# Python 简介

1. 程序员：

程序设计人员。

1. 程序：

一组计算机能识别和执行的指令，是实现某种需求的软件。

1. 操作系统：

管理和控制计算机软件与硬件资源的程序；

隔离不同硬件的差异，使开发程序简单化。

例如，Windows，Linux，Unix。

1. 硬件：

主板--计算机的主要电路系统。

CPU --主要负责执行程序指令，处理数据。

硬盘--持久化存储数据的记忆设备，容量大，速度慢。

内存—临时存储数据的记忆设备，容量小，速度快。

只存储正在运行的程序数据。

IO设备--键盘、鼠标、显示器。

## Python 定义



是一个免费、开源、跨平台、动态、面向对象的编程语言。

## Python程序的执行方式

### 交互式

在命令行输入指令，回车即可得到结果。

1. 打开终端
2. 进入交互式：python3
3. 编写代码：print(“hello world”)
4. 离开交互式：exit()

### 文件式 （实际工程项目都用这种方式）

将指令编写到.py文件，可以重复运行程序。

1. 编写文件。
2. 打开终端
3. 进入程序所在目录：cd 目录
4. 执行程序： python3 文件名

## Linux常用命令

1. pwd：查看当前工作目录的路径
2. ls：查看指定目录的内容或文件信息
3. cd：改变工作目录（进入到某个目录）

练习：

1. 在指定目录创建python文件.

--目录：/home/tarena/1906/month01

--文件名：exercise01.py

2. 在文件中写入：print(“你好，世界!”)

3. 运行python程序

## 执行过程



计算机只能识别机器码(1010)，不能识别源代码(python)。

1. 由源代码转变成机器码的过程分成两类：编译和解释。
2. 编译：在程序运行之前，通过编译器将源代码变成机器码，例如：C语言。

-- 优点：运行速度快

-- 缺点：开发效率低，不能跨平台。

1. 解释：在程序运行之时，通过解释器对程序逐行翻译，然后执行。例如Javascript

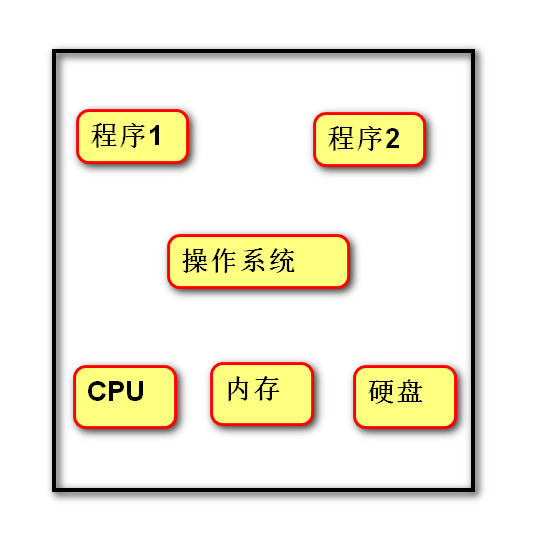
-- 优点：开发效率高，可以跨平台；

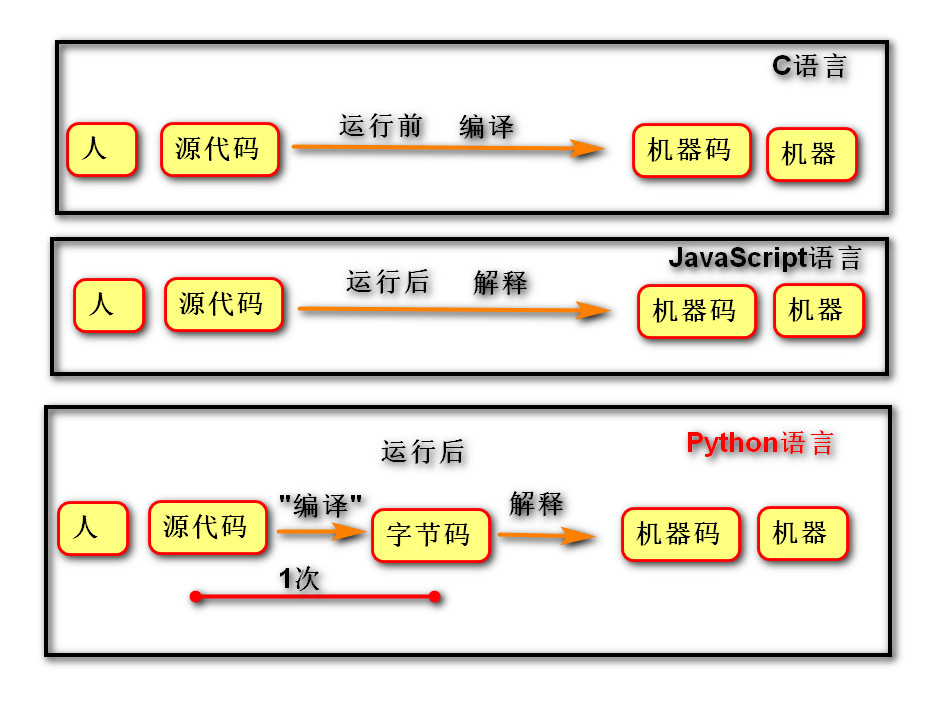
-- 缺点：运行速度慢。

1. python是解释型语言，但为了提高运行速度，使用了一种编译的方法。编译之后得到pyc文件，存储了字节码（特定于Python的表现形式，不是机器码）。

源代码 -- 编译 --> 字节码 -- 解释 --> 机器码

|————1次———| 只在第一次运行时编译一次，之后的运行则无



**

*"""*

*函数*

*"""*

# input

# 功能：让用户在终端中输入信息

# 参数：提示的信息

# 结果：用户输入的信息

# = 赋值号:　将右边的结果，复制一份，给左边.

result = input("请输入...:")

# print

# 功能：在终端中打印(输出／显示)信息

# 参数：需要打印的信息

# 结果：没有

print("你好，世界")

# 交互：问好的案例

name = input("请输入姓名：")

# +: 拼接

print("您好："+name+".")

## 解释器类型

1. CPython（C语言开发)
2. Jython (java开发)
3. IronPython (.net开发)

# 数据基本运算

## pycharm常用快捷键

1. 移动到本行开头：home键
2. 移动到本行末尾：end键盘
3. 注释代码：ctrl + /
4. 复制行：ctrl +d
5. 选择列：鼠标左键 + alt
6. 移动行：shift + ctrl + 上下箭头
7. 智能提示：Ctrl + Space
8. 代码规范：Ctrl + Alt +l

## 注释

给人看的，通常是对代码的描述信息。

1. 单行注释：以#号开头。
2. 多行注释：三引号开头，三引号结尾。

## 函数

表示一个功能，函数定义者是提供功能的人，函数调用者是使用功能的人。

例如：

1. print(数据) 作用：将括号中的内容显示在控制台中
2. 变量 = input(“需要显示的内容”) 作用：将用户输入的内容赋值给变量

## 变量

1. 定义：关联一个对象的标识符。
2. 命名：必须是字母或下划线开头，后跟字母、数字、下划线。

不能使用关键字(蓝色)，否则发生语法错误：SyntaxError: invalid syntax。

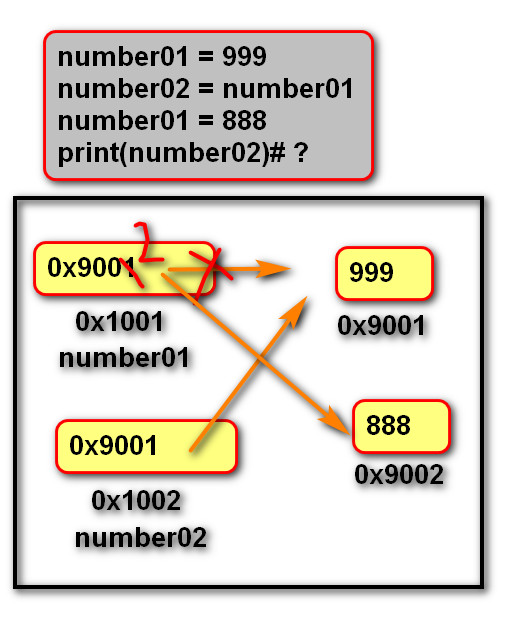
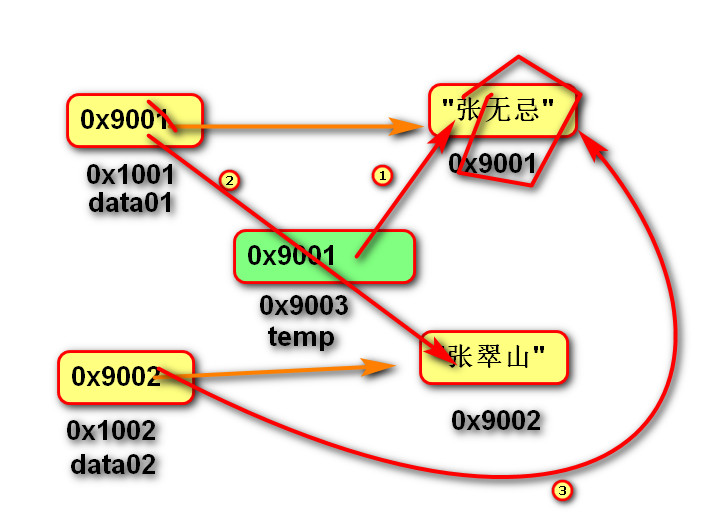
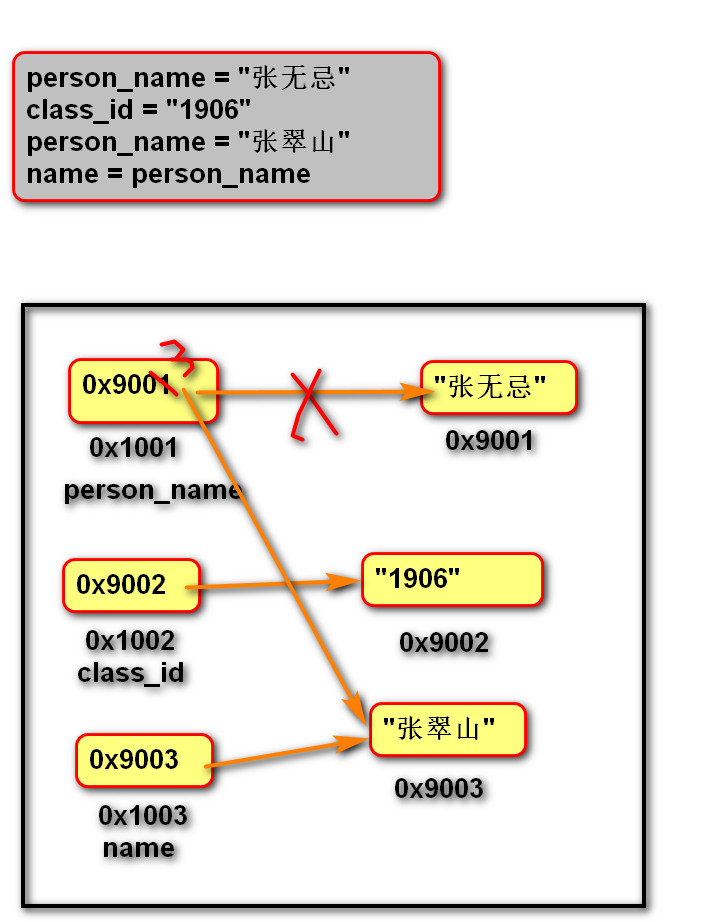
1. 建议命名：字母小写，多个单词以下划线隔开。

class\_name

1. 赋值：创建一个变量或改变一个变量关联的数据。
2. 语法：变量名 = 数据

变量名1 = 变量名2 = 数据

变量名1, 变量名2, = 数据1, 数据2

**

person\_name = "张无忌"

class\_id = "1906"

person\_name = "张翠山"

name = person\_name

# 将括号中的内容输出到终端中

print(person\_name)

# 格式：xx班级的xx.

print(class\_id + "班级的" + person\_name)

data01 = input("请输入第一个变量：")

data02 = input("请输入第二个变量：")

temp = data01

data01 = data02

data02 = temp

print(data01)

print(data02)

number01 = 999

number02 = number01

number01 = 888

print(number02)

## del 语句

1. 语法:

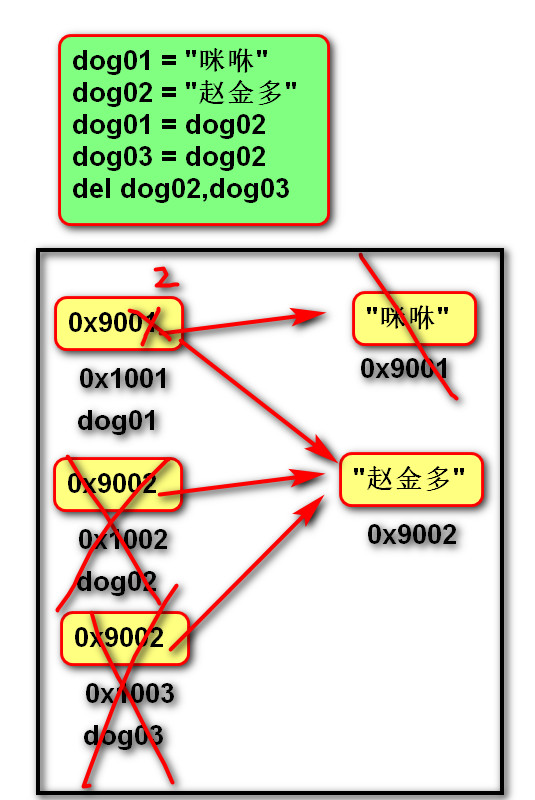
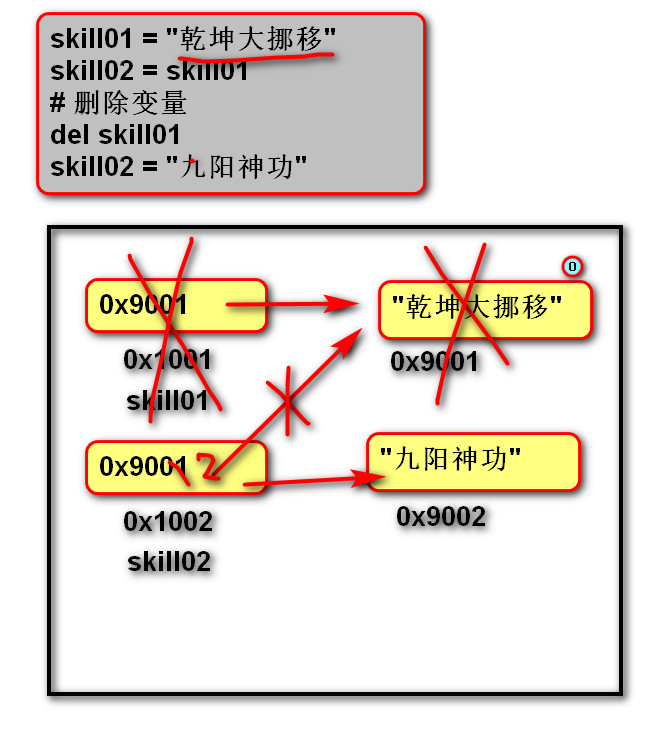
del 变量名1, 变量名2

1. 作用：

用于删除变量,同时解除与对象的关联.如果可能则释放对象。

1. 自动化内存管理的引用计数：

每个对象记录被变量绑定(引用)的数量,当为0时被销毁。



skill01 = "乾坤大挪移"

skill02 = skill01

# 删除变量

del skill01

skill02 = "九阳神功"

dog01 = "咪咻"

dog02 = "赵金多"

dog01 = dog02

dog03 = dog02

del dog02,dog03

# "金多"　还在(因为变量dog01引用着)

print(dog01)

# 访问已经删除的变量，会引发错误.

# print(dog02)

## 核心数据类型

1. 在python中变量没有类型，但关联的对象有类型。
2. 通过type函数可查看。

### 空值对象 None

1. 表示不存在的特殊对象。
2. 作用：占位和解除与对象的关联。

### 整形int

1. 表示整数，包含正数、负数、0。

如： -5, 100, 0

1. 字面值：

十进制：5

二进制：0b开头，后跟1或者1

八进制：0o开头，后跟0~7

十六进制：0x开头，后跟0~9,A~F,a~f

1. 小整数对象池：CPython 中整数 -5 至 256,永远存在小整数对象池中,不会被释放并可重复使用。

### 浮点型float

1. 表示小数，包含正数、负数，0.0)。
2. 字面值：

小数：1.0 2.5

科学计数法：e/E (正负号) 指数

1.23e-2 (等同于0.0123)

1.23456e5(等同于123456.0)

### 字符串str

是用来记录文本信息(文字信息)。

字面值：双引号

### 复数complex

由实部和虚部组成的数字。

虚部是以j或J结尾。

字面值： 1j 1+1j 1-1j

### 布尔bool

用来表示真和假的类型

True 表示真(条件满足或成立)，本质是1

False 表示假(条件不满足或不成立)，本质是0

## 数据类型转换

1. 转换为整形: int(数据)
2. 转换为浮点型:float(数据)
3. 转换为字符串:str(数据)
4. 转换为布尔:bool(数据)

结果为False：bool(0) bool(0.0) bool(None)

1. 混合类型自动升级：

1 + 2.14 返回的结果是 3.14

1 + 3.0 返回结果是: 4.0

*核心数据类型*

*变量没有类型，关联的对象才有类型.*

*"""*

# 1. None 空 NoneType

# 占位:只希望有个变量，指向的对象还不确定。

name = None

skill01 = "乾坤大挪移"

# 解除"乾坤大挪移" 与　变量 skill01 的关系

skill01 = None

# 2. 整形(整数)int

# 十进制：每逢十进一位　0 1 2 3 ...10

number01 = 250

# 二进制：每逢二进一位　0 1 10 11 100 101

number02 = 0b100

# 八进制：每逢八进一位　0 1 2 .. 7 10 11

number03 = 0o10

# 十六进制：每逢十六进一位0 1 3 ...9 a(10) f(15)

number04 = 0xf

print(number04)

# 3. 浮点型(小数)float

number05 = 10.5

# 科学计数法

number06 = 1.23456e5

print(number06) # 123456.0

number07 = 0.00000000000000000000000005

number08 = 5e-26

print(0.00001)

# 4. 字符串str

message01 = "我爱编程"

number09 = "100.5"

print(type(name))

# 5. \*复数 complex

num01 = 10 + 1.5j

print(type(num01))

# 6. 类型转换

# input 函数的结果是字符串类型

str\_age = input("请输入年龄：") # "18" + 1

# str --> int

int\_age = int(str\_age) # 19

# int --> str

print("明年是：" + str(int\_age)) # "明年是：" 19

## 运算符

### 算术运算符

+ 加法

- 减法

\* 乘法

/ 除法：结果为浮点数

// 地板除：除的结果去掉小数部分

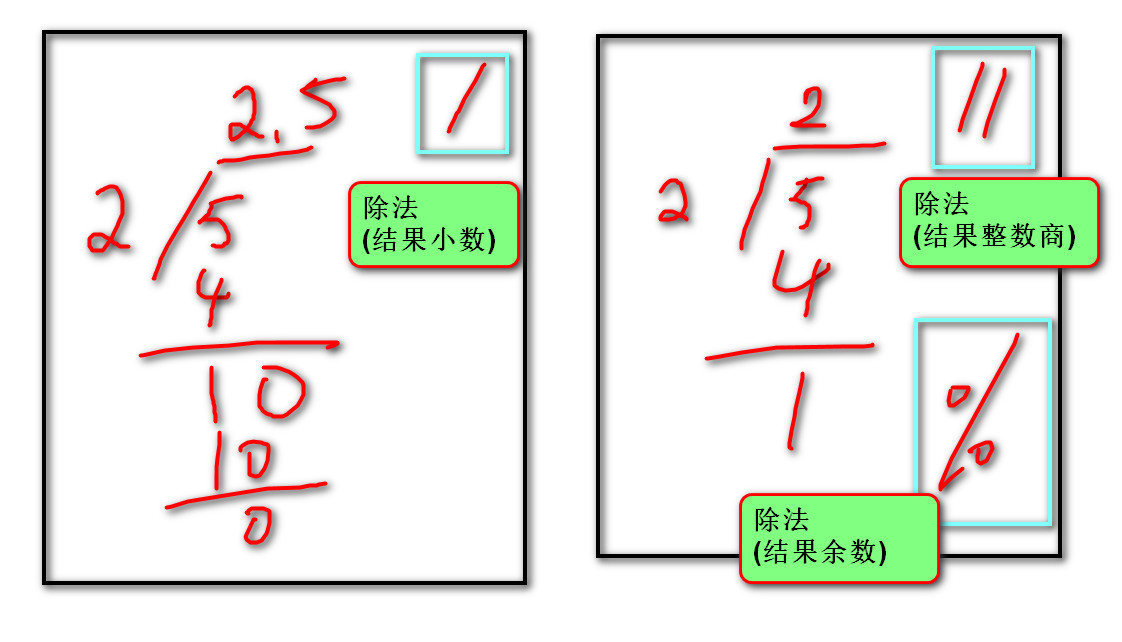
% 求余

\*\* 幂运算

优先级从高到低： ()

\*\*

\* / % //

 + -

### 增强运算符

y += x 等同于 y = y + x

y -= x 等同于 y = y - x

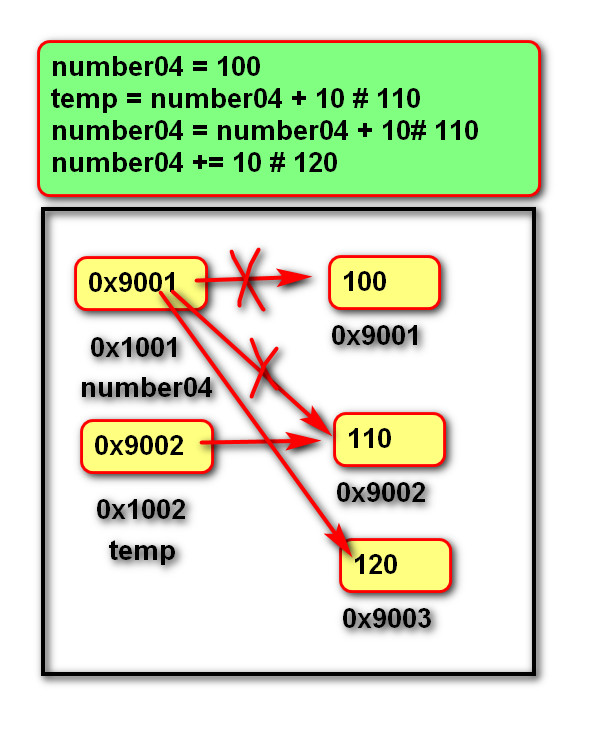
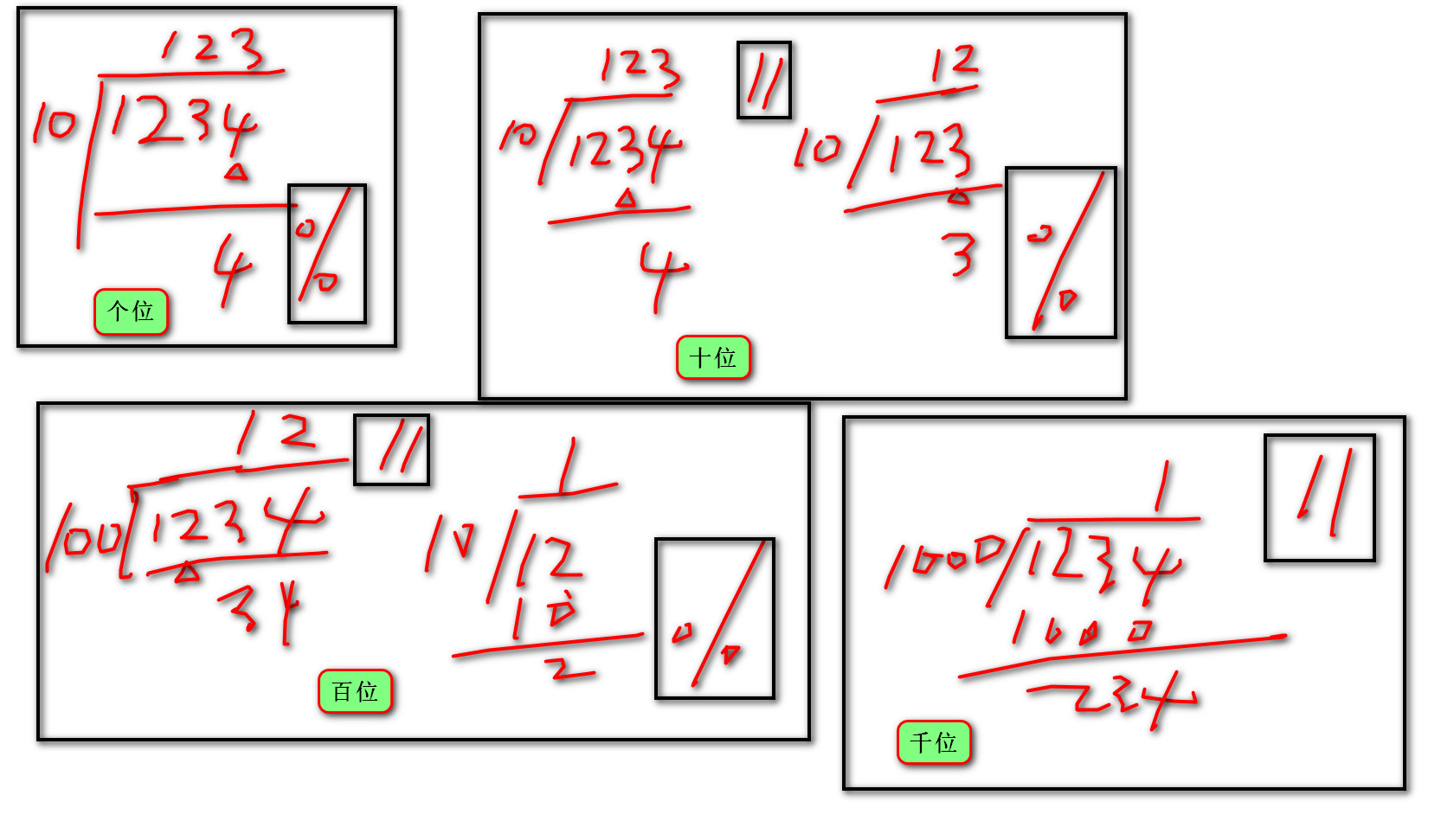
y \*= x 等同于 y = y \* x

