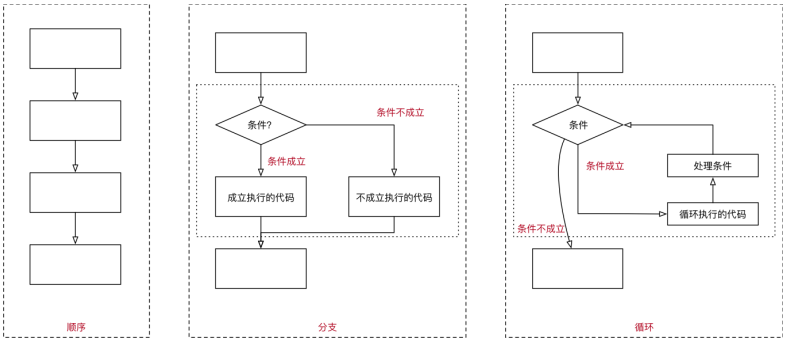
Day-3

- Video-108-132循环
  - 程序的三大流程



    - 顺序：从上向下，顺序执行代码
    - 分支：根据条件判断，决定执行代码的**分支**
    - 循环：让**特定代码重复**执行
  - While循环：让执行的代码按照指定的次数重复执行
  - 基本语法：While语句以及缩进部分是一个**完整的代码块**

```
初始条件设置 —— 通常是重复执行的 计数器

while 条件(判断 计数器 是否达到 目标次数):
    条件满足时，做的事情1
    条件满足时，做的事情2
    条件满足时，做的事情3
    ...(省略)...

    处理条件(计数器 + 1)
```

  - 死循环：没有修改循环的判断条件，导致循环持续执行，程序无法终止
  - 赋值运算符：运算符中间不能添加空格

运算符	描述	实例
=	简单的赋值运算符	c = a + b 将 a + b 的运算结果赋值为 c
+=	加法赋值运算符	c += a 等效于 c = c + a
-=	减法赋值运算符	c -= a 等效于 c = c - a
*=	乘法赋值运算符	c *= a 等效于 c = c * a
/=	除法赋值运算符	c /= a 等效于 c = c / a
//=	取整除赋值运算符	c //= a 等效于 c = c // a
%=	取 模 (余数)赋值运算符	c %= a 等效于 c = c % a
**=	幂赋值运算符	c **= a 等效于 c = c ** a

  - Python中的计数方法：程序计数法，从0开始
  - 循环计算：利用循环重复计算

- break & continue
  - break：某一条件满足时，退出循环，不再执行后续重复的代码
    - **会直接跳出整个循环**
  - continue：某一条件满足时，不执行后续重复的代码
    - **会直接跳到循环的条件判断位置**
  - break & continue都只针对**当前所在的循环**
- 循环嵌套

```
while 条件 1:
    条件满足时，做的事情1
    条件满足时，做的事情2
    条件满足时，做的事情3
    ...(省略)...

    while 条件 2:
        条件满足时，做的事情1
        条件满足时，做的事情2
        条件满足时，做的事情3
        ...(省略)...

        处理条件 2

    处理条件 1
```

  - print函数：默认情况下，在内容输出后，**会自动在内容末尾增加换行**
    - 不希望末尾增加换行

```
# 向控制台输出内容结束之后，不会换行
print("*", end="")

# 单纯的换行
print("")
```

    - 转义字符
      - `\t` 在控制台输出一个 **制表符**，协助在输出文本时 **垂直方向** 保持对齐
      - `\n` 在控制台输出一个 **换行符**

