# 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。

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()

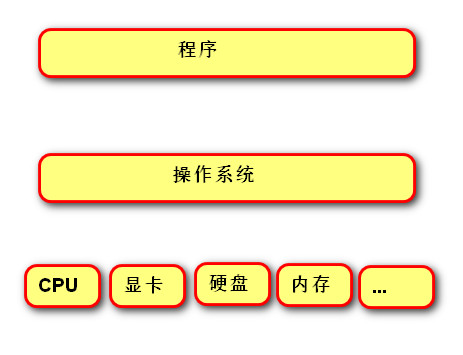
### 文件式

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

内存图：

１．计算机构成图：



## 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/1905/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；

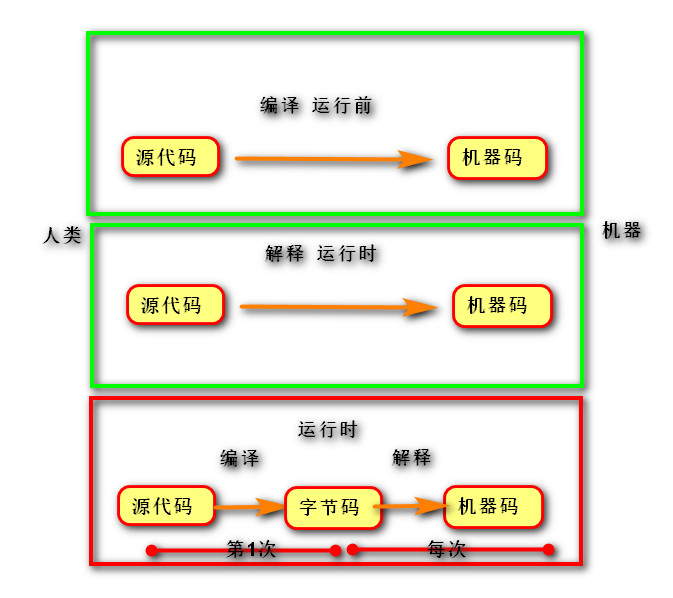
-- 缺点：运行速度慢。

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———|

执行过程图：



## 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 + alt + 上下箭头
7. 智能提示：Ctrl + Space

## 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

1.变量交换

# 在控制台中获取两个变量，然后交换数据，最后显示结果.

# “请输入第一个变量：”

# “请输入第二个变量：”

# 交换

# “第一个变量是：”

# “第二个变量是：”

data01 = input("请输入第一个变量：")

data02 = input("请输入第二个变量：")

