山脚课程学习笔记

第0关 print()函数与变量–千与千寻的名字

课程目标

1. 了解python：用途、组成
2. 熟练使用函数：print()
3. 掌握换行的三种方法
4. 了解变量的意义，熟练给变量赋值
5. 认识转义字符，了解转义字符的使用方法

课程难点

1. print()函数以及引号的使用
2. 换行的两种简便方法
3. 区分赋值与等于

课程重要内容重现

一、最基本的一条代码

1. print()函数 print()函数由两部分构成：

- 指令：print；
- 指令的执行对象：在print后面的括号里的内容

2. 引号的用法

单引号和双引号都可以使用，但需要匹配，并且配合使用可以区分开原文和print()函数的结构。例如，print(“Let’s go”)，双引号的作用是函数结构，单引号是英文语法。

不用引号时，括号内必须是数字或者数字运算，这是计算机可以理解的内容。例如：print(1+1)，最后输出是2。

注意：python中所有的符号都是英文状态下的，并且会区分大小写。

二、换行

1. 重复使用print()函数，将不同行的语句放在不同的函数中输出。（事倍功半）
2. 使用三引号：用三引号将需要分行的内容括起来，并且在引号内使用回车进行段落排版。

例如：

```
print('''我愿意留在汤婆婆的澡堂里工作两年，
                第一年在锅炉房和锅炉爷爷一起烧锅炉水，
                第二年在澡堂给客人搓背，
```

```
                如果我违背工作内容的话，
                将在这个世界变成一头猪。''')
```

3. 使用\n，此时是不允许回车换行的！例如：

```
print('我愿意留在汤婆婆的澡堂里工作两年,\n第一年在锅炉房和锅炉爷爷一起烧锅炉水,\n第二年在澡堂给客人搓背,\n如果我违背工作内容的话,\n将在这个世界变成一头猪。')
```

三、转义字符

对于可作为结构性符号的，例如单引号，感叹号，若想直接使用，可在符号前加一个反斜线\。则对于之前的例子：print(“Let’s go”)，也可以写作print(‘Let\’s go’),中间的单引号由于使用了反斜线，所以作为整条语句的内容而不是print()函数的结构。

四、变量与赋值

1. 变量是我们自己创建的，命名要求：
2. 只能是一个词
3. 只能包含字母、数字、下划线（下划线可以用于连接多词）
4. 不能以数字开头
5. 尽量描述包含的数据内容（抽象概括存储的内容）

代码中的=（等号）是用于赋值而逻辑上的等于要使用两个等号，即1+1==2。

变量的特点：保存的数据是可以随意变化的，储存的值永远都是最新的那个。例如：

```
name='魔法少女千酱'
name='夏目千千'
name='千寻'
print(name)
```

这段代码输出是结果是‘千寻’

第1关 数据类型与转换–萌新的进化

课程目标

1. 熟练掌握三种数据类型及其转换
2. 掌握数据拼接的方法
3. 了解查询数据类型的type()函数

课程难点

1. 区分字符串下的数字与整数、浮点数下的数字
2. 使用【+】进行数据拼接时，连接的数据类型必须为同数据类型
3. 使用函数进行数据类型的转换时，int()与float()函数括号内的数据必须为纯数字型文本

课程重要内容重现

一、常见的三种数据类型

1、字符串

特点：被引号括起来的文本。（注意引号要使用英文状态下的单引号或者双引号、三引号）

```
例：
slogan = '命运！不配做我的对手！'
attack = "308"
gold = "48g"
blood = '''+101'''
achieve = "First Blood!"
```

先将内容以字符串形式赋值给变量，最后使用print()函数输出变量即可。

2、整数

整数英文为integer，简写做int。是正整数、负整数和零的统称，是没有小数点的数字。

特点：无需配合引号使用，可进行计算。如：108（整数）‘108’（字符串）‘6小灵童’（字符串）但若存在文字类数据，则必须使用引号，将其变为字符串类型。例：

```
print(6小灵童)
print(6skr)
#打印数据

SyntaxError: invalid syntax
#终端显示结果：报错：无效语法
```

具体的计算符号：（优先级与日常算数一致）

3、浮点数

相对于整数而言，浮点数就是带小数点的数字。英文名是float，与整数int()和字符串str()不同，浮点数没有简写。

二、查询数据类型——type()函数

作用：查询数据类型

例：print(type('查询内容'))

```
achieve = 'Penta Kill'
print(type(achieve))

#结果显示：<class 'str'>
```

