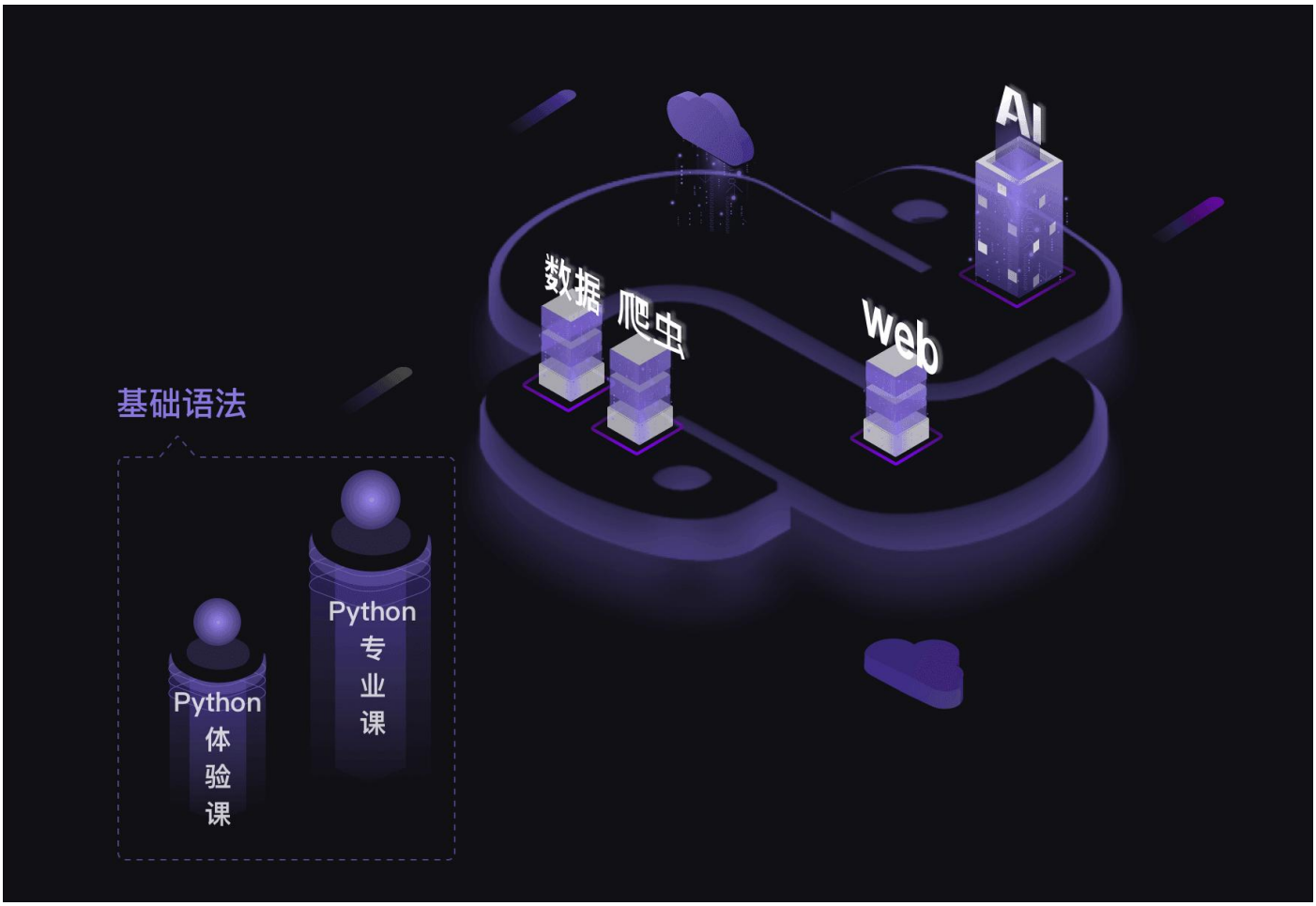
Python 小课|学习笔记 JS



基础语法 新手村(0-3) 山脚村(4-9) 山腰村(10-14) 山顶村(15-19)

目录

0 千寻的名字

0.1 print()打印函数

```
print(520)                  #打印 520
#>> 520
print('千寻')               #原样打印加''
#>> 千寻
print("Let's play")         #可以用双引号""
#>> Let's play
print(1+1)                  #可以运算
#>> 2
print('1+1')                #原样打印
#>> 1+1
print('''第一行
第二行''')                  #换行用三引号'''   '''
#>> 第一行
#>> 第二行
print('第三行\n 第四行')     #转义字符\n 换行
#>> 第三行
#>> 第四行
print('Let\'s go')          #转义字符\使第二个'变为纯粹的符号
#>> Let's go
```

0.2 变量和赋值

```
name = '千寻'               #千寻赋值给了变量 name
name = '小千'               #小千赋值给了变量 name，覆盖了千寻
print(name)                 #此时打印变量 name，  结果是小千
#>> 小千
```

1 萌新的进化

1.1 数据类型 type()

数据类型		
数据类型	**案例**	**说明**
字符串(str)	’千寻’、’love’、’2’	用引号括起来的内容
整数(int)	2、40、-11	普通的整数数字
浮点数(float)	12.0、5.5、-0.13	带小数点的数字

```
print(type('查询的内容'))          #type()为查询数据类型函数
#>> <class 'str'>                #'查询的内容'数据类型为 str 字符串
print(type(520))
#>> <class 'int'>                #520 数据类型为 int 整数
print(type(5.20))
#>> <class 'float'>              #5.20 数据类型为 float 浮点数
```