# 版本１：所有语言通用思想

# temp = data01

# data01 = data02

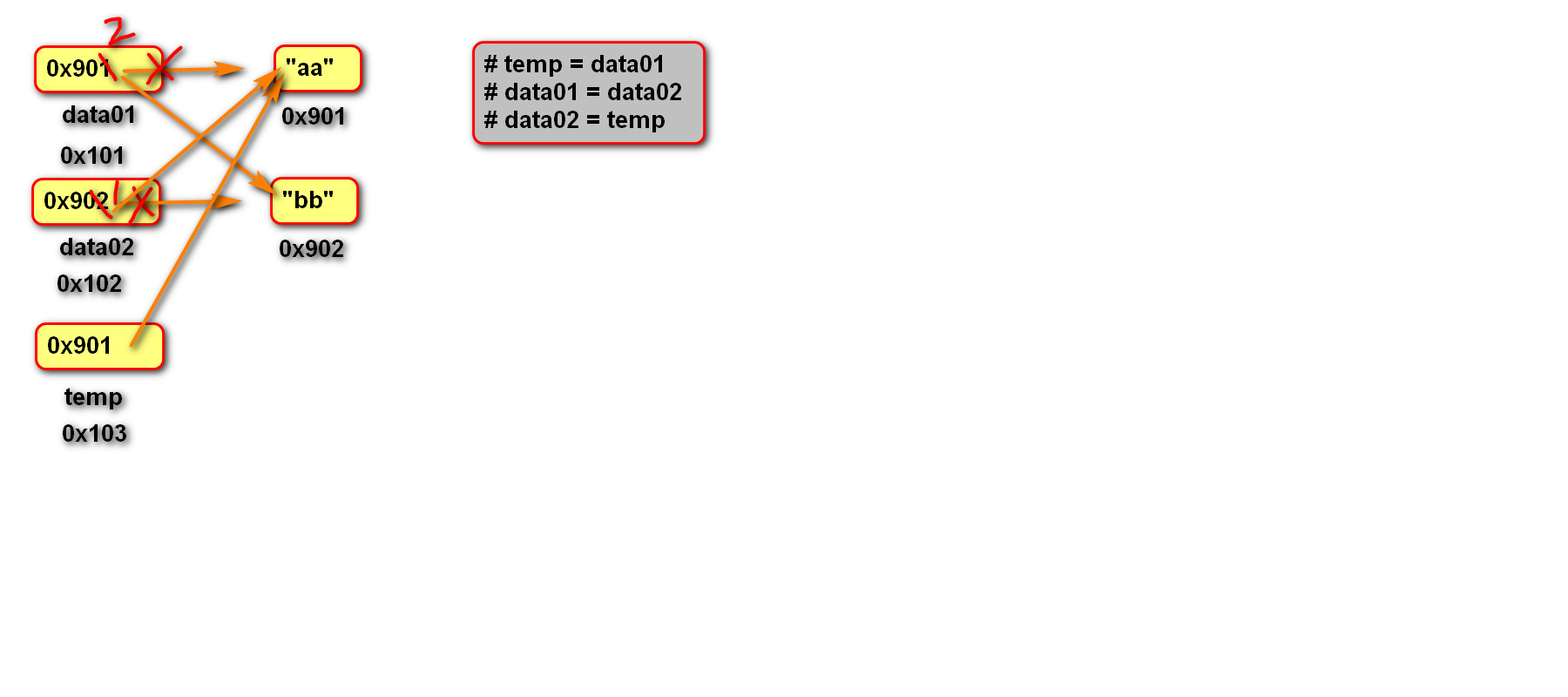
# data02 = temp

# 版本２：适合ｐｙｔｈｏｎ

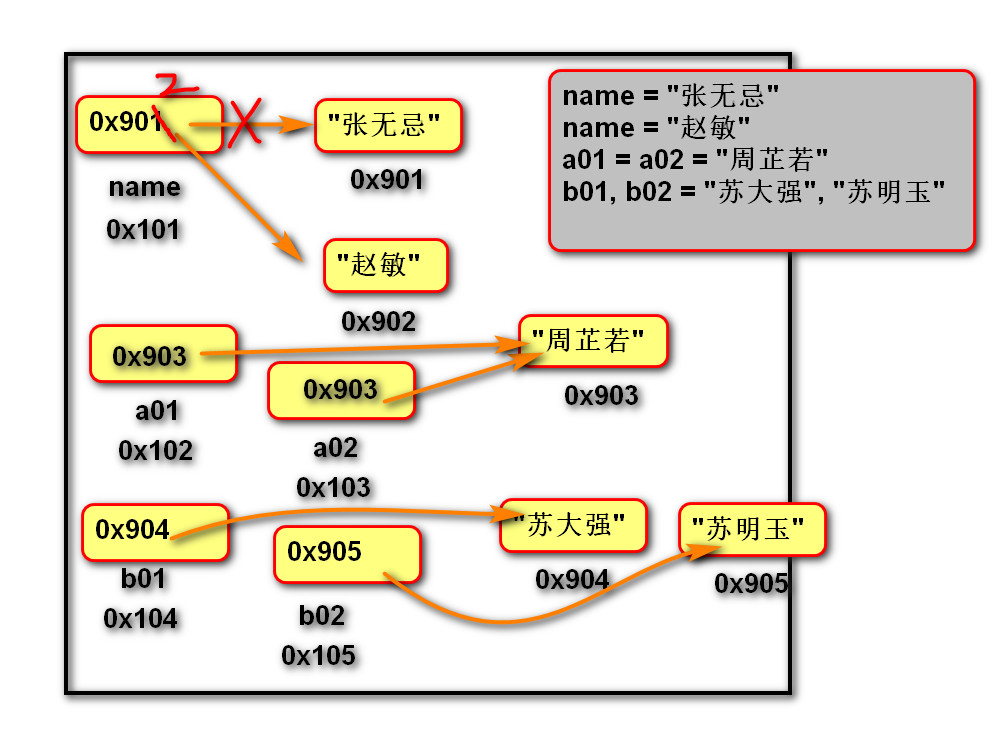
data01, data02 = data02, data01

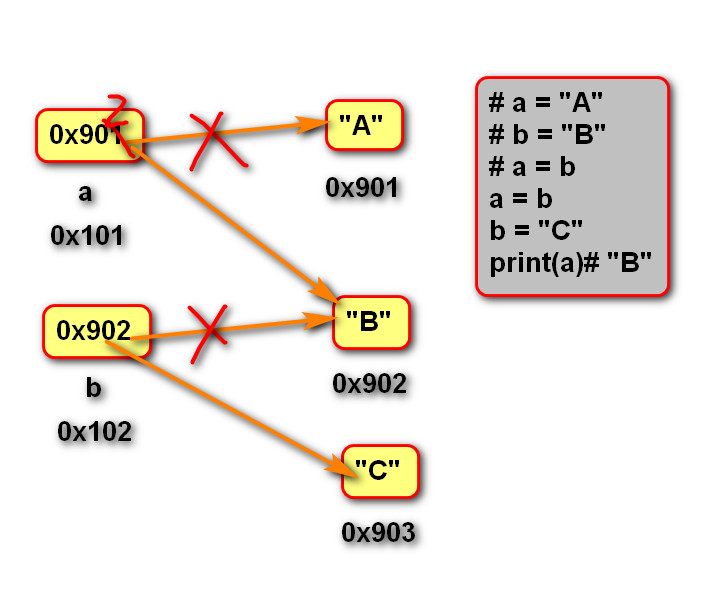
print("第一个变量是：" + data01)

print("第二个变量是：" + data02)



3.变量内存图：





demo01

"""

print

input

"""

# 字面意思：输入功能

# 作用：从终端中获取输入的信息，存到程序(变量)中。

name = input("请输入姓名：")

# 字面意思：打印功能

# 作用：将括号中的内容，显示到终端。

print(name)

demo02:

"""

变量：在内存中存储数据

"""

# 语法：

# 变量名称　= 对象

# 例如：

name = "张无忌"

print(name)

# 语义：内存图

# 变量名：真实内存地址的别名

# 见名知意

# 赋值号：将右边对象的地址复制给左边内存空间。

name = "赵敏"

a01 = a02 = "周芷若"

b01, b02 = "苏大强", "苏明玉"

class\_name = "1905"

## 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。

demo01:

a = "悟空"

b = a

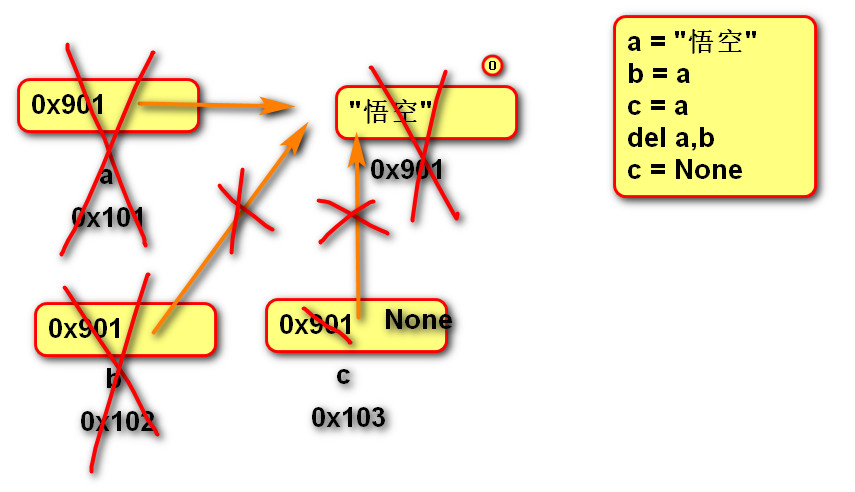
c = a

# 删除变量a,b,不释放对象"悟空"

del a,b

# 由于变量ｃ不再引用对象"悟空",而悟空的引用计数为０，所以被标记为可回收。

c = None



## 核心数据类型

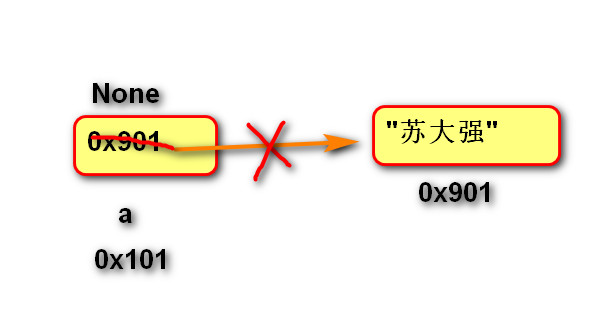
1. 在python中变量没有类型，但关联的对象有类型。
2. 通过type函数可查看。

### 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。

内存图：

１．None



# 1. Ｎｏｎｅ

a01 = "苏大强"