      > 制表符 的功能是在不使用表格的情况下在 **垂直方向** 按列对齐文本

转义字符	描述
\\	反斜杠符号
\'	单引号
\"	双引号
\n	换行
\t	横向制表符
\r	回车

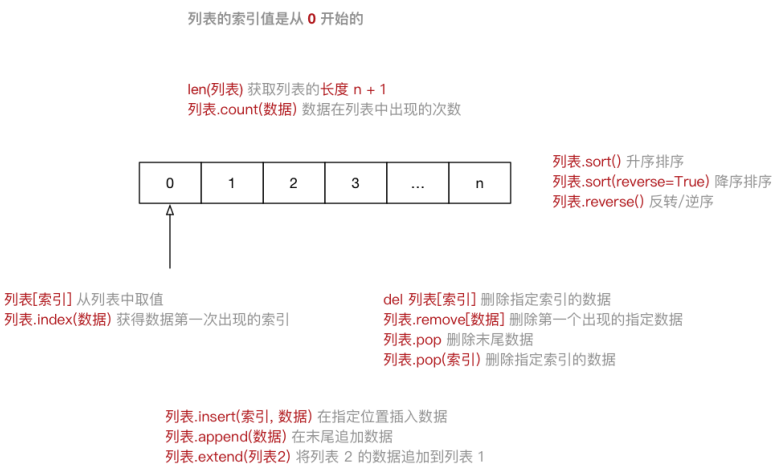
- Video-133-157函数基础
  - 函数：把**具有独立功能的代码块**组织为一个小模块，在需要的时候调用
    - 使用步骤
      - 1.定义函数---封装独立的功能
      - 2.调用函数---享受封装的成果
    - 作用：提高编写的效率，以及代码的崇勇
  - 函数的定义
    - 语法：函数名的命名应该符合标识符的命名规则

```
def 函数名():

    函数封装的代码
    ......
```

  - 函数的调用：函数名()
    - 函数调用不能放在函数定义的上方
  - Pycharm调试工具
    - step over：直接跳过整个函数
    - step into：进入函数中调试
  - 函数的文档注释
    - 给函数添加注释，应该在定义函数的下方，使用连续三对引号
  - 函数的参数：调用函数时，将外部变量传递到函数内部中，进行函数运算
    - 参数的使用：在函数名的后面的小括号内部填写参数，多个参数用 , 分隔
    - 参数的作用：**增加函数的通用性**，针对相同的数据处理逻辑，能够适应更多的数据
    - 形参：定义函数时，小括号中的参数，**是用来接收参数用的，在函数内部作为变量使用**
    - 实惨：调用函数时，小括号中的参数，是**用来把数据传递到函数内部用的**
  - 函数的返回值：函数完成工作后，最后给调用者的一个结果
    - return关键字，后续的代码都不会被执行
    - 调用函数，可以用变量来接收函数的返回结果
  - 函数的嵌套调用：函数又调用了另外一个函数
  - 使用模块中的函数：Python程序架构的一个核心概念
    - 模块：工具包，使用时需要用 import导入模块
    - 每一个以py结尾的python源代码文件都是一个模块
    - 在模块中定义的**全局变量、函数**都是模块能够提供给外界直接使用的工具
    - 模块名也是一个标识符
  - Pyc文件：Python解释器将**模块的源码**转换为**字节码**

        - 字节码：速度优化，二进制，会被处理成机器码，然后通过CPU处理
- Video-158-201高级变量类型
    - 非数字型变量特点
        - 都是一个 **序列** sequence，也可以理解为 **容器**
        - **取值** []
        - **遍历** for in
        - **计算长度、最大/最小值、比较、删除**
        - **链接** + 和 **重复** *
        - **切片**
    - 列表
        - 定义：专门用于存储一串信息，用 **[]** 定义，数据之间用 **,** 分隔
        - **索引从0开始**



        - 常用操作

- 输入 `name_list.` 按下 `TAB` 键， `ipython` 会提示 **列表** 能够使用的 **方法** 如下：

```
In [1]: name_list.
name_list.append   name_list.count    name_list.insert   name_list.reverse
name_list.clear    name_list.extend   name_list.pop      name_list.sort
name_list.copy     name_list.index    name_list.remove
```

