目录

[一、 数据类型 2](#_Toc6991)

[1. 查看数据类型 2](#_Toc31949)

[2. 数据类型转换 2](#_Toc25774)

[3. 字符串 2](#_Toc1618)

[二、 变量 3](#_Toc21889)

[三、 运算符 3](#_Toc23526)

[四、 input语句 4](#_Toc14296)

[五、 布尔类型（bool）和比较运算符结合 4](#_Toc14804)

[六、 if判断语句if和else后要有： 4](#_Toc2263)

[七、 循环语句 7](#_Toc19821)

[八、 函数 10](#_Toc21738)

[1. 函数传入参数 10](#_Toc14931)

[2. 函数返回值 11](#_Toc31728)

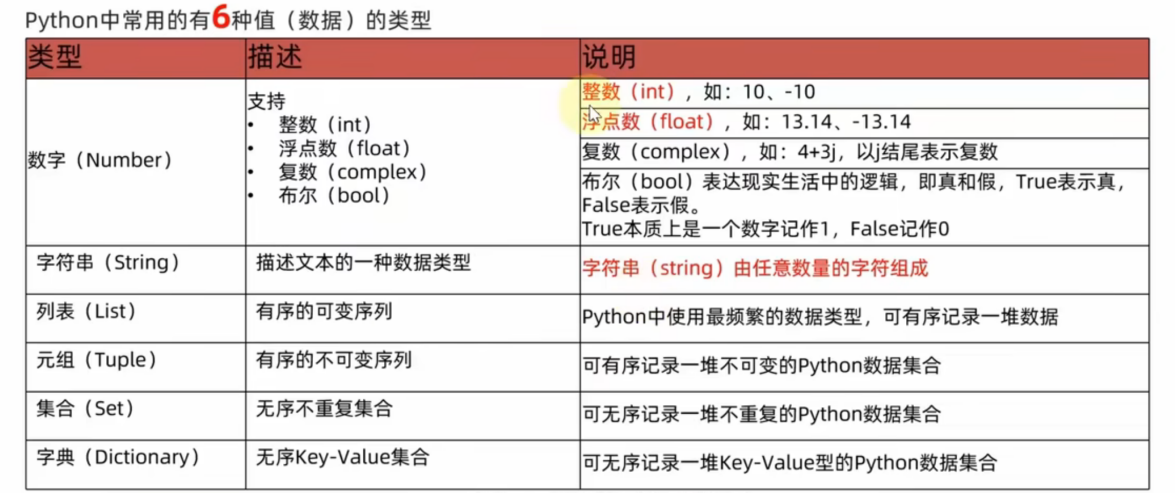
[3. 函数的说明文档 11](#_Toc21149)

[4. 函数嵌套调用 11](#_Toc29536)

[5. 作用域 12](#_Toc29312)

[九、 数据容器 13](#_Toc30956)

1. 数据类型



1. 查看数据类型
2. 用print输出数据类型信息

**print(type(60))**

1. 使用变量存储type语句的结果

**a=type(“你好”)**

**print(a)**

1. 使用type语句查看变量的数据类型

**B =666**

**print(type(B))**

1. 数据类型转换

int(x) float(x) str(x)

将数字类型转换成字符串

**A =str(6)**

**print(type(A))**

1. 字符串
2. 字符串三种定义方式：’ ’ ; “ ” ; “”” ””” ;

如果输出内容想包含引号，用转义字符\ 例如”\’你好”