# 解除变量与数据的绑定关系

a01 = None

# 使用Ｎｏｎ占位

sex = None

### 整形int

1. 表示整数，包含正数、负数、0。

如： -5, 100, 0

1. 字面值：

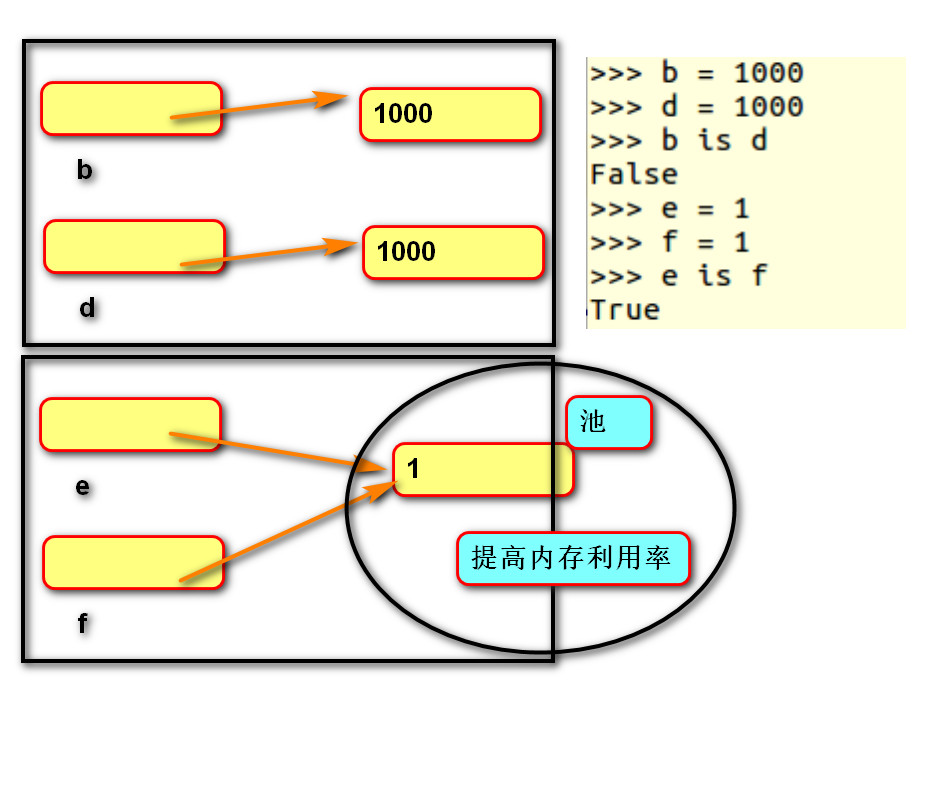
十进制：5

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。

内存图：

# 2. 整形int

# 十进制

num01 = 20

# 二进制：０　　１　　１０　　１１　　 　１００

print(0b10)# 2

# 八进制：0 1　．．7 10 11 ..

print(0o10)# 8

# 十六进制：0 -- 9 a(10) - f(15)

print(0x10)# 16

### 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)

# 3.　浮点数float

print(1.5)

# 科学计数法:表示过小或过大的值很明确

# 1.23e-25

print(0.000000000000000000000000123)

### 字符串str

是用来记录文本信息(文字信息)。

字面值：双引号

# 4. 字符串str

print("1.5")

a = 10

print(a)# 打印变量　　10

print("a")# 打印字符串 a

### 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

# 1. 数据类型转换

# int float str

str\_usd = input("请输入美元：")

# 类型转换str --> int

int\_usd = int(str\_usd)

result = int\_usd \* 6.9

# str + 数值 --> str + str(数值)

# result = str(result)

# print("结果是："+result)

print("结果是："+str(result))

exercise03:

# 在控制台中，录入一个商品单价。25

# 再录入一个数量 2

# 最后获取金额，60 计算应该找回多少钱。60 - 25\*2

price = input("请输入商品单价：")

price = float(price)

count = int(input("请输入数量："))

money = float(input("请输金额："))

result = money - price \* count

print("应该找回：" + str(result))

exercise04:

# 练习２：在控制台中获取分钟

# 　　　　再获取小时

# 　　　　再获取天

# 计算总秒数

minute = int(input("请输入分钟："))

hour = int(input("请输入小时："))

day = int(input("请输入天："))

result = minute \* 60 + hour \* 60 \* 60 + day \* 24 \* 60\*60

print("总秒数是:"+str(result))

exercise05:

"""

古代的秤一斤是１６两 33 二斤　　一两

练习：在控制台中获取两，计算是几斤零几两。

　　　显示几斤零几两

"""

weight\_liang = int(input("请输入两："))

jin = weight\_liang // 16

liang = weight\_liang % 16

print(str(jin) + "斤零" + str(liang) + "两")

exercise06:

# 在控制台中录入距离，时间，初速度，计算加速度。

# 匀变速直线运动的位移与时间公式：

# 加速度　＝　(距离 - 初速度　×　时间) \* 2 / 时间平方

distance = float(input("请输入距离："))

time = float(input("请输入时间："))

initial\_velocity = float(input("请输入初速度："))

accelerated\_speed = (distance - initial\_velocity

\* time) \* 2 / time \*\* 2

print("加速度是：" + str(accelerated\_speed))

## 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： ()

\*\*

\* / % //

+ -

# 2. 算数运算符

# print(1+2)

# print(1-2.5)

# 地板除（保留整数）

print(5 // 2)# 商2

# 余

print(5 % 2)# 余1

# 除

print(5 / 2)# 2.5

# 获取整数的个位

print(27 % 10 )# 7

# 幂运算

# 5的2次方:5\*5

print(5\*\*2)

# 5的3次方:5\*5\*5

print(5\*\*3)

### 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

# 3. 增强运算符

# number01 = 200

# print(number01 + 1)

# print(number01)# 200

number01 = 200

# 变量加上另外一个数，再赋值给自身

# number01 = number01 + 1

# 累加(在自身基础上增加)

number01 += 1

print(number01)# 201

exercise07:

# 在控制台中录入一个四位整数：1234

# 计算每位相加和。　　１＋２＋３＋４

# 显示结果。10

number = int(input("请输入４位整数：")) # 1234

# 方法１：分别计算出每位，再相加

# 个位 17:00

unit01 = number % 10

# 十位 1234 // 10 -> 123 % 10 -> 3

unit02 = number // 10 % 10

# 百位 1234 // 100 -> 12 % 10 -> 2

unit03 = number // 100 % 10

# 千位

unit04 = number // 1000

result = unit01 + unit02 + unit03 + unit04

print("结果是：" + str(result))

# 方法２：累加每位

# 个位

result = number % 10

# 累加十位

result += number // 10 % 10

# 累加百位

result += number // 100 % 10

# 累加千位

result += number // 1000

print("结果是：" + str(result))

### 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

# １．　bool 类型

# 取值：(真，对的，满足条件)True (假，错的，不满足条件)False

# 命题:带有判断性的陈述句。

# 例如：我是个男人。

# 1 > 2 --> False

print(1 > 2)

### 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

# －－　与：一假俱假,表示并且(都得满足)的关系

print(True and True) # True　都得满足条件，结论才满足条件。

print(False and True) # False

print(True and False) # False

print(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

# －－　或：一真俱真,表示或者(一个满足就行)的关系

print(True or True) # True

print(False or True) # True

print(True or False) # True

print(False or False) # False

#### 非 not

表示取反

例如：

not True # 返回False

not False # 返回True

# －－　非：取反

print(not True)

# 练习：判断年份是否为闰年。

# 闰年Ｔｒｕｅ:年份能被4整除，但是不能被100整除。

# 能被400整除

# 平年Ｆａｌｓｅ

year = int(input("请输入年份："))

result = year % 4 == 0 and year % 100 != 0 or year % 400 == 0

print(result)

#### 短路运算

一但结果确定，后面的语句将不再执行。

# 短路逻辑：逻辑运算时，尽量将复杂(耗时)的判断放在后边。

num = 1

# and 发现Ｆａｌｓｅ，就有了结论,后续条件不再判断。

# re = num > 1 and input("请输入：") == "a"

# or 发现Ｔｒｕｅ，就有了结论,后续条件不再判断。

re = num + 1 > 1 or input("请输入：") == "a"

### 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

demo02:

a = 800

b = 1000

# id函数，可以获取变量存储的对象地址。

print(id(a))

print(id(b))