三、数据拼接

利用数据拼接符号【+】，将需要拼接的变量连在一起。注意：变量内的数据类型必须为字符串型才可进行拼接！如：

```
hero = '亚瑟'
enemy = '敌方'
action = '团灭'
gain = '获得'
achieve = 'ACE称号'
#结果显示为
#亚瑟团灭敌方获得ACE称号


print(hero+action+enemy+gain+achieve)
hero = '亚瑟'
enemy = '敌方'
action = '秒杀'
gain = '获得'
number = 5
achieve = 'Penta Kill'

print(hero+action+number+enemy+gain+achieve)

#结果显示报错：TypeError：can only concatenate str (not "int") to str
#类型错误：只能将字符串与字符串拼接
```

四、数据类型转换

1、转换为字符串类型

str()函数能将数据转换成其字符串类型。只要将所需数据放到括号里，这个数据就能成为字符串类型。

用引号将数据括起来也能达到同样结果。

例如：

```
hero = '亚瑟'
enemy = '敌方'
action = '秒杀'
```

```
gain = '获得'
number = 5
achieve = 'Penta Kill'

print(hero+action+str(number)+enemy+gain+achieve)
print(hero+action+'5'+enemy+gain+achieve)
#使用str()函数将变量number里的数字5变成了字符串5。
```

2、转换为整数

int()函数的使用，与str()类似。注意一点：只有符合整数规范的字符串类数据，才能被int()强制转换。

```
print(int('3.8'))

#运行后显示结果：ValueError: invalid literal for int() with base 10: '3.8'
```

小数型字符串会直接报错，而浮点数会被强制转换：

```
print(int(3.8))

#运行后结果显示：3
```

也就是说，对于浮点数，int()会保留其整数部分。注意：不是四舍五入！

3、转换为浮点数

float()函数的使用与int()、str()类似。如果括号里面的数据是字符串类型，那这个数据一定得是数字形式。

第2关 条件判断与条件嵌套-灭霸的选择

课程目标

1. 熟练掌握if条件判断语句，包括单向、双向以及多向判断。
2. 熟悉语句间的级别关系，不同级别之间要有缩进
3. 熟悉计算机执行指令的顺序，能看懂并得出代码输出结果以及写出if嵌套语句

课程难点

1. 逻辑判断应使用逻辑符号等于【==】
2. 条件语句后一定要记得接冒号【:】，注意观察冒号之后的语句是否缩进以及同级别的语句是否在格式上处于并列状态。
3. 在执行变量的判断之前，要注意变量是否已经被赋值

课程重要内容重现

一、条件判断

即中文逻辑语句“如果...就...”。在进行判断之前，一定要先对变量进行赋值！条件判断就是针对不同的可能性，进行不同操作。赋值情况的前提不满足if的条件时，自动跳过，执行下一行命令。

其次，每一个判断语句之后要使用冒号【:】，表示接下来的内容是只有满足条件才运行的。若不是条件下的语句，要记得删除缩进。

1、单向判断

要是if之后的条件不满足，就跳过if语句进行下一命令。格式：

if xxx(判断的条件):

    如果满足上述条件，就执行的操作语句

2、双向判断

要是if之后的条件不满足，就执行else里的。if与else平级（缩进一致，在else前必须有一个平级的前提）。每一个条件不能有重合部分，是互斥的，格式：

if xxx(判断的条件):

    如果满足上述条件，就执行的操作语句

else:

    如果不满足if之后的语句，就执行的操作语句

```
weight=101
#要先为酱酱的体重赋值，酱酱的体重是101斤

if weight>100:
#如果体重超过100斤的条件下，就……(条件后需加冒号)
  print('不吃了')
  #就打印结果：不吃了！(注意检查是否自动缩进)

else:
#如果体重没有超过100斤的条件，就……(else条件前无缩进，条件后需加冒号)
  print('放心吃吧')
  #就打印：放心吃吧(注意检查是否自动缩进)
```

3、多向判断

if、elif和else平级。可以存在多个elif，数量根据整体能分成的所需选项数来定。注意：每一个条件不能有重合部分，是互斥的，即x<10与9<x<15，这样的两个条件是不可行的。如果不满足if的条件，就判断是否满足elif下的条件，若所有elif的条件都不满足，就执行else下的语句。并且elif之后可以不接else，格式：

if xxx(判断的条件):

    如果满足上述条件，执行的操作语句

elif xxx(与前一个if互斥的另一个条件):

    如果满足elif后的条件，就需要执行的语句

else:

    若if、elif后面的条件都不满足，则会执行的语句

```
stonenumber=1
#一定要先为宝石数量赋值

if stonenumber>=6: #注意冒号
    #条件：如果你拥有的宝石数量大于等于6个
    print('你拥有了毁灭宇宙的力量') #注意缩进

elif 3<stonenumber<=5:
    #条件：如果宝石数量在4至5个
    print('红女巫需要亲手毁掉幻视额头上的心灵宝石')

else:
```

```
    #条件：当赋值不满足if和elif条件时，执行else下的命令，即宝石数量在3个以下
    print('需要惊奇队长逆转未来')
```