1.2 数据拼接和转换 str()、int()、float()

```
#-----数据拼接-----#
name = '我爱'
number = 999
print(name + str(number))          #数据拼接用【+】，str()函数把 number 转换为字符串
#>> 我爱 999
print(name + '999')                #引号内的 int 已经为字符串，想同类型的数据才能拼接
#>> 我爱 999

#-----数据转换-----#
bug = '666'
hair = '0'
money1 = 1.5
money2 = '2.2'
print(int(bug) + int(hair))        #只有符合整数规范的【字符串类】数据，才能被 int()强制转换
#>> 666
print(int(money1))                 #浮点数转换成整数会被强制抹零取整
#>> 1
print(float(bug))                  #整数类型字符串转换成浮点数
#>> 666.0
print(float(money2))               #浮点类型字符串转换成浮点数
#>> 2.2
```

1.3 Python 算术运算符

Python 算术运算符		
运算符	表示	例子
+	加	2+1 输出结果 3
-	减	1-2 输出结果 -1
*	乘	1*2 输出结果 2
/	除	1/2 输出结果 0.5
%	取模--返回除法的余数	5%2 输出结果 1
**	幂--返回 x 的 y 次幂	2**3 为 2 的 3 次方
//	取整除--返回商的整数部分	11//2 输出结果 5

2 灭霸的选择

2.1 条件判断 if...elif...else...

```
#-----单向判断 if...-----#
stonenumber = 6                                    #为宝石数量赋值
if stonenumber >= 6:                               #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')
    #结果：显示‘你拥有了毁灭宇宙的力量’的结果
#>> 你拥有了毁灭宇宙的力量

#-----双向判断 if...else...-----#
```

```
weight = 90                                                    #为体重赋值 90
if weight >= 100:                                               #条件：如果体重大于等于 100
    print("不吃了")                                             #结果：显示 if 下的结果
else:
#条件：当不满足 if 条件时，执行 else 下的命令
    print("放心吃吧")                                           #结果：显示 else 下的结果
#>> 放心吃吧

#-----多向判断 if...elif...else...------#
money = 10                                                     #为 money 赋值 10
if money >= 100:                                                #条件：当 money 大于等于 100
    print('买保时捷吧！')                                       #结果：显示 if 下的结果
elif 30 <= money < 100:                                         #条件：当 money 大于等于 30，且小于 100
    print('买奔驰吧！')                                         #结果：显示 elif 下的结果
else:                                                          #条件：if 和 elif 都不满足
    print('买什么车，单车了解一下')                             #结果：显示 else 下的结果
#>> 买什么车，单车了解一下
```

2.2 if 嵌套

```
#-----if 嵌套-----#
historyscore=26                                                #赋值语句：为历史成绩赋值
if historyscore>=60:                                            #条件：当历史成绩大于等于 60 时
    print('你已经及格')                                         #结果：输入及格的结果
    if historyscore>=80:
    #条件：当历史成绩大于等于 60，且历史成绩大于等于 80
        print('你很优秀')                                       #结果：输出你很优秀的结果
    else:                                                      #条件：当历史成绩大于等于 60，且小于 80 时
        print('你只是一般般')                                   #结果：输出你只是一般般的结果
else:                                                          #条件：当历史成绩小于 60 时
    print('不及格')                                             #结果：输出不及格的结果
    if historyscore<30:                                         #条件：当小于 60，且小于 30 时
        print("学渣")                                           #结果：输出学渣结果
    else:                                                      #条件：当小于 60，且大于等于 30 时
        print("还能抢救一下")                                   #结果：输出还能抢救一下的结果
print('程序结束')                                               #打印最终结果
#>> 不及格
#>> 学渣
#>> 程序结束
```

3 霍格沃茨来信

3.1 input()函数

input 函数		
使用	有问有答，有来有往	终端区输入
赋值	函数好用，赋值第一	必须赋值
数据类型	返回类型，必为 str	输入值必为字符串
结果转换	想要整数，源头转换	需要整数，需要转换 int(input(...))

```
#-----input()使用-----#
input('请输入你的名字：')
#input()函数，有问有答，需要在终端输入信息

#-----input()赋值-----#
name = input('请输入宠物的名字：')                    #用 input()搜集信息
print('I Love ' + name)                              #打印内容

#-----input()数据类型-----#
choice = input('请输入您的选择：')
#输入值，永远会被强制转换为字符串类型
if choice == '1':                                    #条件判断:条件 1
    print('霍格沃茨欢迎您的到来。')                    #条件 1 的结果
else:                                                #条件判断：其他条件
    print('您可是被梅林选中的孩子，我们不接受这个选项。')    #其他条件的结果
#>> 请输入您的选择：1
#>> 霍格沃茨欢迎您的到来。

#-----input()结果的强制转换-----#
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
#>> 你一个月工资多少钱？：1
#>> 我负责赚钱养家，你负责貌美如花～
```

3.2 课后练习：古灵阁金币兑换

小精灵：您好，欢迎古灵阁，请问您需要帮助吗？需要 or 不需要？

你：需要

小精灵：请问您需要什么帮助呢？1 存取款；2 货币兑换；3 咨询

你：2

小精灵：金加隆和人民币的兑换率为 1:51.3，即一金加隆=51.3 人民币

小精灵：请问您需要兑换多少金加隆呢？

（你说了一个数字 N）

小精灵：好的，我知道了，您需要兑换（你说的数字 N）金加隆。

小精灵：那么，您需要付给我（你说的数字 N*51.3）人民币。

注 1：如果选择不需要帮助，小精灵会礼貌地说'好的，再见。'

