Week06

```
(base) Administrator@DESKTOP-2HD7O7S MINGW64 ~/Desktop/首经贸/2024-2025第二学期
程与计算/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): 55
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): 33
猜的数字太大了，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): 22
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): 11
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): 3
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): 2
恭喜你🎉，猜对了！
游戏结束，再见👋。
```

写代码，先排除特殊情况

```python
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
```

再开始进行循环判断

```python
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
```

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

text = "hello"
for i in text:
    print(i)
```

```
for 迭代循环示例：
apple
banana
cherry
h
e
l
l
o
```

```python
# while 条件循环 (conditional loop)
print("\nwhile 条件循环示例:")
count = 0
while count < 5:
    print(count)
    count += 1
num =[1,2,3,4,5]
while num:
    print(num.pop())
```

```
while 条件循环示例：
0
1
2
3
4
5
4
3
2
1
```

```python
# break 打断跳出循环
print("\nbreak 打断跳出循环示例:")
numbers = [1, 2, 3, 4, 5]
for num in numbers:
    if num == 3:
        break
    print(num)
```

```
break 打断跳出循环示例：
1
2
```

```python
# continue 跳至下一轮循环
print("\ncontinue 跳至下一轮循环示例:")
for num in numbers:
    if num == 3:
        continue
    print(num)
```

```
continue 跳至下一轮循环示例：
1
2
4
5
```
跳过 print（3）这一步直接进入下次循环

```python
# for...else 循环未被打断的处理
print("\nfor...else 循环未被打断的处理示例:")
for num in numbers:
    print(num)
else:
    print("循环正常结束，未被 break 打断。")
```

```
for...else 循环未被打断的处理示例：
1
2
3
4
5
```

```python
# 模拟被 break 打断的情况
for num in numbers:
    if num == 3:
        break
    print(num)
else:
    print("循环正常结束，未被 break 打断。")
```

```python
# if 条件分支
print("\nif 条件分支示例:")
x = 10
if x > 5:
    print("x 大于 5")
```

```
if 条件分支示例：
x 大于 5
```

```python
# if...elif[...elif] 多重条件分支
print("\nif...elif 多重条件分支示例:")
score = 75
if score >= 90:
    print("成绩为 A")
elif score >= 80:
    print("成绩为 B")
elif score >= 70:
    print("成绩为 C")
else:
    print("成绩为 D")
```

```
if...elif 多重条件分支示例：
成绩为 C
```

```python
# if...else 未满足条件的处理
print("\nif...else 未满足条件的处理示例:")
age = 15
if age >= 18:
    print("你已成年，可以投票。")
else:
    print("你还未成年，不能投票。")
```

```
if...else 未满足条件的处理示例：
你还未成年，不能投票。
```

```python
# try...except[...except...else...finally] 捕捉异常的处理
print("\ntry...except 捕捉异常的处理示例:")
try:
    result = 10 / 0
except ZeroDivisionError:
    print("错误:除数不能为零。")
else:
    print("没有发生异常，结果是:", result)
finally:
    print("无论是否发生异常，都会执行此语句。")
# 另一个 try...except 示例，处理不同类型的异常
try:
    num_list = [1, 2, 3]
    print(num_list[3])
except IndexError:
    print("错误:索引超出列表范围。")
except TypeError:
    print("错误:类型错误。")
```

```
try...except 捕捉异常的处理示例：
错误：除数不能为零。
无论是否发生异常，都会执行此语句。
错误：索引超出列表范围。
```

```python
# raise 主动抛出异常
print("\nraise 主动抛出异常示例:")
def divide(a, b):
    if b == 0:
        raise ZeroDivisionError("除数不能为零。")
    return a / b
try:
    result = divide(10, 0)
    print(result)
```

```
    except ZeroDivisionError as e:
        print(e)
```

```
raise 主动抛出异常示例：
除数不能为零。
```

```
    def func1():
        x = 50
        y =x**0.5-7
        print(y)

    def func2():
        x = 70
        y =x**0.5-7
        print(y)
        return y
        return y+1
    def func3(x):
        y =x**0.5-7
        print(y)
    def func4(x=50):
        y =x**0.5-7
        print(y)