二、if嵌套

在基础条件满足的情况下，再在基础条件底下增加额外的条件判断。在编写if嵌套语句时，同样的，可以按照框架，从大到小，依次往不同的大条件中补充额外条件。

```
historyscore=26
if historyscore>=60:
    print('你已经及格')
    if historyscore>=80:
        print('你很优秀')
    else:
        print('你只是一般般')
else:
    print('不及格')
    if historyscore<30:
        print('学渣')
    else:
        print('还能抢救一下')
print('程序结束'）
#结果显示为 ：
#不及格
#学渣
#程序结束

```

每一个级别下的条件都只能执行一个！（互斥）elif与if类似。

第3关 input()函数–霍格沃茨来信

课程目标

1. 理解输入函数input()的意义

2. 熟练掌握input()函数返回的数据类型，以及结果赋值、数据类型转换的方法

课程难点

1. input()函数括号内的内容会被输出，但需要输入对应数据才能继续执行之后代码
2. input()函数的结果必须赋值给变量，且数据类型为字符串型

课程重要内容重现

input()函数

1、定义

input()函数是输入函数，与print()函数类似，input()函数括号里面的内容是会显示出来的，但不同在于我们需要输入对应的内容，回车后才能继续运行。

2、input()函数赋值

在括号内用引号括起提示语，例：

```
input('请铲屎官输入宠物的名字：')
#运用input函数搜集信息
```

输入的内容被储存在计算机内，需要将结果赋值给变量。例：

```
print（'那么，您的选择是什么？"1"接受，还是"2"放弃呢？'）
choice = input('请输入您的选择：')
#变量赋值
if choice == '1':
    print('霍格沃茨欢迎你的到来')
else:
    print('您可是被梅林选中的孩子，我们不接受这个选项。)
```

3、input()函数的数据类型

对于input()函数来说，不管输入的是整数1234，还是字符串‘我爱摩卡’，input()函数的输入值（搜集到的回答），永远会被强制性地转换为字符串类型。（Python3固定规则）所以，不管我们在终端区域输入什么，input()函数的返回值一定是字符串，将结果赋值给变量后，变量的数据类型也一定是字符串。

4、input()函数的数据类型转换

使用数据类型转换函数，int()，float()可以从源头强制转换为对应类型。但是要注意，此时的input()函数返回值一定要是纯数字型！例：

```
money = int(input('你一个月工资多少钱？'))
#将输入的工资数（字符串），强制转换为整数

if money >= 10000:
    #当工资数（整数）大于等于10000（整数）时
    print('土豪我们做朋友吧！')
    #打印if条件下的结果

else:
    #当工资数（整数）小于等于10000（整数）时
```

```
    print('我负责赚钱养家，你负责貌美如花～')
    #打印else条件下
```

注：输入值会运用到计算时，千万记得用int()转换！

第4关 列表和字典–收纳的艺术

课程目标

1. 熟练掌握列表、字典中元素的增删改查
2. 理解列表和字典的区别

课程难点

1. 列表与字典增删改查的异同
2. 正确使用切片，深刻理解切片时冒号左右数字的意义

课程重要内容重现

一、列表

1. 代码格式

students是列表名，数据存储在中括号[]里，用逗号隔开并使用等号赋值给列表。中括号里面的每一个数据叫作“元素”。

列表中的元素是有自己明确的“位置”的，元素相同，在列表中排列顺序不同，就是两个不同的列表。

列表中字符串、整数、浮点数都可以存储。

```
list = ['小明',17,5.2]
```

1. 提取元素

1）、偏移量：元素在列表中的编号。

- 偏移量是从0开始的；
- 列表名后加带偏移量的中括号，就能取到相应位置的元素。

2）、切片：用冒号来截取列表元素的操作。

- 冒号左边空（或者为0），：m，表示从头取m个元素；
- 右边空（或者为0），n：，跳过前n个元素把剩下的取完；
- 冒号左右都有数字时，n：m，表示跳过前n个元素，取到第m个。（取出前m个元素中除了前n个后剩下的那些）

切片截取了列表的一部分，所以得到的结果仍然是一个列表。（即使只有一个元素，也是在列表里的，与用偏移量取单个元素区别开）

```
students = ['小明','小红','小刚']
print(students[2])
#使用偏移量提取单一元素，结果显示:
```

```
#小刚
print(students[2:])
#使用切片，即使结果仍然只有一个元素，但显示为列表：
#['小刚']
```

3）特别地，a,b,c=students，也可以提取出列表中的元素，变量分别用逗号隔开，且变量的数量与列表元素数一致，最终列表元素会分别赋值给变量，例如：

```
appetizer = ['话梅花生','拍黄瓜','凉拌三丝']
a,b,c=appetizer

print(a)
print(b)
print(c)
print(a,b,c)
#结果显示为
#话梅花生
#拍黄瓜
#凉拌三丝
#话梅花生 拍黄瓜 凉拌三丝
```