注 2：如果选择帮助【1 存取款】，小精灵会推荐你去存取款窗口；如果选择帮助【3 咨询】，小精灵会推荐你去咨询窗口。

```
choice1 = input('小精灵：您好，欢迎古灵阁，请问您需要帮助吗？·需要·或·不需要·？ ')
if choice1 == '需要':
    choice2 = int(input('小精灵：请问您需要什么帮助呢？1 存取款；2 货币兑换；3 咨询'))
    if choice2 == 1:
        print('小精灵：您好，推荐您去存取款窗口')
    elif choice2 == 2:
        print('小精灵：金加隆和人民币的兑换率为 1:51.3，即 1 金加隆=51.3 人民币')
        choice3 = int(input('小精灵：请问您需要兑换多少金加隆呢？ '))
        print('小精灵：好的，我知道了，您需要兑换 '+str(choice3)+' 金加隆')
        print('小精灵：那么您需要付给我 '+str(choice3*51.3)+' 人民币')
    else:
        print('小精灵：您好，推荐您去咨询窗口')
else:
    print('小精灵：好的，再见。')
```

4 收纳的艺术

计算机利用数据的三种方式		
直接使用数据	计算和加工数据	用数据做判断
print(3) #>>> 3	print(2*3) #>>> 6	i=0;if i>0:;else:

4.1 列表 append()、del

```
#-----列表-----#
list1 = ['小明',18,1.7]                          #列表名、赋值号、中括号、逗号
print(list1)                                     #打印列表 list1
#>> ['小明',18,1.7]
students = ['小明','小红','小刚']                  #列表中每个元素都有偏移量，0,1,2...
print(students[0])                               #打印 students 列表中的第[0]个元素
#>> 小明
print(students[:1])                              #列表切片后还是列表
#>> ['小明']
list2 = [5,6,7,8,9]
print(list2[:])                                  # 打印出[5,6,7,8,9]
print(list2[2:])                                 # 打印出[7,8.9]
print(list2[:2])                                 # 打印出[5,6]
print(list2[1:3])                                # 打印出[6,7]
#总结为：左右空，取到头；左要取，右不取。

#-----列表增加删除元素 append()、del-----#
students = ['小明','小红','小刚']
students.append('小美')
#给列表增加'小美'元素，append()函数每次只能加一个
print(students)
#>> ['小明', '小红', '小刚', '小美']
del students[3]                                  #删除列表中的 3 号元素
```

```
print(students)
#>> ['小明', '小红', '小刚']
del students[:2]                                    #删除列表中的 0、1 号元素
print(students)
#>> ['小刚']
```

4.2 字典、元组 len()

```
#-----字典、查询长度函数 len()-----#                  #字典中的键具有唯一性，而值可以重复
scores = {'小明':95,'小红':90,'小刚':90}
#字典名、赋值号、大括号、逗号，键值对（键、冒号、值）
print(len(scores))                                  #查询长度函数 len()，字典有 3 个键值对，所以>>3
#>> 3
print(scores['小明'])                                #从字典提取元素，提取键'小明'的值
#>> 95
album = {'周杰伦':'七里香','王力宏':'心中的日月'}
del album['周杰伦']                                  #删除字典键值对
print(album)                                        #>> {'王力宏': '心中的日月'}
album['周杰伦'] = '十一月的萧邦'                       #增加键值对
print(album)
#>> {'王力宏': '心中的日月', '周杰伦': '十一月的萧邦'}
print(album['周杰伦'])                               #>> 十一月的萧邦

#-----元组-----#
tuple1 = ('A','B')                                  #元组和列表类似，不过他是用（）来包的
list2 = [('A','B'),('C','D'),('E','F')]
print(tuple1[0])                                    #元组都是序列，提取方式也是偏移量
#>> A
print(list2[1][1])                                  #元组也支持嵌套
#>> D
```

4.3 课后练习：找出那只狼

```
#-----课后练习：找出那只狼-----#
'''
在未来世界，一个新建的童话镇吸引了不少人入住。
不过，在人群里隐藏着一只狼，会威胁大家的安全。
童话镇的镇长希望你能找到它，并揭发其身份。
用程序语言就是说：列表中有个字符串是“狼”，将其打印出来吧。
'''
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'])
```

5 消灭该死的循环（上）

for 循环 VS while 循环		
	for 循环	while 循环
循环次数明确	✔	
循环次数不明确		✔
把一件事做 N 遍	✔	✔

5.1 for 循环、while 循环

```
#-----for...in...循环(遍历)-----#
for i in [1,2,3,4,5]:                #空房间 i，列表中一群人排队办业务
   print(i)                          #办事流程
   #>> 1、2、3、4、5（换行）
print(i)                             #for 循环结束后还能用这个房间，最后一个进去的留在了房间内
#>> 5

#有一个空房间：i
#有排队办业务元素：这里是列表[1,2,3,4,5]
#办事流程：他们中的每一个被叫到号(for i in)，轮流进去一个空房间办业务
#每一个数字进去房间之后，都对计算机说：“喂，我要办这个业务：帮忙把我自己打印出来”，也就是 print(i)
#然后计算机忠实的为每一个数字提供了打印服务，将 1,2,3,4,5 都打印在了屏幕上

dict1 = {'日本':'东京','法国':'巴黎'}
for i in dict1:                      #从字典中提取键
    print(i)                        #>> 日本；法国
for i in dict1:                      #从字典中提取值
    print(dict1[i])                  #>> 东京；巴黎

#-----while 循环-----#
a = 0                 #先定义变量 a，并赋值
while a < 5:          #设定一个放行条件：a 要小于 5，才能办事
    a = a + 1         # 满足条件时，就办事：将 a+1
    print(a)          # 继续办事：将 a+1 的结果打印出来
```

