

金融编程作业 week06

1. Fork 第 06 周打卡仓库至我的名下，然后将名下的这个仓库 Clone 到本地计算机；然后用 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

```
(base) zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo
$ cd week06

(base) zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week06 (main)
$ ls -l
total 24
-rw-r--r-- 1 zhu77 197121 18805  4月  20 21:01 LICENSE
-rw-r--r-- 1 zhu77 197121  2239  4月  20 21:01 README.md

(base) zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week06 (main)
$ cp ../week05/environment.yml ./

(base) zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week06 (main)
$ ll
total 25
-rw-r--r-- 1 zhu77 197121    220  4月  20 21:03 environment.yml
-rw-r--r-- 1 zhu77 197121 18805  4月  20 21:01 LICENSE
-rw-r--r-- 1 zhu77 197121  2239  4月  20 21:01 README.md
```

激活

```
(base) zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week06 (main)
$ conda activate week06
(week06)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week06 (main)
$ conda list
# packages in environment at D:\zhu77\Anaconda\envs\week06:
#
# Name                        Version           Build    Channel
bzip2                         1.0.8             h2466b09_7    conda-forge
ca-certificates               2025.1.31         h56e8100_0    conda-forge
libexpat                      2.7.0             he0c23c2_0    conda-forge
libffi                        3.4.6             h537db12_1    conda-forge
liblzma                       5.8.1             h2466b09_0    conda-forge
libsqlite                     3.49.1            h67fdade_2    conda-forge
libzlib                       1.3.1             h2466b09_2    conda-forge
openssl                       3.5.0             ha4e3fda_0    conda-forge
pip                           25.0.1            pyh8b19718_0    conda-forge
python                        3.12.10           h3f84c4b_0_cpython    conda-forge
setuptools                    78.1.0            pyhff2d567_0    conda-forge
tk                             8.6.13            h5226925_1    conda-forge
tzdata                        2025b             h78e105d_0    conda-forge
ucrt                          10.0.22621.0      h57928b3_1    conda-forge
vc                             14.3              h2b53caa_26    conda-forge
vc14_runtime                  14.42.34438       hfd919c2_26    conda-forge
wat-inspector                 0.4.3             pyhff2d567_0    conda-forge
wheel                         0.45.1            pyhd8ed1ab_1    conda-forge
(week06)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week06 (main)
```

2. 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：

```
import random
```

```

def guessing_game():
    # 生成 1 到 100 之间的随机整数
    secret_number = random.randint(1, 100)
    n = 0

    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。")

    while True:
        n += 1
        # 获取玩家输入
        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() # 去除多余空白字符

        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

        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

        raise NotImplementedError

    print("游戏结束，再见。")

```

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

```
(Pdb) n
> c:\users\zhu77\repo\week06\guessing_game.py(50)<module>()
-> guessing_game()
(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。
(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): |
```

```
*** NameError: name 'n' is not defined
(Pdb) n
> c:\users\zhu77\repo\week06\guessing_game.py(50)<module>()
-> guessing_game()
(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。
(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
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): 46
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): 99
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): 80
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): 88
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): 85
猜的数字太小了，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): 86
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): 87
恭喜你 🎉，猜对了！
游戏结束，再见 🙋。
--Return--
> c:\users\zhu77\repo\week06\guessing_game.py(50)<module>()->None
-> guessing_game()
(Pdb) |
```

```
> c:\users\zhu77\repo\week06\guessing_game.py(7)guessing_game()
-> n = 0
(Pdb) p secret_number
37
(Pdb) n
> c:\users\zhu77\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\zhu77\repo\week06\guessing_game.py(11)guessing_game()
-> while True:
(Pdb)
> c:\users\zhu77\repo\week06\guessing_game.py(12)guessing_game()
-> n += 1
(Pdb)
> c:\users\zhu77\repo\week06\guessing_game.py(14)guessing_game()
-> guess = input(
(Pdb)
> c:\users\zhu77\repo\week06\guessing_game.py(15)guessing_game()
-> 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): "
```

3. 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): 在这个例子中，`for` 循环会依次取出列表 `fruits` 中的每个元素，赋值给变量 `fruit`，然后执行循环体中的 `print` 语句，输出对每种水果的喜爱。

```
→ week06
! environment.yml U  guessing_game.py U  f
flow_controls.py > ...
1  fruits = ["apple", "banana", "cherry"]
2  for fruit in fruits:
3      print(f"I like {fruit}")
4
5  message = "hello"
6  for char in message:
7      print(char)
8

(week06)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week06 (main)
$ python flow_controls.py
I like apple
I like banana
I like cherry
(week06)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week06 (main)
$ python flow_controls.py
I like apple
I like banana
I like cherry
h
e
l
l
o
(week06)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week06 (main)
$ |
```

