day17

1.今日内容

- 迭代器（3*）
- 生成器（4*）
- 装饰器（5*）
- 项目结构
- logging模块

2.内容回顾 & 作业

2.1 内容回顾

2.1.1 函数（内置/自定义）

- 基本函数结构

```
def func(a1,a2):
    pass
```

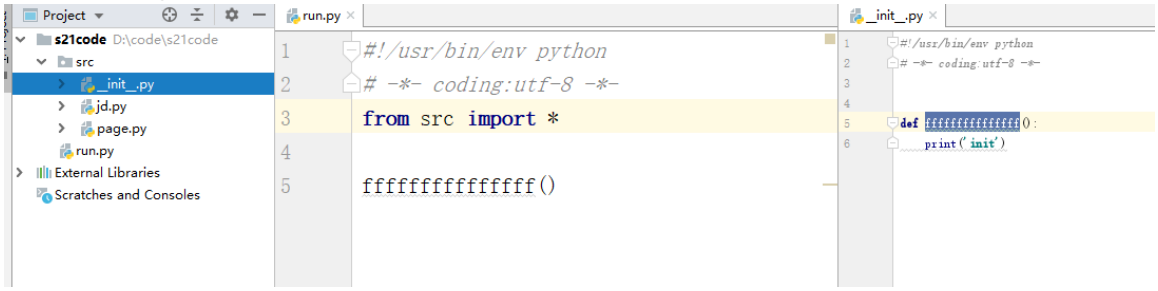
  - 参数
  - 返回值
  - 执行函数
- 函数小高级
  - 函数做变量
  - 函数做参数
- 函数中高级
  - 函数做返回值
  - 函数的嵌套
- 装饰器 & 闭包
- 递归
- 匿名函数
- 内置函数

2.1.2 模块（内置/第三方/自定义）

- 定义模块
  - 内置：time/json/datetime/os/sys ... 【re模块】
  - 第三方：
    - 安装：
      - pip包管理工具： pip install xlrd

        - 源码安装：
            - 下载源码包：压缩文件。
            - 解压文件
            - 打开cmd窗口，并进入此目录：cd C:\Python36\Lib\site-packages
            - 执行：python36 setup.py build
            - 执行：python36 setup.py install
        - 安装路径：C:\Python36\Lib\site-packages
        - 你了解的第三方模块：
            - xlrd
            - requests
    - 自定义
        - py文件
        - 文件夹 `__init__.py`
- 调用模块
    - import
        - import 模块1 模块1.函数()
        - import 模块1.模块2.模块3 模块1.模块2.模块3.函数()
    - from xx import xxx
        - from 模块.模块 import 函数 函数()
        - from 模块.模块 import 函数 as f f()
        - from 模块.模块 import * 函数1() 函数2()
        - from 模块 import 模块 模块.函数()
        - from 模块 import 模块 as m m.函数()
    - 特殊情况：
        - import 文件夹 加载`__init__.py`