5.2 range()、pop()

```
#-----range()函数-----#
for i in range(2):
    print(i)                    #>> 0 和 1；range(b)为 0~b-1 的整数序列
for i in range(11,13):
    print(i)                    #>> 11 和 12；range(a,b)为 a~b-1 的整数序列
for i in range(21,25,2):
    print(i)                    #>> 21 和 23；range(a,b,c)为 a~b-1 之间、计数间隔为 c 的序列

#-----pop()函数-----#
'''
列表中的 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。
可以将其理解为提取和删除的融合：①提取：取到元素，对列表没有影响；②删除：删除列表的元素。
```

```
而移除，则是同时做到取到元素，并且删除列表中的元素。
'''
# 提取只取不删
list1 = ['0','1','2','3']
print(list1[3])                          #>> 3
print(list1)                             #>> ['0','1','2','3']

# 删除(del)只删不取
list1 = ['0','1','2','3']
del list1[3]
print(list1)                             #>> ['0','1','2']

# 移除（pop）又取又删
list1 = ['0','1','2','3']
print(list1.pop())              #>> 3 #默认删除最后一个元素，并返回该元素的值。
print(list1)                    #>> ['0','1','2']
print(list1.pop(0))             #>> 0 #也可指定删除某个元素，并返回该元素的值。
print(list1)                    #>> [1','2']
```

5.3 课后练习：数数字，不要 4

小美想要用今天学到的循环打印数字 1-7，不过，她不喜欢 4 这个数字

用不同的循环方式来帮小美实现“打印 1-7，但是不要 4”这个愿望。

```
# while 循环
n = 0
while n < 7:
    n = n+1
    if n != 4:          # 当 num != 4，执行打印语句；等于 4 时不打印。
        print(n)

# for 循环
for num in range(1,8):  # 为同时能运行两个循环，新取参数 num。
    if num != 4:        # 当 num != 4，执行打印语句；等于 4 时不打印。
        print(num)
```

5.4 课后练习：前排轮流坐

小明、小红、小刚是同班同学，且坐在同一排，分别坐在第一位、第二位、第三位。

由于他们的身高都差不多，所以，老师计划让他们三个轮流坐在第一位。

每次换座位的时候，第一位变第三位，后面两位都往前一位。

```
# 常规方法
students = ['小明','小红','小刚']
for i in range(3):
    student1 = students[0]       # 获取第一个座位的学生 student1
    students = students[1:]      # 让 student1 暂时离开，后面的学生座位都进一位。
    students.append(student1)    # 将 student1 安排到最后一个座位
    print(students)
```

```
# 使用移除 pop()函数
students = ['小明','小红','小刚']
for i in range(3):
    student1 = students.pop(0)    # 运用 pop()函数，同时完成提取和删除。
    students.append(student1)     # 将移除的 student1 安排到最后一个座位。
    print(students)
```

6 消灭该死的循环（下）

6.1 布尔值和布尔运算 bool()

python 中的真和假(True 和 False 是布尔值)	
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:'春天',2:'秋天'}
None	

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	<=

```
#布尔值有两种：True 或 False

#【第一种计算：两个数值做比较】
#一共有六种比较方式：==、!=、>、<、>=、<=
print(3==3.0)        #判断为 True
print('a'!='a')      #判断为 False
print(3>5)           #判断为 False
print(3<5)           #判断为 True
print(100>=101)      #判断为 False
print(100<=101)      #判断为 True

#【第二种计算：直接用数值做运算】
print(bool(False))    #判断为 False
print(bool(0))        #判断为 False
print(bool(''))       #判断为 False
print(bool([]))       #判断为 False
print(bool({}))       #判断为 False
print(bool(None))     #判断为 False
```

```
print(bool(True))           #True
print(bool(5))              #True
print(bool('abc'))          #True
print(bool([1,2]))          #True
print(bool({1:1,2:2}))      #True

#【第三种计算：布尔值之间的运算】
#一共有 5 种计算方式：and、or、not、in、not in
#and 要求条件都满足才为 True
#or 只要求其中一个条件满足就为 True
a = 1
b = -1
print(a==1 and b==1)      #判断为假
print(a==1 or b==1)       #判断为真
#not 计算会反转真假
a = True
print(not a)              #判断为 False
#in 判断一个元素是否在一堆数据中
#not in 判断一个元素是否不在一堆数据中
a = [1,2,3,4,5]
print(0 in a)             #判断为 False
print(1 in a)             #判断为 True
print(0 not in a)         #判断为 True
print(1 not in a)         #判断为 False
```