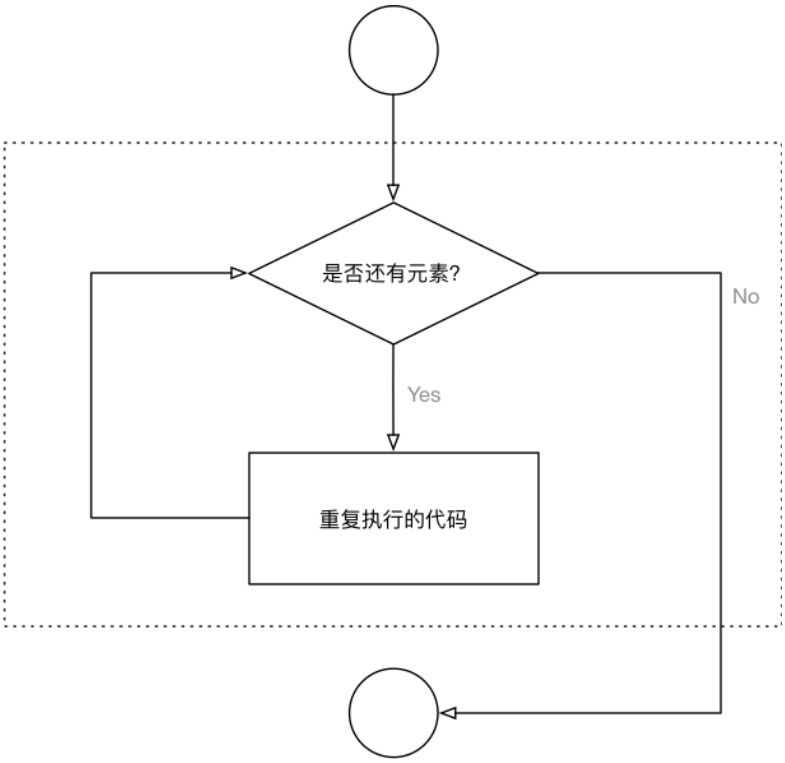
序号	分类	关键字 / 函数 / 方法	说明
1	增加	列表.insert(索引, 数据)	在指定位置插入数据
		列表.append(数据)	在末尾追加数据
		列表.extend(列表2)	将列表2 的数据追加到列表
2	修改	列表[索引] = 数据	修改指定索引的数据
3	删除	del 列表[索引]	删除指定索引的数据
		列表.remove[数据]	删除第一个出现的指定数据
		列表.pop	删除末尾数据
		列表.pop(索引)	删除指定索引数据
		列表.clear	清空列表
4	统计	len(列表)	列表长度
		列表.count(数据)	数据在列表中出现的次数
5	排序	列表.sort()	升序排序
		列表.sort(reverse=True)	降序排序
		列表.reverse()	逆序、反转

- 关键字、函数和方法
  - 关键字：Python内置的、具有特殊意义的标识符；**关键字后面不需要使用括号**
  - 函数：封装了独立功能，可以直接调用
  - 方法：和函数类似，同样是封装了独立的功能；**方法需要通过对象来调用**，表示针对这个对象要做的操作
    - 对象.方法名(参数)
- 循环遍历：从头到尾依次从列表中获取数据；在循环体内部针对每一个元素执行相同的操作
  - 使用 `for` 就能够实现迭代遍历

```
# for 循环内部使用的变量 in 列表
for name in name_list:

    循环内部针对列表元素进行操作
    print(name)
```

- 应用场景
  - 1.列表存储相同类型的数据
  - 2.通过迭代遍历，在循环体内部，针对列表中的每一项元素，执行相同的操作



- 元组：表示多个元素组成的序列
  - **元组的元素不能修改**
  - 元组的定义
    - 元组的索引从 0 开始
    - 元组通常保存不同类型的变量
    - 元组中只包含一个元素时，需要在后面加上，
  - 元组的常用操作
    - count，计数
    - index，取索引
    - 循环遍历：可以使用for循环，遍历循环各种类型的数据
  - 应用场景
    - 1.**函数的参数和返回值**，一个函数可以接收任意多个参数，或者一次返回多个数据
    - 2.**格式字符串**
    - 3.**让列表不可以被修改，以保护数据安全**
      - 列表--->元组：tuple(列表)
      - 元组--->列表：list(元组)
- 字典：通常用于存储**描述一个物体的相关信息**
  - 与列表的区别
    - 1.**列表是有序**的对象集合
    - 2.**字典是无序**的对象集合
  - 字典的定义：使用**键值对**存储数据

        - 1.键key：索引
        - 2.值value：数据
        - 3.键和值之间使用：隔离
        - **4.键必须是唯一的**
        - 5.值可以取任何数据类型，**键只能使用字符串、数字或元组**
    - 字典的常用操作

len(字典) 获取字典的 键值对数量

key	value
name	小明
age	18
gender	True
height	1.75

字典.keys() 所有 key 列表
字典.values() 所有 value 列表
字典.items() 所有 (key, value) 元组列表

字典[key] 可以从字典中取值，key 不存在会报错
字典.get(key) 可以从字典中取值，key 不存在不会报错

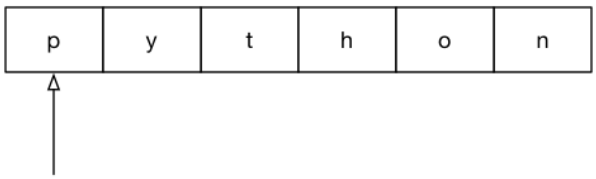
del 字典[key] 删除指定键值对，key 不存在会报错
字典.pop(key) 删除指定键值对，key 不存在会报错
字典.popitem() 随机删除一个键值对
字典.clear() 清空字典