1. 字符串的拼接：可以将变量与字符串拼接在一起输出，要严格控制输出类型一致

**num=10**

**print(“我有”+str(num)+”元”)**

1. 字符串格式化 占位拼接

%s将内容转换为字符串占位 %d将内容转换为整数占位%f将内容转换为浮点数占位

**Name=”张三”**

**Age=10**

**Message=”我是%s,我今年%d岁” % (Name,Age)**

**print(Message)**

1. 字符串快速格式化 （只占位，不进行精度控制）

**name=”张三”**

**grade=92.5**

**print(f“我是{name},成绩{grade}分”)**

1. 字符串格式化 精度控制 用m.n控制宽度和精度 小数点和空格都算入宽度内

%5d控制宽度为5位 %5.2f控制宽度为5位，保留2位小数

不限制宽度只保留两位小数%.2f

**a=11.1236  
print("数字为%7.3f"%a)**

输出结果：数字为 11.124（其中有一个空格）

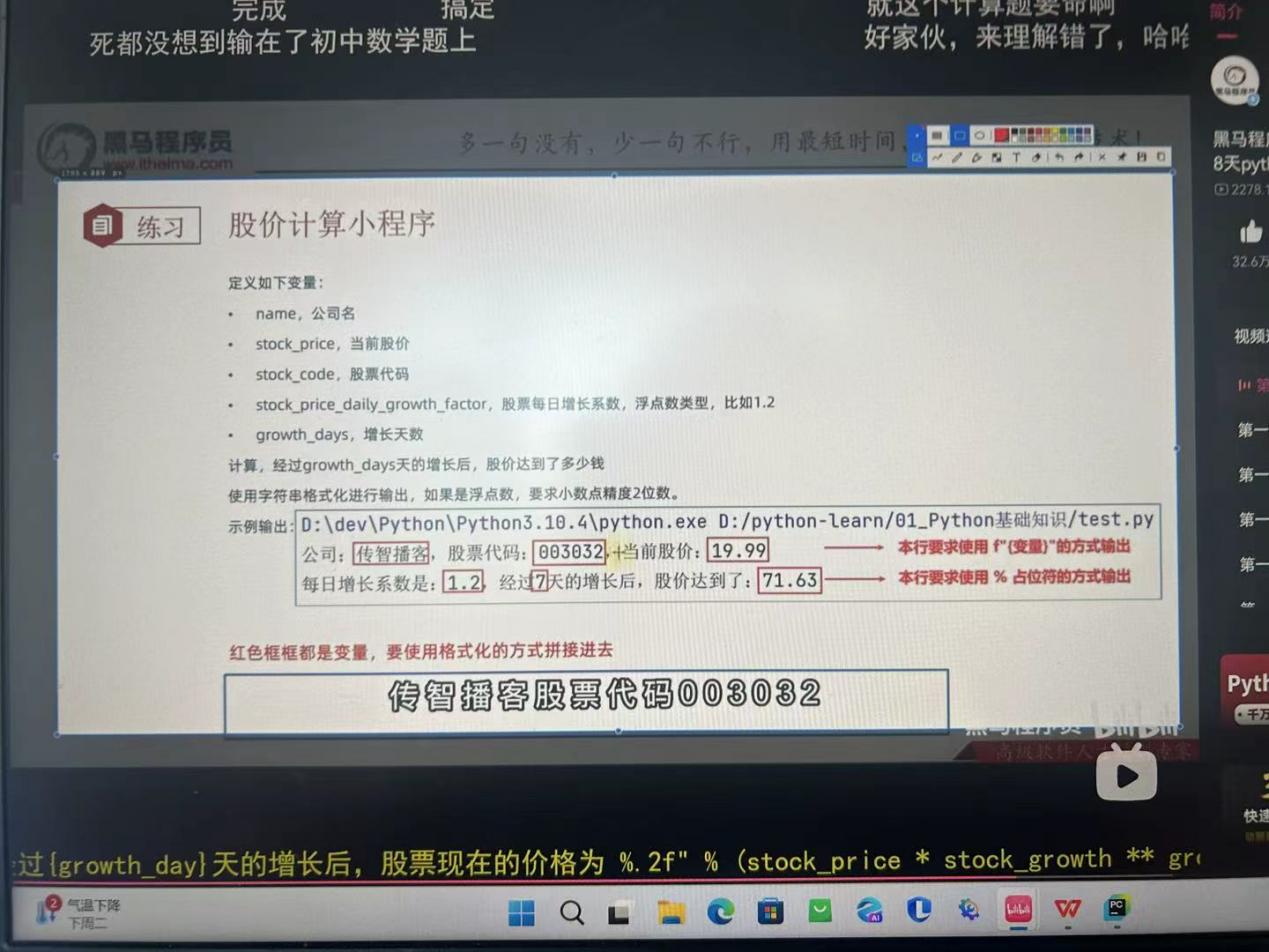
1. 表达式的格式化

**print(“1\*1的结果是:%d”%(1\*1))**

**print(f“1\*1的结果是:{1\*1}”)**

**print(“字符串在python中的类型名是：%s”%type(“字符串”))**

字符串格式化练习



Python代码实现



1. 变量
2. 定义变量： age=20
3. 输出变量内容： print(“年龄为：”,age)
4. 命名规则：由字母、数字、下划线组成，只能由字母下划线开头
5. 注意：只能由字母下划线开头，区分大小写，不能用关键字命名
6. 运算符
7. + - \*正常计算 /除 //取整除 %取余 \*\*指数
8. 赋值运算符：a=2，将等号右边的值或结果赋给左边的变量
9. 复合赋值运算符：a+=1，意思是a=a+1
10. 比较运算符：等于== 不等于!= 大于等于>=
11. input语句