# flase

print(a is b) # is 的本质就是通过ｉｄ函数进行判断的

c = a

print(id(c))

print(c is a)

d = 1000

print(d is b)

### 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。

demo03:

# 三个物理行，三个逻辑行

a = 1

b = a + 2

c = a + b

# 一个物理行，三个逻辑行（不建议）

a = 1;b = a + 2;c = a + b

# 一个物理行，一个逻辑行

d = 1 + 2 +3 + 4

# 一个物理行，两个逻辑行

# \ 表示折行符

d = 1 + 2 \

+3 + 4

# 四个物理行，一个逻辑行

d = (1 +

2 +

3 +

4)

## 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。

demo04:

sex = input("请输入性别：")

if sex == "男":

print("您好，先生！")

elif sex == "女":

print("您好，女士！")

else:

print("性别未知")

print("后续逻辑")

# 调试：让程序中断，逐语句执行。

# 　--　目的：审查程序运行时变量取值

# 审查程序运行的流程

# --　步骤：

# 1. 加断点(可能出错的行)

# 2. 调试运行 Ｓｈｉｆｔ＋Ｆ９

# 3. 执行一行　F8

# ４. 停止　Ctrl +F2

exercise01:

# 练习:当钱不够时，提示"金额不足",

# 钱够时，提示"应找回"

# 调试程序.

price = input("请输入商品单价：")

price = float(price)

count = int(input("请输入数量："))

money = float(input("请输金额："))

result = money - price \* count

if result >= 0:

print("应该找回：" + str(result))

else:

print("金额不足")

exercise02:

"""

练习:在控制台中获取一个季度(春夏秋冬)，

显示相应的月份。

春 --> １月２月３月

夏 --> ４月５月６月

秋 --> ７月８月９月

冬 --> １０月１１月１２月

14:18

"""

season = input("请输入季度：")

# if season == "春":

# print("１月２月３月")

# if season == "夏":

# print("４月５月６月")

# if season == "秋":

# print("７月８月９月")

# if season == "冬":

# print("１０月１１月１２月")

# 相比上面代码的优点：如果前面条件满足，后续条件不再判断。

if season == "春":

print("１月２月３月")

elif season == "夏":

print("４月５月６月")

elif season == "秋":

print("７月８月９月")

elif season == "冬":

print("１０月１１月１２月")

exercise03:

# 练习２:在控制台中录入一个数字,

# 再录入一个运算符(+ - \* /)，最后录入一个数字。

# 根据运算符，计算两个数字。

# 要求:如果运算符，不是加减乘除，则提示"运算符有误"

number\_one = float(input("请输入第一个数字："))

operator = input("请输入运算符:")

number\_two = float(input("请输入第二个数字："))

if operator == "+":

print(number\_one + number\_two)

elif operator == "-":

print(number\_one - number\_two)

elif operator == "\*":

print(number\_one \* number\_two)

elif operator == "/":

print(number\_one / number\_two)

else:

print("运算符输入有误")

exercise04:

# 练习３:在控制台中分别录入４个数字

# 打印最大的数字

# 5

# 2

# 9

# 1

# 将第一个数字记在心里，然后与第二个比较

# 如果第二个大于心中的，则心中记录第二个

# 然后与第三个比较．．．．

number01 = float(input("请输入第1个数字："))

number02 = float(input("请输入第2个数字："))

number03 = float(input("请输入第3个数字："))

number04 = float(input("请输入第4个数字："))

# 　假设第一个是最大值

max\_value = number01

# 以此与后面进行比较

if max\_value < number02:

# 发现更大的，则替换假设的。

max\_value = number02

if max\_value < number03:

max\_value = number03

if max\_value < number04:

max\_value = number04

print(max\_value)

exercise05:

"""

在控制台中录入一个成绩，

判断等级（优秀／良好／及格／不及格/输入有误）。

"""

score = int(input("请输入成绩："))

# if score >= 90 and score <= 100:

# print("优秀")

# elif score >= 80 and score < 90:

# print("良好")

# elif score >= 60 and score < 80:

# print("及格")

# elif score >= 0 and score < 60:

# print("不及格")

# else:

# print("输入有误")

# if 90 <= score <= 100:

# print("优秀")

# elif 80 <= score < 90:

# print("良好")

# elif 60 <= score < 80:

# print("及格")

# elif 0 <= score < 60:

# print("不及格")

# else:

# print("输入有误")

if score > 100 or score < 0:

print("输入有误")

elif 90 <= score:

print("优秀")

elif 80 <= score:

print("良好")

elif 60 <= score:

print("及格")

else:

print("不及格")

erxercise06:

# 在控制台中获取一个月份

# 打印天数,或者提示"输入有误".

# 1 3 5 7 8 10 12 --> 31天

# 4 6 9 11 --> 30天

# 2 --> 28天

# 16:05

month = int(input("请输入月份："))

if month < 1 or month > 12:

print("输入有误")

elif month == 2:

print("２８天")

elif month == 4 or month == 6 or month == 9\

or month == 11:

print("３０天")

else:

print("３１天")

### if 语句的真值表达式

if 100:

print("真值")

等同于

if bool(100):

print("真值")

### 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。

Demo05:

"""

真值表达式

if 数据:

语句

本质就是使用bool函数操作数据

条件表达式

练习:exercise07.py

17:10

"""

# 1. 真值表达式

if "a":

# if bool("a"):

print("真值")

str\_input = input("请输入：")

if str\_input:

print("输入的字符串不是空的")

# 2. 条件表达式:有选择性的为变量进行赋值

# sex = None

# if input("请输入性别:") == "男":

# sex = 1

# else:

# sex = 0

# print(sex)

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

print(sex)

exercise07:

# 　练习１：在控制台中获取一个整数，

# 如果是偶数为变量state赋值"偶数",否则赋值"奇数"

number = int(input("请输入整数："))

# if number % 2 == 1:

# state = "奇数"

# else:

# state = "偶数"

# if number % 2:

# state = "奇数"

# else:

# state = "偶数"

state = "奇数" if number % 2 else "偶数"

print(state)

# 练习2: 在控制台中录入一个年份，

# 如果是闰年，给变量day赋值29，否则赋值28.

year = int(input("请输入年份："))

# result = year % 4 == 0 and year % 100 != 0 or year % 400 == 0

# if result:

# day = 29

# else:

# day = 28

# if year % 4 == 0 and year % 100 != 0 or year % 400 == 0:

# day = 29

# else:

# day = 28

# 代码可读性　很差

# if not year % 4 and year % 100 or not year % 400:

# if year % 4 == 0 and year % 100 != 0 or year % 400 == 0:

# day = 29

# else:

# day = 28

day = 29 if year % 4 == 0 and year % 100 != 0 or year % 400 == 0 else 28

print(day)

## 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。

Demo06:

# 死循环：循环条件永远是满足的。

while True:

usd = int(input("请输入美元："))

print(usd \* 6.9)

if input("输入q键退出:"):

break # 退出循环体

demo07:

# 需求:执行三次

count = 0

while count < 3: # 0 1 2

count += 1

usd = int(input("请输入美元："))

print(usd \* 6.9)

exercise08:

# 　练习:使下列代码循环执行，按e键退出。

# 调试程序

while True:

season = input("请输入季度：")

if season == "春":

print("１月２月３月")

elif season == "夏":

print("４月５月６月")

elif season == "秋":

print("７月８月９月")

elif season == "冬":

print("１０月１１月１２月")

if input("输入e键退出:") == "e":

break

exercise09:

# 练习1:在控制台中输出0 1 2 3 4 5

# 练习2:在控制台中输出2 3 4 5 6 7

# 练习3:在控制台中输出0 2 4 6

# count = 0

# while count < 6:

# print(count)

# count += 1

# count = 2

# while count < 8:

# print(count)

# count += 1

count = 0

while count <= 6:

print(count)

count += 2

例１：

"""

猜数字2.0:

最多猜３次，如果猜对提示"猜对了，总共猜了?次"

如果超过次数，提示"游戏结束".