字典[key] = value
    如果 key 存在，修改数据
    如果 key 不存，新建键值对
字典.setdefault(key, value)
    如果 key 存在，不会修改数据
    如果 key 不存在，新建键值对
字典.update(字典2) 将字典 2 的数据合并到字典 1

        - 循环遍历：依次从字典中获取所有键值对
    - 应用场景
        - 1.使用多个键值对，描述一个物体的相关信息---描述更复杂的数据信息
        - 2.将多个字典放在一个列表中，再进行遍历，在循环体内部针对每一个字典进行相同的处理
- 字符串：一串字符，是**编程语言中表示文本的数据类型**
    - 字符串的定义
        - 建议使用“”来定义字符串
        - 可以使用索引获取一个字符串中的指定位置的字符，索引从0开始
        - 也可以使用for循环进行迭代循环
    - 常用操作
        - 计数操作

字符串的索引值是从 0 开始的

len(字符串) 获取字符串的长度
字符串.count(字符串) 小字符串在大字符串中出现的次数



字符串[索引] 从字符串中取出单个字符
字符串.index(字符串) 获得小字符串第一次出现的索引

- 判断类型-9

方法	说明
string.isspace()	如果 string 中只包含空格，则返回 True
string.isalnum()	如果 string 至少有一个字符并且所有字符都是字母或数字则返回 True
string.isalpha()	如果 string 至少有一个字符并且所有字符都是字母则返回 True
string.isdecimal()	如果 string 只包含数字则返回 True， `全角数字`
string.isdigit()	如果 string 只包含数字则返回 True， `全角数字` 、 `⑴` 、 `\u00b2`
string.isnumeric()	如果 string 只包含数字则返回 True， `全角数字` ， `汉字数字`
string.istitle()	如果 string 是标题化的(每个单词的首字母大写)则返回 True
string.islower()	如果 string 中包含至少一个区分大小写的字符，并且所有这些(区分大小写的)字符都是小写，则返回 True
string.isupper()	如果 string 中包含至少一个区分大小写的字符，并且所有这些(区分大小写的)字符都是大写，则返回 True

- 查找和替换-7

方法	说明
string.startswith(str)	检查字符串是否是以 str 开头，是则返回 True
string.endswith(str)	检查字符串是否是以 str 结束，是则返回 True
string.find(str, start=0, end=len(string))	检测 str 是否包含在 string 中，如果 start 和 end 指定范围，则检查是否包含在指定范围内，如果是返回开始的索引值，否则返回 `-1`
string.rfind(str, start=0, end=len(string))	类似于 find()，不过是从右边开始查找
string.index(str, start=0, end=len(string))	跟 find() 方法类似，不过如果 str 不在 string 会报错
string.rindex(str, start=0, end=len(string))	类似于 index()，不过是从右边开始
string.replace(old_str, new_str, num=string.count(old))	把 string 中的 old_str 替换成 new_str，如果 num 指定，则替换不超过 num 次

- 大小写转换-5

方法	说明
string.capitalize()	把字符串的第一个字符大写
string.title()	把字符串的每个单词首字母大写
string.lower()	转换 string 中所有大写字符为小写
string.upper()	转换 string 中的小写字母为大写
string.swapcase()	翻转 string 中的大小写

- 文本对齐-3

方法	说明
string.ljust(width)	返回一个原字符串左对齐，并使用空格填充至长度 width 的新字符串
string.rjust(width)	返回一个原字符串右对齐，并使用空格填充至长度 width 的新字符串
string.center(width)	返回一个原字符串居中，并使用空格填充至长度 width 的新字符串

- q去除空白字符-3

方法	说明
string.lstrip()	截掉 string 左边（开始）的空白字符
string.rstrip()	截掉 string 右边（末尾）的空白字符
string.strip()	截掉 string 左右两边的空白字符