while 条件循环 (conditional loop): 在这个例子中，定义了一个变量 `count` 并初始化为 0。`while` 循环的条件是 `count < 5`，只要这个条件为真，循环体就会执行。每次循环中，打印当前的计数值，然后将 `count` 加 1。当 `count` 等于 5 时，循环条件变为假，循环结束。

```
8
9  count = 0
10 while count < 5:
11     print(f"当前计数: {count}")
12     count += 1
13

0
当前计数: 0
当前计数: 1
当前计数: 2
当前计数: 3
当前计数: 4
(week06)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week06 (main)
$ |
```

break 打断跳出循环

continue 跳至下一轮循环: 在这个例子中，`while` 循环的条件是 `fruits`，因为非空列表在布尔上下文中为 `True`，所以只要 `fruits` 列表不为空，循环就会继续。在循环体中，使用 `pop()` 方法从列表末尾取出一个元素，并打印出取出的元素以及剩余的列表内容。当列表为空时，循环条件为假，循环结束。

```
13
14 count = 0
15 while count < 10:
16     count += 1
17     if count == 5:
18         continue # 跳过本次循环剩余部分，继续下一次循环
19     if count == 8:
20         break # 终止整个循环
21     print(f"当前计数: {count}")
22

0
当前计数: 0
当前计数: 1
当前计数: 2
当前计数: 3
当前计数: 4
当前计数: 1
当前计数: 2
当前计数: 3
当前计数: 4
当前计数: 6
当前计数: 7
(week06)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week06 (main)
$ |
```

for...else 循环未被打断的处理: 在这个例子中，`for` 循环遍历 `numbers` 列表，尝试查找 `target` 变量指定的数字。由于 `target` 的值 60 不在列表中，循环会正常结束，不会遇到 `break` 语句。因此，`else` 子句会被执行，输出 “没有找到目标数字 60”。


```

numbers = [10, 20, 30, 40, 50]
target = 60
for num in numbers:
    if num == target:
        print(f"找到了目标数字 {target}")
        break
    else:
        print(f"没有找到目标数字 {target}")

```

```

当前计数: 7
没有找到目标数字 60
(week06)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week06 (main)
$ |

```

if 条件分支: 在这个例子中, 定义了一个变量 `age` 并赋值为 18。if 语句的条件是 `age >= 18`, 如果这个条件为真, 就会执行缩进在 if 语句块内的 `print` 语句, 输出 “你已经成年了, 可以参加投票。”。

```

age = 18
if age >= 18:
    print("你已经成年了, 可以参加投票。")

```

```

你已经成年了, 可以参加投票。
你的成绩等级是 B。
(week06)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/re
$ |

```

if...elif...elif 多重条件分支: 在这个例子中, if 语句首先判断 `score` 是否大于等于 90, 如果是则输出相应的等级信息。如果 if 条件为假, 会依次检查每个 elif 语句的条件, 直到找到一个为真的条件并执行对应的语句块。如果所有 if 和 elif 条件都为假, 就会执行 else 语句块内的代码, 输出 “你的成绩不及格。” (在这个例子中, 会输出 “你的成绩等级是 B。”)。

```

35
36 score = 85
37 if score >= 90:
38     print("你的成绩等级是 A。")
39 elif score >= 80:
40     print("你的成绩等级是 B。")
41 elif score >= 70:
42     print("你的成绩等级是 C。")
43 elif score >= 60:
44     print("你的成绩等级是 D。")
45 else:
46     print("你的成绩不及格。")
47

```

```

当前计数: 7
没有找到目标数字 60
你已经成年了, 可以参加投票。
你的成绩等级是 B。
(week06)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week06 (m
$ |

```

if...else 未满足条件的处理: 在这个例子中, if 语句的条件是 `num > 0`, 由于 `num` 的值为 -3, 不满足该条件, 所以会执行 else 语句块中的代码, 输出 -3 是负数或者零。

```

48 num = -3
49 if num > 0:
50     print(f"{num} 是正数")
51 else:
52     print(f"{num} 是负数或者零")

```

```

-3 是负数或者零
(week06)
zhu77@LAPTOP-72KG9NFN MINGW64
$ |

```

try...except[...except...else...finally] 捕捉异常的处理:

Try...except...: 在这个例子中, try 块中尝试执行 `10 / 0`, 这会引发 `ZeroDivisionError` 异常。当异常发生时, 程序会立即跳转到 except 块中执行, 输出 “除数不能为零!”。

```

try:
    num = 10 / 0
    print(num)
except ZeroDivisionError:
    print("除数不能为零!")