"""

import random

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

print(random\_number)

count = 0

while count < 3:

# 三次以内

count += 1

input\_number = int(input("请输入数字："))

if input\_number > random\_number:

print("大了")

elif input\_number < random\_number:

print("小了")

else:

print("猜对了，总共猜了" + str(count) + "次")

break# 退出循环体，不会执行else语句。

else:# while的条件不满足

# 三次以外

print("失败")

例２：

# 练习:循环根据成绩判断等级,如果录入空字符串则退出程序.

# 如果成绩录入错误次数达到３．则退出成绩并提示"成绩错误过多"

count = 0

while count<3:

str\_score = input("请输入成绩：")

if str\_score == "":

break# 不会执行else语句

score = int(str\_score)

if score > 100 or score < 0:

print("输入有误")

count += 1

elif 90 <= score:

print("优秀")

elif 80 <= score:

print("良好")

elif 60 <= score:

print("及格")

else:

print("不及格")

else:

print("成绩错误过多")

课后作业：

（扩展)一个小球从１００ｍ的高度落下

　　每次弹回原高度的一半．

　　计算：总共弹起来多少次（最小弹起高度0.01ｍ）．

总共走了多少米

height = 100

count = 0

# 经过距离

distance = height

# 弹起前高度 大于　最小弹起高度

# while height > 0.01:

# 弹起来的高度 大于　最小弹起高度

while height / 2 > 0.01:

count += 1

# 弹起

height /= 2

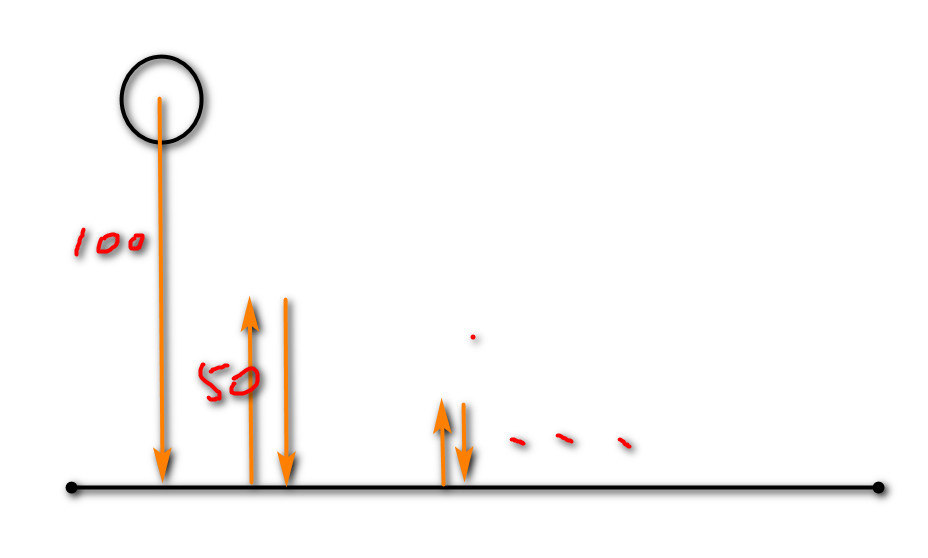
print("第%d次弹起来的高度是%f." % (count, height))

# 累加起/落高度

distance += height \* 2

print("总共弹起来%d次" % count)

print("总共经过的距离是%.2f" % distance)



课后作业：

"""

彩票　双色球：

红球:6个，1 -- 33 之间的整数 不能重复

蓝球:1个，1 -- 16 之间的整数

(1) 随机产生一注彩票[6个红球１个蓝球].

"""

import random

list\_ticket = []

# ６个不重复的红球

while len(list\_ticket) < 6:

random\_number = random.randrange(1, 33)

# 如果随机数不存在，则存储。

if random\_number not in list\_ticket:

list\_ticket.append(random\_number)

# 1个蓝球

list\_ticket.append(random.randrange(1, 16))

print(list\_ticket)

""

(2) 在控制台中购买一注彩票

提示：

"请输入第1个红球号码："

"请输入第2个红球号码："

"号码不在范围内"

"号码已经重复"

"请输入蓝球号码："

"""

# ６个1--33范围内的不重复红球号码

list\_ticket = []

while len(list\_ticket) < 6:

number = int(input("请输入第%d个红球号码:" % (len(list\_ticket) + 1)))

if number < 1 or number > 33:

print("号码不在范围内")

elif number in list\_ticket:

print("号码已经重复")

else:

list\_ticket.append(number)

# １个1--16范围内的蓝球号码

while len(list\_ticket) < 7:

number = int(input("请输入蓝球号码:"))

if 1 <= number <= 16:

list\_ticket.append(number)

else:

print("号码不在范围内")

print(list\_ticket)

### 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。

Demo02:

"""

for: 适合执行预定次数。

while:适合根据条件循环执行。

"""

# for 变量 in 可迭代对象:

# 循环体

str01 = "我叫苏大强!"

# item 存储的是字符串中每个字符的地址

for item in str01:

print(id(item))

# 整数生成器: range(开始值,结束值,间隔)

# for + range ：　更善于执行预定次数。

for item in range(5):#01234

print(item)

# 需求：折纸１０次

thickness = 0.0001

for item in range(10):

thickness\*=2

print(thickness)

exercise05.py

# 练习1:累加1--100的和 1+2+3+..+100

# 用于存储累加和的变量

sum\_value = 0

for item in range(1,101):

#0 += 1

#1 += 2

#3 += 3

#6 += 4

sum\_value += item

print(sum\_value)

# 练习2:累加1--100之间偶数和 2+4+6+8+...+100

sum\_value = 0

for item in range(2,101,2):

sum\_value += item

print(sum\_value)

# 练习3:累加10--36之间的和

sum\_value = 0

for item in range(10,37):

sum\_value += item

print(sum\_value)

# 如果不懂，断点调试　＋　内存图

exercise06.py

"""

练习:随机加法考试

随机产生两个数字(1--10),

在控制台中获取两个数相加的结果

如果用户输入正确得１０分

　总共３道题，最后输出得分.