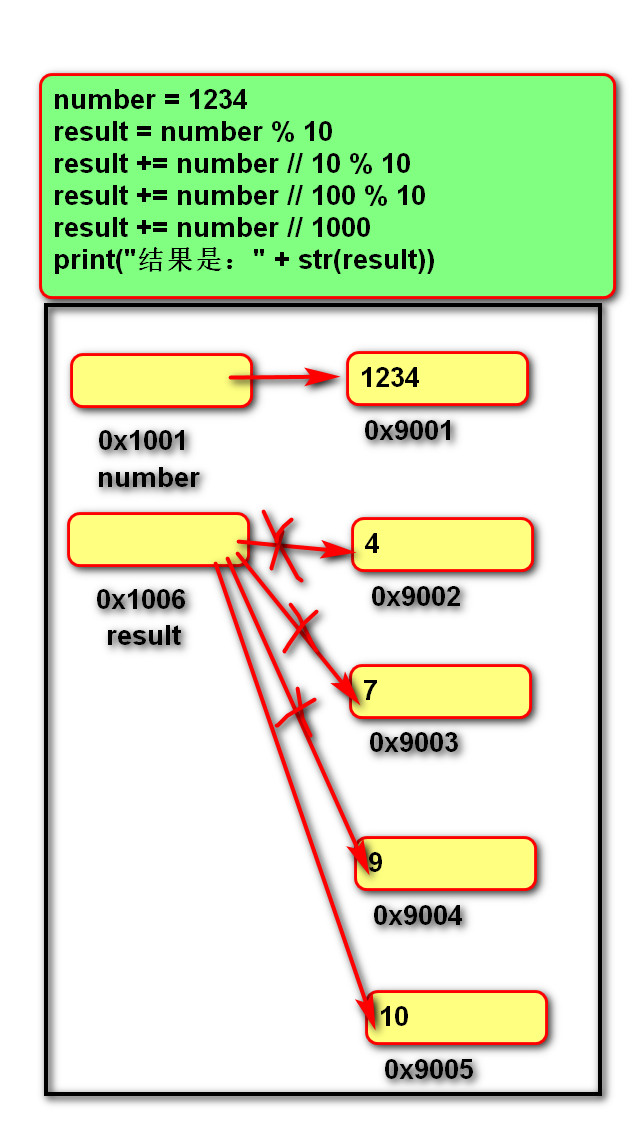
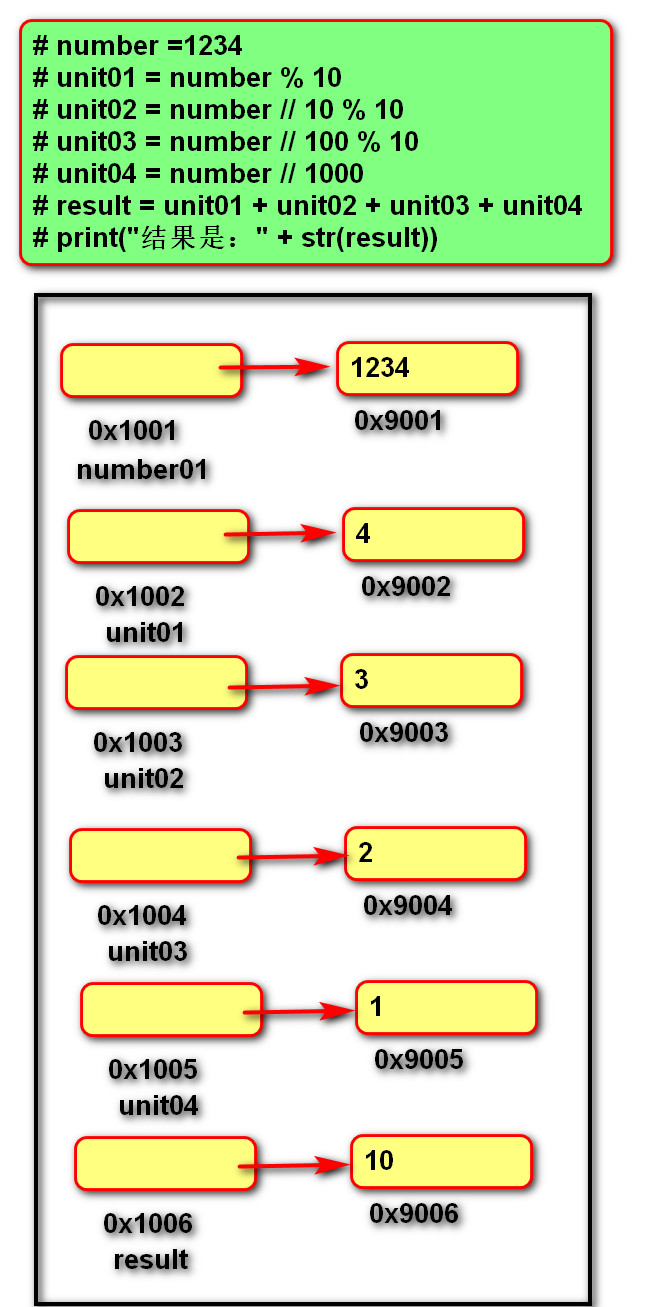
y /= x 等同于 y = y / x

y //= x 等同于 y = y // x

y %= x 等同于 y = y % x

y \*\*= x 等同于 y = y \*\* x





### 比较运算符

< 小于

<= 小于等于

> 大于

>= 大于等于

== 等于

!= 不等于

返回布尔类型的值

比较运算的数学表示方式:0 <= x <= 100

*运算符*

*算数运算符*

*+ - \* / // % \*\**

*增强运算符*

*+= -= \*= /= //= %= \*\*=*

*练习:exercise05.py*

*exercise06.py*

*exercise07.py*

*exercise08.py*

*exercise09.py*

*"""*

number01 = 5

number02 = 2

# result = number01 / number02 # 2.5

# result = number01 // number02 # 2

result = number01 % number02 # 2

print(result) # ?

number03 = 4 \*\* 2

print(number03)# 4 \* 4

# 增强运算符

number04 = 100

# print(number04 + 10)# 110

# print(number04)# ?100

temp = number04 + 10 # 110

print(number04)# 100

# # 变量 = 变量　+ 数据

number04 = number04 + 10

print(number04)# 110

# 变量 += 数据(改变自身变量)

number04 += 10

print(number04)# 110

### 逻辑运算符

#### 与and

表示并且的关系，一假俱假。

示例:

True and True # True

True and False # False

False and True # False

False and False # False

#### 或or

表示或者的关系，一真俱真

示例:

True or True # True

True or False # True

False or True # True

False or False # False

#### 非 not

表示取反

例如：

not True # 返回False

not False # 返回True

*bool*

*比较运算符*

*逻辑运算符*

*练习:exercise10.py*

*"""*

# 1. bool　 int 类型

# True False 10 20 30 .. 数据

# 命题：带有判断性质的陈述句.

# 我是个男人 对的/真的(True)

# 我是个坏人 错的/假的(False)

# 1 > 2 错的/假的(False)

# 2. 比较运算符 > < >= <= == != 结果是bool类型

# 结果 = 变量1 > 变量2

print(1 > 2) # False

# 3. 逻辑运算符 与and　　或or　　非not

# 判断两个命题(bool值)关系

# 总结：一假俱假　　表达的是"并且”的关系(必须都满足)

print(True and True)# True

print(False and True)# False

print(True and False)# False

print(False and False)# False

# 总结：一真俱真　表达的是"或者”的关系(满足一个就行)

print(True or True)# True

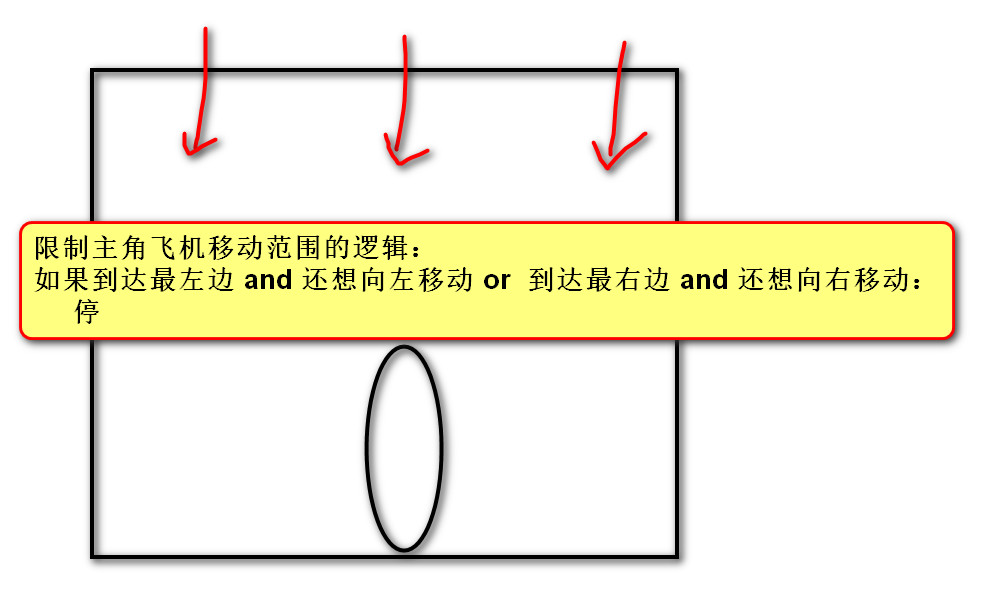
print(False or True)# True

print(True or False)# True

print(False or False)# False

# 非

print(not True)



#### 短路运算

一但结果确定，后面的语句将不再执行。将耗时的运算放在后面，可提高性能。

### 身份运算符

语法:

x is y

x is not y

作用：

is 用于判断两个对象是否是同一个对象,是时返回True,否则返回False。

is not 的作用与is相反

*身份运算符*

*"""*

a = 500

b = a

c = 1000

print(id(a)) # 返回变量所指向的数据的内存地址

print(id(b))

print(id(c))

print(a is b) # 内部：判断的就是id函数的结果

print(a is c)

