cp 1week04/environment.yml

conda init
conda activate week05
执行 python use_of _str.py
Pdb 中 import wat wat / a
Id(x) 返回对象在虚拟内存中的地址，不是同一个对象地址不一样，如果 id（a）==id（b），那么 a is b
type()，类型
isinstance(变量,类型)，判断对象是否属于某个类型 (a,(str,float))是否属于这些类型里
dir() -- 返回对象所支持的属性 (attributes) 的名称列表
str()，转化为字符串
assert 语句查验某个表达式 (expression) 为真，否则报错 (AssertionError) 退出
例：assert isinstance(a,str)
try 语句拦截报错，避免退出

```
try:
    assert isinstance(a,str)
except AssertionError:
    print("type error")
```

breakpoint() 函数暂停程序运行，进入 pdb 调试 (debug) 模式
import wat
wat / s
python -m pdb use_of_str.py

s = f"name:{x}" x 是变量名 输出：name：x
s='a\tb' print('TAB',S) 输出结果 TAB a b 三个空格 \n 换行
{x+y} 可以运算
{p1:.2f} 保留两位小数
{:*^25,.2f}

1. *：填充字符
填充字符用于在值的长度不足指定宽度时，在值的两侧进行填充。这里指定的填充字符是 *，意味着当值的长度小于指定宽度时，会用 * 来补齐。
2. ^：对齐方式
对齐方式指定了值在指定宽度内的对齐方式，有以下几种选择：
<：左对齐（默认值） >：右对齐 ^：居中对齐
这里使用 ^ 表示将值居中对齐，即值会被放置在指定宽度的中间位置，两侧用填充字符补齐。
3. 25：宽度
宽度指定了格式化后的字符串的总宽度。如果值的长度小于这个宽度，会根据填充字符和对齐方式进行填充；如果值的长度大于这个宽度，则会原样输出。这里指定的宽度是 25，意味着格式化后的字符串总长度至少为 25 个字符。
4. ,：千位分隔符
逗号 , 用于在数字中插入千位分隔符，以提高数字的可读性。例如，将 1000000 格式化为 1,000,000。

5. .2：精度
精度指定了浮点数小数点后的位数。这里指定的精度是 .2，意味着会将浮点数保留两位小数。
6. f：类型
类型指定了要格式化的值的类型，常见的类型有：
s：字符串 d：十进制整数 f：浮点数
这里使用 f 表示要格式化的值是浮点数。

“““ xxx
Yyy”””三个引号换行符都会保留

```
s = "-"
s = s * 10
print(s * 20)
```

s=’hello’
s[3] 索引 从 0 开始

s.upper() 大写，括号无内容

```
t = "name:{}，age {}"
t1 = t.format("Wendy", "31")   插入两个值
print(t1)
```

字符串连接 + *可以

```
a = "red "
b = "velvet"
c = a + b
assert c == "red velvet"
print(c)
```

```
try:
    print(a - b)
except TypeError as e:
    print(e)
```

assert ‘ ‘ 会报错 iter() 是否可迭代
a.capitalize() 大写首字母 a.count(“值”)
a.endwith(‘k’)以 k 为结尾 "0base1".isalnum()是否以数字和字母组成（空格下划线等显示 false）
a.index(‘b’) 索引位置
变量名不能以数字开头

列表中间用*连接

```
q = ["it", "pops", "like", "candy"]
print("*".join(q))
```

删除两边空格/右边或左边

```
print(" song ha ".strip())
print(" song ha ".rstrip())
```

将字符串以空格输出成列表元素

```
a = "love is cosmic"
print(a.split(" "))
```

Partition 将字符串按第一个出现的空格分开

```
assert a.partition(" ") == ("love", " ", "is cosmic")
```

字节串

```
s = b"hello"
print(s)
print(s[0])
```

显示 h 的 ascii　　bin()转换为二进制

- **encode()**：用于将字符串（str）编码为指定编码格式的字节对象（bytes）。基本语法是 string.encode(encoding, errors='strict')，string 是要编码的字符串；encoding 是编码格式，像 UTF-8、GBK 等；errors 处理编码错误，默认 'strict'，即出错抛异常，也可设为 'ignore'（忽略错误）、'replace'（用 ? 替换）等。比如 s = "你好"; s.encode('utf-8')，把 s 按 UTF-8 编码成字节对象。
- **decode()**：和 encode() 相反，将字节对象（bytes）解码为字符串（str）。基本语法 bytes.decode(encoding, errors='strict')，bytes 是待解码字节对象，encoding 需和编码时一致，errors 处理解码错误方式同 encode()。如 b = b'\xe4\xbd\xa0\xe5\xa5\xbd'; b.decode('utf-8')，将 UTF-8 编码的字节对象 b 解码成字符串。

s.encode("utf-8")