- 拆分和连接-5

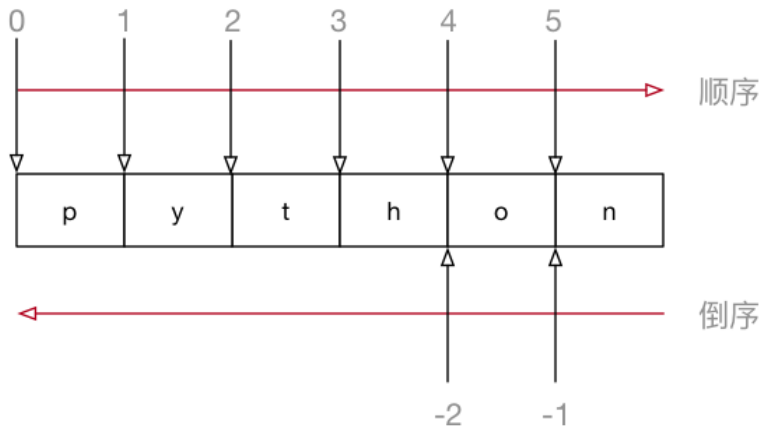
方法	说明
string.partition(str)	把字符串 string 分成一个 3 元素的元组 (str前面, str, str后面)
string.rpartition(str)	类似于 partition() 方法，不过是从右边开始查找
string.split(str="", num)	以 str 为分隔符拆分 string，如果 num 有指定值，则仅分隔 num + 1 个子字符串，str 默认包含 '\r', '\t', '\n' 和空格
string.splitlines()	按照行('\r', '\n', '\r\n')分隔，返回一个包含各行作为元素的列表
string.join(seq)	以 string 作为分隔符，将 seq 中所有的元素（的字符串表示）合并为一个新的字符串

- 字符串切片
  - 切片方法适用于字符串、列表、元组
  - 字符串[开始索引:结束索引:步长]；左闭右开

    注意：

    1. 指定的区间属于 **左闭右开** 型 `[开始索引，结束索引）` => `开始索引 >= 范围 < 结束索引`
       - 从 `起始` 位开始，到 `结束` **位的前一位** 结束（**不包含结束位本身**）
    2. 从头开始，**开始索引 数字可以省略，冒号不能省略**
    3. 到末尾结束，**结束索引 数字可以省略，冒号不能省略**
    4. 步长默认为 `1`，如果连续切片，**数字和冒号都可以省略**

  - 顺序&倒序索引



- 公用方法
  - 内置函数

函数	描述	备注
len(item)	计算容器中元素个数	
del(item)	删除变量	del 有两种方式
max(item)	返回容器中元素最大值	如果是字典，只针对 key 比较
min(item)	返回容器中元素最小值	如果是字典，只针对 key 比较
cmp(item1, item2)	比较两个值，-1 小于/0 相等/1 大于	Python 3.x 取消了 cmp 函数

注意

- **字符串** 比较符合以下规则： "0" < "A" < "a"

- 切片：使用索引值来限定范围
- 运算符

运算符	Python 表达式	结果	描述	支持的数据类型
+	[1, 2] + [3, 4]	[1, 2, 3, 4]	合并	字符串、列表、元组
*	["Hi!"] * 4	['Hi!', 'Hi!', 'Hi!', 'Hi!']	重复	字符串、列表、元组
in	3 in (1, 2, 3)	True	元素是否存在	字符串、列表、元组、字典
not in	4 not in (1, 2, 3)	True	元素是否不存在	字符串、列表、元组、字典
> >= == < <=	(1, 2, 3) < (2, 2, 3)	True	元素比较	字符串、列表、元组

注意

- `in` 在对 **字典** 操作时，判断的是 **字典的键**
- `in` 和 `not in` 被称为 **成员运算符**

成员运算符

成员运算符用于 **测试** 序列中是否包含指定的 **成员**

运算符	描述	实例
in	如果在指定的序列中找到值返回 True，否则返回 False	`3 in (1, 2, 3)` 返回 `True`
not in	如果在指定的序列中没有找到值返回 True，否则返回 False	`3 not in (1, 2, 3)` 返回 `False`

注意：在对 **字典** 操作时，判断的是 **字典的键**

  - + vs extend：+方法会返回一个新的变量，而extend是直接在原来变量中修改
  - append：是直接将变量作为独立元素插入
- 完整的for循环语法

```
for 变量 in 集合:

    循环体代码
else:
    没有通过 break 退出循环，循环结束后，会执行的代码
```

  - 应用场景
    - 在 **迭代遍历** 嵌套的数据类型时，例如 **一个列表包含了多个字典**
    - 需求：要判断 某一个字典中 是否存在 指定的 值
      - 如果 **存在**，提示并且退出循环
      - 如果 **不存在**，在 **循环整体结束** 后，希望 **得到一个统一的提示**

以上内容整理于 幕布文档