        - from 文件 import *



2.1.3 其他

- 两个值数据交换
- 推导式
  - 列表（*）
  - 字典
  - 集合

总结

- 基础知识
- 逻辑能力
- 面试题
- 下阶段目标：锻炼编码能力。

2.2作业

1. 思维导图
2. Python学习笔记
3. 本周作业
4. 复习：从前到后
   - 笔记
   - 作业题

3.今日内容

> 类和对象

3.1 迭代器

自己不会写迭代器，只用。

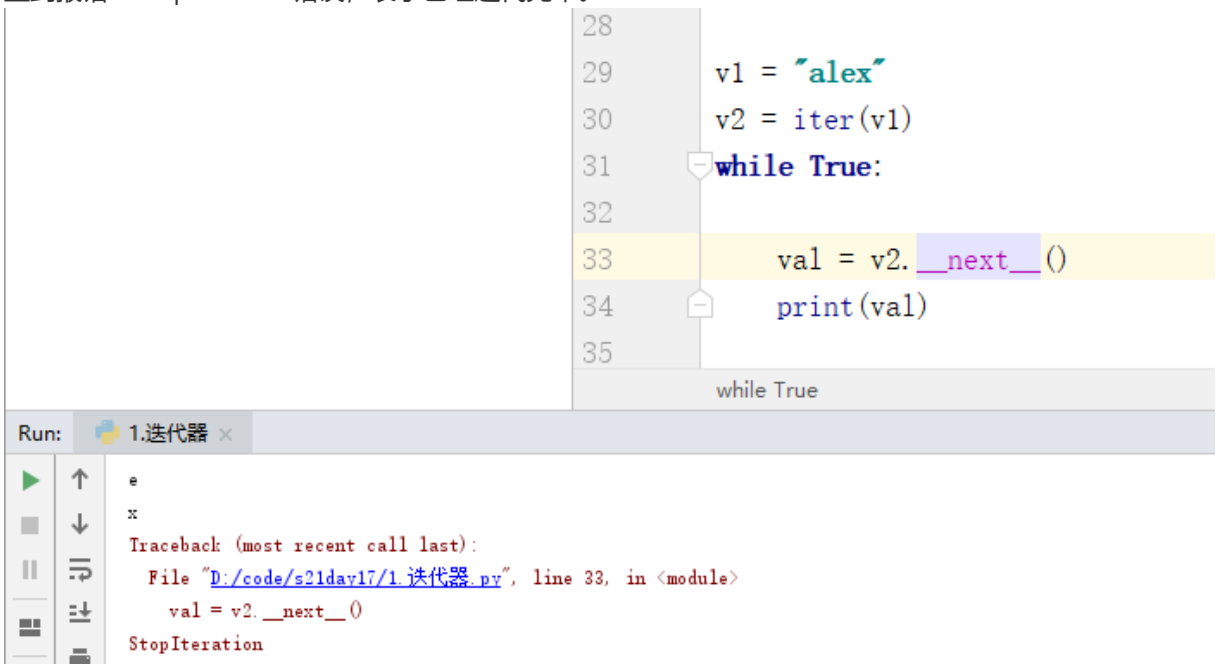
任务：请展示列表中所有的数据。

- while + 索引 + 计数器
- 迭代器，对 某种对象(str/list/tuple/dict/set类创建的对象)-可迭代对象 中的元素进行逐一获取，表象：具有 `__next__` 方法且每次调用都获取可迭代对象中的元素（从前到后一个一个获取）。
  - 列表转换成迭代器：
    - v1 = iter([11,22,33,44])
    - `v1 = [11,22,33,44].__iter__()`
  - 迭代器想要获取每个值：反复调用 `val = v1.__next__()`

```
v1 = [11,22,33,44]

# 列表转换成迭代器
v2 = iter(v1)
result1 = v2.__next__()
print(result1)
```

```
result2 = v2.__next__()
print(result2)
result3 = v2.__next__()
print(result3)
result4 = v2.__next__()
print(result4)
result5 = v2.__next__()
print(result5)
"""
# v1 = "alex"
# v2 = iter(v1)
# while True:
#     try:
#         val = v2.__next__()
#         print(val)
#     except Exception as e:
#         break
```

- 直到报错：StopIteration错误，表示已经迭代完毕。



- 如何判别一个对象是否是迭代器：内部是否有`__next__方法`。

- for循环

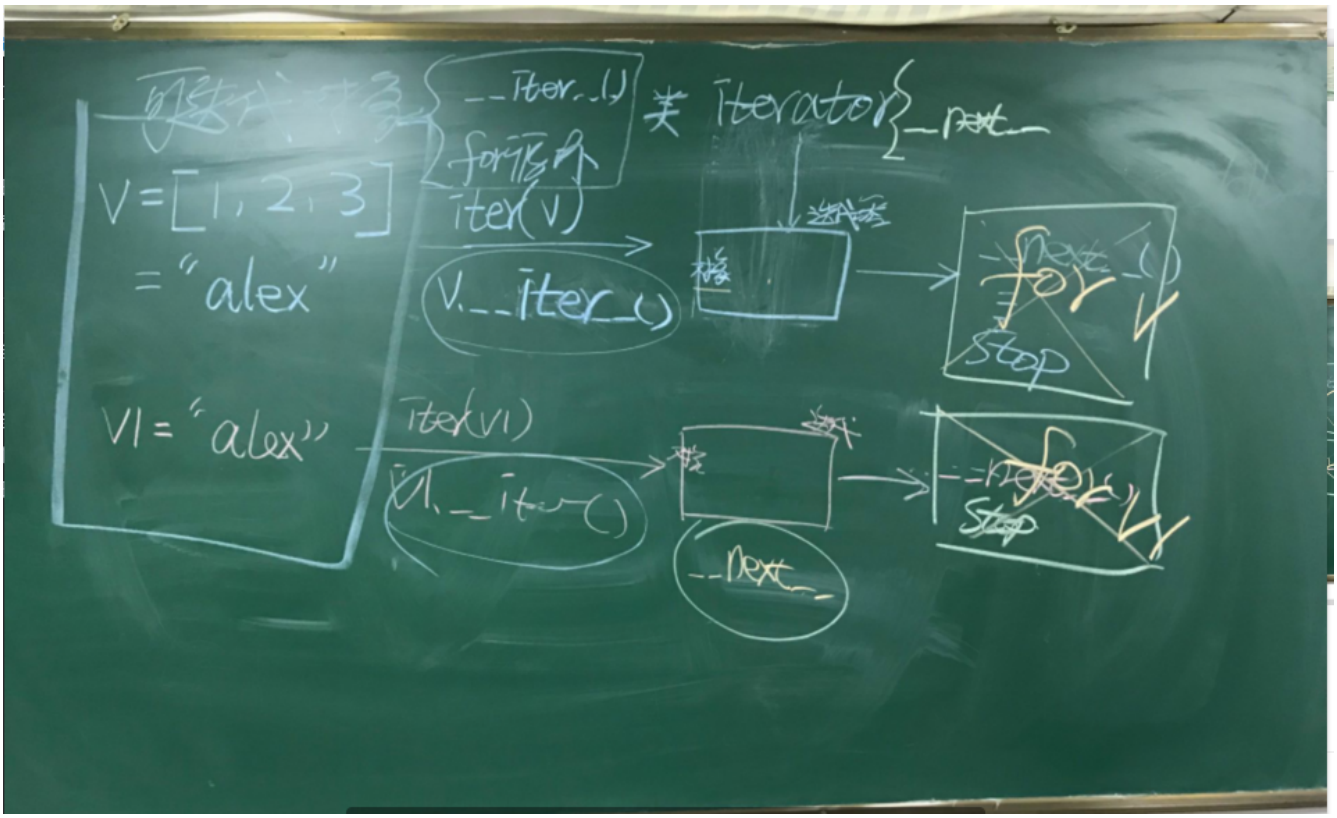
```
v1 = [11,22,33,44]

# 1.内部会将v1转换成迭代器
# 2.内部反复执行 迭代器.__next__()
# 3.取完不报错
for item in v1:
    print(item)
```