6.2 四种新语句 break、continue、pass、else

```
#-----break-----# 从循环内跳出，必须和 if 语句连用
for i in range(5):
    print('明日复明日')
    if i==3:              # 当 i 等于 3 的时候触发
        break             # 结束循环,实际打印 4 次

i = 0
while i<5:
    print('明日复明日')
    i = i+1
    if i==3:              # 当 i 等于 3 的时候触发
        break             # 结束循环，实际打印 3 次

while True:                          #开启一个无限循环
    p = input('请输入你的密码:')    #循环内部获取用户数据
    if p == '小龙女':                #直到用户输入正确密码
        break                        #结束循环
print('通过啦')
```

```
#-----continue-----# 跳跃到循环开头，必须和 if 语句连用
# continue 语句搭配 for 循环
for i in range(5):
```

```
    print('明日复明日')
    if i==3:                                # 当 i 等于 3 的时候触发
        continue                            # 回到循环开头
    print('这句话在 i 等于 3 的时候打印不出来')

# continue 语句搭配 while 循环
i = 0
while i<5:
    print('明日复明日')
    i = i+1
    if i==3:                                # 当 i 等于 3 的时候触发
        continue                            # 回到循环开头
    print('这句话在 i 等于 3 的时候打印不出来')

#西夏公主挑驸马的故事，只有连续答对才行，否则任何一题打错，都要从头开始回答问题。
while True:
    q1 = input('第一问：你一生之中，在什么地方最是快乐逍遥？')
    if q1 != '黑暗的冰窖':
        continue
    print('答对了，下面是第二问：')
    q2 = input('你生平最爱之人，叫什么名字？')
    if q2 != '梦姑':
        continue
    print('答对了，下面是第三问：')
    q3 = input('你最爱的这个人相貌如何？')
    if q3 == '不知道':
        break
print('都答对了，你是虚竹。')
```

```
#-----pass-----#pass 为跳过，什么也不做，常用在 if 语句下
a = int(input('请输入一个整数:'))
if a >= 100:
    pass            #如果输入的数字>=100，则直接跳过，什么也不做
else:
    print('你输入了一个小于 100 的数字')
```

```
#-----else-----#
#else 不但可以和 if 配合使用，也可以跟 for 循环 while 循环使用，正常结束未 break 则执行 else
for i in range(5):
    a = int(input('请输入 0 来结束循环，你有 5 次机会:'))
    if a == 0:
        print('你触发了 break 语句，循环结束，导致 else 语句不会生效。')
        break
else:
    print('5 次循环你都错过了，else 语句生效了。')
```

```
#-----课堂练习-----#
```

```
'''
一个人在心里想好一个数，比如 24，然后让另一个人猜。如果他猜的数比 24 小，告诉他“太小了”，
如果他猜的数比 24 大，告诉他“太大了”，这个游戏只能猜 3 次，3 次都猜不中，就告诉他“失败了”。
'''
secret = 24
for i in range(3):                                    #3 次机会开启 3 次循环
    guess = input('guess which number is my secret:')
    if  int(guess) ==secret:
        print('正确！你很棒哦。')                        #若猜对输出结果
        break
    elif int(guess)>secret:
        print('你猜的太大了，请重新猜猜~')
    else:
        print('你猜的太小了，请重新猜猜~')
else:
    print('给你 3 次机会都猜不到，你失败了。')              #3 次机会都失败则执行 else
```

6.3 课后练习：囚徒困境

假设有两名囚徒 A 和 B 因为合伙犯罪被抓捕，因没有确凿可以指认罪行的证据，审判者准备单独审判两位囚徒。

若两人都认罪，则两人各判 10 年；若一个认罪一个抵赖，则认罪的人判 1 年，抵赖的人判 20 年；若两人都抵赖，则各判 3 年。

当两人都抵赖时，打印判决，代码结束；若为其他结果，则在打印判决后继续循环。

```
while True:
    a = input('A，你认罪吗？请回答认罪或者不认')
    b = input('B，你认罪吗？请回答认罪或者不认')
    if a == '认罪' and b == '认罪':
        print('两人都得判 10 年，唉')
    elif a == '不认' and b == '认罪':
        print('A 判 20 年，B 判 1 年，唉')
    elif a == '认罪' and b == '不认':
        print('A 判 1 年，B 判 20 年')
    else:
        print('都判 3 年，太棒了')
        break                                  # 当满足开头提到的条件时，跳出循环。
```

6.4 课后练习：困境中的选择

我们将“囚徒困境”写成了代码，让程序收集两名囚犯的认罪情况，进而决定他们的判决：

两人都认罪，则各判 10 年；一个认罪一个抵赖，则前者判 1 年，后者判 20 年；两人都抵赖，各判 3 年。只有两人都不认罪，程序才会停止。

现在有一个社会学家，在不同的人群中做这个实验，一旦遇到都不认罪的情况，就停止该人群中的实验。

同时，他希望程序能记录每一对实验者的选择，以及记录第几对实验者都选择不认罪。请你帮帮他吧。

```
n = 0
list_answer = []

while True:
    n += 1
    a = input('A，你认罪吗？请回答认罪或者不认：')
    b = input('B，你认罪吗？请回答认罪或者不认：')
    list_answer.append([a,b])                    # 用列表嵌套的方式来存放实验者的选择，也可用元组或字典。
    if a == '认罪' and b == '认罪':
        print('两人都得判 10 年，唉')
    elif a == '不认' and b == '认罪':
        print('A 判 20 年，B 判 1 年，唉')
    elif a == '认罪' and b == '不认':
        print('A 判 1 年，B 判 20 年')
    else:
        print('都判 3 年，太棒了')
        break

print('第' + str(n) + '对实验者选了最优解。')

for i in range(n):                               # 注意数据类型的转换，以及计数起点的不同（0 和 1）
    print('第' + str(i+1) + '对实验者的选择是：' + str(list_answer[i]))
```

6.5 课后练习：演员的作品

我很喜欢看电影，我回忆了一下，这两年我觉得还不错的国产电影。

下面，会将电影的影片名和主演放在字典里，如 movie = {'妖猫传':['黄轩','染谷将太']}。

需要你补充一些代码，让其他人只要输入演员名，就打印出：××出演了电影××。

```
movies = {
'妖猫传':['黄轩','染谷将太'],
'无问西东':['章子怡','王力宏','祖峰'],
'超时空同居':['雷佳音','佟丽娅'],
}

actor = input('你想查询哪个演员？')
for  movie in movies:                    # 用 for 遍历字典
    actors = movies[movie]               # 读取各个字典的主演表
    if actor in actors:
        print(actor + '出演了电影' + movie)
        break
else:
    print('我不认识')
```

7 小游戏大学问(项目实操)

7.1 调用模块 import、格式化字符串 format()

```
#-----time.sleep()-----#
import time                    #调用 time 模块
time.sleep(secs)
#使用 time 模块下面的 sleep()函数，括号里填的是间隔的秒数（seconds，简称 secs）
#time.sleep(1.5)就表示停留 1.5 秒再运行后续代码

#-----random.randint()-----#
import random                  #调用 random 模块
a = random.randint(1,100)
print(a)
# 随机生成 1-100 范围内（含 1 和 100）的一个整数，并赋值给变量 a
```