无论输入的是什么类型的数据，得到的都是字符串类型

如果要变为整型就int(input()) 其他的同理

**name=input(“你是：”)**

**print(“我是：%s”%name)**

1. 布尔类型（bool）和比较运算符结合
2. 布尔类型 True是真为1，False是假为0
3. 通过比较运算符计算得到布尔类型结果

**num1=10**

**num2=6**

**print(f”10>=6的结果是：{num1>=num2}”)**

1. if判断语句if和else后要有：
2. if，else条件判断

结果为True时，代码才被执行，

结果为False时，执行else后的，else要和if对齐，且输出语句前要有缩进

**age=20**

**if age>=18 :**

**print(“已成年”)**

**else：**

**print(“未成年”)**

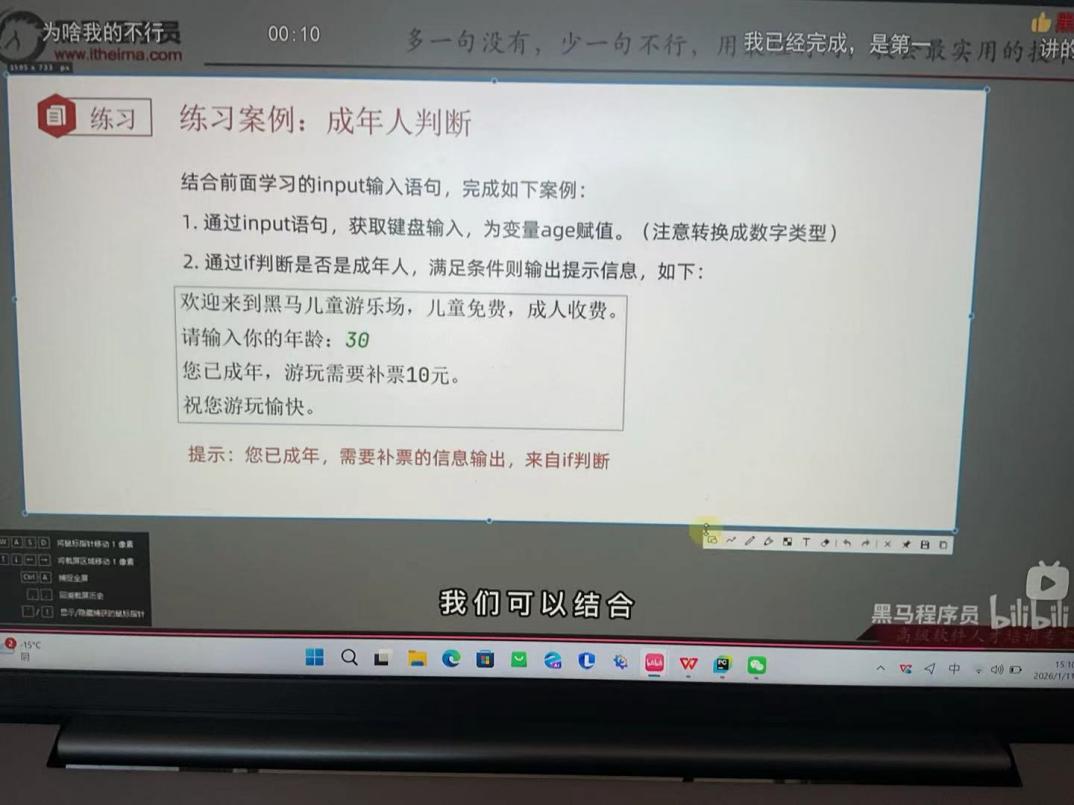
**print(“你好”)**

如果age=20，输出的结果为已成年和你好

如果age=10，结果为你好

if的输出要有缩进，if为True时，缩进后的语句才会被打印出来

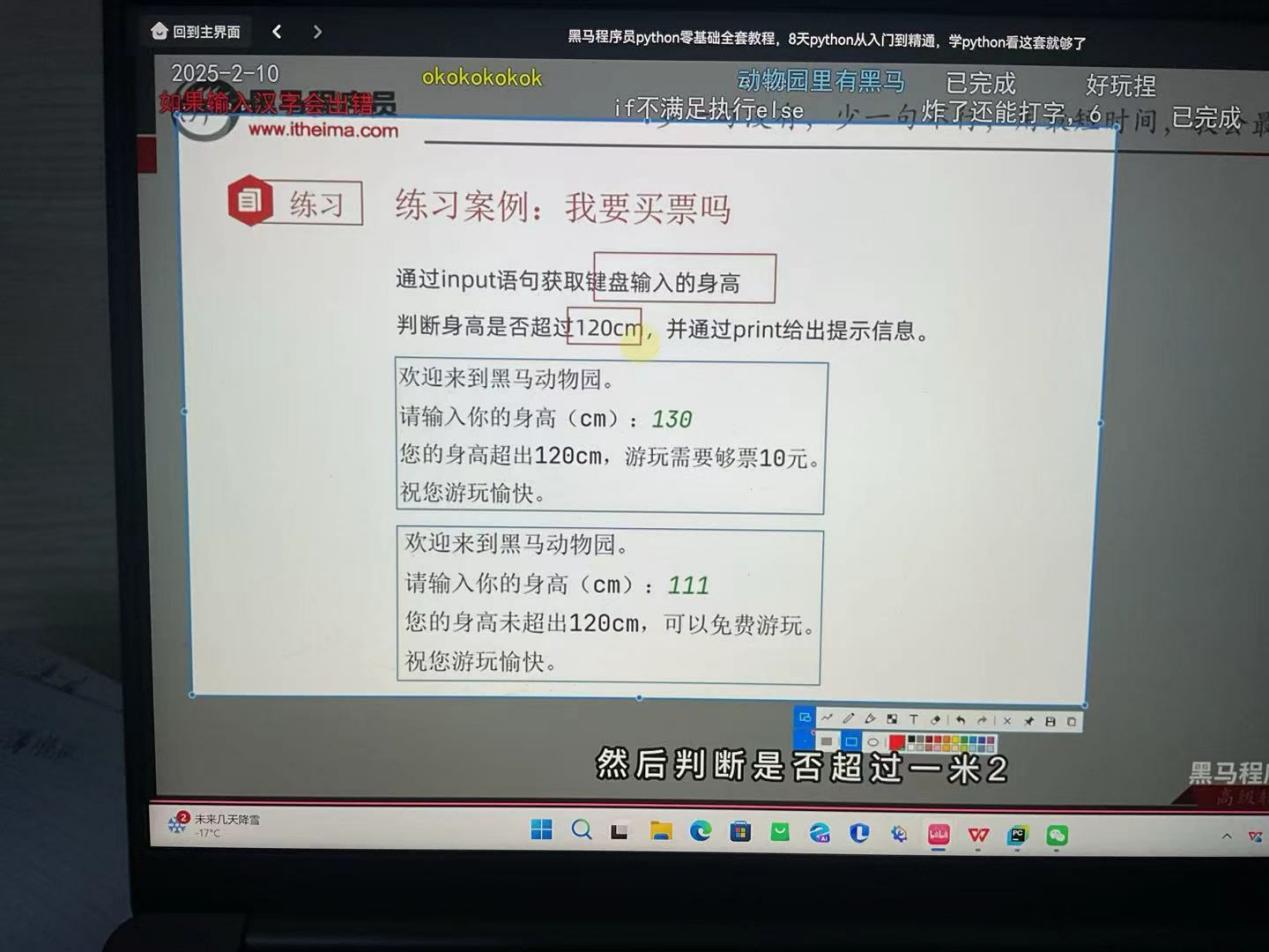
if判断语句练习



Python代码实现



if else判断语句练习



Python代码实现



1. elif加入if，else条件判断 可以写多个elif if，else，elif只有一个能执行

从上到下依次执行，第一个条件不满足就看第二个，第二个不满足就看第三个，

全都不满足就执行else

**age=int(input(“请输入你的年龄(岁)：”))**

**vip\_level=int(input(“请输入vip等级(1~5)：”))**

**if age<18 :**

**print(“未成年,免费游玩”)**

**elif vip\_level >=3:**

**print(“vip3及以上免费游玩”)**

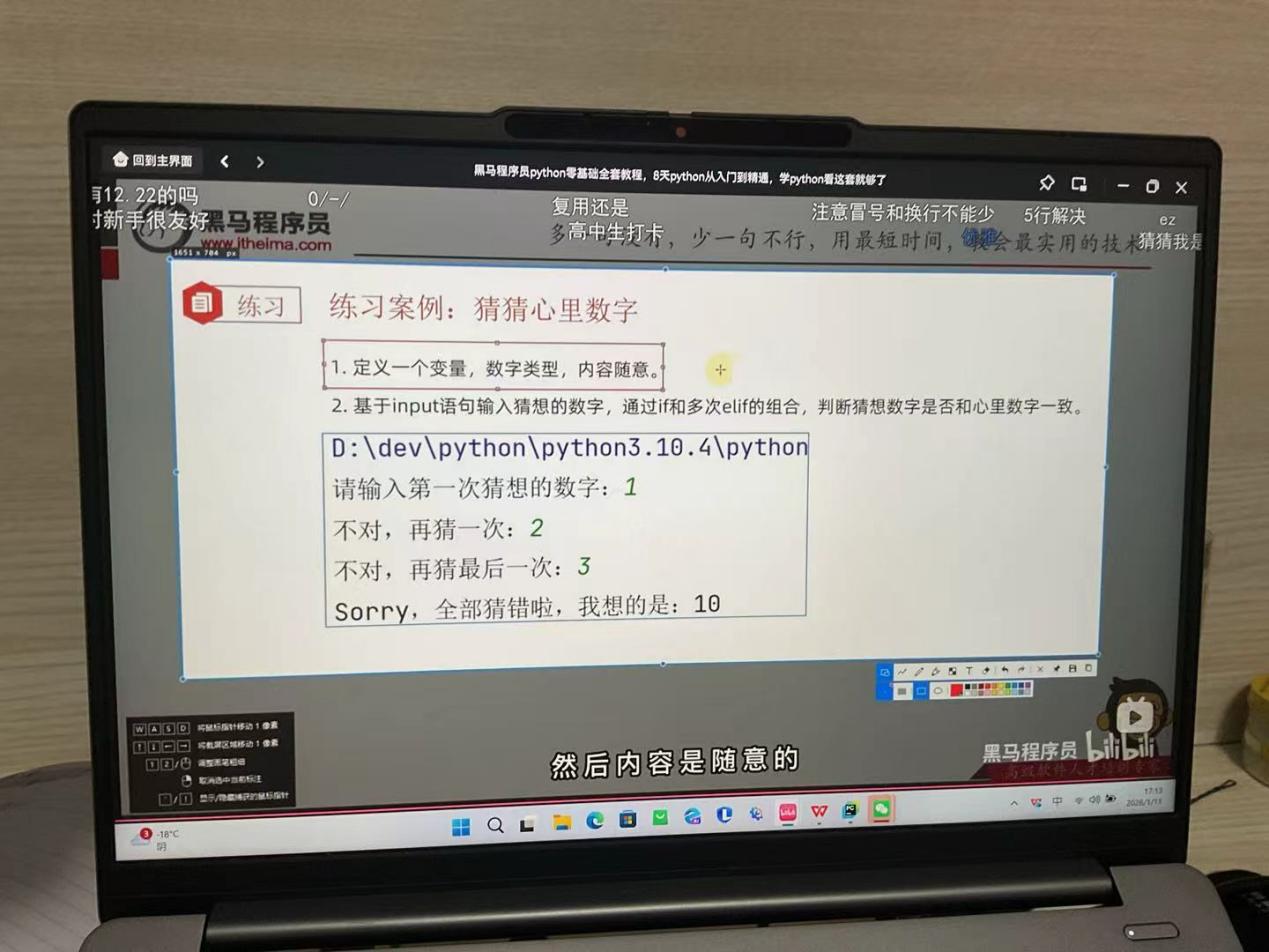
**else：**

**print(“已成年，补票10元”)**

如果age=20，vip\_level=3输出的结果为vip3及以上免费游玩

