**day03笔记**

day03(上午)：

**1.列表(list)**

特点：**有序的**(有索引、定义和显示的顺序是一致的)、**可变的**(既可以改变元素的内容也可以自动(扩缩)容)、**可重复的**、可以存储任何数据类型的元素

lt = [**'宋江'**,**'李逵'**,**'鲁智深'**,**'武松'**,**'西门庆'**,**'潘金莲'**,**'时迁'**,[**'刘备'**,**'关羽'**,**'张飞'**]]

1.1.如何访问列表中的元素？

1).获取元素

列表名配合下标获取 --> 例如：**lt[0],lt[-1]**

2).设置元素(和str有区别，str属于不可变数据类型，所以不可设置元素)

列表名配合下标 = ... --> 例如：**lt[4] = "武大郎"**（即将西门庆改变为武大郎元素）

1.2.列表的+和\*操作：

对于+操作而言：将两个列表对象中的元素都组合起来以一个新的列表返回

对于\*操作而言：将列表中的元素重复出现n次，以新的列表返回

lt1=[1,2,3,4]  
lt2=[5,6,7,8]  
lt3 = lt1 + lt2  
print(lt3,type(lt3),len(lt3)) *#[1,2,3,4,5,6,7,8]<class 'list'> 8*print(lt2 \* 3) *#[5,6,7,8,5,6,7,8,5,6,7,8]*

**【注意事项】：**

lt1 = ['妲己','安琪拉','王昭君','诸葛亮','司马懿','甄姬']

需求：将"嬴政"加入到lt1中

思考：以下代码会不会有问题？ lt4 = lt1 + "嬴政"

会报错，因为"嬴政"是一个字符串数据，字符串和字符串之间只能做拼接操作，和需求不符；

1.3.和列表有关的一些函数：

**append(obj)：将obj元素追加到列表的末尾**

例如：以下代码可以实现将"嬴政"加入到lt1中

lt1.append(**"嬴政"**)  
print(lt1)

clear()：清空列表，元素没了，列表对象还在（内部不能放东西，即没有参数）

lt.clear()  
print(lt)

显示结果为[ ] 就是框架还在

1.4.del关键字配合列表对象/元素使用

目的：回收列表对象，并且回收其中的元素内容（即可以回收列表对象，也可以回收列表元素）

lt2 = [**'李白'**,**'露娜'**,**'韩信'**,**'元歌'**,**'阿珂'**]

**del** lt2[-2],lt2[1]  
print(lt2)  
显示结果为lt2 = [**'李白'**,**'韩信'**,**'阿珂'**]  
以下代码就是全部删除了 连框架都没了

**del** lt2  
print(lt2)

**演示：列表对象可以存储任何类型的数据，并且可以重复**lt5=[0,50,3.14,**True**,**"abcd"**,[1,2,3],(10,20),{**"haha"**:**"hehe"**},{5,6,7},**None**,0,3.14,**True**]  
print(lt5)

1.5.列表的切片操作：

格式和特点和str一模一样...

lt6 = ['葫芦娃','黑猫警长','熊出没','灌篮高手','柯南','海尔兄弟']

lt7 = lt6[1:5]

print(lt7,type(lt7))

print(lt6,type(lt6))

print(lt6[2:-2])

print(lt6[:])

print(lt6[::-1])

以下代码相当于删除,'黑猫警长','熊出没','灌篮高手'

lt6[1:-2] = []  
print(lt6)

以下代码相当于删除全部数据，但结构还在  
lt6[:] = []  
print(lt6)

1.6.归纳总结列表的定义格式

**格式一：列表名 = [值1,值2,...,值n]**

**格式二：列表名 = [变量1,变量2,...,变量n] = [值1,值2,...,值n]**

**格式三：列表名 = []**

例如：lt7 = [1,2,3,4,5]  
lt8 = [a,b,c] = [10,20,30]  
print(lt8,type(lt8))  
print(a,b,c)  
lt9 = []  
print(lt9)  
lt10 = [**'习近平'**]  
print(lt10)

**2.元祖(tuple)**

特点：**有序的(有索引、定义和显示的顺序是一致的)、不可变的(和str一样理解)、**

**可重复的、可以存储任何数据类型的元素**

2.1.如何访问元祖中的元素？

1).获取元素

tp = (**'秦始皇'**,**'汉高祖'**,**'隋炀帝'**,**'唐太宗'**,**'宋太祖'**,**'明太祖'**,**'皇太极'**)

元祖名配合下标获取 --> 例如：**tp[0],tp[-1]**

2).设置元素(和str保持一致)