格式化字符串	
格式符+类型码	含义
%s	字符串显示
%f	浮点数显示
%d	整数显示

```
#-----格式化字符串-----#
# % 格式化：str % ()
print('%s%d'%('数字：',0))
print('%d，%d'%(0,1))
print('%d，%d，%d'%(0,1,0))

name1 = 'Python'
print('I am learning %s'% name1)  # 注：当只跟一个数据时，%后可不加括号，format()一定要有。

# format()格式化函数：str.format()
print('{}{}'.format('数字：',0))    # 优势 1：不用担心用错类型码。
print('{}，{}'.format(0,1))         # 不设置指定位置时，默认按顺序对应。
print('{1}，{0}'.format(0,1))       # 优势 2：当设置指定位置时，按指定的对应。
print('{0}，{1}，{0}'.format(0,1))  # 优势 3：可多次调用 format 后的数据。

name2 =  'Python 基础语法'
print('我正在学{}'.format(name2))  # format()函数也接受通过参数传入数据。
```

```
#-----英文变量名：CODELF（https://unbug.github.io/codelf）-----#
```

7.2 项目实操

这个游戏中，会随机生成玩家和敌人的属性，同时互相攻击，直至一方血量小于零。

另外，这样的战斗会持续三局，采取三局两胜制，最后输出战斗结果，公布获胜方。


```
#-----版本 1.0：自定属性，人工 PK-----#

print('【玩家】\n 血量：100\n 攻击：50')  # 自定义玩家角色的血量和攻击，用换行符'\n'来优化视觉
print('------------------------')  # 辅助功能，起到视觉分割的作用，让代码的运行结果更清晰

print('【敌人】\n 血量：100\n 攻击：30')
print('------------------------')

print('你发起了攻击，【敌人】剩余血量 50')  # 人工计算敌人血量：100-50=50
print('敌人向你发起了攻击，【玩家】剩余血量 70')  # 人工计算玩家血量：100-30=70
print('------------------------')

print('你发起了攻击，【敌人】剩余血量 0')  # 双方同时攻击，若血量出现小于等于 0，游戏结束
print('敌人向你发起了攻击，【玩家】剩余血量 40')
print('------------------------')

print('敌人死翘翘了，你赢了！') # 打印结果
```

```
#-----版本 2.0：随机属性，自动 PK-----#

import time,random

player_life = random.randint(100,150)
player_attack = random.randint(30,50)
enemy_life = random.randint(100,150)
enemy_attack = random.randint(30,50)

print('【玩家】\n'+'血量：'+str(player_life)+'\n 攻击：'+str(player_attack))
print('------------------------')
time.sleep(1)
print('【敌人】\n'+'血量：'+str(enemy_life)+'\n 攻击：'+str(enemy_attack))
print('------------------------')
time.sleep(1)

while (player_life >0) and (enemy_life > 0):
```

```
    player_life = player_life - enemy_attack
    enemy_life = enemy_life - player_attack
    print('你发起了攻击，【玩家】剩余血量'+str(player_life))
    #player_life 是整数，所以拼接时要先用 str()转换
    print('敌人向你发起了攻击，【敌人】剩余血量'+str(enemy_life))
    print('------------------------')
    time.sleep(1.5)
    # 为了体现出战斗回合，这里停顿 1.5 秒
```

```
#-----版本 3.0：打印战果，三局两胜-----#

import time
import random

player_victory = 0
enemy_victory = 0

for i in range(1,4):
    time.sleep(1.5)
    print('  \n————现在是第 %s 局————' % i)
    #对比之前：(' \n————现在是第'+str(i)+'局————')
    player_life = random.randint(100,150)
    player_attack = random.randint(30,50)
    enemy_life = random.randint(100,150)
    enemy_attack = random.randint(30,50)

    print('【玩家】\n 血量：%s\n 攻击：%s' % (player_life,player_attack))
    print('------------------------')
    time.sleep(1)
    print('【敌人】\n 血量：%s\n 攻击：%s' % (enemy_life,enemy_attack))
    print('------------------------')
    time.sleep(1)

    while player_life > 0 and enemy_life > 0:
        player_life = player_life - enemy_attack
        enemy_life = enemy_life - player_attack
        print('你发起了攻击，【敌人】剩余血量%s' % enemy_life)
        print('敌人向你发起了攻击，【玩家】的血量剩余%s' % player_life)
        print('------------------------')
        time.sleep(1.2)

    if player_life > 0 and enemy_life <= 0:
        player_victory += 1
        print('敌人死翘翘了，你赢了！')
    elif player_life <= 0 and enemy_life > 0:
        enemy_victory += 1
        print('悲催，敌人把你干掉了！')
    else:
```

```
        print('哎呀，你和敌人同归于尽了！')

if player_victory > enemy_victory :
    time.sleep(1)
    print('\n【最终结果：你赢了！】')
elif enemy_victory > player_victory:
    print('\n【最终结果：你输了！】')
else:
    print('\n【最终结果：平局！】')
```