可以在条件判断中直接写input语句，节省代码量

if elif else练习猜数



Python代码实现



1. 判断语句嵌套 **嵌套的关键点在于空格缩进，通过空格缩进判断层次关系**

if 条件1：

满足条件1，做的事情1

满足条件1，做的事情2

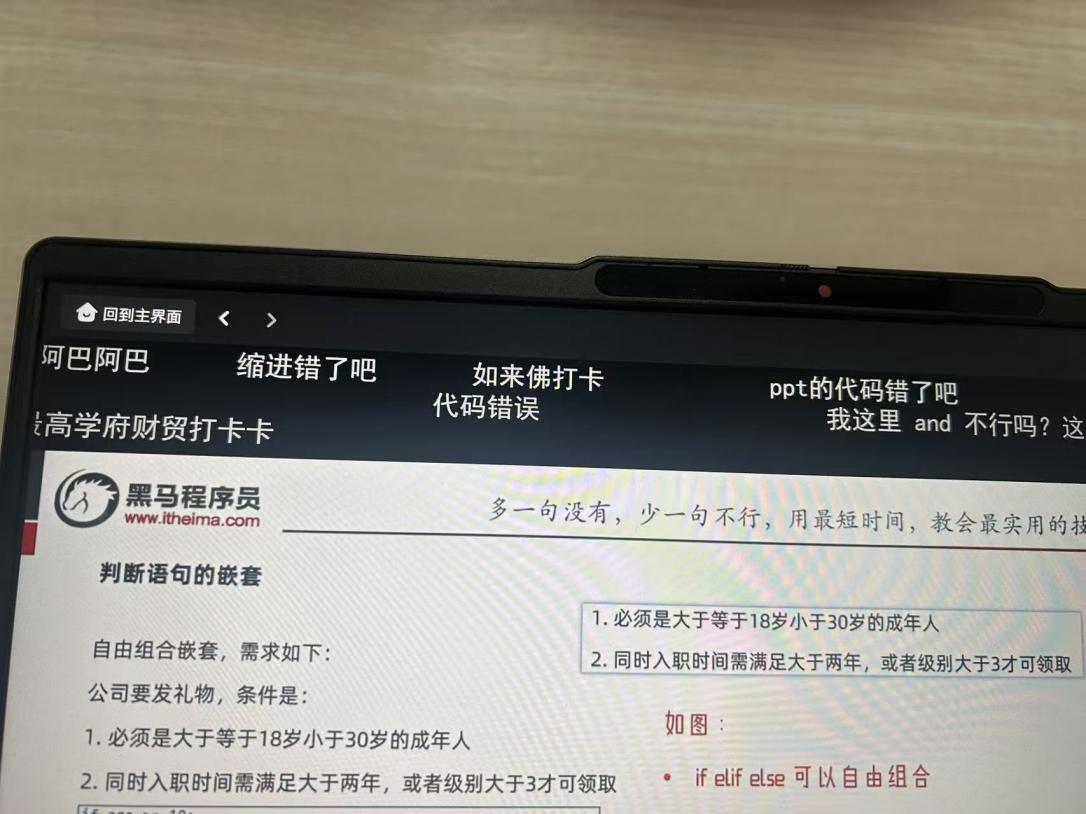
if 条件2：

满足条件2，做的事情1

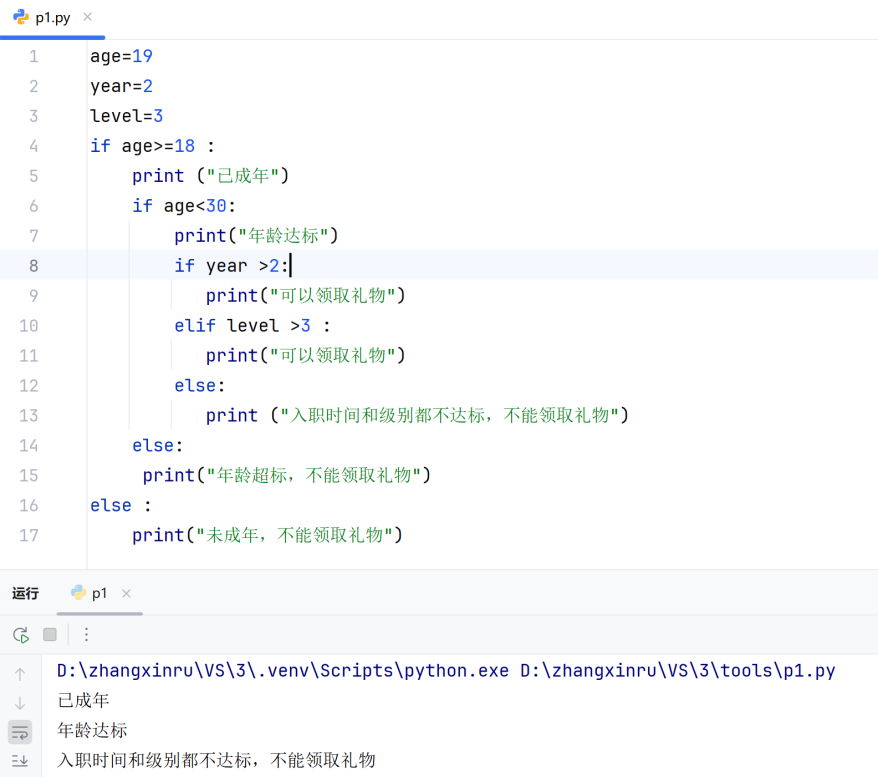
满足条件2，做的事情2

第二个if，属于第一个if内，只有第一个if满足条件，才会执行第二个if

嵌套条件判断



Python代码实现



if判断语句，猜数练习



1. 循环语句
2. While循环(条件必须提供布尔类型结果，True就继续，False就退出循环)

**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**

**...**

1. 用while循环计算从1到100的累加和。

**sum=0  
i=1  
while i <=100 :  
 sum+=i  
 print(sum)  
 i=i+1**

1. 用while循环猜数，范围是1到100

要求：无限次机会，直到猜中为止

每一次猜，会提示猜大了还是小了

猜完数字后，会提示猜了几次

**import random  
num=random.randint(1,100)  
print(f"数字为：{num}")  
*#定义变量记录猜的次数*count=0  
*#通过布尔类型的变量，判断循环是否继续*flag=True  
while flag:  
 guess\_num = int(input("猜数字："))  
 count+=1  
 if guess\_num==num:  
 print("猜对了")  
 *#设置终止循环的条件* flag=False  
 else:  
 if guess\_num > num:  
 print("猜大了")  
 else:  
 print("猜小了")  
print(f"猜了{count}次")**