没有此操作，是非法的，原因元祖是不可变的数据类型

2.2.元祖的+和\*操作：

对于+操作而言：将两个元祖对象中的元素都组合起来以一个新的元祖返回

对于\*操作而言：将元祖中的元素重复出现n次，以新的元祖返回

tp=(1,2,3,4)  
tp1=(5,6,7,8)  
print(tp + tp1) *#(1,2,3,4,5,6,7,8)*print(tp1 \* 3) *#(5,6,7,8,5,6,7,8,5,6,7,8)*

【注意事项】：元祖没有额外函数

思考：元祖有没有类似于列表的append或者clear函数？

没有，因为append和clear这类函数改变的都是容器对象内部的元素内容，这和不可变所冲突

2.3.del关键字配合元祖对象使用**对于tuple类型而言，del关键字不能回收其内部任何的元素，但是可以直接回收整个元祖对象**

目的：回收整个元祖对象，但是无法回收其任何一个元素

2.**4.元祖的切片操作：**

格式和特点和str一模一样...

print(tp[:6:2])  
print(tp[:])

*# ('秦始皇','隋炀帝','宋太祖')  
# ('秦始皇','汉高祖','隋炀帝','唐太宗','宋太祖','明太祖','皇太极')*

2.5.归纳总结元祖的定义格式

格式一：元祖名 = (值1,值2,...,值n)

格式二：元祖名 = (变量1,变量2,...,变量n) = (值1,值2,...,值n)

格式三：元祖名 = ()

格式四：元祖名 = (值1,) **【注意】：定义元祖并且只有一个元素，必须在元素后显示的加上一个逗号**

tp1 = (1,2,3,4,5,6)  
tp2 = (a,b,c) = (10,20,30)  
tp3 = () **tp4 = ('习近平',)**  
tp5 = (**'习近平'**,**'江泽民'**)

print(tp4) *#('习近平',) 如果tp4里面没有逗号，则得到结果为 习近平*

3.**字典(dict)**

**特点：无序性(没有索引，通过键找值)、可变的数据类型(值可变、键不可变)、**

字典的结构：键值对组合 --> key-value对象

键(key)：唯一性(不可重复)、不可变的数据类型(例如：list就不能作为键)

值(value)：没有任何要求，可以是任何类型的数据，甚至可以重复

3.1.如何访问字典元素？

dic1 = {**"name"**:**"韩梅梅"**,**"age"**:23,**"height"**:165.0}

1).获取值？

字典名配合键名可以得到对应的值

dic1[**'name'**]

2).设置值？

字典名配合键名 = ...

dic1[**"name"**] = **"李雷"**

**得到结果如下：{'name': '李雷', 'age': 23, 'height': 165.0}**

**【注意事项】：**

**1).在定义字典对象的时候如果出现了重复的键，保留键名的同时，将后者的值覆盖前面的值**

**2).在添加新的键值对的过程中，如果原字典对象中没有此键，那么就将其键值对一并加入进去，称为一个元素**

**dic2 = {"name":"韩梅梅","age":23,"height":165.0,"name":"李雷"}**

**dic2['weight'] = 120**

**print(dic2)**

**得到结果如下：{'name': '李雷', 'age': 23, 'height': 165.0, 'weight': 120}**

**思考：字典有没有+和\*操作？  
没有，是非法的，和特点不符...**

3.2.字典相关函数：

clear()：清空字典元素，保留空字典对象

dic3.clear()  
print(dic3)

3.3.和**关键字del配合使用：**

**目的：回收键值对组合，或者整个字典对象 （可以回收键值对也可以回收整个字典对象）**

**del** dic2[**'age'**] 删除年龄键值对

**del** dic2 删除所有（数据和整个字典结构）

**注意事项：**键**：必须满足不可变的数据类型；枚举：int、float、bool、str、None等都可以充当键 满足唯一性（不可重复）**

值**：没有要求，任何类型都可以,甚至可重复**

3.4.对于通过键查找值的情况可以分为以下两种方式来实现：

方式一：字典名[键名] 如果键不存在，报错；出现KeyError类型的异常

print(dic2[**"name1"**])

方式二：字典吗.get(键名) 如果键不存在，不会报错，会返回给程序一个None值；此方式比较安全/友好(更加好用)  
print(dic2.get(**"name1"**))