1. 增加/删除元素

1）增加元素

列表名.append()。注意这里是.不是空格！

append后的括号里只能接受一个参数，结果是让列表末尾新增一个元素。列表长度可变，理论容量无限，所以支持任意的嵌套。

```
list3 = [1, 2]
list3.append(3)
print(list3)
#添加‘3’这个元素
#结果显示：
#[1,2,3]

list3.append(4, 5)
list3.append([4, 5])
print(list3)
#添加‘[4, 5]’这个列表，也就是append()的参数为一个列表，也是一个参数，所以不会报错
#结果显示：
#[1, 2, 3, [4, 5]]
```

但是append(4,5)会报错，因为给了两个元素（没有作为一个整体，所以算两个参数）。注意！！千万不能：a=student.append(3)。

2）删除元素

del 列表名[元素的索引]。注意这里是空格不是.了！

与append()函数类似，能删除单个元素、多个元素（切片）、整个列表。

3）修改元素

使用偏移量修改对应位置的元素。

```
list1 = ['小明','小红','小刚','小美']
list1[1] = '小蓝'
print(list1)
```

```
#结果显示
#['小明','小蓝','小刚','小美']
```

二、字典

1. 代码格式

- 字典外层是大括号{}，用等号赋值；
- 列表中的元素是自成一体的，而字典的元素是由键值对构成的，用英文冒号连接。有多少个键值对就有多少个元素。如'小明':95，其中我们把'小明'叫键（key），95叫值(value)。
- 键值对间用逗号隔开

所存储的两种数据若存在一一对应的情况，用字典储存会更加方便。唯一的键和对应的值形成的整体，我们就叫做【键值对】。键具备唯一性，而值可重复。

字典中数据是随机排列的，调动顺序也不影响。所以列表有序，要用偏移量定位；字典无序，便通过唯一的键来取值。

（注：len()函数用于获取数据长度）

```
students = ['小明','小红','小刚']
scores = {'小明':95,'小红':90,'小刚':90}
print(len(students))
print(len(scores))
#结果显示
#3
#3
#字典的元素个数，数冒号就行了
```

2. 提取元素

字典没有偏移量，所以在提取元素时，中括号中应该写键的名称，即字典名[字典的键]。提取出来的是key对应的value，而不会显示键的数据！

```
scores = {'小明': 95, '小红': 90, '小刚': 90}
print(scores['小明'])
```

3. 增加/删除元素、

1）新增元素

字典名[键] = 值。每次只能新增一个键值对。scores['小刚','小美']=92,85，这样是不对的，最终会输出('小刚','小美':(92,85))作为一个键值对。

```
album = {'周杰伦':'七里香'}
album['王力宏'] ='心中的日月'
album['林俊杰'] = '小酒窝'
print(album)
print(album['周杰伦'])
#结果显示
#{'周杰伦':'七里香', '王力宏':'心中的日月', '林俊杰':'小酒窝'}
#七里香
```

2）删除元素

del 字典名[键]

```
album = {'周杰伦':'七里香','王力宏':'心中的日月'}
```

```
del album['周杰伦']
print(album)
#结果显示
#{'王力宏':'心中的日月'}
```

3）修改元素

如果不是整个键值对都不需要，只需要改变对应key下的value，修改就可以，不需要del。

```
dict1 = {'小明':'男'}
dict1['小明'] = '女'
print(dict1)
#字典没有偏移量，只能通过key找到元素位置
```

三、列表与字典的嵌套

1. 列表与列表

列表中有两个列表元素，[1]表示取第二个元素（列表），[2]表示取第二个元素中的第三个元素（偏移量为2）。

```
student=[['小红','小黄','小橙'],['小绿','小蓝','小紫','小青']]
print(student[1][2])
#结果显示为
#小紫
```

1. 字典与字典

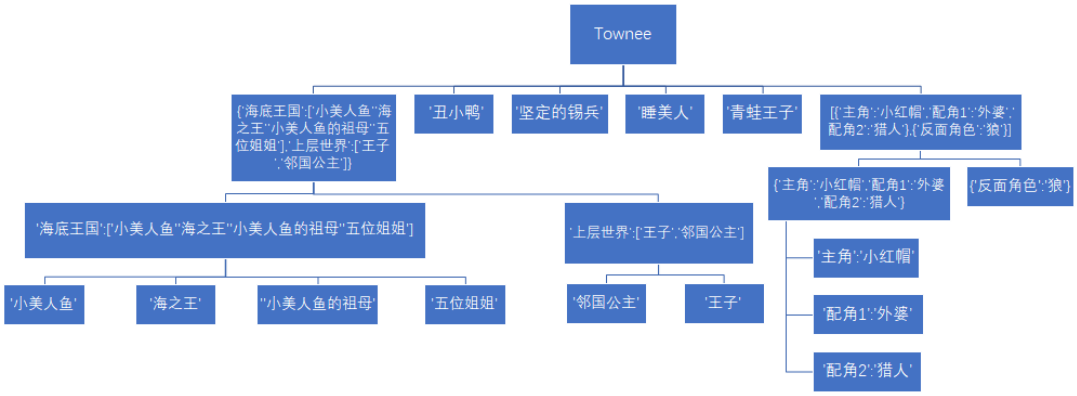
字典中存储了两个字典，所以提取数据时只能用key值。

```
scores={'第一组':{'小明':95,'小红':96},'第二组'{'小刚':94',小青':99}}
print(scores['第一组']['小红'])
#结果显示：
#96
```