　例如:"请输入8+3=?" 10 不得分

　　　　"请输入4+3=?" 7 得10分

　　　　"请输入4+4=?" 8 得10分

　”总分是20“

"""

import random

score = 0

for item in range(3):

random\_number01 = random.randint(1, 10)

random\_number02 = random.randint(1, 10)

input\_number = int(input("请输入" + str(random\_number01) + "+" + str(random\_number02) + "=?"))

if input\_number == random\_number01 + random\_number02:

score += 10

print("总分：" + str(score))

exercise07.py

在控制台中获取一个整数，判断是否为素数。

# 素数:只能被１和自身整除的正数.

# 思路：排除法,使用２到当前数字之间的正数判断，如果存在被整除，则不是素数.

# 判断9：

# 能否被2　-- 8 之间的数字整除,其中3可以，所以不是素数.

# 判断８:

# 能否被2　-- 7 之间的数字整除,其中2可以，所以不是素数.

# 判断7:

# 能否被2　-- 6 之间的数字整除,其中没有，所以是素数.

# 2 3 5 7 11 13 15 ....

# ---------思考过程----------------

# 假设判断11

# 　　　2 -- 10 之间的数字整除

# if 11 % 2 == 0:

# print("不是素数")

# if 11 % 3 == 0:

# print("不是素数")

# if 11 % 4 == 0:

# print("不是素数")

# 备注：没有判断２以下的数字

number = int(input("请输入整数:"))

#判断2 到number之间的数字，能否整除number.

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

if number % item == 0:

print("不是素数")

break # 如果发现满足条件的数字，就不再判断后面的了。

else:

print("是素数")

课后作业：

在控制台中获取一个整数作为边长．

　　根据边长打印矩形．

例如：４

\*\*\*\*

\* \*

\* \*

\*\*\*\*

6

\*\*\*\*\*\*

\* \*

\* \*

\* \*

\* \*

\*\*\*\*\*\*

"""

number = int(input("请输入整数:")) # 4

print("\*" \* number)

for item in range(number - 2): #

print("\*" + " " \* (number - 2) + "\*")

print("\*" \* number)

### 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

## 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。

Demo03:

#　累加1--100之间,能被５整除的数字

# sum\_value = 0

# for item in range(1,101):

# 满足条件则累加

# if item % 5 == 0:

# sum\_value+=item

# print(sum\_value)

sum\_value = 0

for item in range(1,101):

# 不满足条件则跳过本次循环,继续下次循环。

if item % 5 != 0:

continue

sum\_value+=item

print(sum\_value)

exercise08:

# 累加10-50之间个位不是2,5,9的整数.

# 15:40

sum\_value = 0

for item in range(10, 51):

unit = item % 10

# 个位是2,5,9的整数　则　跳过.

if unit == 2 or unit == 5 or unit == 9:

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,一但不同则返回比较结果。

str01 = "悟空"

str02 = "八戒"

# 字符串拼接

str03 = str01 + str02

# 　字符串累加

str01 += str02

print(str01)

# 重复生成元素

print(str02 \* 3)#八戒八戒八戒

str02 \*=3

print(str02)

# 依次比较两个容器中元素,一但不同则返回比较结果。

print("a悟空" > "b八戒")

### 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。

print("我叫" in "我叫齐天大圣")

### 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)。

# 索引

message = "我叫齐天大圣"

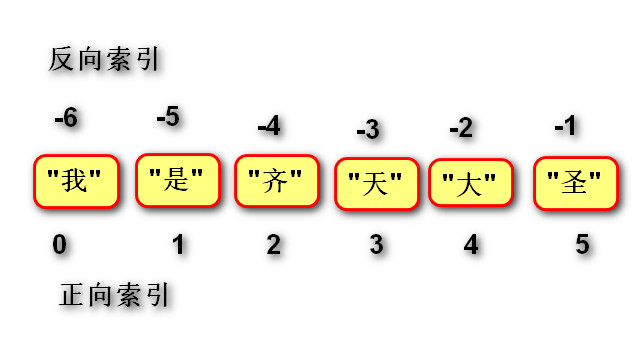
# 获取正数第三个字

print(message[3])

# 获取最后一个字

print(message[-1])

内存图：



### 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

# 切片

print(message[0:2])# 我叫

# 开始值默认为开头

print(message[:2])# 我叫

# 结束值默认为末尾

print(message[-2:])# 大圣

print(message[:])#我叫齐天大圣

print(message[-2:-5:-1])#大天齐

print(message[::-1])# 圣大天齐叫我

print(message[1:1])# 空

print(message[3:1])# 空

print(message[-2:1])# 空

# 索引不能越界

# print(message[7])

# 切片越界不报错

print(message[1:7])# 叫齐天大圣

# 练习:在控制台中获取一个字符串

# 打印第一个字符

# 打印最后一个字符

# 打印倒数第三个字符

# 打印前两个字符

# 倒序打印字符

# 如果字符串长度是奇数，则打印中间字符.

str01 = "我叫齐天大圣."

print(str01[0])

print(str01[-1])

print(str01[-3])

print(str01[:2])

print(str01[::-1])

if len(str01) % 2 == 1:

# print(str01[3])

print(str01[len(str01) // 2])

课后作业：

在控制台中录入一个字符串，判断是否为回文．

判断规则:正向与反向相同．

　　　上海自来水来自海上

message = "上海自来水来自海上"

if message == message[::-1]:

print("是回文")

else:

print("不是回文")

### 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：

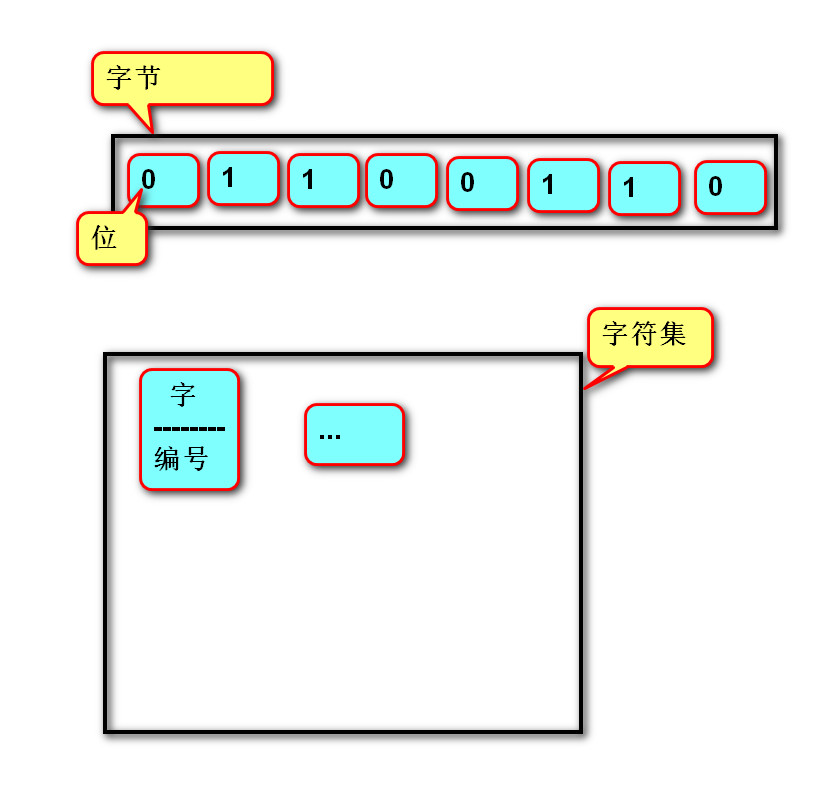
--ASCII编码：包含英文、数字等字符，每个字符1个字节。

--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。

字符集内存图：



#### 相关函数

1. ord(字符串):返回该字符串的Unicode码。

例：# 字　－－> 数

num01 = ord("a")

print(num01)