**思考：可不可以通过值找到键？  
不可以，因为值是可能重复的，不能精确的找到唯一的键**

4**.集合(set)**

特点：无序性(**没有索引**，每一次显示的效果都不一样)、唯一性(不可重复)、

可变的、只能存放不可变的数据 例如list,dict等类型对象

使用场景**：在实际开发过程中，经常用于对数据进行去重(过滤)**

s1 = {**"貂蝉"**,**"西施"**,**"王昭君"**,**"玉环"**}

4.1.集合的一些常用操作如下：以下的操作都是在两个集合对象中使用

符号：

**&**：取交集

**|**：取并集

**-**：取差集

**^**：先取并集再舍弃交集部分

s2 = {**"李白"**,**"杜甫"**,**"李商隐"**,**"王安石"**,**"白居易"**}  
s3 = {**"李白"**,**"曾巩"**,**"苏轼"**,**"杜甫"**,**"王安石"**}

s4 = s2 & s3  
print(s4) *#显示结果 {'杜甫', '李白', '王安石'}*

s5 = s2 | s3  
print(s5) *#显示结果 {'杜甫','曾巩','苏轼','李商隐','李白','王安石','白居易'}*

**注意：对于取差集的操作，需要关注的是谁在-号的前面(左侧)，返回的就是谁独有的数据**s6 = s2 - s3

print(s6) *#显示结果 {'李商隐', '白居易'}*  
s7 = s3 - s2

print(s7) *#显示结果 {'苏轼', '曾巩'}*

演示^的使用：先取并集再舍弃交集部分  
print(s2 ^ s3)  *#显示{'白居易', '李商隐', '苏轼', '曾巩'}*

print(s3 ^ s2) *#显示{'白居易', '李商隐', '苏轼', '曾巩'}*

**思考：对于集合和集合之间有+操作吗？  
没有如下操作，会出现TypeError类型的异常对象；**

**思考：删除"貂蝉"这个数据，该怎么做到？可以使用remove(obj)函数来实现...**s1.remove(**"貂蝉"**)  
print(s1)

**假如不使用remove(obj)函数，还可以怎么做到！  
可以做到，步骤如下：  
第一步：先将集合转换为列表 --> 使用转换函数list()  
第二步：使用循环的思想，来处理列表对象，一旦匹配上了"貂蝉"，就可以使用del关键字进行数据删除的行为**

lt = list(s1) #*将集合转换为列表*i = 0  
**while** i < len(lt):  
 **if** lt[i] == **"貂蝉"**:  
 print(lt[i])  
 **del** lt[i]  
 i += 1  
print(lt)

**需求：将如下的数据存入到列表中，在转换为集合对象，查看去重效果；  
10,0,3.14,'abc',10,3.14,'eeffdd'，*只要是能重复的元素就不能放在里面转集合***

lt = [10,0,3.14,**'abc'**,10,3.14,**'eeffdd'**,(1,2,3)]  
print(lt)  
s2 = set(lt)  
print(s2) *#{0,3.14,10,'eeffdd','abc',(1, 2, 3)}*

**需求：定义一个空集合？  
s3 = set()**

4.2.集合相关的函数：

clear()：清空集合，保留外壳

4.3.和关键字del配合使用：

目的：回收集合元素，或者回收整个集合对象

s1 = {**"貂蝉"**,**"西施"**,**"王昭君"**,**"玉环"**}  
s1.clear()  
print(s1) *#set()*  
**del** s1  
print(s1) *#直接全删了,什么都不剩,所以报错NameError:name's1'is not defined*

-----------------------------------------------------------------------------------------------

day03(下午)：

一、运算符

1.算数运算符：

符号：+ - \* / %(取余、取模) //(取整) \*\*(开方)

2.比较运算符：

符号：> >= < <= ==(全等) !=(不等于)

特点：**比较运算的结果一定是布尔值(True/False)**

print(7 >= 3) ***#True***print(7 < 3) ***#False***

3.赋值&混合赋值运算符：

赋值运算符符号：=

混合赋值运算符符号：+= -= \*= /= %=

a += 4 ***#相当于：a = a + 4***

a += b ***#相当于：a = a + b***

4.逻辑运算符：

需求：a = 6 3<a<9(数学的实现，但是多数计算机语言不支持，python支持)

拆分：a > 3 a < 9 两个表达式之间需要使用逻辑运算符来进行关联

符号：and(逻辑与，并且，去交集)、or(逻辑或，或者，取并集)、not(逻辑非，取反)

特点：运算符两侧的数据类型一定是布尔型，结果也是布尔型

模板：

对于and操作而言：

True and True 结果：True

True and False 结果：False

False and True 结果：False

False and False 结果：False

print(**True and True**)

......