### 优先级

高到低：

算数运算符

比较运算符

快捷运算符

身份运算符

逻辑运算符

# 语句

## 行

1. 物理行：程序员编写代码的行。
2. 逻辑行：python解释器需要执行的指令。
3. 建议一个逻辑行在一个物理行上。
4. 如果一个物理行中使用多个逻辑行，需要使用分号；隔开。

5. 如果逻辑行过长，可以使用隐式换行或显式换行。

隐式换行：所有括号的内容换行,称为隐式换行

括号包括: () [] {} 三种

显式换行：通过折行符 \ (反斜杠)换行，必须放在一行的末尾，目的是告诉解释器,下一行也是本行的语句。

# 三个物理行，三个逻辑行

a = 10

b = 20

c = a + b

# 一个物理行，三个逻辑行(不建议)

a = 10;b = 20;c = a + b

# 三个物理行，一个逻辑行(适用于一行太长时使用)

# 折行符

d = 1+\

2+3+\

4+5

# 括号是天然的折行符

e = 1+(2+3

+4)\*5

## pass 语句

通常用来填充语法空白。

## 选择语句

### If elif else 语句

1. 作用:

让程序根据条件选择性的执行语句。

1. 语法:

if 条件1:

语句块1

elif 条件2:

语句块2

else:

语句块3

1. 说明:

elif 子句可以有0个或多个。

else 子句可以有0个或1个，且只能放在if语句的最后。

*选择语句*

*if 条件bool:*

*满足条件执行的代码*

*else:*

*不满足条件执行的代码*

*if 条件1:*

*满足条件1执行的代码*

*elif 条件2：*

*满足条件2执行的代码*

*else:*

*以上都不满足执行的代码*

*练习:exercise01.py ~ exercise06.py*

*"""*

sex = input("请输入性别：")

# 缩进不是tab，而是四个空格.

if sex == "男":

print("您好，先生！")

elif sex == "女":

print("您好，女士！")

else:

print("性别未知")

# 调试：让程序在指定的行中断，逐语句执行.

# 查看程序执行过程，查看变量的取值。

# 步骤：

# 1. 在可能出错的行，加断点.

# 2. 开始调试Shift + F9.

# 3. 逐语句执行 F8.

# 4. .....

# 5. 停止调试Ctrl + F2

### if 语句的真值表达式

if 100:

print("真值")

等同于

if bool(100):

print("真值")

*if的真值表达式*

*if 变量:*

*变量存在数据则执行*

*条件表达式：*

*有选择性的为变量进行赋值*

*练习:exercise07.py*

number = 10

if number != 0:

print("不是零")

# 1. if的真值表达式

if number:

# if bool(number):

print("不是零")

# 2. 条件表达式：

if input("请输入性别：") == "男":

sex\_id = 1

else:

sex\_id = 0

sex\_id = 1 if input("请输入性别：") == "男" else 0

print(sex\_id)

### 条件表达式

语法：变量 = 结果1 if 条件 else 结果2

作用：根据条件(True/False) 来决定返回结果1还是结果2。

## 循环语句

### while语句

1. 作用:

可以让一段代码满足条件，重复执行。

1. 语法:

while 条件:

满足条件执行的语句

else:

不满足条件执行的语句

1. 说明:

else子句可以省略。

在循环体内用break终止循环时,else子句不执行。

*循环语句*

*while 条件:*

*循环体*

# 死循环：循环条件永远满足

while True:

str\_usd = input("请输入美元：")

int\_usd = int(str\_usd)

rmb = int\_usd \* 6.86

print("人民币是："+ str(rmb))

if input("按下q键退出:") == "q":

break # 退出循环

*while 执行预定次数*

count = 0

while count < 5: # 0 1 2 3 4

print("跑圈")

# print(count)

count += 1

*"""*

*猜数字2.0版本*

*如果猜错3次，退出循环，提示 次数到达3次.*

*练习：exercise04.py*

*"""*

# 生成随机数的工具

import random

# 产生一个随机数

random\_number = random.randint(1, 100)

print(random\_number)

count = 0

while count < 3:

count += 1

input\_number = int(input("请输入："))

if input\_number > random\_number:

print("大了")

elif input\_number < random\_number:

print("小了")

else:

print("恭喜猜对了，总共猜了" + str(count) + "次。")

break

else:# 可以判断循环是从条件离开的，还是从循环体中离开(break)的.

print("次数到达3次")

print("后续逻辑")

### for 语句

1. 作用:

用来遍历可迭代对象的数据元素。

可迭代对象是指能依次获取数据元素的对象，例如：容器类型。

1. 语法:

for 变量列表 in 可迭代对象:

语句块1

else:

语句块2

1. 说明:

else子句可以省略。

在循环体内用break终止循环时,else子句不执行。

*"""  
 for 循环嵌套  
 练习：exercise04.py  
 练习：exercise05.py  
 练习：exercise06.py  
 练习：exercise07.py  
"""*# 外层循环做1次，内层循环做3次  
# 外层循环控制行  
# 内层循环控制列  
for r in range(2):# 0 1  
 for c in range(3):#012 012  
 print("\*",end =" ")# 一行打印  
 print()# 换行

### range 函数

1. 作用:

用来创建一个生成一系列整数的可迭代对象(也叫整数序列生成器)。

1. 语法:

range(开始点，结束点，间隔)

1. 说明:

函数返回的可迭代对象可以用for取出其中的元素

返回的数字不包含结束点

开始点默认为0

间隔默认值为1

*for*

*for 变量* *in 可迭代对象:*

*循环体*

*练习：exercise05.py*

*exercise06.py*

# for item in "我爱Python":

# print(item)

# range(开始值,结束值,步长) --> 整数生成器

# 不包含结束值

# for item in range(0,5,1):# 0 1 2 3 4

# print(item)

# range(结束值)

# for item in range(5):# 0 1 2 3 4

# print(item)

# range(开始值,结束值)

# for item in range(2,6):# 2 3 4 5

# print(item)

# 倒序

# for item in range(6,3,-1):#6 5 4

# print(item)

# 跳着

# for item in range(2,11,2):#2 4 6 8 10

# print(item)

# for + range 特别适合做预定次数的循环.

for item in range(50):

print(item)

## 跳转语句

### break 语句

1. 跳出循环体，后面的代码不再执行。
2. 可以让while语句的else部分不执行。

### continue 语句

跳过本次，继续下次循环。

*continue*

*练习:exercise07.py*

*"""*

# 累加1 -- 50 之间，能被6整除的数字.

# sum\_value = 0

# for item in range(1,51):

# # 如果能被6整除则累加

# if item % 6 == 0:

# sum\_value += item

#

# print(sum\_value)

sum\_value = 0

for item in range(5,51):

# 如果不能被6整除则跳过

if item % 6 != 0:

continue

sum\_value += item

print(sum\_value)

# 容器类型

## 通用操作

### 数学运算符

1. +：用于拼接两个容器
2. +=：用原容器与右侧容器拼接,并重新绑定变量
3. \*：重复生成容器元素
4. \*=：用原容器生成重复元素, 并重新绑定变量
5. < <= > >= == !=：依次比较两个容器中元素,# 方法一：一但不同则返回比较结果。

#1. 数学运算符

# 拼接 + +=

str01 = "悟空" +"八戒"

print(id(str01))# 139637893711808

# 产生了新的字符串"悟空八戒唐僧"

str01 += "唐僧"# 139637893711984

print(id(str01))

# 乘法：重复产生

str02 = "白雪公主"

str02 \*= 10

print(str02)

# 比较运算：依次比较两个容器中元素,一但不同则返回比较结果

# 字符串比较的是编码值

str03 = "ac悟空"

str04 = "b唐僧"

print(str03 > str04)

### 成员运算符

1. 语法：

数据 in 序列

数据 not in 序列

1. 作用：

如果在指定的序列中找到值，返回bool类型。

# 2. 成员运算

re = "八戒" in str01

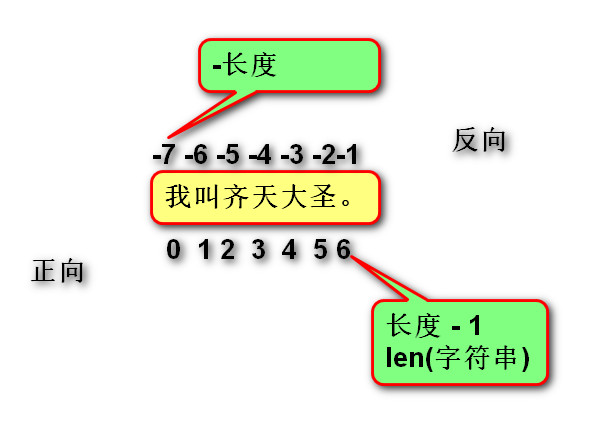
print(re)

### 索引index

1. 作用：访问容器元素
2. 语法：容器[整数]
3. 说明：

正向索引从0开始，第二个索引为1，最后一个为len(s)-1。

反向索引从-1开始,-1代表最后一个,-2代表倒数第二个,以此类推,第一个是-len(s)。



# 3. 索引:定位单个元素

message = "我叫齐天大圣。"

# 获取第一个字符

print(message[0])

print(message[-len(message)])

# 获取最后一个字符

print(message[-1])

print(message[len(message)-1])

# IndexError: string index out of range

# 索引越界

# print(message[500])

### 切片slice

1. 作用：

从容器中取出相应的元素重新组成一个容器。

1. 语法：

容器[(开始索引):(结束索引)(:(步长))]

1. 说明：

小括号()括起的部分代表可省略

结束索引不包含该位置元素

步长是切片每次获取完当前元素后移动的偏移量

# 4. 切片:定位多个元素

print(message[2:6])# 齐天大圣

print(message[2:6:2])# 齐大

# 开始索引，默认从头开始。

print(message[:6])# 我叫齐天大圣

# 结束索引，默认到末尾。

print(message[2:])# 齐天大圣。

# 正向获取

print(message[:])# 我叫齐天大圣。

# 反向获取

print(message[::-1])# 。圣大天齐叫我

print(message[-3:-6:-1])# 大天齐

# 可以同时使用反向与正向索引

print(message[-3:1:-1])# 大天齐

print(message[3:1])# 空

print(message[3:1:-1])# 天齐

print(message[1:1])# 空

print(message[-2:1])# 空

print(message[1:500])# 叫齐天大圣。

### 内建函数

1. len(x) 返回序列的长度
2. max(x) 返回序列的最大值元素
3. min(x) 返回序列的最小值元素
4. sum(x) 返回序列中所有元素的和(元素必须是数值类型)

## 字符串 str

### 定义

由一系列字符组成的不可变序列容器，存储的是字符的编码值。

### 编码

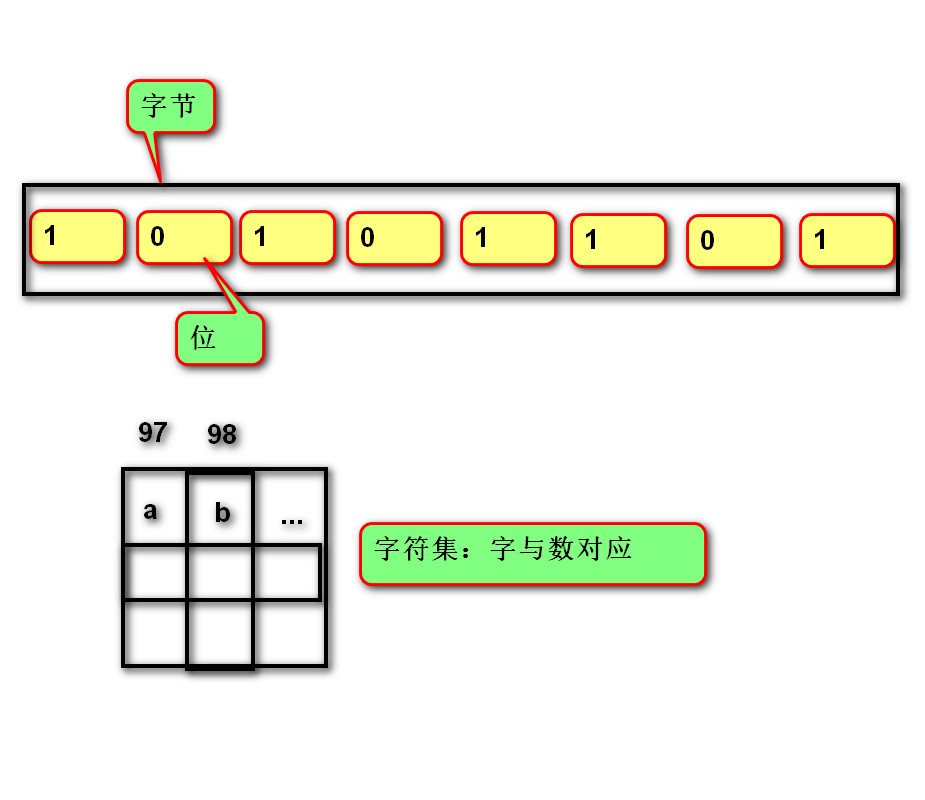
1. 字节byte：计算机最小存储单位，等于8 位bit.
2. 字符：单个的数字，文字与符号。
3. 字符集(码表)：存储字符与二进制序列的对应关系。
4. 编码：将字符转换为对应的二进制序列的过程。
5. 解码：将二进制序列转换为对应的字符的过程。
6. 编码方式：