    # 定义一个包含命名形参的函数
    def func5(name,  message="Hello"):  ##注意位置参数必须排在命名参数之前
        return f"{message}, {name} , !"
    # 定义一个包含命名形参的函数
    def func6(name, /, b, message="Hello"):  ##注意位置参数必须排在命名参数之前
        return f"{message}, {name} , {b}!"
    def func7(name, /, b,*,  message="Hello"):  ##注意位置参数必须排在命名参数之前
        return f"{message}, {name} , {b}!"
    def func8(*numbers):
        total = 0
        for num in numbers:
            total = total + num
        return total
    def func9(**abs):
        for key, value in abs.items():
            print(f"{key}: {value}")
    def func10(arg1, arg2, named_arg='default'):
        """
        此函数接受两个位置形参和一个命名形参。
        :param arg1: 第一个位置形参
        :param arg2: 第二个位置形参
```

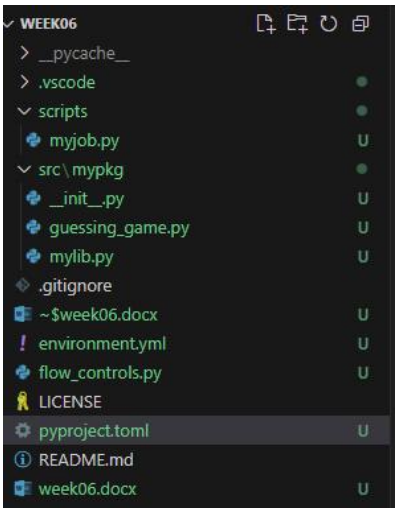
```
    :param named_arg: 命名形参，默认值为 'default'
    :return: 包含传入参数的元组
    """
    print(f"arg1: {arg1}, arg2: {arg2}, named_arg: {named_arg}")
    return arg1, arg2, named_arg
def func12(num1: int, num2: int) -> int:
    """
    此函数用于将两个整数相加，并返回它们的和。
    参数:
    num1 (int): 第一个要相加的整数。
    num2 (int): 第二个要相加的整数。
    返回:
    int: 两个整数相加的结果。
    """
    return num1 + num2
```

```
import mylib  # noqa: F401

y = mylib.func1()
print(y)
y = mylib.func2()
print(y)
y = mylib.func3(45)
print(y)
y = mylib.func3(x=47)  #命名形参
print(y)
y = mylib.func4(48)
print(y)
y = mylib.func4( )
print(y)
result1 = mylib.func5(name="Alice", message="Hi")
print("使用命名形参调用结果 1:", result1)
print(mylib.func6("Bob", "how are you", message="Hi"))
print(mylib.func7("Bob", "how are you", message="Hi"))
print(mylib.func8(1,2,3))
print(mylib.func8())
print(mylib.func9(city='New York', country='USA', zipcode='10001'))
positional_args_tuple = (10, 20)
result1 = mylib.func10(*positional_args_tuple, named_arg='custom')
print("使用元组解包调用结果:", result1)
positional_args_list = [30, 40]
result2 = mylib.func10(*positional_args_list)
print("使用列表解包调用结果:", result2)
print(mylib.func12(3,5))
```

运算结果

```
$ python myjob.py
0.0710678118654755
None
1.3666002653407556
1.3666002653407556
-0.2917960675006306
None
-0.1443453995989561
None
-0.07179676972449123
None
0.0710678118654755
None
使用命名形参调用结果1: Hi, Alice , !
Hi, Bob , how are you!
Hi, Bob , how are you!
6
0
city: New York
country: USA
zipcode: 10001
None
arg1: 10, arg2: 20, named_arg: custom
使用元组解包调用结果：(10, 20, 'custom')
arg1: 30, arg2: 40, named_arg: default
使用列表解包调用结果：(30, 40, 'default')
8
```




```
$ pip install -e .
Looking in indexes: https://mirrors.tuna.tsinghua.edu.cn/pypi/web/simple
Obtaining file:///C:/Users/lululu~yaoyao/Desktop/%E9%A6%96%E7%BB%8F%E8%B4%B8/2024-2025%E7%AC%AC%E4%
BA%8C%E5%AD%A6%E6%9C%9F%E4%BD%9C%E4%B8%9A/%E9%87%91%E8%9E%8D%E7%BC%96%E7%A8%8B%E4%B8%8E%E8%AE%A1%E7
%AE%97/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
Building wheels for collected packages: mypackage
  Building editable for mypackage (pyproject.toml) ... done
  Created wheel for mypackage: filename=mypackage-2025.4.14-py2.py3-none-any.whl size=7216 sha256=b
3a7f202d8097a3f51ee0b13de1511bbf135a9c935273d2e48da1b831a88ecf9
  Stored in directory: C:\Users\lululu~yaoyao\AppData\Local\Temp\pip-ephem-wheel-cache-lw8e042p\whe
els\25\95\e9\1f226019a202263b5f20111ea90fff6963d31a24f4a2f5b885
Successfully built mypackage
Installing collected packages: mypackage
Successfully installed mypackage-2025.4.14
(week06)
```