1. while循环嵌套（注意缩进）

**while 条件1 :**

**条件1满足时，做的事情1**

**条件1满足时，做的事情2**

**条件1满足时，做的事情3**

**while 条件2 :**

**条件2满足时，做的事情1**

**条件2满足时，做的事情2**

**条件2满足时，做的事情3**

打印乘法表



1. for循环（只能从被处理的数据集中，依次处理，不能定义循环条件）

for临时变量 in 待处理的数据集（专业叫法为序列类型） :

循环满足条件时执行的代码

1. 数一数itheima is a brand of itcast中有多少个a

**name="itheima is a brand of itcast"  
count=0  
for x in name:  
 if x=='a':  
 count=count+1  
print(f"{count}个")**

1. for循环嵌套（注意缩进）

for循环和while循环可以相互嵌套使用

用for循环打印乘法表

**for i in range(1,10):  
 for j in range(1,10):  
 if j<=i:  
 print(f"{j} \* {i}= {i\*j}\t",end="")  
 print()**

1. range语句
2. 语法1

**range(num)** #获取一个从0开始，到num结束的数字序列（不包含num本身）

如**range(5)取得的数据是：[0,1,2,3,4]**

1. 语法2

**range(num1,num2)** #获得一个从num1开始，到num2结束数字序列（不含num2本身）

如 **range(5，10) 结果是：[5,6,7,8,9]**

1. 语法3

**range(num1,num2,step)**#获得一个从num1开始，到num2结束的数字序列（不含num2）

数字之间的步长，以step为准，step默认为1

如 **range(5,10,2) 结果是：[5,7,9]**

1. 练习，求从1到100（不含100）共有多少个偶数

要求使用range语句，获取从1到num的序列，用for循环遍历

**count=0  
for x in range(1,100):  
 if x%2==0:  
 count+=1  
print(f"有{count}个偶数")**

1. continue和break（可以用在while和for循环）
2. continue 中断本次循环，直接进入下一次循环

**for i in range(1,100) 遇到continue结束本次循环，进行下一次，**

**语句1 不会执行语句2**

**continue**

**语句2**

1. break 中断循环，直接跳到外循环

**for i in range(1,100)**

**语句1 遇到break，结束循环体，不再执行语句2，**

**语句2**

**break 直接执行语句3**

**语句3**

1. 某公司，账户余额有1W元，给20名员工发工资。

员工编号从1到20，从编号1开始，依次领取工资，每人可领取1000元

领工资时，财务判断员工的绩效分（1-10）（随机生成），如果低于5，不发工资，换下一位，如果工资发完了，结束发工资。

**money=10000  
 import random  
 score = random.randint(1, 10)  
 if score<5:  
 print(f"员工{i}，绩效分{score}，低于5，不发工资，下一位")  
 continue  
 if money >=1000:  
 money-=1000  
 print(f"向员工{i}，发放工资1000元，账户余额还剩余{money}元")  
 else:  
 print(f"余额不足，当前余额{money}元")  
 break**

1. 函数

函数概念：是组织好的，可重复使用的，用来实现特定功能的代码段。

函数定义：

**def 函数名（传入参数）：**

**函数体**

**return 返回值**

注意：参数之间用逗号进行分隔，参数数量不限，形参和实参要一一对应。

先定义函数，后调用函数。 参数和返回值可以省略

1. 函数传入参数

def add(x,y):

result=x+y

print(f“{x}+{y}的结果是：{result}”)

#调用函数

add(1,2)

函数定义中的参数x，y为形参（形式参数），

函数调用中为实参（实际参数）为函数执行时真正使用的参数值

练习：体温判断

**def wen\_du(x):  
 if x<=37.5:  
 print(f"体温是：{x},体温正常")  
 else:  
 print(f"体温是：{x},需要隔离")  
 wen\_du(38.5)**

1. 函数返回值

返回值定义：程序中函数完成事情后，最后给调用者的结果。

**def 函数名（传入参数）：**

**函数体**

**return 返回值**

**变量=函数（参数）**

1. 函数返回值None类型
2. 如果函数没有使用return语句返回数据，函数也有返回值。None(空值)
3. 函数的说明文档

**def func(x,y):**

**“””**

**函数说明：**

**:param x:形参x的说明**

**:param y:形参y的说明**

**:return :返回值的说明**

**“””**

**函数体**

**return 返回值**

1. 函数嵌套调用

一个函数里又调用了另一个函数

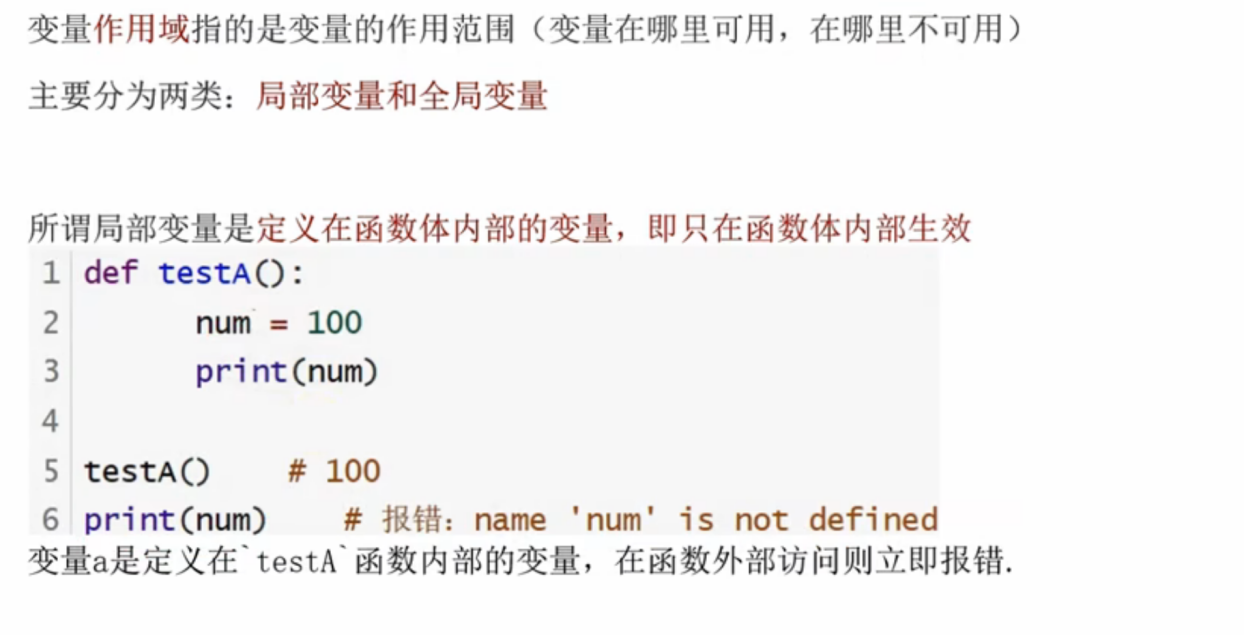
1. **def func\_b():  
    print("---2---")  
   def func\_a():  
    print("---1---")  
    func\_b()  
    print("---3---")  
   func\_a()**