--ASCII编码：包含英文、数字等字符，每个字符1个字节。

--GBK编码：兼容ASCII编码，包含21003个中文；英文1个字节，汉字2个字节。

--Unicode字符集：国际统一编码，旧字符集每个字符2字节，新字符集4字节。

-- UTF-8编码：Unicode的存储与传输方式，英文1字节，中文3字节。



#### 相关函数

1. ord(字符串):返回该字符串的Unicode码。
2. chr(整数):返回该整数对应的字符串。

*str 编码*

*练习:exercise08.py*

*"""*

name = "悟空"

name = "孙悟空"

# 数 --> 字

str01 = chr(97)

print(str01)

# 字 --> 数

number = ord("b")

print(number)

### 字面值

#### 单引和双引号的区别

1. 单引号内的双引号不算结束符
2. 双引号内的单引号不算结束符

#### 三引号作用 （所见即所得）

1. 换行会自动转换为换行符\n
2. 三引号内可以包含单引号和双引号
3. 作为文档字符串

#### 转义字符

1. 改变字符的原始含义。

\’ \” \””” \n \\ \t \0 空字符

1. 原始字符串：取消转义。

a = r”C:\newfile\test.py”

*str 字面值*

*"""*

name = "悟空"

name = '悟空'

# 可见即所得

name = """悟空"""

name = '''

悟

空

'''

message = """我叫"齐'天'大圣"."""

print(message)

# 转义符:改变原有字符含义的特殊字符

# \" \' \n换行 \t tab键 \\

message ="我\n叫\"齐天\t大圣\"."

print(message)

url = "C:\\appData\\boaming\\ciniconfig"

# 原始字符串:没有转义符的字符串

url = r"C:\appData\boaming\ciniconfig"

print(url)

#### 字符串格式化

1. 定义：

生成一定格式的字符串。

1. 语法：

字符串%(变量)

"我的名字是%s,年龄是%s" % (name, age)

1. 类型码：

%s 字符串 %d整数 %f 浮点数

*字符串格式化*

*"..%s..%s.."%(变量,变量)*

*练习:exercise10*

*"""*

name = "孙悟空"

age = 800

score = 85.62

# 我叫：xx，今年:xx岁.

str01 = "我叫：" + name + "，今年:" + str(age) + "岁."

str02 = "我叫：%s，今年:%d岁,成绩是:%.1f。"%(name, age, score)

print(str02)

## 列表 list

### 定义

由一系列变量组成的可变序列容器。

### 基础操作

1. 创建列表：

列表名 = [元素]

列表名 = list(可迭代对象)

1. 添加元素：

列表名.append(元素)

列表.insert(索引，元素)

1. 定位元素：

索引、切片

1. 遍历列表：

正向：

for 变量名 in 列表名:

变量名就是元素

反向：

for 索引名 in range(len(列表名)-1,-1,-1):

列表名[索引名]就是元素

1. 删除元素：

列表名.remove(元素)

del 列表名[索引或切片]

*列表：*

*适用性:存储多个数据.*

*练习:exercise01.py*

*exercise02.py*

*exercise03.py*

*"""*

# 1. 创建列表

# 空列表

list01 = []

list01 = list()

# 具有默认值的列表

list02 = [1,"a",True]

# list02 = list(可迭代对象)

# str --> 列表

list02 = list("齐天大圣")

# range --> 列表

list02 = list(range(5))

# 2. 增加

# 追加

list02.append("qtx")

# 插入(索引,元素)

list02.insert(3,"lzmly")

# 3. 删除

# 根据元素删除

list02.remove(4)

# 根据索引删除

del list02[3]

# 4. 修改

list02[0] = "第一个元素"

print(list02)

# 5. 切片:定位多个元素

# 获取 (重新创建新列表)

# 前三个元素

list03 = list02[:3]

# 修改

# 修改后两个元素

list02[-2:] = ["倒数第2个元素","倒数第1个元素"]

print(list02)

# 通过切片定位2个元素，修改为3个元素

list02[2:4] = ["a","b","c"]

# 通过切片定位3个元素，修改为0个元素 [删除]

list02[2:5] = []

print(list02)

# 6. 查询

# 获取所有元素

for item in list02:

print(item)

# 倒序获取所有

# 不建议通过切片(因为重新创建新列表,索引不建议.)

# for item in list02[::-1]:

# print(item)

# 建议通过索引

#2 1 0

# for i in range(2,-1,-1):

for i in range(len(list02)-1, -1, -1):

print(list02[i])

*内存图*

*练习：exercise04.py*

*exercise05.py*

*exercise06.py*

*"""*

list01 = ["无忌","翠山","三丰"]

# 将list01中存储的列表地址，赋值给list02.

list02 = list01

# 修改列表第一个元素

# list01[0] = "张无忌"

# 修改变量list01的引用(指向)

list01 = "张无忌"

print(list02[0])

# ----------------练习------------------------

# 15:15上课

list01 = [800,900,1000]

list02 = list01

list03 = list01

list01[0] = "八百"

print(list02[0])# "八百"

list03 = "九百"

print(list02)# ["八百",900,1000]

list01 = [800,900,1000]

# 通过切片创建新列表，赋值给变量list02

list02 = list01[:]

list01[0] = "八百"

print(list02[0])#?

list01 = [800,900,1000]

list02 = list01

# 定位第二个元素，然后进行修改.

list01[1:2] = ["a","b"]

print(list02)# [800, 'a', 'b', 1000]

list01 = [100,[200,300]]

list02 = list01

list01[1][0] = 500

print(list02[1][0])# 500

#------------------------

list01 = [100,[200,300]]

# list02 = list01[:]# 浅拷贝

list02 = list01.copy()# 浅拷贝

list01[1][0] = 500

print(list02[1][0])# 500

list01 = [100,200]

list02 = [100,200]

print(list01 == list02)#true 比较的是内容

print(list01 is list02)#false 比较的是地址 id(list01) == id(list02)

# 准备拷贝工具

import copy

list01 = [100,[200,300]]

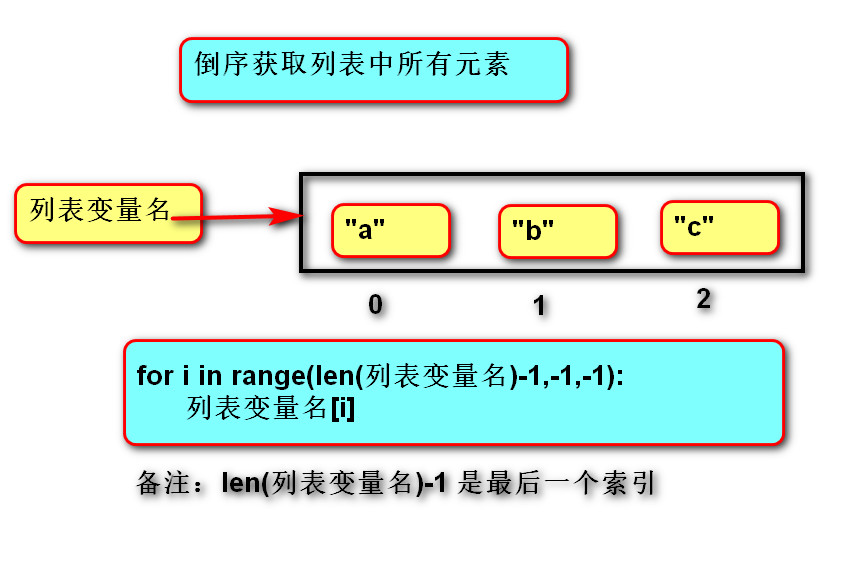
# 深拷贝(“划清界限”：拷贝前与拷贝后的对象互不影响)

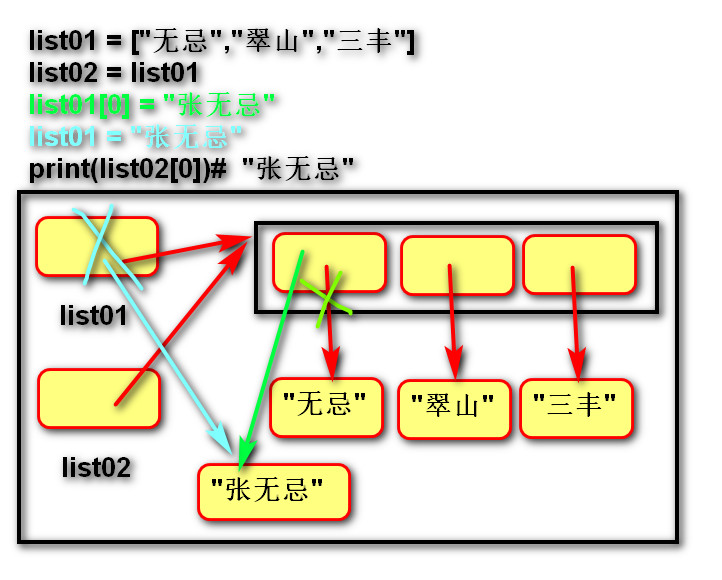
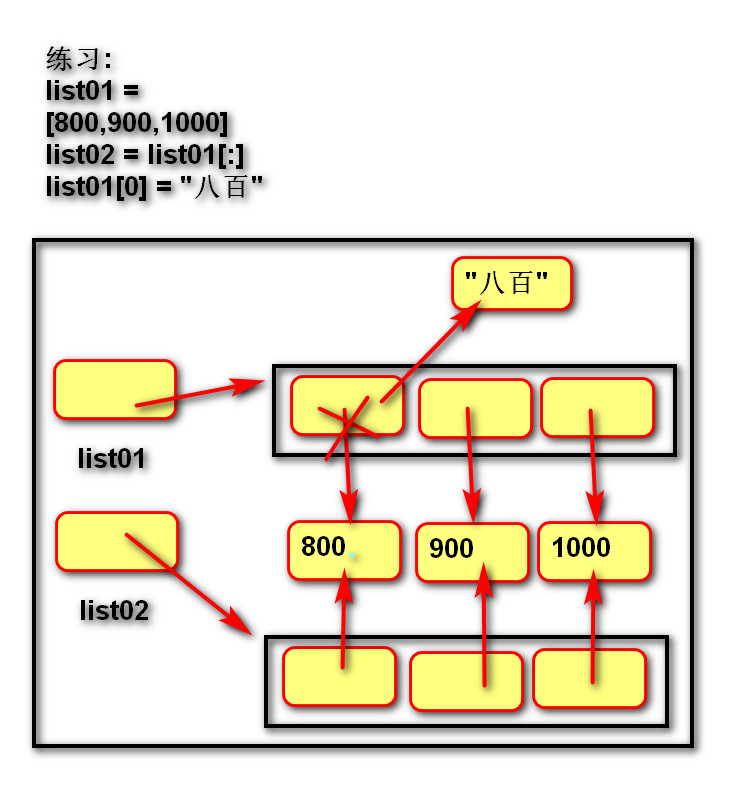
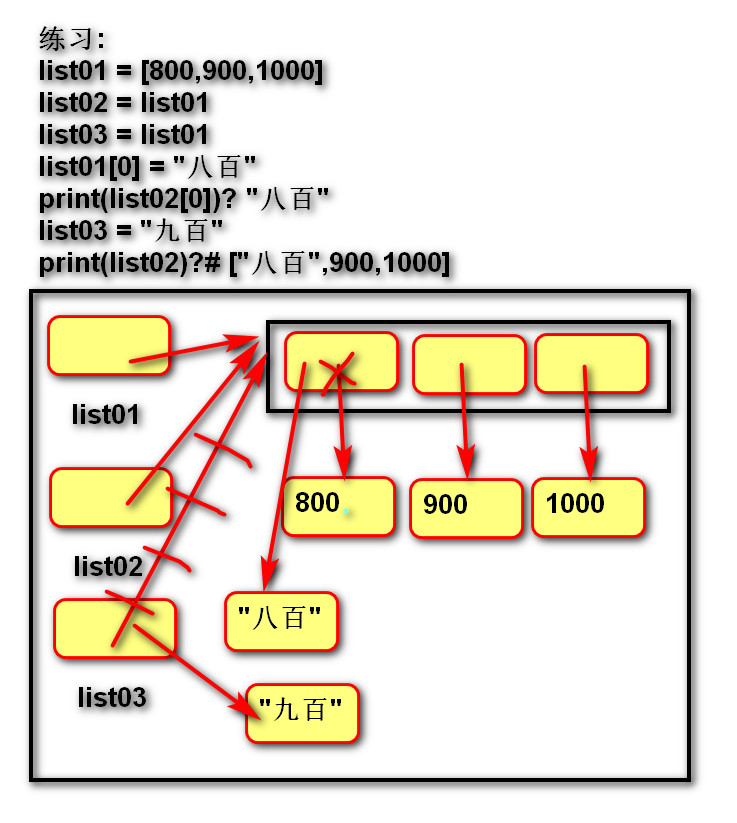
# 注意：深拷贝可能占用内存过大.

