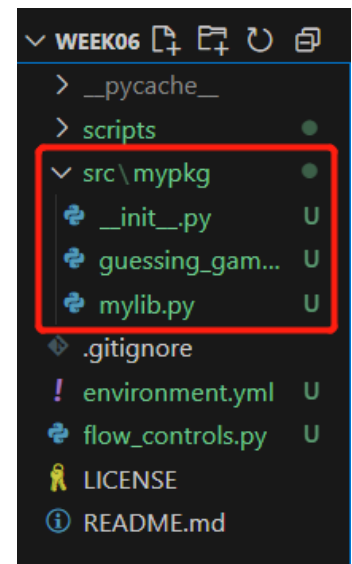
The screenshot shows the 'pyproject.toml' file with two sections highlighted by red boxes. The first box, labeled '软件包基本信息' (Basic Package Information), covers the [project] section. The second box, labeled '软件包的构建配置' (Package Build Configuration), covers the [build-system] and [tool.hatch.build.targets.wheel] sections.

```
pyproject.toml
[project]
name = "mypackage"
version = "2025.4.19"
dependencies = [
    'openpyxl',
]
requires-python = ">=3.12"
authors = [
    {name = "kaki", email = "kaki@example.com"},
]
description = "for practice"

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

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

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



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

```
(week06)
71970@1326 MINGW64 ~/repo/week06 (main)
$ pip install --editable .
Looking in indexes: https://mirrors.tuna.tsinghua.edu.cn/pypi/web/simple
Obtaining file:///C:/Users/71970/repo/week06
Installing build dependencies ... done
Checking if build backend supports build_editable ... done
Getting requirements to build editable ... done
Installing backend dependencies ... done
Preparing editable metadata (pyproject.toml) ... done
Collecting openpyxl (from mypackage==2025.4.19)
```

5. 将 week06 环境删除，重新安装前修改 `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    - pip:
9      - '--editable .' #- '-e .'
10
```

6. 重新安装环境，并试用本地可编辑软件包

```
(week06)
71970@1326 MINGW64 ~/repo/week06 (main)
$ python
Python 3.12.10 | packaged by conda-forge | (main, Apr 10 2025, 22:08:16) [MS
C v.1943 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import mypkg
>>> 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): 80
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): 50
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): 65
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): 60
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): 55
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): 52
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): 51
恭喜你 🎉，猜对了！
游戏结束，再见 🙌。
```