```

```

-3 是负数或者零
除数不能为零!
请输入一个整数: 56
计算结果是: 0.17857142857142858
无论是否有异常, 这个都会被执行。
(week06)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week06
$ |

```

try...except...else...finally: 在这个综合例子中, try 块尝试获取用户输入并进行计算; except 块处理可能的异常; 如果没有异常发生, else 块会输出计算结果; 而 finally 块中的代码无论如何都会被执行, 输出提示信息。

```
try:
    num = int(input("请输入一个整数: "))
    result = 10 / num
except ZeroDivisionError:
    print("除数不能为零!")
except ValueError:
    print("输入的不是一个有效的整数!")
else:
    print(f"计算结果是: {result}")
finally:
    print("无论是否有异常, 这个都会被执行。")
```

请输入一个整数: 56
计算结果是: 0.17857142857142858
无论是否有异常, 这个都会被执行。
(week06)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week06 (main)
\$ |

raise 主动抛出异常

(1) 抛出内置异常类型: 在这个例子中, `check_age` 函数用于检查输入的年龄。如果年龄小于 0, 就使用 `raise` 语句抛出一个 `ValueError` 异常, 并附带相应的错误信息。在 `try` 块中调用 `check_age` 函数, 当捕获到 `ValueError` 异常时, 会执行 `except` 块中的代码, 打印出异常的错误信息。

```
72
73
74 def check_age(age):
75     if age < 0:
76         raise ValueError("年龄不能为负数")
77     return f"你的年龄是 {age} 岁"
78
79
80 try:
81     result = check_age(-5)
82     print(result)
83 except ValueError as e:
84     print(e)
```

年龄不能为负数
输入的参数必须是数字类型 (int 或 float)
(week06)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week06
\$ |

(2) 抛出 `TypeError` 异常: `add_numbers` 函数用于计算两个数的和。在函数中, 通过 `isinstance` 检查输入的参数类型, 如果参数不是 `int` 或 `float` 类型, 就抛出 `TypeError` 异常。在 `try` 块中调用函数并传入不合适的参数类型, `except` 块会捕获并处理该异常。

```
def add_numbers(a, b):
    if not isinstance(a, (int, float)) or not isinstance(b, (int, float)):
        raise TypeError("输入的参数必须是数字类型(int 或 float)")
    return a + b

try:
    result = add_numbers(5, "3")
    print(result)
except TypeError as e:
    print(e)
```

无论是否有异常, 这个都会被执行。
年龄不能为负数
输入的参数必须是数字类型 (int 或 float)
(week06)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week06 (mai
\$ |

4. 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:

定义函数 `func1`, 没有形参, 没有返回值

```
environment.yml U  guessing_game.py U  flow_controls.py U  myjob.py
1 import mylib # noqa: F401
2
3 breakpoint()
4
```

zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week06 (main)
\$ python myjob.py
--Return--
> c:\users\zhu77\repo\week06\myjob.py(3)<module>()->None
-> breakpoint()
(Pdb) l
1 import mylib # noqa: F401
2
3 -> breakpoint()
[EOF]
(Pdb) p mylib
<module 'mylib' from 'C:\Users\zhu77\repo\week06\mylib.py'>
(Pdb) import wat
(Pdb) wat / mylib

value: <module 'mylib' from 'C:\Users\zhu77\repo\week06\mylib.py'>
type: module

Public attributes:
def func1()
(Pdb) |

```
environment.yml U  guessing_game.py U  flow_controls.py U  n
myjob.py
1 import mylib # noqa: F401
2
3 mylib.func1()
4

zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week06 (main)
$ python myjob.py
0.0710678118654755
(week06)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week06 (main)
$
```

没有返回值的时候，返回 None

```
environment.yml U  guessing_game.py U  flow_controls.py U  n
myjob.py > ...
1 import mylib # noqa: F401
2
3 y = mylib.func1()
4 breakpoint()
5 print(y)
6

zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week06 (main)
$ python myjob.py
0.0710678118654755
> c:\users\zhu77\repo\week06\myjob.py(5)<module>()
-> print(y)
(Pdb) l
1 import mylib # noqa: F401
2
3 y = mylib.func1()
4 breakpoint()
5 -> print(y)
[EOF]
(Pdb) p y
None
(Pdb) p type(y)
<class 'NoneType'>
(Pdb)
```

```
environment.yml U  guessing_game.py U  flow_controls.py U  n
myjob.py > ...
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

(Pdb) >>> quit
(week06)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week06 (main)
$ python myjob.py
0.0710678118654755
None
func1() takes 0 positional arguments but 1 was given
(week06)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week06 (main)
$
```

定义函数 func2，没有形参，有返回值

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)

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)

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

定义函数 func6，在形参列表中使用 / 来限定只接受位置实参的形参

定义函数 func7，在形参列表中使用 * 来限定只接受命名实参的形参

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)

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)

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

定义函数 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 (如字典) 自动解包，按命名实参传入

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