7.3 课后练习：再来一盘

每盘（3 局）游戏结束后，游戏会问我们是否要继续游戏，再加一盘。

我们可以选择再来一盘，也可以选择退出游戏。

```
import time
import random

player_victory = 0
enemy_victory = 0

while True:
    for i in range(1,4):
        time.sleep(1.5)
        print('  \n————现在是第 %s 局————' % i)
        player_life = random.randint(100,150)
        player_attack = random.randint(30,50)
        enemy_life = random.randint(100,150)
        enemy_attack = random.randint(30,50)

        print('【玩家】\n 血量：%s\n 攻击：%s' % (player_life,player_attack))
        print('------------------------')
        time.sleep(1)
        print('【敌人】\n 血量：%s\n 攻击：%s' % (enemy_life,enemy_attack))
        print('------------------------')
        time.sleep(1)

        while player_life > 0 and enemy_life > 0:
            player_life = player_life - enemy_attack
            enemy_life = enemy_life - player_attack
            print('你发起了攻击，【玩家】剩余血量%s' % player_life)
            print('敌人向你发起了攻击，【敌人】的血量剩余%s' % enemy_life)
            print('------------------------')
            time.sleep(1.2)

        if player_life > 0 and enemy_life <= 0:
            player_victory += 1
            print('敌人死翘翘了，你赢了！')
```

```
        elif player_life <= 0 and enemy_life > 0:
            enemy_victory += 1
            print('悲催，敌人把你干掉了！')
        else:
            print('哎呀，你和敌人同归于尽了！')

    if player_victory > enemy_victory :
        time.sleep(1)
        print('\n【最终结果：你赢了！】')
    elif enemy_victory > player_victory:
        print('\n【最终结果：你输了！】')
    else:
        print('\n【最终结果：平局！】')

    a1 = input('要继续游戏吗，请输入 n 退出，输入其他继续：')  # 在 while True 循环中设置跳出条
件。
    if a1 == 'n':
        break
```

8 编程学习的两大瓶颈(编程思维)

学过就忘 → 用法查询笔记 → 深度理解笔记

解题不会 → 分析问题，明确结果 → 思考需要知识 → 思考切入点 → 尝试解决一部分 → 重复

8.1 课堂练习：九九乘法表

```
#=====课堂练习：九九乘法表=====#
'''
1 X 1 = 1
1 X 2 = 2  2 X 2 = 4
1 X 3 = 3  2 X 3 = 6  3 X 3 = 9
1 X 4 = 4  2 X 4 = 8  3 X 4 = 12  4 X 4 = 16
1 X 5 = 5  2 X 5 = 10  3 X 5 = 15  4 X 5 = 20  5 X 5 = 25
1 X 6 = 6  2 X 6 = 12  3 X 6 = 18  4 X 6 = 24  5 X 6 = 30  6 X 6 = 36
1 X 7 = 7  2 X 7 = 14  3 X 7 = 21  4 X 7 = 28  5 X 7 = 35  6 X 7 = 42  7 X 7 = 49
1 X 8 = 8  2 X 8 = 16  3 X 8 = 24  4 X 8 = 32  5 X 8 = 40  6 X 8 = 48  7 X 8 = 56  8 X 8 =
64
1 X 9 = 9  2 X 9 = 18  3 X 9 = 27  4 X 9 = 36  5 X 9 = 45  6 X 9 = 54  7 X 9 = 63  8 X 9 =
72  9 X 9 = 81
'''
```

```
#-----for 循环打印九九乘法表-----#
for i in range(1,10):
    for j in range(1,i+1):
        print( '%d X %d = %d' % (j,i,i*j),end = '  ' )
    print('  ')
```

```
#-----while 循环打印九九乘法表-----#
i = 1
while i <= 9:
    j = 1
    while j <= i:
        print('%d X %d = %d' % (j,i,i*j),end = '  ')
        j += 1
    print('')
    i += 1
```

8.2 课后练习：列表合并 extend()和排序 sort()

一次测评中，老师将 学习小组 A 和 学习小组 B 的测评成绩(满分 100 分)从低到高记录放进两个列表：

A=[91, 95, 97, 99]， B=[92, 93, 96, 98] 。

现在，老师想将两个小组的成绩合并为一个列表，并按照从低到高的顺序排序，你能帮老师完成吗？

```
list1 =  [91, 95, 97, 99]
list2 =  [92, 93, 96, 98]

# 把 A 组成绩赋值给一个新列表，用来存合并的成绩—这个细节要注意！
list3 = list1
list3.extend(list2)
print(list3)

list3.sort()
print(list3)
```

8.3 课后练习：列表平均分

老师有了新的需求：想知道两组的平均分，以及把低于平均分的成绩也打印出来。

所以，在这个练习中，我们会帮老师计算出两组的平均分，并挑出那些在平均分之下的成绩。

```
#-----方法一-----#
scores1 =  [91, 95, 97, 99, 92, 93, 96, 98]
sum = 0
scores2 = []

for score in scores1:
    sum = sum + score
    average = sum/len(scores1)
print('平均成绩是：{}'.format(average))

for score in scores1:
    if score < average:
        scores2.append(score)
        continue  # 少于平均分的成绩放到新建的空列表中，然后继续判断。
print(' 低于平均成绩的有：{}'.format(scores2))
```

```
#-----第二种方法-----# sum 函数
```

```
scores1 =  [91, 95, 97, 99, 92, 93, 96, 98]
average = sum(scores1)/len(scores1)
print('平均成绩是：{}'.format(average))

#-----第三种方法-----#导入 numpy 库，下面出现的 np 即 numpy 库
import numpy as np

scores1 =  [91, 95, 97, 99, 92, 93, 96, 98]
scores2 = []

average = np.mean(scores1)  # 一行解决。
print('平均成绩是：{}'.format(average))

#-----第四种方法-----#一种 NumPy 数组的操作
socres3 = np.array(scores1)
print(' 低于平均成绩的有：{}'.format(socres3[socres3<average]))
```

9 喊出我的名字(def 定义函数)

