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)：1
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)：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)：63
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)：57
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)：53
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
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)：52
恭喜你🎉，猜对了！
游戏结束，再见👋。

```python
# 1. for 迭代循环
print("=== for 循环示例 ===")
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)

# 2. while 条件循环
print("\n=== while 循环示例 ===")
count = 0
while count < 3:
    print(f"计数: {count}")
    count += 1

# 3. break 打断循环
print("\n=== break 示例 ===")
for num in range(5):
    if num == 3:
        break
    print(num)

# 4. continue 跳过当前迭代
print("\n=== continue 示例 ===")
for num in range(5):
    if num == 2:
        continue
    print(num)

# 5. for...else 循环未被打断
print("\n=== for...else 示例 ===")
for num in range(3):
    print(num)
else:
    print("循环正常结束")

# 6. if 条件分支
print("\n=== if 示例 ===")
x = 10
if x > 5:
    print("x 大于 5")

# 7. if...elif...else 多重分支
print("\n=== if...elif...else 示例 ===")
score = 85
if score >= 90:
    print("优秀")
elif score >= 80:
    print("良好")
elif score >= 60:
    print("及格")
else:
    print("不及格")
```

```python
# 8. try...except...else...finally 异常处理
print("\n=== 异常处理示例 ===")
try:
    result = 10 / 2
except ZeroDivisionError:
    print("不能除以零")
else:
    print(f"结果是: {result}")
finally:
    print("执行完成")

# 9. raise 主动抛出异常
print("\n=== raise 示例 ===")
try:
    raise ValueError("这是一个自定义错误")
except ValueError as e:
    print(f"捕获到错误: {e}")
```

```
=== for 循环示例 ===
apple
banana
cherry

=== while 循环示例 ===
计数: 0
计数: 1
计数: 2

=== break 示例 ===
0
1
2

=== continue 示例 ===
0
1
3
4

=== for...else 示例 ===
0
1
2
循环正常结束

=== if 示例 ===
x 大于 5

=== if...elif...else 示例 ===
良好

=== 异常处理示例 ===
结果是: 5.0
执行完成

=== raise 示例 ===
捕获到错误: 这是一个自定义错误
```

```python
# 1. 无参数无返回值
def func1():
    print("func1 被调用")

# 2. 无参数有返回值
def func2():
    return "func2 的返回值"

# 3. 单个位置参数
def func3(param):
    print(f"func3 被调用，参数: {param}")

# 4. 单个命名参数
def func4(param="默认值"):
    print(f"func4 被调用，参数: {param}")

# 5. 多个位置和命名参数
def func5(pos1, pos2, named1="默认1", named2="默认2"):
    print(f"func5: pos1={pos1}, pos2={pos2}, named1={named1}, named2={named2}")

# 6. 仅限位置参数
def func6(pos1, pos2, /):
    print(f"func6: pos1={pos1}, pos2={pos2}")

# 7. 仅限命名参数
def func7(*, named1, named2):
    print(f"func7: named1={named1}, named2={named2}")

# 8. 可变位置参数
def func8(a, b, *args):
    print(f"func8: a={a}, b={b}, args={args}")

# 9. 可变命名参数
def func9(a, **kwargs):
    print(f"func9: a={a}, kwargs={kwargs}")

# 10. 参数解包
def func10(a, b, c=0):
    print(f"func10: a={a}, b={b}, c={c}")

# 11. 字典解包
def func11(a=1, b=2, c=3):
    print(f"func11: a={a}, b={b}, c={c}")

# 12. 带文档和类型注解的函数
def func12(name: str, age: int) -> str:
    """
    返回用户信息字符串

    参数:
        name: 用户名
        age: 用户年龄

    返回:
        格式化后的用户信息字符串
    """
    return f"{name} 今年 {age} 岁"
```

```python
from mylib import *

# 1. 无参数无返回值
func1()

# 2. 无参数有返回值
result = func2()
print(result)

# 3. 单个位置参数
func3("位置参数")  # 位置传参
func3(param="命名参数")  # 命名传参
# func3()  # 不传参会报错

# 4. 单个命名参数
func4("位置传参")  # 位置传参
func4(param="命名传参")  # 命名传参
func4()  # 使用默认值

# 5. 多个位置和命名参数
func5(1, 2)  # 只传位置参数
func5(1, 2, named1="a")  # 混合传参
func5(1, 2, named2="b", named1="a")  # 命名参数顺序可变

# 6. 仅限位置参数
func6(1, 2)  # 必须用位置传参
# func6(pos1=1, pos2=2)  # 这样会报错

# 7. 仅限命名参数
func7(named1=1, named2=2)  # 必须用命名传参
# func7(1, 2)  # 这样会报错

# 8. 可变位置参数
func8(1, 2, 3, 4, 5)  # 3,4,5被打包为元组

# 9. 可变命名参数
func9(1, x=10, y=20)  # x,y被打包为字典

# 10. 参数解包
args = (1, 2, 3)
func10(*args)  # 解包元组

# 11. 字典解包
kwargs = {'a': 10, 'b': 20, 'c': 30}
func11(**kwargs)  # 解包字典

# 12. 带文档和类型注解的函数
info = func12("张三", 25)
print(info)
# 查看文档
print(func12.doc)
```