1. 列表与字典

使用偏移量从最外层括号到最内层括号取数。查找townee列表中的第六个元素中的第2个元素（一定是字典，因为之后用的是键值而不是偏移量）中key为’反面角色’的value。

```
townee = [
    {'海底王国':['小美人鱼''海之王''小美人鱼的祖母''五位姐姐'],'上层世界':['王子','邻国公主']},
    '丑小鸭','坚定的锡兵','睡美人','青蛙王子',
    [{'主角':'小红帽','配角1':'外婆','配角2':'猎人'},{'反面角色':'狼'}]
    ]
print(townee[5][1]['反面角色'])
```



第5关 for循环和while循环–消灭该死的重复（上）

课程目标

1. 掌握for...in...循环的格式与特点，了解其数据传递
2. 熟练掌握for...in...与range()函数的结合使用
3. 掌握while循环的格式与特点
4. 区别for...in...循环与while循环

课程难点

1. range()函数各个参数的含义
2. for...in...循环与while循环的变量更迭
3. while语句如何避免死循环

课程重要内容重现

一、range()函数

1. 使用range()函数，可以生成一个整数序列。
2. 参数的意义：
   - range(n)与range(0,n)一样，都是生成一个从0到n–1的整数序列
   - range(m,n)，n>m，生成从m到n–1的序列。取头不取尾。
   - range(m,n,p)，n>m，生成m到n–1中间隔为p的整数序列。例如range(3,30,5)的意思就是从3开始每间隔5取一个数，直到29，结果为[3,8,13,18,23,28]。（只有两个参数时，p默认为1。）

二、for...in...循环

1. i是变量名，可以根据需要取。
2. 遍历：逐一访问全部数据。

3. 用于处理已知循环次数或循环固定次数的问题。
4. 格式：
   - for...in...语句最后需要接冒号
   - for...in...语句内部语句需要缩进，会被重复执行

```
for i in [1,2,3,4,5]:
   print(i)
#显示结果为：
#1
#2
#3
#4
#5
for i in '吴承恩':
    print(i)
#显示结果为：
#吴
#承
#恩
```

5. in后可接列表、字典和字符串，但不允许接整数、浮点数。
   - 列表里的元素值依次赋值给变量i，i最终的值取决于最新一次的赋值。
   - 若是字典，则赋值给变量的是字典中的key，而不是value。
   - 对于字符串而言，会依次输出字符。

```
d = {'小明':'醋','小红':'油','小白':'盐','小张':'米'}
for i in d:
    print(i)
#显示结果为：
#小明
#小红
#小白
#小张

d = {'小明':'醋','小红':'油','小白':'盐','小张':'米'}
for i in d:
    print(d[i])
#显示结果为：
#醋
#油
#盐
#米
```

6. range()函数配合使用

   可以用for...in...语句遍历range()函数生成的整数序列并打印。

```
for i in range(13,17):
    print(i)
#显示结果为：
#13
#14
#15
#16
```

for i in range(n)：，表示循环n次。如下面例题中i=0，i=1，i=2时都执行一次语句内的代码，也就是3次。

```
for i in range(3):
    print('我很棒')
#结果显示为：
#我很棒
#我很棒
#我很棒
```

三、while循环

1. 与if条件判断类似，while后条件若满足，会进入语句内部循环直至条件不再满足或内部打断。
2. 用于处理未知循环次数或循环固定次数的问题。
3. 格式

- 在while循环前要定义变量
- 为避免陷入死循环，在循环内必须更新变量，如自加：a=a+1，也可以写成a+=1。
- 循环体内部需要缩进！

```
a = 0
#非格式要求，但在while循环前必须要定义变量

while a < 5:
      a = a + 1
      print(a)
#显示结果为：（不显示0是因为a先自加再输出，此时赋值已更新）
#1
#2
#3
#4
#5
a = 0
e循环前必须要定义变量
while a < 5:
      a = a + 1
print(a)
#显示结果为：（因为print()在循环外，只输出最终a的赋值）
#5
```

