Python基础DAY05

一、字典概述

- 声明方式: 变量名= {键:值,键:值....}
- 作用:
  - 存储大量数据,比列表存储的数据量还大
  - 将数据关联起来
- 字典的键必须是可哈希的数据类型,即不可变的数据类型,且要是唯一的,若不唯一,则后者会覆盖前者
- 字典的值可以是任意数据
- 哈希:只有不可变的数据类型可以哈希,可变的数据类型不可哈希
- 字典是无序的,不支持索引.Python3.6以后按照用户添加的顺序排列
- 总结:字典是无序,可变,可迭代的数据结构.唯一一种键值对的数据.使用非常频繁,也非常重要
- 字典和列表的区别:
  - 字典的查找更加方便
  - 字典的查找速度更快

二、字典的增删改查

设有字典dic = {"name":"guo","age":"21","sex":"m"}

> 首先要声明的一点是对字典的操作都是对字典的键进行操作

- 字典的增加:
  - 直接增加,暴力增加: dic["school"] = "sxau"
  - 通过setdefault增加:dic.setdefalut(键,值)
    - 若该键已存在,例如dic.setdefault("name","mu"),则该操作不起作用,并且返回字典中该键对应的值
    - 若该键不存在,例如dic.setdefault("addr","cz"),则会在字典中增加一对键值对,并且返回增加的值
    - setdefault的执行过程
      - 1.首先先到字典中寻找该键是否存在,若存在才会执行下一步
      - 2.将键值添加到该字典中
- 字典的删除:
  - pop:dic.pop("sex"),删除该键及对应的值,并且返回该值
  - popitem:dic.popitem("sex"),官方文档描述为随机删除一个键值对,但实际上现在会删除最后一个键值对,返回被删除的键值对
  - clear:dic.clear(),将字典清空
  - del dic["sex"]:删除该键值对,无返回值
- 字典的修改:
  - 直接修改:dic["name"] = "mu",若该键不存在则新增加该键值对,但若已存在,则修改该值
  - update:dic.update(字典):参数里填一个字典,并且该字典要高于被修改的原字典.该字典中的键若在原字典中已存在则修改,不存在则添加例如:

dic.update({"name":"xin","birth":"0714"})则name内容会被修改为"xin",生日则会添加到原字典中

这个方法也可以叫做两个字典的合并

- 字典的查找:
  - 直接找:print(dic["name"]),返回该键对应的值,若该键不存在则会报错
  - get:print(dic.get("name","提示信息")),返回该键对应的值,若该键不存在则会返回一个None,参数中的提示信息为若找不到该键时输出该提示信息,替代None
- 字典的其他操作:
  - 若用for遍历字典,默认遍历字典的键
  - dic.keys():获取包含该字典所有键的一个高仿列表
  - dic.values():获取包含该字典所有值的一个高仿列表
  - dic.items():获取包含该字典所有键值对的一个高仿列表,每一对键值对都以元组形式存在
  - 高仿列表:形式上与列表一样,但只可以迭代,不支持索引

三、解构

- 如果想要为多个变量同时赋值,一个一个赋会太过麻烦,代码冗余,使用解构的方式就会变得简单,例如:

  要为a赋值10,为b赋值20,为c赋值30,若采用传统方式:a = 10 b = 20 c = 30

  若采用解构的方式赋值:a,b,c = 10,20,30
- 若要给变量赋的值类型是字典,则会将键赋给变量
- 要注意的是解构使用时,等号前后元素数量要保持一致,即一一对应,否则会报错
- 但也有一种方式可以不需要一一对应,即采用聚合

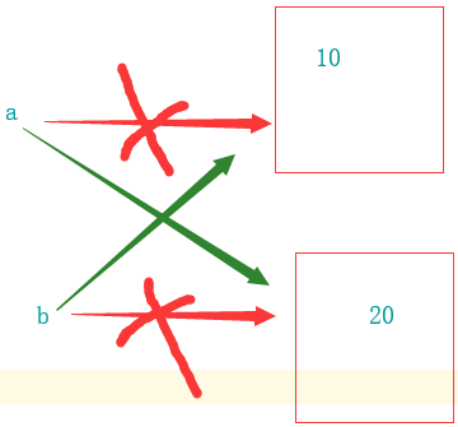
  a,b,*c = [1,2,3,4,5,6,7].采用这种方式可以将a和b所需的元素之外的元素全部存在 * c里,如果输出c则结果为一个列表,如果输出 * c则结果变成多个元素
- 解构在字典中的应用:循环键值对:

  for k,v in dic.items():

  print(k)

  print(v)
- 面试题:用一行代码将a = 10,b = 20的值互换

  a,b = b,a



四、字典的嵌套

查找时通过键进行查找,一层一层查找