整数：不能 len 索引　x.to_bytes()

生成随机数

```
x = random.random()
print(x)
```

缺失值

```
nan = float("nan")
```

科学计数法：3.14e2
无穷大的数

```
pinf = float("inf")
```
同理-inf

布尔值：是整数，0 or 1

列表连接

```
a = ["Sherry", "Shiho"]
b = [26, 4869]
print(a + b)
print(b + a)
```

更改列表元素

```
a[0] = 10
```

```
a = [1, 2, 3]
b = [i**2 for i in a]
print(b)
```

添加 a.append()

```
  def append(object, /) # Append object to the end of the list
.
  def clear() # Remove all items from list.
  def copy() # Return a shallow copy of the list.
  def count(value, /) # Return number of occurrences of value.
  def extend(iterable, /) # Extend list by appending elements
from the iterable.
  def index(value, start=0, stop=9223372036854775807, /) # Ret
urn first index of value.…
  def insert(index, object, /) # Insert object before index.
  def pop(index=-1, /) # Remove and return item at index (defa
ult last).…
  def remove(value, /) # Remove first occurrence of value.…
  def reverse() # Reverse *IN PLACE*.
  def sort(*, key=None, reverse=False) # Sort the list in asce
nding order and return None.…
```

字典
提取键和值　d.values()　d.items()

```
for a in d:
    print(a)

for a in d:
    print(d[a])
```

```
for k, v in d.items():
    print(k, v)
```

循环的另一种表达方式

```
l = [a for a in d.items()]
print(l)
```

```
Public attributes:
  def clear(…) # D.clear() -> None.  Remove all items from D.
  def copy(…) # D.copy() -> a shallow copy of D
  def fromkeys(iterable, value=None, /) # Create a new diction
ary with keys from iterable and values set to value.
  def get(key, default=None, /) # Return the value for key if
key is in the dictionary, else default.
  def items(…) # D.items() -> a set-like object providing a vi
ew on D's items
  def keys(…) # D.keys() -> a set-like object providing a view
 on D's keys
  def pop(…) # D.pop(k[,d]) -> v, remove specified key and ret
urn the corresponding value.…
  def popitem() # Remove and return a (key, value) pair as a 2
-tuple.…
  def setdefault(key, default=None, /) # Insert key with a val
ue of default if key is not in the dictionary.…
  def update(…) # D.update([E, ]**F) -> None.  Update D from m
apping/iterable E and F.…
  def values(…) # D.values() -> an object providing a view on
D's values
```

d.get(键,默认值) 如果不存在，返回默认值

d.pop(键) 移除

d.setdefault('键',0) 若存在，返回其值。不存在，添加一个键值对

d["键"]=值

元组不可变

```
Public attributes:
  def count(value, /) # Return number of occurrences of value.
  def index(value, start=0, stop=9223372036854775807, /) # Ret
urn first index of value.…
```

```
d = {}
d[t] = 5
print(d)
# 元组可以作为键
```

创建元组

```
a = 4, 8, 6, 9
print(type(a))
```

列表和字典可变

集合（唯一化）

```
a = [1, 1, 2, 2]
print(a)
b = set(a)
print(b)
```

并集和交集和差

```
a = [1, 1, 2, 2]
print(a)
b = set(a)
print(b)
print(2 in a)

c = {2, 3}
print(b | c)
print(b & c)
print(b ^ c)
```

Path

```python
from pathlib import Path
from pprint import pprint

p = Path(".")  # 当前目录
print(p)
print(p.exists())
print(p.absolute())  # 绝对路径
print(list(p.iterdir()))  # 列举文件夹
pprint(list(p.iterdir()))

# 新建文件夹
p = Path("./data1")
print(p.exists())
p.mkdir(exist_ok=True)  # 创建文件夹 若目录已存在不会抛出异
print(p.exists())
print(p.is_dir())

p = Path(".")
p2 = p / "readme.md"
print(p2)
p3 = p2.absolute()
print(p3)
```

Date 类型

```python
from datetime import date, datetime, timedelta  # noqa:

print(date.today())
t1 = date.today()
t2 = date(2014, 8, 10)
td = t1 - t2
print(type(td))
print(td.days)

s1 = "2012-04-08"
s2 = "2014-08-10"
d1 = datetime.strptime(s1, "%Y-%m-%d")
d2 = datetime.strptime(s2, "%Y-%m-%d")
print(d1)
print(d2)
```