list02 = copy.deepcopy(list01)

list01[1][0] = 500

print(list02[1][0])# 200





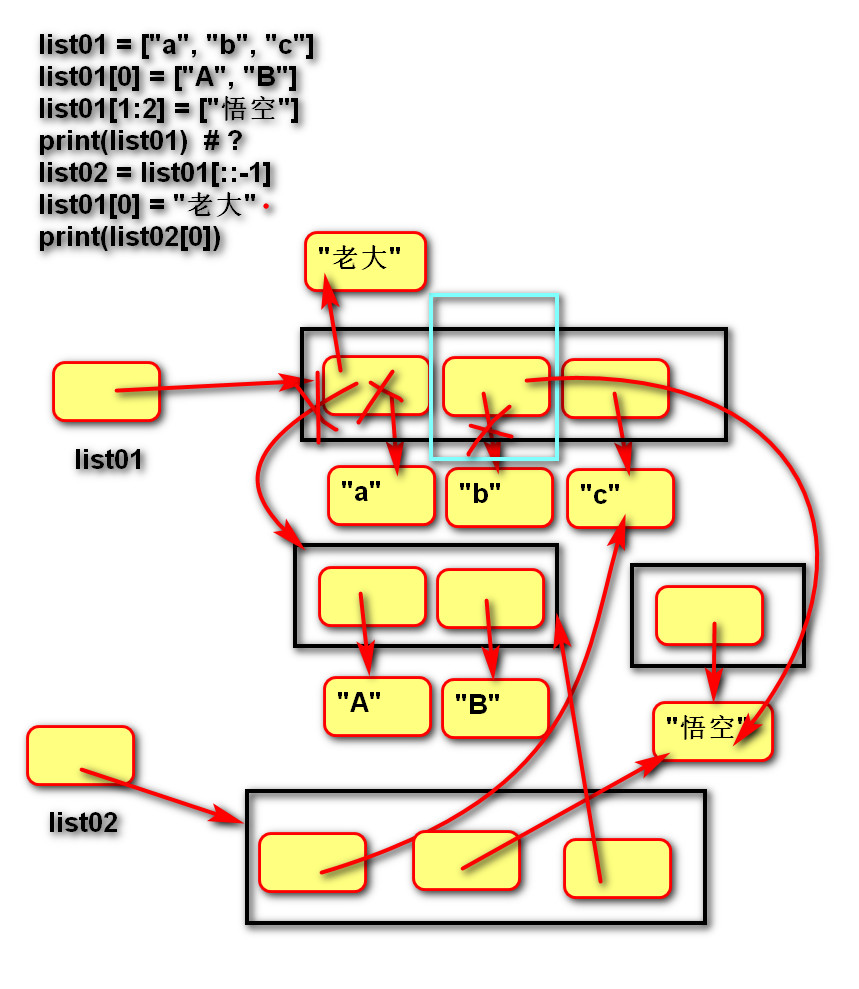
### 

### 

### 深拷贝和浅拷贝

浅拷贝：复制过程中,只复制一层变量,不会复制深层变量绑定的对象的复制过程。

深拷贝：复制整个依懒的变量。 注意： 拷贝后占用内存过大。



### 列表VS字符串

1. 列表和字符串都是序列,元素之间有先后顺序关系。
2. 字符串是不可变的序列,列表是可变的序列。
3. 字符串中每个元素只能存储字符,而列表可以存储任意类型。
4. 列表和字符串都是可迭代对象。
5. 函数：

将多个字符串拼接为一个。

result = "连接符".join(列表)

将一个字符串拆分为多个。

列表 = “a-b-c-d”.split(“分隔符”)

*list --> str*

*练习:exercise07.py*

*"""*

list01 = ["a","b","c"]

str01 = "+".join(list01)# a+b+c

print(str01)

# 案例:根据某些逻辑，拼接字符串.

# str\_result = ""

# for item in range(10):#0 1 2 3 4 5 6 7 8 9

# # str\_result += str(item)

#

# # "" + "0" --> 产生一个新字符串对象 "0"

# # "0" + "1" --> 产生一个新字符串对象 "01"

# # "01" + "2" --> 产生一个新字符串对象 "012"

# # ...

# str\_result = str\_result + str(item)

#

# print(str\_result)

# 核心思想：通过可变对象(list)收集需要拼接的字符串,最后再转换为字符串。

list\_result = []

for item in range(10):#0 1 2 3 4 5 6 7 8 9

list\_result.append(str(item))

str\_result = "".join(list\_result)

print(str\_result)

*str -->list*

*练习:exercise08.py*

*"""*

str01 = "张无忌\_张翠山\_张三丰"

list\_result = str01.split("\_")

print(list\_result)

### 列表推导式

1. 定义：

使用简易方法，将可迭代对象转换为列表。

1. 语法：

变量 = [表达式 for 变量 in 可迭代对象]

变量 = [表达式 for 变量 in 可迭代对象 if 条件]

1. 说明:

如果if真值表达式的布尔值为False,则可迭代对象生成的数据将被丢弃。

*列表推倒式*

*练习：exercise01.py*

# 需求1：将lie01中所有数据+1再存储另外一个列表

list01 = [5,15,55,6,6,7]

# list02 = []

# for item in list01:

# list02.append(item + 1)

#

# list03 = [item + 1 for item in list01]

# print(list02)

# print(list03)

# 需求2：将lie01中大于10的数据，+1再存储另外一个列表

list02 = []

for item in list01:

if item > 10:

list02.append(item + 1)

list03 = [item + 1 for item in list01 if item > 10]

print(list02)

print(list03)

### 列表推导式嵌套

1. 语法：

变量 = [表达式 for 变量1 in 可迭代对象1 for 变量2 in可迭代对象2]

1. 传统写法：

result = []

for r in ["a", "b", "c"]:

for c in ["A", "B", "C"]:

result.append(r + c)

1. 推导式写法：

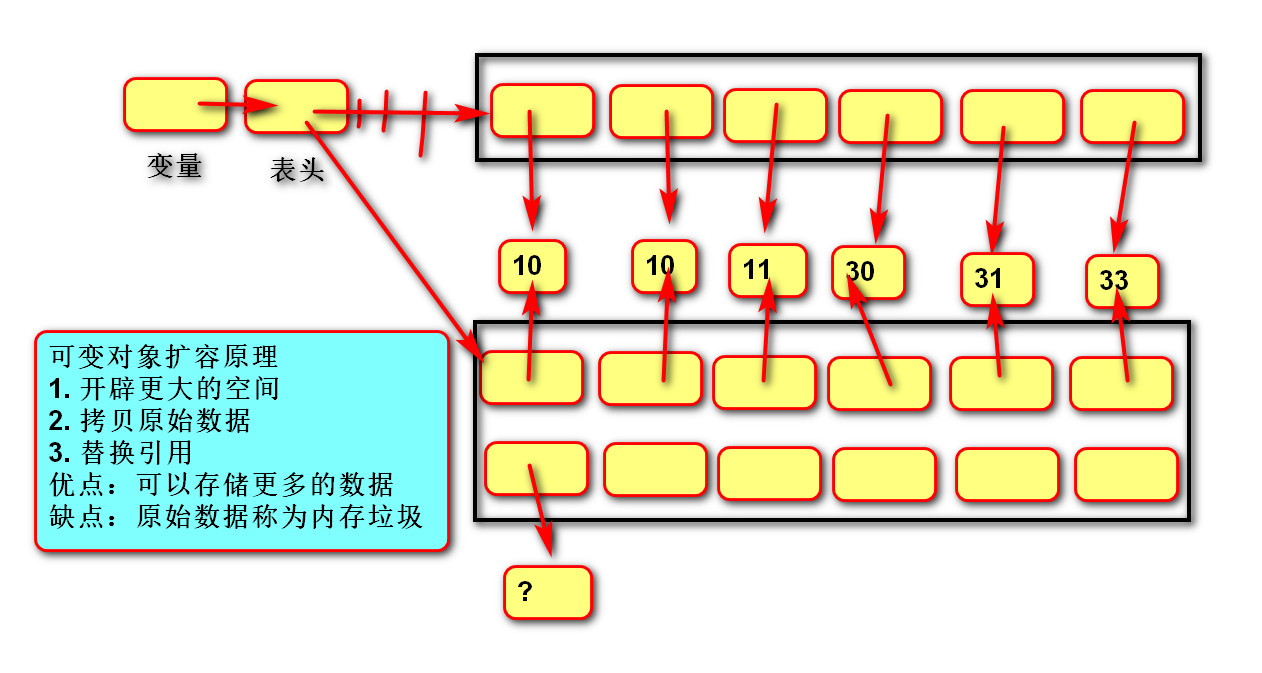
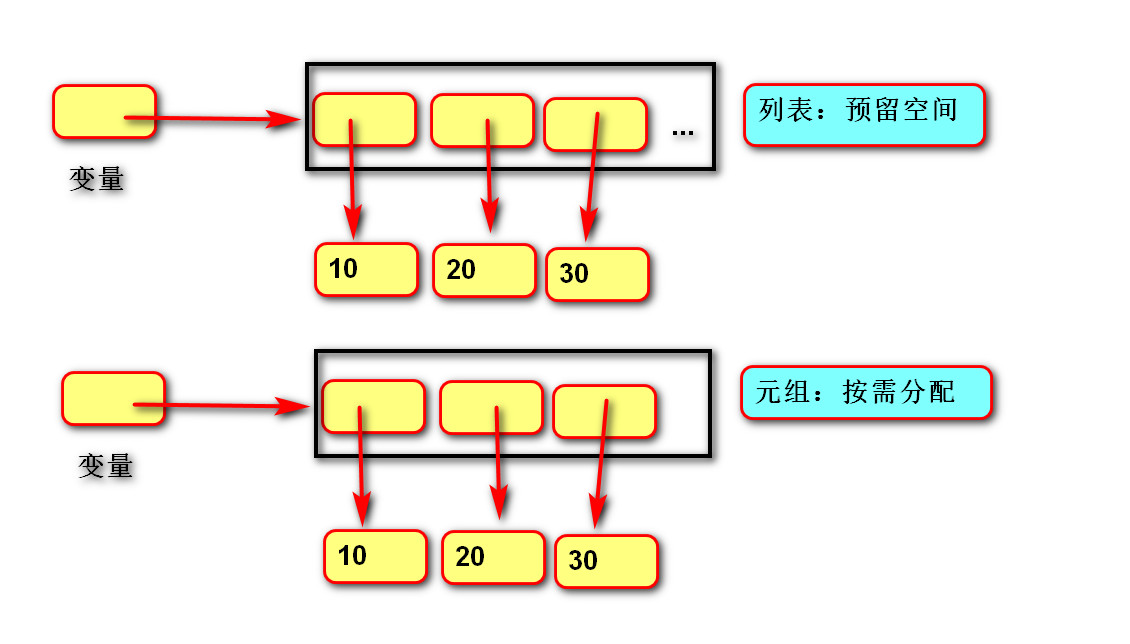
result = [r + c for r in list01 for c in list02]