结果为---1---

---2---

---3---

1. 调用顺序：先执行a，在执行调用的b，等b全部执行完，再去执行a剩余的部分
2. 作用域
3. 局部变量：定义在函数体内部的变量，只在函数体内部生效



1. 全局变量：函数体内和函数体外都能生效

***#定义全局变量a*num=100  
def testA():  
 print(num) *#访问全局变量num，并打印*def testB():  
 print(num)*#访问全局变量num，并打印*testA()  
testB()**

1. global关键字

global关键字，使函数内定义的变量声明为全局变量

**num=100  
def testA():  
 print(num) *#访问全局变量num，并打印*def testB():**

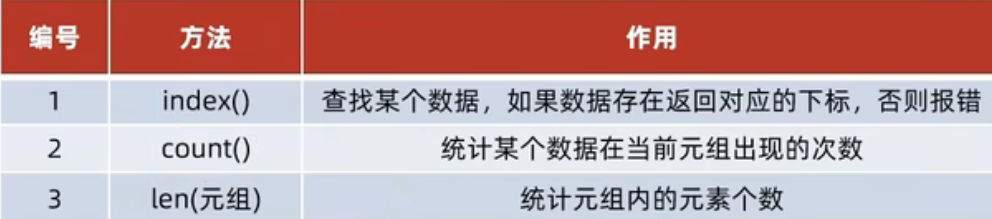
**global num**

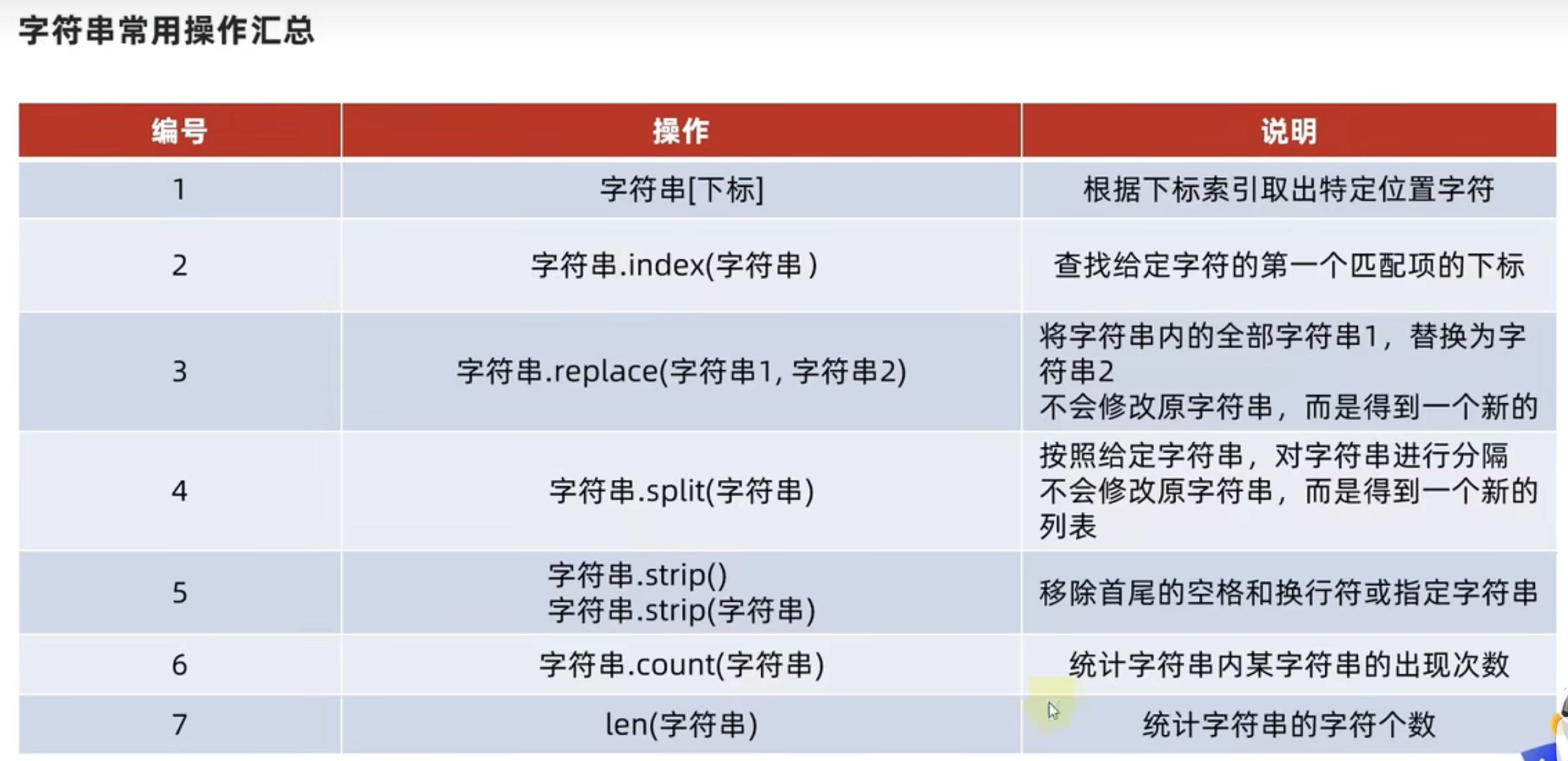
**num=200  
 print(num)*#访问全局变量num，并打印*testA()  
testB()**

1. 数据容器

数据容器的五个类型：列表[ ]、元组( )、字符串” ”、字典{ }、集合{ }











1. 列表（list）

列表中的元素类型可以不同，列表可以被修改。可嵌套使用

1. 语法

**#字面量**

**[元素1，元素2，元素3，元素4，...]**

**#定义变量**

**变量名称=[元素1，元素2，元素3，元素4，...]**

**#定义空列表**

**变量名称 = [ ]**

**变量名称 = list( )**

例：

my\_list=[“my”,666,True]