四、*pop()函数

用于移除列表中的一个元素（默认最后一个元素），并且返回该元素的值。

可以指定移除元素，列表使用偏移量为参数，字典使用key作为参数。例如：students.pop(0)是删除并提取students这个列表中偏移量为0的元素并返回这个元素。

```
students = ['小明','小红','小刚']
student1 = students.pop(0)
#运用pop()函数，同时完成第一个元素的提取和删除。并将pop()的返回值存在student1这个变量中。
students.append(student1)
#将移除的student1安排到最后一个座位。
```

```
print(students)
#显示结果为：
#['小红','小刚','小明']
```

第6关 布尔值和四种语句–消灭该死的重复（下）

课程目标

1. 熟悉布尔值与布尔运算
2. 掌握break语句、continue语句如何与if结合使用

课程难点

1. 数值本身作为判断条件时，何为真何为假
2. break语句、continue语句与if结合使用时的缩进量

课程重要内容重现

一、布尔值与布尔运算

1. 布尔值：True（判断为真）和False（判断为假）。为真，则可继续运行下去，为假，条件不成立，不会执行接下来的语句。在使用True与False时首字母要大写！
2. 布尔运算：用数据做逻辑运算。（进行布尔运算后得到的结果为布尔值。例如：下方print()函数括号内进行了布尔运算，输出的值为布尔值。）

   1）用数值做比较，使用比较运算符号：（注意区别=与==）

Python中的比较运算符	
等于	==
不等于	!=
大于	>
小于	<
大于等于	>=
小于等于	<=

```
print(3<5)
print(3>5)
print('长安'=='长安')
print('长安'!='金陵')
#显示结果为：
#True
#False
#True
#True
```

   2）直接用数值做运算

   数值本身作为判断条件时：（none代表空值）

假的	其他都是真的
False	True
0	5（任意整数） 1.0（任意浮点数）
''（空字符串）	'苏东坡'（字符串）
[]（空列表）	[1,2,3]
{}（空字典）	{1:'春风',2:'秋分'}
None	

3）布尔值之间的运算

and（与）、or（或）、not（非）、in（判断一个元素是否在一组数据中）、not in（判断一个元素是否不在一组数据中。


```
list = [1,2,3,4,5]
a = 1
# 做一次布尔运算，判断“a是否在列表中”
print(bool(a in list))
print(bool(a not in list))

dict = {'法国':'巴黎','日本':'东京','中国':'北京'}
a = '法国'
#做一次布尔运算，判断“字典中是否有a这个键”
print(bool(a in dict))

```

3. bool()函数

使用bool()函数可查看一个数据的布尔值，用法与type()类似。

二、break语句

break的意思是如果满足了某一个条件，就提前结束循环，只能在循环内部使用。所以要注意break前的缩进！Tab键和空格键不能同时混用。

```
for...in...:
    ...
    if ...:
        break
# break语句搭配while循环
while...(条件):
    ...
    if ...:
        break
```

三、continue语句

在循环内部使用，当条件满足时，触发continue语句，将跳过之后的代码，直接回到循环的开始，即结束本次循环，开启下次循环。

```
for...in...:
    ...
    if ...:
        continue
# continue语句搭配while循环
while...(条件):
    ...
    if ...:
        continue
```

四、pass语句

常与if配合使用。

为了保持代码结构的完整性，pass不做任何操作，只是充当了一个占位语句。当没想好结构中具体的代码时，可以先用pass占位，保证程序正常运行不报错。

五、else语句

当循环中没有碰到break语句、continue语句等跳出循环的操作时，就会执行循环后面的else语句，否则就不会执行。

第7关 项目实操：PK小游戏（1）– 小游戏大学问

课程目标

能够独立完成一个完整的项目

课程难点

1. 模块的调用
2. 使用格式符%实现数据类型的转换

课程重要内容重现

一、完成一个项目的流程

1. 明确项目目标
2. 分析过程，拆解项目
3. 逐步执行，代码实现

二、调用模块

1. import 模块名
2. 调用模块里的函数：模块名.函数名(参数)
3. time模块与random模块

1）延时函数：time.sleep(1.5) 表示暂停1.5秒再运行下一行代码
2）随机生成一个数字：random.randint(n,m) 表示随机从n-m之间抽取一个整数
3）随机从列表或字典抽取一个元素：random.choice(列表或字典名)

三、不同数据类型的拼接——格式符%

先占一个位置，之后再填上实际的变量，可以省去使用数据转换函数的麻烦。例如print('血量：%s 攻击：%s' % (player_life,player_attack))。实际变量之间用逗号隔开。

1. %s ： 占一个位置用来放字符串
2. %f：占一个位置用来放浮点数，%.2f表示保留两位小数，'.'后的数字表示小数位数
3. %d：占一个位置用来放整数

四、format()函数

用来占位的是大括号{}，不用区分类型码（%+类型码）。格式是'str.format()'。例如： print('【玩家】\n血量：{}\n攻击：{}'.format(player_life,player_attack))