总结：and操作，满足一假即假的特点，理解为：对False敏感

对于or操作而言：

True or True 结果：True

True or False 结果：True

False or True 结果：True

False or False 结果：False

总结：or操作，满足一真即真的特点，理解为：对True敏感

对于not操作而言：

not True 结果为：False

not False 结果为：True

5.位运算符：(了解)

长处：是运算效率最高的运算符，直接是底层二进制的运算

符号：&(按位与)、|(按位或)、^(按位异或)（相异就为一）、<<(左移)、>>(右移)

模板：

对于按位与操作而言：

9 & 5，结果为多少？ 1

9：0000 1001

5：0000 0101

结果： 0000 0001

对于按位或操作而言：

9 | 5，结果为多少？ 13

9：0000 1001

5：0000 0101

结果： 0000 1101

对于按位异或操作而言：

9 ^ 5，结果为多少？ 12

9：0000 1001

5：0000 0101

结果： 0000 1100

对于<<(左移)操作而言：

9 << 2，结果为多少？ 36

9：0000 1001

结果：00 100100

对于>>(右移)操作而言：

9 >> 2，结果为多少？ 2

9：0000 1001

结果：0000 0010

6.**成员运算符：**

作用：运用对于容器对象的判断，结果都是布尔值；判断某个元素是否存在于容器中

符号：**in / not in**

【注意】：对于dict对象，只能判断键是否存在，对于值是无效的(永远返回False)

补充：何为表达式？

大白话：凡是计算可以得出个值的就是表达式

何为条件表达式？

大白话：在表达式的基础上，最终结果一定是布尔值;

str1 = **"abcdefg1234567"**print(**"c" in** str1) *#True*  
print(**"cde" in** str1) *#True*  
print(**"cdeg" in** str1) *#False，一旦断裂了，就返回False*print(**"cdeg" not in** str1) *#True*  
lt = [10,20,30,**"aa"**,**"bb"**,**"cc"**]  
print(10 **in** lt) *#True*  
print(**"aa" in** lt) *#True*  
print(**"AA" in** lt) *#False对于字符串数据内容层面，严格区分大小写* **对于dict类型对象的判断，只对键有效，值永远False(无效)**dic1 = {**"name"**:**"Tom"**,**"age"**:30}  
print(**"name" in** dic1) *#True*  
print(**"name1" in** dic1) *#False*  
print(**"Tom" in** dic1) *#False*print(30 **in** dic1) *#False*

7.**三元/三目运算符：**

格式：**表达式1 if 条件表达式 else 表达式2**

执行流程：

会先执行条件表达式，

如果执行结果为True，那么就会执行表达式1，并且将表达式1的结果作为整个三元运算的最终结果

如果执行结果为False，那么就会执行表达式2，并且将表达式2的结果作为整个三元运算的最终结果

三运运算符是python的语法糖之一

**需求：从键盘读入两个整数值，得到较大值**num1 = int(input(**'请输入第一个整数：'**))  
num2 = int(input(**'请输入第二个整数：'**))  
result = (num1 if num1 > num2 else num2)  
 print("两个数中的较大值为：%d" %result)  
或者print("两个数中的较大值为：{}".format(result))

二、流程控制语句

分类：1).顺序结构

2).判断结构

3).循环结构

**判断结构：**

特点：如果 ... 否则 ...

分类：三种格式

格式一：

**①**

**if 条件表达式:**

**语句块**

**②**

执行流程：会先执行①，遇到if关键字执行条件表达式；如果条件表达式的结果为True，那么就立即进入其内部执行语句块，知道语句块全部执行完毕，认为if结构结果了；最后一定会执行②；如果条件表达式的结果为False，那么就不会进入其内部执行语句块，可以认为整个if结构已经结束了；但是最终一定会执行②；

**需求：定义一个变量接受一个正整数，判断其奇偶性**  
a = 11  
a = 10

**if** a % 2 == 0:  
 print(**'是偶数...'**)

**if** a % 2 == 1:  
 print(**'是奇数...'**)  
print(**'程序结束了...'**)

格式二：

**①**

**if 条件表达式:**

**语句块1**

**else:**

**语句块2**

**②**

执行流程：会先执行①，遇到了if关键字执行条件表达式；如果条件表达式的结果为True，那么就立即进入if的内部执行语句块1，执行完毕以后整个if结构就结束了，最终一定会去执行②如果条件表达式的结果为False，那么就立即进入else的内部执行语句块2，执行完毕以后整个if结果也结束了，最终一定会去执行②

age = 13  
**if** age > 18:  
 print(**'恭喜你成年了...'**)  
**else**:  
 print(**'对不起，你还不能看成人电影'**)  
print(**'但是你可以到老郭家里看...'**)