print(my\_list) #打印列表

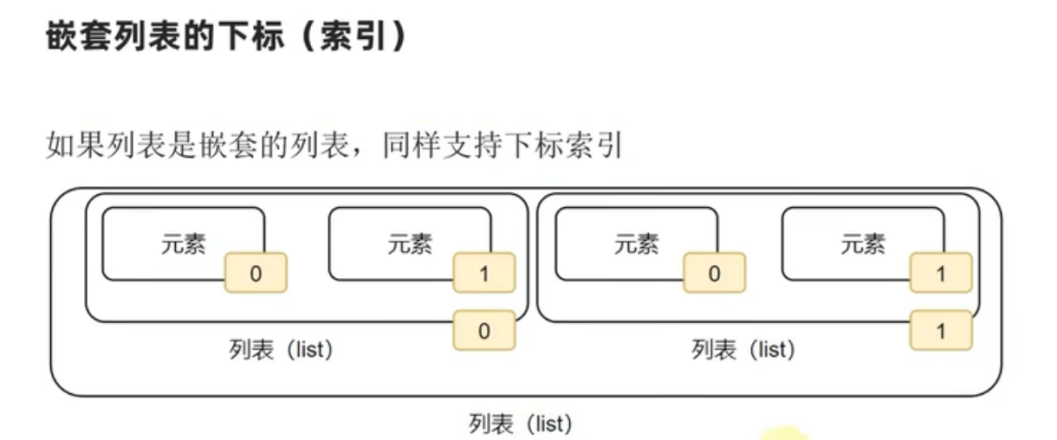
print(type(my\_list)) #打印列表类型

1. 列表的下标索引
2. **语法：(正向：第一个元素为0，从左向右递增)，**

**(反向：最后一个元素为-1，从右向左递减)**

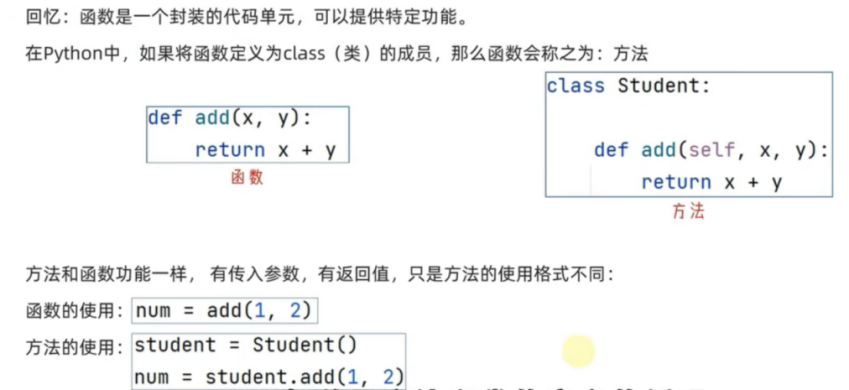
name\_list=["tom","lily","mike"]  
print(name\_list[0])  
print(name\_list[1])  
print(name\_list[2])

1. 嵌套列表的下标索引



**list2=[(1,2,3),(4,5)]  
*#取出元素5*num=list2[1][1]  
print(f"取出的元素为{num}")**

1. List列表的常用操作



1. 查找某元素在列表内的下标索引

**语法： 列表.index(元素)**

name\_list=["tom","lily","mike"]

index=name\_list.index(“tom”)

print(f"下标索引的值是：{index}")

1. 修改特定下标索引的元素的值

**语法： 列表[下标]=值**

name\_list=["tom","lily","mike"]  
name\_list[0]=5  
print(name\_list)

1. 在特定索引之间插入新的元素

**语法： 列表.insert(下标，元素) 在指定的下标位置，插入指定的元素**

name\_list=["tom","lily","mike"]  
name\_list.insert(1,"rose")  
print(name\_list)

1. 追加元素

**语法1： 列表.append(元素) 将指定的单个元素，追加到列表的尾部**

name\_list=["tom","lily","mike"]  
name\_list.append("rose")  
print(name\_list)

**语法2： 列表.extend(一些元素/列表) 将指定的一些元素（列表），追加到列表的尾部**

name\_list=["tom","lily","mike"]  
name\_list.extend([1,2,3])  
print(name\_list)

**或者**

name\_list=["tom","lily","mike"]  
list2=[1,2,3]  
name\_list.extend(list2)  
print(name\_list)

1. 删除元素

**语法1：del 列表[下标] 只能进行删除**

name\_list=["tom","lily","mike"]  
del name\_list[2]  
print(name\_list)

**语法2：列表.pop(下标) 可以通过返回值得到删去的元素**

name\_list=["tom","lily","mike"]  
element=name\_list.pop(2)  
print(f"删除后的列表：{name\_list},删去的元素：{element}")

**语法3： 列表.remove(元素) 删除重复元素在列表中的第一个匹配项**

name\_list=["tom","lily","tom","mike"]  
name\_list.remove("tom")  
print(f"删除后的列表：{name\_list}")

1. 清空列表

**语法：列表.clear( )**

1. 统计列表内某元素的数量

**语法：列表名.count(元素)**

1. 统计列表一共有多少个元素

**语法：len(列表)**

1. 列表的遍历

while循环可以自定义循环条件，并自行控制。

for循环不能自定义循环条件，只可以从容器内取出数据。

1. while循环

循环控制变量通过索引来控制，默认为0

每一次循环将下标索引变量+1

循环条件：下标索引变量<列表的元素数量

**name\_list=["tom","你好","mike"]**

**#定义一个变量，用来标记列表的下标  
index=0  
while index < len(name\_list):**

**#通过index变量取出对应下标的元素  
 element=name\_list[index]  
 print(f"列表的元素：{element}")  
 index += 1**

1. for循环

**list=[1,2,3,4,5]  
index=0  
for element in list:  
 print(f"列表的元素：{element}")**

练习：取出偶数，并放入一个新的列表

for循环

**list=[1,2,3,4,5,6,7,8,9,10]  
new\_list=[]  
for element in list:  
 if element%2==0:  
 oushu=element  
 print(f"偶数为：{oushu}")  
 new\_list.append(oushu)  
print(new\_list)**

练习：取出偶数，并放入一个新的列表

while循环

**list2=[1,2,3,4,5,6,7,8,9,10]  
new\_list=[]  
index=0  
while index < len(list2):  
 element2=list2[index]  
 if element2%2==0:  
 oushu2=element2  
 print(f"偶数为：{oushu2}")  
 new\_list.append(oushu2)  
 index += 1  
print(new\_list)**

1. 元组（tuple）

元组中元素的数据类型可以不同，元组不可以被修改，可嵌套使用

定义语法：

**#定义元组字面量**

**(元素，元素，元素，...，元素)**

**#定义元组变量**

**变量名称=(元素，元素，元素，...，元素)**

**#定义空元组**

**变量名称=( ) #方式1**

**变量名称=tuple( ) #方式2**

注意：定义单个元素的元素，后面要跟一个逗号和空 t1=(“hello”, )

1. 元组的操作

元组的操作和列表相同，只是不能修改、增加、删除。

元组中嵌套列表，可以修改列表，但是不能修改元组的元素

循环遍历也可以

1. 字符串（str）

定义语法：str=”字符串1，字符串2”

只读，无法修改，查找和元组中一样

1. **字符串的替换**（前面的替换成后面的，并不是修改了字符串本身，而是得到了一个新的）

语法：字符串.replace(字符串1，字符串2)

str="nihao hello"  
new\_str=str.replace("nihao","hi")  
print(f"将字符串{str}，替换成{new\_str}")