# 练习1:在控制台中，获取一个字符串.

# 打印每个字符的编码值

str\_input = input("请输入文字：")

for item in str\_input:

print(ord(item))

　2.　chr(整数):返回该整数对应的字符串。

　　 例： # 数　－－> 字

str01 = chr(97)

print(str01)

# 练习2:在控制台中，重复录入一个编码值，然后打印字符.

# 如果输入空字符串，则退出程序.

while True:

str\_code = input("请输入编码值:")

if str\_code == "":

break

code\_value = int(str\_code)

print(chr(code\_value))

### 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

message = '我叫"苏大强"。'

1. 双引号内的单引号不算结束符

message = "我叫'苏大强'。"

#### 三引号作用

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

例：

# 可见即所得

name = '''苏大强'''

name = """

苏

大

强

"""

print(name)

#### 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”

# 转义符：\"　\n \t \\

message = "我叫\"苏大强\"。"

message = "我叫\n苏大强。" # 换行

message = "我叫\t苏大强。" # 水平制表格ｔａｂ键

url = "C:\\nltk\_data\\aodels\\bmt15\_eval"

# 原始字符串(没有转义符)

url = r"C:\nltk\_data\aodels\bmt15\_eval"

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

a = "1"

b = "2"

# "请输入" + str(a) + "+" + str(b) + "=?"

# 在字符串中插入变量

# 请输入1+2=?

# 字符串拼接（缺点：乱）

str01 = "请输入" + a + "+" + b + "=?"

str02 = "请输入%s+%s=?" % (a, b)

str03 = "请输入%s+%.1f=?" % ("1", 10.5678)

print(str03)

## 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 = []

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):

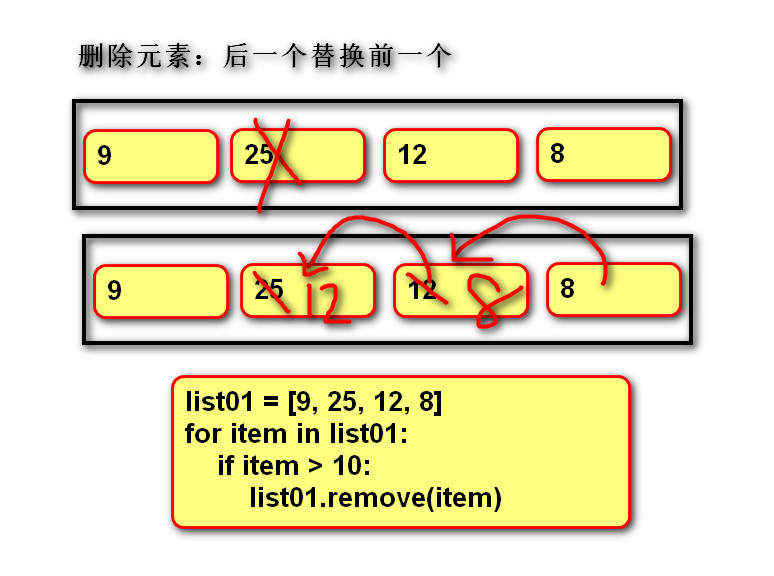
列表名[索引名]就是元素

1. 删除元素：

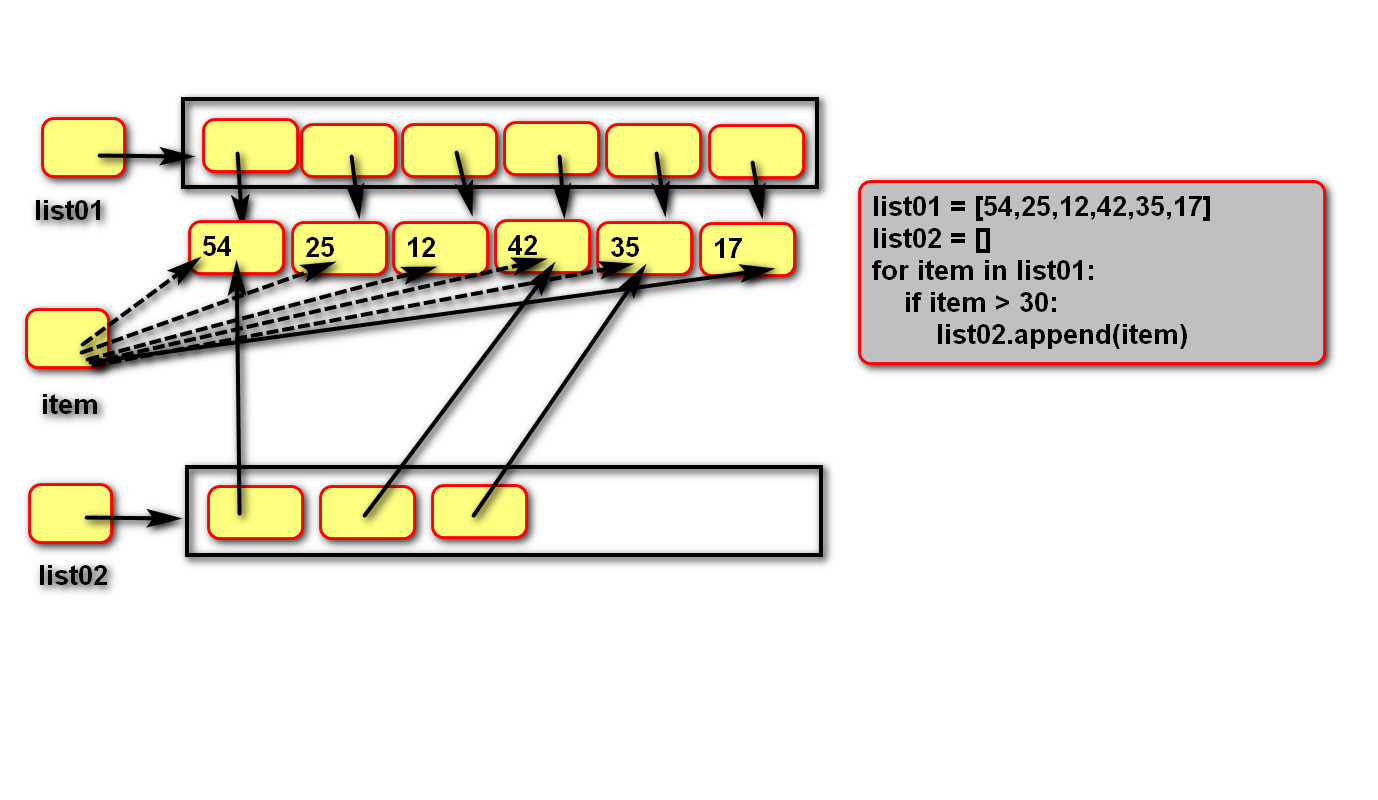
列表名.remove(元素)

del 列表名[索引或切片]