9.1 def 定义函数

```
#=====def 定义函数=====#
#函数是组织好的、可以重复利用的、用来实现单一功能的代码。
#函数名：最好是取体现函数功能的名字，一般用小写字母和单下划线、数字等组合
def greet(name):
    #参数：根据函数功能，括号里可以有多个参数，也可以不带参数，命名规则与函数名相同
    #规范：括号是英文括号，后面的冒号不能丢
    print(name+'早上好')
    #函数体：函数体就是体现函数功能的语句，要缩进，一般是四个空格
    return
    #return 语句，返回值，不写默认返回 None
greet('皮卡丘')
#>> 皮卡丘早上好
```

```
#-----函数的参数类型-----#
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')
# menu()函数中，appetizer,course 为位置参数，默认按位置传递，也可自定义传递
# dessert=''为默认参数，默认传递，也可自定义传递
# *barbeque 为不定长参数，上述('烤鸡翅','烤茄子','烤玉米')为一个元组会整体传递给*barbeque
# 元组是可迭代对象，可以用 for 循环遍历
```

```
order=('烤鸡翅','烤茄子','烤玉米')
def menu(*barbeque):
    print(barbeque)
menu(*order)
#元组的长度没有限制,可先生成元组再传入参数
```

```
#-----函数的 return 语句-----#
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'))
print(main('新垣结衣','长泽雅美'))
# ①dream_face='李若彤'，②face('李若彤')，③return 李若彤的脸蛋
# 1、如果不是立即要对函数返回值做操作，那么可以使用 return 语句保留返回值
# 2、需要多次调用函数时，可以再定义一个主函数 main(),调用非主函数的返回值
# 3、返回值可以是多个，多个返回值的数据类型是元组（tuple）
# 4、若无 return 语句，函数会返回空值（None）,例如 print()
# 5、一旦函数内部遇到 return 语句，会停止执行并返回结果，后面的 return 无效
```

```
#-----变量作用域-----#
quantity = 108              #这里的 quantity 是【全局变量】
def egg():
   print(quantity)          #函数内的局部作用域，可以访问全局变量
egg()

def egg():
    global quantity         #【global 语句】可以将局部变量声明为全局变量
    quantity = 108          #这里的 quantity 是【局部变量】
egg()
print(quantity)             #本来不能访问局部变量，但有 global 把他变为了全局变量
```

9.2 新知识：列表生成、extend()新用法

```

#-----新知识：列表生成方式、extend 新用法-----#
# 知识 1：一种新的列表生成方式
num1 = [1,2,3,4,5]          # 想一想，如果用这个方法生成一个 1-100 的列表……
num2 = list(range(1,6))
print(num1)                 #>> [1,2,3,4,5]
print(num2)                 #>> [1,2,3,4,5]

# 知识 2：extend 的新用法
num2.extend(['ABCDE'])
```

```
num2.extend('ABCDE')        # extend 后面是列表的话会将其合并，后面是字符串的话会将每个字符当成一个列表中的元素。
print(num2)                 #>> [1,2,3,4,5,'ABCDE','A','B','C','D','E']

# 知识点 3：列表生成式
list1 = [i for i in range(3)]                          # 规定列表中元素的范围
print(list1)                                           #>> [0,1,2]
list2 = [m+n for m in ['天字', '地字'] for n in '一二']  # 列表元素可以是组合，分别规定范围。
print(list2)                                           #>> ['天字一','天字二','地字一','地字二']
list3 = [n*n for n in range(1,11) if n % 3 == 0]       # 元素既可规定范围，也可附加条件。
print(list3)                                           #>> [9,36,81]

# 假设用普通的方法来生成上面的列表：
'''
list1 = []
for i in range(3):
    list1.append(i)
print(list1)

list2 = []
for m in ['天字', '地字']:
    for n in '一二':
        list2.append(m+n)
print(list2)

list3 = []
for i in range(1,11):
    if i % 3 == 0:
        list3.append(i*i)
print(list3)
'''
```

9.3 课后练习：抽奖器、扑克牌

```
#-----小游戏：打印圣诞树-----#
def tree(Height):
    print('Merry Christmas!')
    for i in range(Height):
        print((Height-i)*2*' '+'o'+ i*'~x~o')
        print(((Height-i)*2-1)*' '+(i*2+1)*'/'+'|'+(i*2+1)*'\\')
tree(8)
```

```
#-----课后练习：抽奖器（练习封装函数）-----#
import random
import time
# 将抽奖程序封装成函数
def choujiang(q,w,e):              # 定义一个抽奖函数，带有 3 个参数（候选人）
    luckylist = [q,w,e]            # 定义一个中奖名单的列表
```

```
    a = random.choice(luckylist)  # random.choice()为随机选择函数
    print('开奖倒计时',3)
    time.sleep(1)                 # 调用 time 模块，控制打印内容出现的时间
    print('开奖倒计时',2)
    time.sleep(1)
    print('开奖倒计时',1)
    time.sleep(1)
    image = '''
    /\_)o<
    |      \\
    | O . O|
    \_____/
    '''                           # 使用三引号打印 hellokitty 的头像
    print(image)
    print('恭喜'+a+'中奖！ ')       # 使用 print 函数打印幸运者名单
choujiang('虚竹','萧峰','段誉')    # 调用函数
```

```
#-----课后练习：生成一幅扑克牌-----#
'''
生成扑克牌：返回一个扑克牌列表，里面有 52 个元组，对应 52 张牌。
'''
def cards():
  color = ['红心', '方块', '梅花','黑桃']     # 将花色放在一个列表中待用
  num = list(range(2, 11))
  num.extend('JQKA')                        # 通过两行代码，生成一个 2-A 的数字列表。
  return [(x, y) for x in color for y in num ]  # 用列表生成式完成扑克牌的生成。

print(cards())
# 注：花色对应的正式单词是：suit 和 rank，上面为了好理解所以用了 color。
```