第8关 编程思维：如何解决问题-编程学习的两大思维瓶颈

略

第9关 函数-喊出我的名字

课程目标

1. 了解函数的作用
2. 熟练掌握函数的参数类型以及参数传递
3. 掌握函数的返回值及return语句的运用
4. 了解全局变量和局部变量，以及将局部变量转换为全局变量的方法

课程难点

1. 函数中参数的传递顺序
2. 函数的返回值及return语句的运用

课程重要内容重现

一、函数

函数是组织好的、可以重复使用的、用来实现单一功能的代码。写为：函数名(参数名)，注：括号内可以为空，也可以为多个参数，多个参数间用逗号隔开即可。

1. 参数

函数中需要使用的变量，数量视情况而定。
1）位置参数

位置参数是描述总体最常用的一种参数。参数的顺序和个数要和函数定义中一致。例如：

```
def  menu(appetizer,course):
    print('一份开胃菜：'+appetizer)
    print('一份主食：'+course)

menu('话梅花生','牛肉拉面')
#'话梅花生'和'牛肉拉面'是对应参数appetizer和course的位置顺序传递的，所以被叫作【位置参数】
```

2）默认参数

默认参数必须放在位置参数之后。若调用函数时没有传递参数就为默认值，但如果调用时向默认参数传递了数据，则默认参数被新参数代替。

```
def  menu(appetizer,course,dessert='绿豆沙'):
    print('一份开胃菜：'+appetizer)
    print('一份主食：'+course)
    print('一份甜品：'+dessert)
menu('话梅花生','牛肉拉面')
#结果显示为：因为调用时只给了两个参数，第三个参数为默认值
#一份开胃菜：话梅花生
#一份主食：牛肉拉面
#一份甜品：绿豆沙
menu('话梅花生','牛肉拉面','银耳羹')
#结果显示为：因为调用时给了三个参数，第三个参数被更新
#一份开胃菜：话梅花生
#一份主食：牛肉拉面
#一份甜品：银耳羹
```

3）不定长参数

一个星号*加上参数名。当传入此处的参数数量不确定时使用，数据类型为元组（元组（tuple）：写法是把数据放在小括号()中，它的用法和列表用法类似，主要区别在于列表中的元素可以随时修改，但元组中的元素不可更改。列表一样，元组是可迭代对象，这意味着我们可以用for循环来遍历它）。

- 当默认参数在不定长参数后时，若想更改默认参数，需要注明dessert=’银耳羹’，例如：

```
def menu(appetizer,course,*barbeque,dessert='绿豆沙'):
    print('一份开胃菜：'+appetizer)
    print('一份主菜：'+course)
    print('一份甜品：'+dessert)
    for i in barbeque:
        print('一份烤串：'+i)

menu('话梅花生','牛肉拉面','烤鸡翅','烤茄子','烤玉米',dessert='银耳羹')
#结果显示为：
#一分开胃菜：话梅花生
#一份主菜：牛肉拉面
#一份甜品：银耳羹
#一份烤串：烤鸡翅
#一份烤串：烤茄子
#一份烤串：烤玉米
```

- 当默认参数在不定长参数之前，则默认按顺序传递参数（默认参数会改变），剩下的参数均传递给不定长参数。

```
def menu(appetizer,course,dessert='绿豆沙',*barbeque):
    print('一份开胃菜：'+appetizer)
    print('一份主菜：'+course)
    print('一份甜品：'+dessert)
    for i in barbeque:
        print('一份烤串：'+i)

menu('话梅花生','牛肉拉面','银耳羹','烤鸡翅','烤茄子','烤玉米')
#结果显示为：
#一分开胃菜：话梅花生
#一份主菜：牛肉拉面
#一份甜品：银耳羹
#一份烤串：烤鸡翅
#一份烤串：烤茄子
#一份烤串：烤玉米
```

2. 定义函数

语法：

```
def 函数名(参数1，参数2，...参数n):
  函数体
  return 语句
```

函数内部一旦遇到return语句，就会停止执行并返回结果。没有return语句的函数，Python也会在末尾隐性地加上return None，即返回None值（return None可以简写为return。）

3. 调用函数

1）语法：函数名(参数)

参数从0个到多个都可以实现。但当参数个数为0时，括号也不能省略。

2）math(10)的意思是将整数10赋值给参数x并运行该函数。函数执行完毕后最终返回了y的值即110，然后将这个结果赋值给变量a，再用print()将变量a打印出来。

```
def math(x):
    y =  x ** 2 + x
    return y
a = math(10)
print(a)
```

3）也可以只用一行代码print(math(10))来表示同样的意思。

二、return语句

1. return是返回值，当输入参数给函数，函数就会返回一个值，事实上每个函数都会有返回值。
2. 如果不是立即要对函数返回值做操作，可以用return语句保留返回值。