1. **字符串的分割** （按指定的字符串，将字符串分割，并存入列表中）（字符串本身不变）

语法：字符串.split(分隔符字符串)

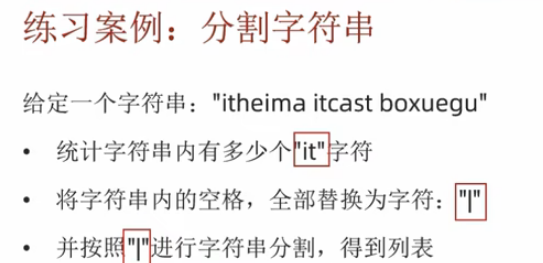
str="nihao hello"  
list=str.split(" ")  
print(f"将字符串{str}，替换成{list}")

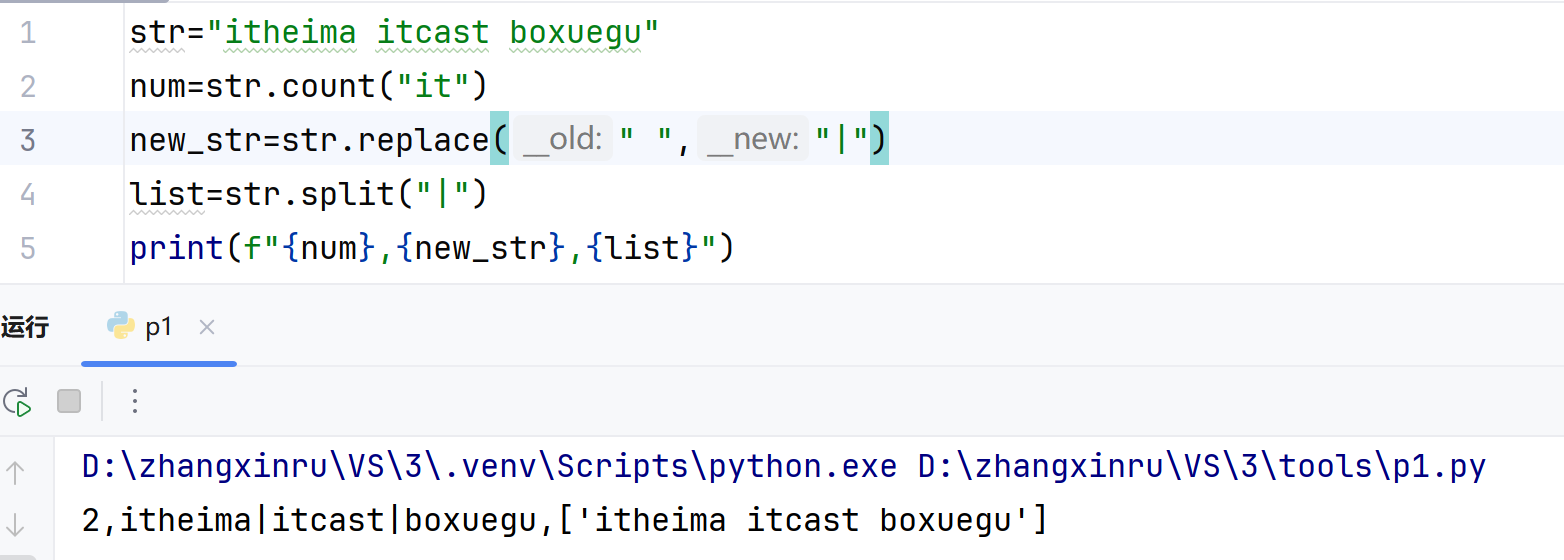
1. 字符串规整操作（去前后空格）

语法：字符串.strip( )

1. 字符串规整操作（去前后字符串）

语法：字符串.strip(字符串)





1. 字典（dict）

Key不可以重复，后面的会把前面覆盖，没有下标索引，

通过字典的key可以找到对应的value，字典遍历只能用for循环，不能用while循环

key不可以是字典，value可以是字典可以进行嵌套。

定义语法：

#定义字典字面量

{key：value，key：value，...，key：value}

#定义字典变量

my\_dict={key:value,key:value,...}

#定义空字典

my\_dict={ } #空字典定义方式1

my\_dict=dict{ } #空字典定义方式2



1. **字典的查询**

dict={"张三":99,"李四":98}

score=dict["张三"] print(f"成绩为：{score}")

1. **字典的嵌套**

dict={"张三": {"数学":99,"语文":96},  
 "李四":{"数学": 95,"语文": 93}}  
score=dict["张三"]  
print(f"成绩为：{score}")

1. **新增元素** 语法：字典[key]=value
2. **更新元素**

语法：字典[key]=value

因为字典的key不能重复，所以新增同一个key值即为更新

1. **删除元素**

字典名.pop(key)

1. **清空元素** 字典名.clear( )
2. **获取全部的key**

语法：字典.keys( )  
keys=dict.keys()

print(f"{keys}")

1. **字典的遍历**

方式一：for key in keys:

print(f”字典的key是：{key},字典的value是：{value}”)

方式二：for key in 字典名：

1. **字典的元素数量**

num=len(字典名)

1. 集合 (set)

元素不能重复，不能用下标索引，打印出的顺序不能保证，可以修改元素

定义语法：

**#定义集合字面量**

**{元素，元素，....，元素}**

**#定义集合变量**

**变量名称={元素，元素，...，元素}**

**#定义空集合**

**变量名称=set( )**

1. **添加元素**

语法：集合.add(元素)

1. **移除元素**

语法：集合.remove(元素)

1. **从集合中随机取出元素**

语法：集合.pop( )

1. **清空集合** 语法：集合.clear( )
2. **取两个集合的差集**

语法：集合1.difference (集合2) (功能：取出集合1有，集合2没有的)

1. **消除两个元素的差集**

语法：集合1.difference\_update (集合2) (结果：集合1被修改，集合2不变)

1. **两个集合合并**

语法：集合1.union(集合2)

1. **统计集合中元素的数量**

len(集合)

语法：集合1.union(集合2) (结果：得到新的集合，集合1和集合2不变)

1. 切片
2. 序列的概念：内容连续，有序，可使用下标索引的一类容器。

列表，元组，字符串都可以视为序列

1. 切片：从一个序列中取出子序列 （不会影响序列本身，而是得到一个新的序列）
2. 语法：序列[起始下标：结束下标：步长]

（包含起始下标，不包含结束下标） 步长默认为1可以不写

步长： 步长为1，一个一个取，步长为2，跳一个取，步长为N，跳N-1个取

步长为负数，反向取，注意，起始下标和结束下标，起始下标反向的

比如 要反向取ami

str="itheima"  
result=str[6:3:-1]  
print(f"{result}")

反转字符串str=”01234567”

result=str[: : -1]

print(“{result}”)

