第六周建立自己的软件包

显示当前 conda 所有的类型：conda env list

需要删除某一环境： conda env remove -n prj1

函数 input 返回的内容都是字符串

guess=guess.strip()用于去除 guess 字符串中的空格

使用 pdb 调试器检测猜数字游戏的运行具体内容：

```
lqy0929@LAPTOP-CDK7DEUF MINGW64 /d/repo/week06 (main)
$ python -m pdb guessing_game.py
> d:\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，你可以开始猜啦。")
 10
 11         while True:
(Pdb) n
> d:\repo\week06\guessing_game.py(4)<module>()
-> def guessing_game():
(Pdb)
> d:\repo\week06\guessing_game.py(49)<module>()
-> if __name__ == "__main__":
(Pdb)
> d:\repo\week06\guessing_game.py(50)<module>()
-> guessing_game()
(Pdb) s
--Call--
> d:\repo\week06\guessing_game.py(4)guessing_game()
-> def guessing_game():
```

按 s（step）进入函数的内部

for 迭代循环 (iteration loop)

while 条件循环 (conditional loop)

break 打断跳出循环：会直接跳出离 break 最近的一个循环

continue 跳至下一轮循环，会返回上一轮的 while 循环语句

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

要考虑用户输入的各种可能性

越少见的情况越要先写，最后使用 raise 主动抛出异常，万一出现没有考虑到的情况直接报错‘ raise NotImplementedError’

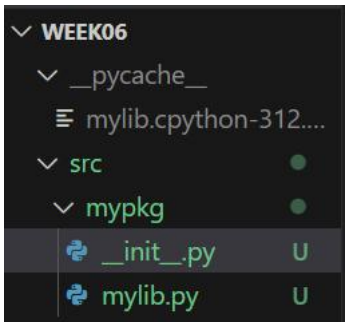
python 数数是从 0 开始的
Break 和 continue 都可以用于循环内部

```
(Pdb) import wat
(Pdb) wat /mylib

value: <module 'mylib' from 'D:\\repo\\week06\\mylib.py'>
type: module

Public attributes:
  def func1()
```

在 mylib（模块）里定义函数；在 myjob（脚本）中使用 import mylib 调用 mylib 中的函数
可以在调试器中使用 wat ./func 查看对应函数的注解
模块转化为软件包
当把脚本放在别的文件夹里的时候，再 import 就会显示失败，因为没有变成软件包



当文件夹中有__init__.py，python 才能识别这个文件夹是软件包

Writing your `pyproject.toml`

`pyproject.toml` is a configuration file used by packaging tools, as well as other tools such as linters, type checkers, etc. There are three possible TOML tables in this file.

- The `[build-system]` table is **strongly recommended**. It allows you to declare which build backend you use and which other dependencies are needed to build your project.
- The `[project]` table is the format that most build backends use to specify your project's basic metadata, such as the dependencies, your name, etc.
- The `[tool]` table has tool-specific subtables, e.g., `[tool.hatch]`, `[tool.black]`, `[tool.mypy]`. We only touch upon this table here because its contents are defined by each tool. Consult the particular tool's documentation to know what it can contain.

[build-system]为构建信息，和[project]可前可后

大版本号变动，改动更新更大，有 breaking change 升级要小心；小版本号更新只是一些小 bug 的修复，有更新就尽快更新；
软件包的 pyproject.toml 至少有 name,version 的软件包信息