*"""  
 列表推导式嵌套  
 练习:exercise08.py  
"""*list01 = ["a","b"]  
list02 = ["A","B","C"]  
list\_result = []  
  
# list01[0] + list02[0]  
# list01[0] + list02[1]  
# list01[0] + list02[2]  
  
# for c in range(3):  
# list\_result.append(list01[0] + list02[c])  
#  
# for c in range(3):  
# list\_result.append(list01[1] + list02[c])  
  
# for r in range(2):  
# for c in range(3):  
# list\_result.append(list01[r] + list02[c])  
  
for r in list01:  
 for c in list02:  
 list\_result.append(r + c)  
  
list\_result = [r + c for r in list01 for c in list02]  
print(list\_result)

## 元组 tuple

### 定义

1. 由一系列变量组成的不可变序列容器。
2. 不可变是指一但创建，不可以再添加/删除/修改元素。
3. 元组：按需分配 列表：预留空间



### 基础操作

1. 创建空元组：

元组名 = ()

元组名 = tuple()

1. 创建非空元组：

元组名 = (20,)

元组名 = (1, 2, 3)

元组名 = 100,200,300

元组名 = tuple(可迭代对象)

1. 获取元素：

索引、切片

1. 遍历元组：

正向：

for 变量名 in 元组名:

变量名就是元素

反向：

for 索引名 in range(len(元组名)-1,-1,-1):

元祖名[索引名]就是元素

*元组*

*"""*

# 1. 创建元组

# 空元组

tuple01 = ()

tuple01 = tuple()

# 具有默认值元组

tuple02 = (1,2)

# 预留空间的存储机制 --> 按需分配的存储机制

list01 = ["a","b"]

tuple02 = tuple(list01)

print(tuple02)

# 按需分配的存储机制 --> 预留空间的存储机制

list02 = list(tuple02)

print(list02)

# tuple02 = ("a")# 不是元组

tuple02 = ("a",)# 注意:如果元组只有一个元素，必须要在末尾添加逗号

print(tuple02)

# 多个变量 = 序列

name01,name02 = ("无忌","翠山")

print(name01)

print(name02)

tuple03 = ("a","b","c","d")

# 获取元素

# 单个元素

print(tuple03[2])

# 多个元素

print(tuple03[1:3])# ("b","c")

# 所有元素

# 正向

for item in tuple03:

print(item)

# 反向

for i in range(len(tuple03)-1,-1,-1):

print(tuple03[i])

### 作用

1. 元组与列表都可以存储一系列变量，由于列表会预留内存空间，所以可以增加元素。
2. 元组会按需分配内存，所以如果变量数量固定，建议使用元组，因为占用空间更小。
3. 应用：

变量交换的本质就是创建元组：x, y = （y, x）

格式化字符串的本质就是创建元祖："姓名:%s, 年龄:%d" % ("tarena", 15)

## 字典 dict

### 定义

1. 由一系列键值对组成的可变散列容器。
2. 散列：对键进行哈希运算，确定在内存中的存储位置，每条数据存储无先后顺序。
3. 键必须惟一且不可变(字符串/数字/元组)，值没有限制。
4. 读写速度最快，但内存占用较大。

Hash，一般翻译做散列、杂凑，或音译为哈希，是把任意长度的[输入](https://baike.baidu.com/item/输入/5481954" \t "_blank)（又叫做预映射pre-image）通过散列算法变换成固定长度的[输出](https://baike.baidu.com/item/输出/11056752" \t "_blank)，该输出就是散列值。这种转换是一种压缩映射，也就是，散列值的空间通常远小于输入的空间，不同的输入可能会散列成相同的输出，所以不可能从散列值来确定唯一的输入值。简单的说就是一种将任意长度的消息压缩到某一固定长度的[消息摘要](https://baike.baidu.com/item/消息摘要/4547744)的函数。

若结构中存在和关键字K相等的记录，则必定在f(K)的[存储](https://baike.baidu.com/item/存储/1582924" \t "_blank)位置上。由此，不需比较便可直接取得所查记录。称这个对应关系f为[散列函数](https://baike.baidu.com/item/散列函数" \t "_blank)(Hash function)，按这个事先建立的表为[散列表](https://baike.baidu.com/item/散列表" \t "_blank)。

### 基础操作

1. 创建字典：

字典名 = {键1：值1，键2：值2}

字典名 = dict (可迭代对象)

1. 添加/修改元素：

语法:

字典名[键] = 数据

说明:

键不存在，创建记录。

键存在，修改映射关系。

1. 获取元素：

变量 = 字典名[键] # 没有键则错误

1. 遍历字典：

for 键名 in 字典名:

字典名[键名]

for 键名,值名 in 字典名.items():

语句

1. 删除元素：

del 字典名[键]

*字典：*

*"""*

# 1. 创建字典

# 空

dict01 = {}

dict01 = dict()

# 具有默认值

dict02 = {"qtx":18,"ls":20,"ww":23}

list01 = [["a","b"],("c","d")]

dict02 = dict(list01)

print(dict02)

# 2. 添加元素

# 第一次(没有该键)添加

dict02["键"] = "值"

# 3. 修改元素

dict02["键"] = "值2"

dict02["a"] = "B"

print(dict02)

# 4. 删除

del dict02["键"]

print(dict02)

# 5.查找单个元素

# 在查找元素时,如果字典中不存在该键，则错误.

# 所以查找前，一定通过in判断。

if "qtx" in dict02:

print(dict02["qtx"])

else:

print("不存在")

# del dict02["qtx"]#　KeyError:

# 获取所有元素

for key in dict02:

print(key)# 键

print(dict02[key])# 键 --> 值

# # (键,值)

# for item in dict02.items():

# print(item[0])

# print(item[1])

# (键,值)

for key,value in dict02.items():

print(key)

print(value)

# 获取字典中所有值

for value in dict02.values():

print(value)

# 在python3.6以后,字典在功能上体现了加入的顺序.

dict02["三丰"] = 128

dict02["无忌"] = 28

dict02["翠山"] = 35

for item in dict02:

print(item)

### 字典推导式

1. 定义：

使用简易方法，将可迭代对象转换为字典。

1. 语法:

{键:值 for 变量 in 可迭代对象}

{键:值 for 变量 in 可迭代对象 if 条件}

*字典推导式*

*练习：exercise07.py*

*练习：exercise08.py*

*"""*

# 需求:range(10) key 0---9 value 键的平方

dict01 = {}

for item in range(10):

dict01[item] = item \*\* 2

dict02 = {item: item \*\* 2 for item in range(10)}

# 需求:range(10) 大于5的数， key 0---9 value 键的平方

dict01 = {}

for item in range(10):

if item > 5:

dict01[item] = item \*\* 2

dict02 = {item: item \*\* 2 for item in range(10) if item > 5}

print(dict01)

print(dict02)

### 字典 VS 列表

1. 都是可变容器。
2. 获取元素方式不同,列表用索引,字典用键。
3. 字典的插入,删除,修改的速度快于列表。
4. 列表的存储是有序的,字典的存储是无序的。

## 集合 set

### 定义

1. 由一系列不重复的不可变类型变量组成的可变散列容器。
2. 相当于只有键没有值的字典(键则是集合的数据)。

### 基础操作

1. 创建空集合：

集合名 = set()

集合名 = set(可迭代对象)

1. 创建具有默认值集合：

集合名 = {1, 2, 3}

集合名 = set(可迭代对象)

1. 添加元素：

集合名.add(元素)

1. 删除元素：

集合名.discard(元素)

*"""  
 集合  
 不能重复 ---> 将其他具有重复元素的容器转换为集合(去重复)  
 数学运算 --->  
 练习:exercise01.py  
"""*# 创建空集合  
s01 = set()  
print(type(s01))  
  
# 创建具有默认值的集合  
s02 = {"a","b"}  
s02 = set("abcabc")#  
print(s02)  
tuple02 = tuple(s02)  
  
# 增加  
s02.add("qtx")  
s02.add("mly")  
  
# 删除  
s02.remove("mly")# 如果没有该元素 则报错  
print(s02)  
  
# 获取所有  
for item in s02:  
 print(item)

### 运算

1. 交集&：返回共同元素。

s1 = {1, 2, 3}

s2 = {2, 3, 4}

s3 = s1 & s2 # {2, 3}

1. 并集：返回不重复元素

s1 = {1, 2, 3}

s2 = {2, 3, 4}

s3 = s1 | s2 # {1, 2, 3, 4}

1. 补集-：返回只属于其中之一的元素

s1 = {1, 2, 3}

s2 = {2, 3, 4}

s1 - s2 # {1} 属于s1但不属于s2

补集^：返回不同的的元素

s1 = {1, 2, 3}

s2 = {2, 3, 4}

s3 = s1 ^ s2 # {1, 4} 等同于(s1-s2 | s2-s1)

1. 子集<：判断一个集合的所有元素是否完全在另一个集合中
2. 超集>：判断一个集合是否具有另一个集合的所有元素

s1 = {1, 2, 3}

s2 = {2, 3}

s2 < s1 # True

s1 > s2 # True

1. 相同或不同== !=：判断集合中的所有元素是否和另一个集合相同。

s1 = {1, 2, 3}

s2 = {3, 2, 1}

s1 == s2 # True

s1 != s2 # False

子集或相同,超集或相同 <= >=

# 数学运算  
# 交集  
s1 = {1, 2, 3}  
s2 = {2, 3, 4}  
s3 = s1 & s2 # {2, 3}  
print(s3)  
  
# 并集  
s1 = {1, 2, 3}  
s2 = {2, 3, 4}  
s3 = s1 | s2 # {1, 2, 3, 4}  
print(s3)  
  
# 对称补集  
s1 = {1, 2, 3}  
s2 = {2, 3, 4}  
s3 = s1 ^ s2 # {1, 4} 等同于(s1-s2 | s2-s1)  
print(s3)  
  
# 补集  
print(s2 - s1)# {4}  
print(s1 - s2)# {1}  
  
# 子集超集  
s1 = {1, 2, 3}  
s2 = {2, 3}  
print(s2 < s1) # True  
print(s1 > s2) # True

### 集合推导式

1. 定义：

使用简易方法，将可迭代对象转换为集合。

1. 语法:

{表达式 for 变量 in 可迭代对象}

{表达式 for 变量 in 可迭代对象 if 条件}

## 固定集合 frozenset

### 定义

不可变的集合。

#### 作用

固定集合可以作为字典的键,还可以作为集合的值。

#### 基础操作

创建固定集合：frozenset(可迭代对象)

#### 运算

等同于set

*"""  
 固定集合frozenset  
 主要作用：与其他容器互相转换  
 次要作用:可以作为字典的key  
"""*# 创建  
s01 = frozenset(["a","a","b"])  
print(s01)

# 函数 function

## pycharm相关设置

1. “代码自动完成”时间延时设置

File -> Settings -> Editor -> General -> Code Completion -> Autopopup in (ms):0

1. 快捷键：

Ctrl + P 参数信息（在方法中调用参数）

Ctrl + Q 快速查看文档

Ctrl + Alt + M 提取方法

## 定义

1. 用于封装一个特定的功能，表示一个功能或者行为。
2. 函数是可以重复执行的语句块, 可以重复调用。

## 作用

提高代码的可重用性和可维护性（代码层次结构更清晰）。

## 定义函数

1. 语法：

def 函数名(形式参数):

  函数体

1. 说明：

def 关键字：全称是define，意为”定义”。

函数名：对函数体中语句的描述，规则与变量名相同。

形式参数：方法定义者要求调用者提供的信息。

函数体：完成该功能的语句。

1. 函数的第一行语句建议使用文档字符串描述函数的功能与参数。

## 调用函数

1. 语法：函数名(实际参数)
2. 说明：根据形参传递内容。

*"""  
 函数：表示一个功能  
 参数：用功能时提供的信息。  
 练习：exercise09.py  
 练习：exercise10.py  
 17:05上课  
"""*# 做功能(参数)  
def attack(count):  
 *"""  
 攻击* ***:param*** *count: 整数类型的攻击次数  
 """* for i in range(count):  
 print("勾拳")  
 print("直拳")  
 print("天马流星拳拳")  
 print("临门一脚")  
#.....  
  