需要多次调用函数时，可以定义一个主函数main()，调用非主函数的返回值。下面例子中的传参：main()函数中的李若彤→dream_face(位置参数)→face()函数的参数name，最终返回name + ’的脸蛋’即李若彤的脸蛋。

```
def face(name):
    return name + '的脸蛋'
def body(name):
    return name + '的身材'
def main(dream_face,dream_body):
    return '我的梦中情人：' + face(dream_face) + ' + ' + body(dream_body)

print(main('李若彤','林志玲'))
```

1. 若需要返回多个值，在return后用逗号隔开即可，此时返回的是一个元组。元组与列表其实都是数据的“序列”，元组取某个位置的值的操作，与列表是一模一样的，即元组名[i]。

```
def lover(name1,name2):
    face = name1 + '的脸蛋'
    body = name2 + '的身材'
    return face,body
a=lover('李若彤','林志玲')
#此时return的值为元组 ('李若彤的脸蛋', '林志玲的身材')，并赋值给变量 a 。
print('我的梦中情人：'+a[0]+' + '+a[1])
```

4. 没有return语句的函数会默认返回None值。
5. 一旦函数内部遇到return语句，就会停止执行并返回结果。

三、变量的作用域

1. 局部变量

   在函数内定义的变量，仅能在函数内部使用（局部作用域）

2. 全局变量

   所有函数之外赋值的变量，可以在程序的任何位置使用（全局作用域）

3. 局部变量和全局变量不要取相同的名字，以免混淆。
4. global语句：可以将局部变量转换为全局变量，一般写在函数体的第一行。

```
def egg():
    global quantity
#global语句将变量quantity声明为全局变量
    quantity = 108
egg()
print(quantity)
```

四、函数的嵌套

sum = rent + cost() 的结果就是sum = rent + variable_cost （cost()函数的返回值）。

最后调用函数时，也只需调用sum_cost()即可。

```
rent = 3000

def cost():
    utilities = int(input('请输入本月的水电费用'))
    food_cost = int(input('请输入本月的食材费用'))
```

```
    variable_cost = utilities + food_cost
    print('本月的变动成本是' + str(variable_cost))
    return variable_cost

def sum_cost():
    sum = rent + cost()#调用函数
    print('本月的总成本是' + str(sum))

sum_cost()
```

五、其他

1. list()函数

   可以将数据转换为列表。

2. extend()函数

- append() ,方法用于在列表末尾添加新的对象，只能添加一个值（多个值作为一个整体也行）。
- extend() 函数用于在列表末尾一次性追加另一个序列中的多个值（用新列表扩展原来的列表）如果extend的是字符串，则字符串会被拆分成字符数组，如果extend的是字典，则字典的key会被加入到ist中。

```
list1=[1,2]
list2=[2,4,5]
list3=[]
list2.append(list1)
print(list2)
#结果显示为：
#[2,4,5,[1,2]]    list1的元素作为一个列表整体被加入list2
list3.extend(list1)
print(list3)
#结果显示
#[1,2]    list1的元素独立加入list3中
```

3. *reversed()函数 （以下仅做参考）

   reversed()函数可以将数据反转，从后往前迭代。

   reversed()之后，第二次for循环、list()、tuple()以及join()得到的结果都为空，原因就是b不是反转列表本身，而是一个列表反向迭代器，所以直接输出函数返回值会是类似于乱码，且reversed()之后，只在第一次遍历时返回值。

```
#参数是列表1：
a=[1,2,3,4,5]
b=reversed(a)
print(a)
#结果显示为：
#[1, 2, 3, 4, 5]
print(b)
#结果显示为：
#<listreverseiterator object at 0x06A9E930>
for i in b:#第一次遍历
print(i)
#结果显示为：
```

```
#5 4 3 2 1
for i in b:#第二次遍历为空
print(i)
#结果显示为：啥也没有
```

```
#参数是列表2：
a=[1,2,3,4,5]
b=reversed(a)
print(list(b))#第一次转换为列表
#结果显示为：
#[5,4,3,2,1]
print(list(b))#第二次
#结果显示为：
#[]
```

```
#参数是元组
t=(4,5,6)
tt=reversed(t)
print(t)
#结果显示为：
#(4, 5, 6)

print(tt)
#结果显示为：
#<reversed object at 0x06A07E50>

print(tuple(tt))
#第一次转换为元组
#结果显示为
#(6, 5, 4)

print(tuple(tt))
#第二次为空
#结果显示为
```

#()

```
#参数是字符串
s='cba'
ss=reversed(s)
print(s)
#结果显示为
#'cba'
print(ss)
#结果显示为
#<reversed object at 0x06A07E70>
print(list(ss))#第一次转换为列表
#结果显示为
#['a', 'b', 'c']
print(list(ss))#第二次为空
```

```
#结果显示为
#[]
```