3.2 可迭代对象

- 内部具有`__iter__()`方法且返回一个迭代器。 (*)

```
v1 = [11,22,33,44]
result = v1.__iter__()
```

- 可以被for循环



3.3 生成器（函数的变异）

```
# 函数
def func():
    return 123
func()
```

```
# 生成器函数（内部是否包含yield）
def func():
    print('F1')
    yield 1
    print('F2')
    yield 2
    print('F3')
    yield 100
    print('F4')
# 函数内部代码不会执行，返回一个 生成器对象 。
v1 = func()
# 生成器是可以被for循环，一旦开始循环那么函数内部代码就会开始执行。
for item in v1:
    print(item)
```

```
def func():
    count = 1
    while True:
        yield count
        count += 1

val = func()

for item in val:
    print(item)
```

总结：函数中如果存在yield，那么该函数就是一个生成器函数，调用生成器函数会返回一个生成器，生成器只有被for循环时，生成器函数内部的代码才会执行，每次循环都会获取yield返回的值。

```
def func():
    count = 1
    while True:
        yield count
        count += 1
        if count == 100:
            return

val = func()
for item in val:
    print(item)
```

示例：读文件

```
def func():
    """
    分批去读取文件中的内容，将文件的内容返回给调用者。
    :return:
    """
    cursor = 0
    while True:
        f = open('db', 'r', encoding='utf-8')# 通过网络连接上redis
        # 代指   redis[0:10]
        f.seek(cursor)
        data_list =[]
        for i in range(10):
            line = f.readline()
            if not line:
                return
            data_list.append(line)
        cursor = f.tell()
        f.close()  # 关闭与redis的连接


        for row in data_list:
            yield row
```

```
for item in func():
    print(item)
```

redis源码示例

```
    def scan_iter(self, match=None, count=None):
        """..."""
        cursor = '0'
        while cursor != 0:
            # 每次去获取100个
            # cursor取完之后的游标位置
            # data本次取出来100条数据
            cursor, data = self.scan(cursor=cursor, match=match, count=count)
            for item in data:
                yield item
```

其他知识：

- yeild from关键字【欠】
- 生成器推导式【欠】

总结

- 迭代器，对可迭代对象中的元素进行逐一获取，迭代器对象的内部都有一个 **next**方法，用于以一个个获取数据。
- 可迭代对象，可以被for循环且此类对象中都有 **iter**方法且要返回一个迭代器（生成器）。
- 生成器，函数内部有yield则就是生成器函数，调用函数则返回一个生成器，循环生成器时，则函数内部代码才会执行。

  特殊的迭代器（**）：

  ```
  def func():
      yield 1
      yield 2
      yield 3

  v = func()
  result = v.__next__()
  print(result)
  result = v.__next__()
  print(result)
  result = v.__next__()
  print(result)
  result = v.__next__()
  print(result)
  ```

  特殊的可迭代对象：

```
def func():
    yield 1

v = func()
result = v.__iter__()
print(result)
```

作业

1. day15作业
2. 思维导图
3. Python学习笔记
4. 本周作业
5. 复习：从前到后
    - 笔记
    - 作业题