# print("直拳")  
# print("勾拳")  
# print("天马流星拳拳")  
# print("临门一脚")  
  
# 用功能  
attack(1)  
  
# print("直拳")  
# print("勾拳")  
# print("天马流星拳拳")  
# print("临门一脚")  
attack(2)

## 返回值

1. 定义：

方法定义者告诉调用者的结果。

1. 语法：

return 数据

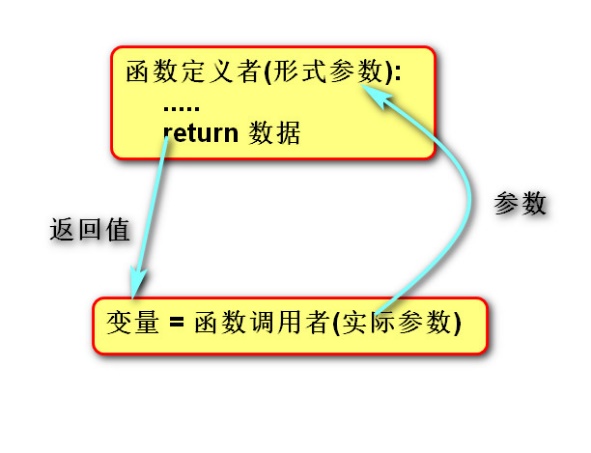
1. 说明：

return后没有语句，相当于返回 None。

函数体没有return，相当于返回None。

return会退出方法，其后的语句不会执行。

函数的返回值类型必须是一种，不能同时返回数字和字符串



*"""  
 函数：【一个】功能  
 返回值：做函数的人，给用函数的人的结果  
 练习：exercise11.py  
 练习：exercise12.py  
"""*# 小李同学  
# 定义两个数字相加的函数  
def add(number\_one, number\_two):  
 # 逻辑处理  
 return number\_one + number\_two # 返回结果，退出方法  
 print("看见我了吗？") # return 以后的语句不在执行  
  
  
# 获取数据  
one = int(input("请输入第一个整数："))  
two = int(input("请输入第二个整数："))  
re = add(one, two)  
# 显示结果  
print("结果是：" + str(re))

## 可变／不可变类型在传参时的区别

1. 不可变类型参数有:

数值型(整数，浮点数,复数)

布尔值bool

None 空值

字符串str

元组tuple

固定集合frozenset

1. 可变类型参数有:

列表 list

字典 dict

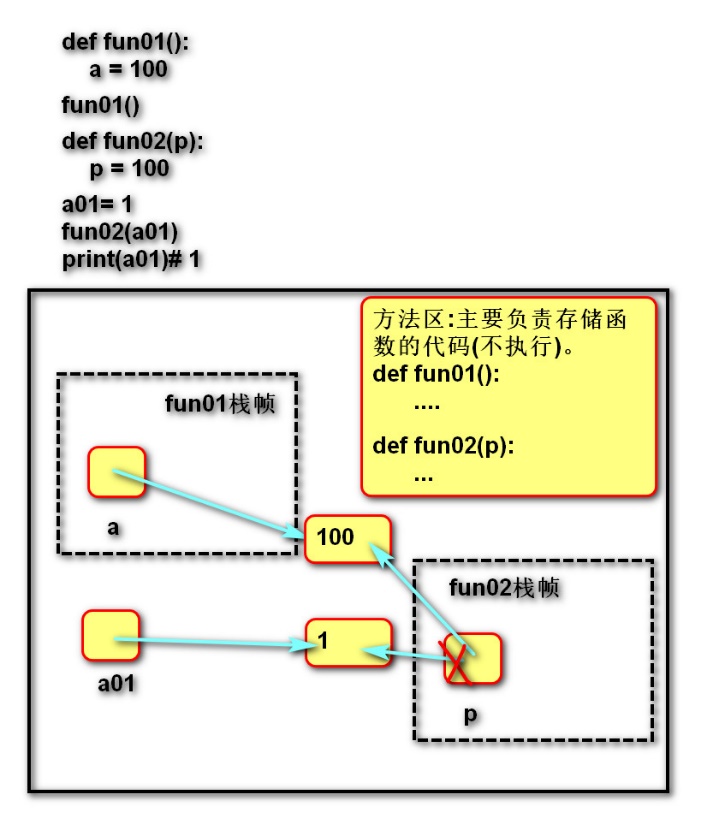
集合 set

1. 传参说明：

不可变类型的数据传参时，函数内部不会改变原数据的值。

可变类型的数据传参时，函数内部可以改变原数据。

1. 总结：
2. 向函数传递可变对象
3. 函数内部修改可变对象的值
4. 函数执行后，不需要通过返回值，也能拿到修改后的结果



*"""  
 函数内存分配  
"""*# 将函数代码存入方法区(不执行函数体)  
def fun01():  
 a = 100  
  
# 开辟栈帧  
fun01()  
  
# 栈帧销毁  
  
def fun02(p):  
 # 修改fun02栈帧中的变量  
 p = 100  
  
a01 = 1  
fun02(a01)  
print(a01) # 1



def fun03(p1, p2):  
 # 修改列表的元素  
 p1[0] = "悟空"  
 # 修改栈帧中的变量  
 p2 = "八戒"  
  
a02 = ["孙悟空"]  
a03 = ["猪八戒"]  
fun03(a02, a03)  
print(a02)  
print(a03)



def fun04(p1, p2):  
 # 修改列表的元素  
 p1[:] = ["悟空"] # 将["悟空"]遍历后，存入p1指向的列表中  
 temp = p2[:] # 浅拷贝p2指向的列表  
 temp[:] = ["八戒"] # 修改拷贝后的列表元素  
  
a02 = ["孙悟空"]  
a03 = ["猪八戒"]  
fun04(a02, a03)  
print(a02) # ['悟空']  
print(a03) # ['猪八戒']

## 函数参数

### 实参传递方式argument

#### 位置传参

定义：实参与形参的位置依次对应。

##### 序列传参

定义：实参用\*将序列拆解后与形参的位置依次对应。

#### 关键字传参

定义：实参根据形参的名字进行对应。

##### 字典关键字传参

1. 定义：实参用\*\*将字典拆解后与形参的名字进行对应。
2. 作用：配合形参的缺省参数，可以使调用者随意传参。

*"""  
 函数参数  
 实际参数  
  
"""*def fun01(a, b, c):  
 print(a)  
 print(b)  
 print(c)  
  
# 1. 位置实参:实参根据位置与形参进行对应  
# positional argument  
fun01(1, 2, 3)  
  
# 2. 序列实参:使用星号将序列中的元素拆开，与形参进行对应.  
# 序列:字符串 列表 元组  
list01 = ["aaa", "bb", "cc"]  
fun01(\*list01)  
  
# 3. 关键字实参：实参根据名称与形参进行对应  
# 备注：作用明天讲。  
fun01(b=2, a=1, c=3)  
  
# 4. 字典实参：使用双星号将字典中的元素拆开,根据键形参进行对应,传递值..  
dict01 = {"c": 33, "a": 11, "b": 22}  
fun01(\*\*dict01)

### 形参定义方式parameter

#### 缺省参数

1. 语法：

def 函数名(形参名1=默认实参1, 形参名2=默认实参2, ...):

函数体

1. 说明：

缺省参数必须自右至左依次存在，如果一个参数有缺省参数，则其右侧的所有参数都必须有缺省参数。

缺省参数可以有0个或多个，甚至全部都有缺省参数。

#### 位置形参

语法：

def 函数名(形参名1, 形参名2, ...):

函数体

##### 星号元组形参

1. 语法：

def 函数名(\*元组形参名):

  函数体

1. 作用：

收集多余的位置实参。

1. 说明：

一般命名为'args'

形参列表中最多只能有一个

#### 命名关键字形参

1. 语法：

def 函数名(\*, 命名关键字形参1, 命名关键字形参2, ...):

函数体

def 函数名(\*args, 命名关键字形参1, 命名关键字形参2, ...):  
  函数体

1. 作用：

强制实参使用关键字传参

##### 双星号字典形参

1. 语法：

def 函数名(\*\*字典形参名):

函数体

1. 作用：

收集多余的关键字传参

1. 说明:

一般命名为'kwargs'

形参列表中最多只能有一个

#### 参数自左至右的顺序

位置形参 --> 星号元组形参 --> 命名关键字形参 --> 双星号字典形参

*函数参数  
 形参  
 默认形参：实参可以不传递数据。  
 位置形参：实参根据位置进行对应  
 星号元组形参：实参数量无限（将实参合并为元组）  
 关键字形参：实参根据名称进行对应  
 双星号字典形参:实参数量无限（将实参合并为字典）  
 命名关键字形参：实参必须是关键字实参*# 1. 默认形参：实参可以不传递数据（从右向左依次存在）  
def fun01(a=0, b="bb", c=1.5):  
 print(a)  
 print(b)  
 print(c)  
  
fun01()  
fun01(1, "b")  
# 关键实参 + 默认形参：调用者可以随意指定参数进行传递  
fun01(b="bbbbb")  
  
# 2. 星号元组形参：让位置实参的数量无限  
def fun02(p1, p2, \*args):  
 print(args)  
  
fun02(1, 2)  
fun02(1, 2, 3)  
fun02(1, 2, 3, 4, 5)  
  
# 3. 命名关键字形参：传递的实参必须是关键字实参。  
# 写法1：星号元组形参以后的参数是命名关键字形参  
# p1 p2  
def fun03(\*args, p1="", p2):  
 print(args)  
 print(p1)  
 print(p2)  
  
fun03(2, 2, p1=111, p2=222)  
fun03(p1=111, p2=222)  
fun03(p2=222)  
# 案例：  
# def print(\*args, sep=' ', end='\n', file=None):  
# 1---fff---3.5---4---55---6---67 ok  
print(1, "fff", 3.5, 4, 55, 6, 67, sep="---", end=" ")  
print("ok")  
  
# 写法2:星号以后的位置形参是命名关键字形参  
def fun04(\*, p1=0, p2):  
 print(p1, p2)  
  
fun04(p1=1, p2=2)  
fun04(p2=2)  
  
# 4. 双星号字典形参:让关键字实参的数量无限  
def fun05(\*\*kwargs):  
 print(kwargs)  
  
fun05(a=1) # {'a': 1}  
fun05(a=1, b=2)  
fun05(a=1, b=2, qtx=3) # {'a': 1, 'b': 2, 'qtx': 3}

# 作用域LEGB

1. 作用域：变量起作用的范围。
2. Local局部作用域：函数内部。
3. Enclosing 外部嵌套作用域 ：函数嵌套。
4. Global全局作用域：模块(.py文件)内部。
5. Builtin内置模块作用域：builtins.py文件。

## 变量名的查找规则

1. 由内到外：L -> E -> G -> B
2. 在访问变量时，先查找本地变量，然后是包裹此函数外部的函数内部的变量，之后是全局变量，最后是内置变量。

## 局部变量

1. 定义在函数内部的变量(形参也是局部变量)
2. 只能在函数内部使用
3. 调用函数时才被创建，函数结束后自动销毁

## 全局变量

1. 定义在函数外部,模块内部的变量。
2. 在整个模块(py文件)范围内访问（但函数内不能将其直接赋值）。

## global 语句。

1. 作用：

在函数内部修改全局变量。

在函数内部定义全局变量(全局声明)。

1. 语法：

global 变量1, 变量2, …

1. 说明

在函数内直接为全局变量赋值，视为创建新的局部变量。

不能先声明局部的变量，再用global声明为全局变量。

*"""  
 变量作用域  
 在局部(函数内部)作用域中定义的变量，就是局部变量.  
 在全局(.py文件中)作用域中定义的变量，就是全局变量.  
 练习：exercise07.py  
"""*# 全局变量  
g01 = "悟空"  
g02 = "八戒"  
g03 = "沙僧"  
  
def fun01():  
 l01 = 100  
 print(l01)# 只能在函数内部使用局部变量  
  
 print(g01)# 在函数内部可以访问全局变量  
  
 # 创建了局部变量g02，覆盖了全局变量g02  
 g02 = "老朱"  
 print("fun01---"+g02)# 老朱  
  
 # 声明全局变量g03  
 global g03  
 g03 = "老沙"  
  
fun01()  
print("全局---"+g02)# 八戒  
print(g03)

## nonlocal 语句

1. 作用：

在内层函数修改外层嵌套函数内的变量

1. 语法

nonlocal 变量名1,变量名2, ...

1. 说明

在被嵌套的内函数中进行使用