删除原理图：



for 列表内存图：



　笔记：

# 1. 创建列表

# 空

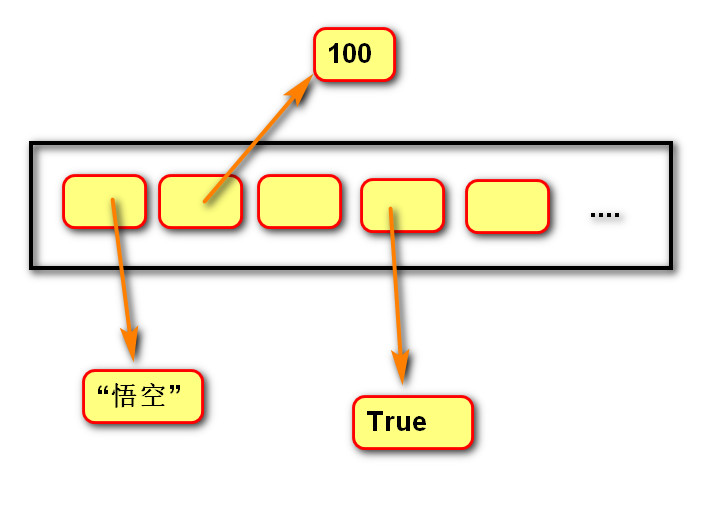
list01 = []

list01 = list()

# 默认值

list02 = ["悟空", 100, True]

list02 = list("我是齐天大圣")



# 2. 获取元素

# 索引

print(list02[2]) # 齐

# 切片

print(list02[-4:]) # ['齐', '天', '大', '圣']

# 3. 添加元素

# 追加(在末尾添加)

list02.append("八戒")

# 插入(在指定位置添加)

list02.insert(1, True) # 在索引为１(第二个)的位置添加True

# 4. 删除元素

# 根据元素删除

list02.remove("是")

# 根据位置删除

del list02[0]

print(list02)

# 5.定义元素，目的：可以增删改查元素。

# 切片

del list02[1:3]

print(list02)

# [True, '大', '圣', '八戒']

# [True, 'a', 'b', '八戒']

list02[1:3] = ["a", "b"]

# [True,'八戒']

# list02[1:3] = []

print(list02)

# 遍历列表

# 获取列表中所有元素

for item in list02:

print(item)

# 倒序获取所有元素

# 不建议

# list02[::-1] 通过切片拿元素，会重新创建新列表.

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

# print(item)

# 3 2 1 0

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

print(list02[i])

# -1 -2 -3 -4

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

print(list02[i])

exercise01:

在控制台中录入，西游记中你喜欢的人物.

输入空字符串，打印(一行一个)所有人物.

list\_person = []

# 录入过程

while True:

str\_input = input("输入在西游记中喜欢的人物:")

if str\_input == "":

break

list\_person.append(str\_input)

# 输出过程

for item in list\_person:

print(item)

exercise02:

# 练习:在控制台中录入，所有学生成绩.

# 输入空字符串，打印(一行一个)所有成绩.

# 打印最高分,打印最低分,打印平均分.

list\_score = []

# 录入过程

while True:

str\_score = input("请输入成绩：")

if str\_score == "":

break

list\_score.append(int(str\_score)) # 输出过程

for item in list\_score:

print(item)

print("最高分：" + str(max(list\_score)))

print("最低分：" + str(min(list\_score)))

print("平均分：" + str(sum(list\_score) / len(list\_score)))

exercise03:

"""

# 在控制台中录入，所有学生姓名.

# 如果姓名重复，则提示"姓名已经存在",不添加到列表中.

# 如果录入空字符串，则倒叙打印所有学生.

list\_name = []

while True:

name = input("请输入姓名:")

if name == "":

break

# 判断变量在列表中是否存在

if name not in list\_name:

list\_name.append(name)

else:

print("姓名已经存在")

# -1 -2 -3

# 2 1 0

for item in range(-1, -len(list\_name) - 1, -1):

print(list\_name[item])

demo02:

list01 = ["张无忌","赵敏"]

list02 = list01

# 修改的是列表第一个元素

list01[0] = "无忌"

print(list02[0])

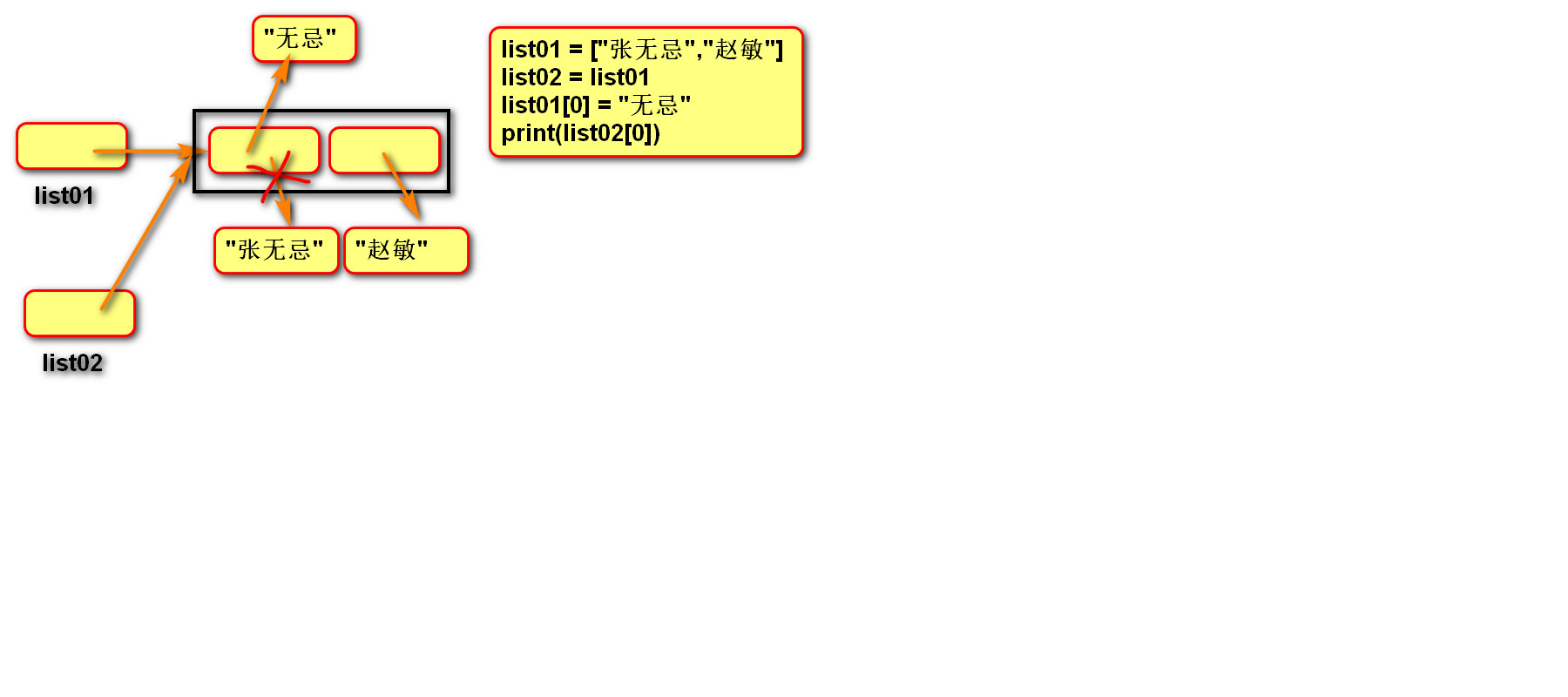
list01 = ["张无忌","赵敏"]

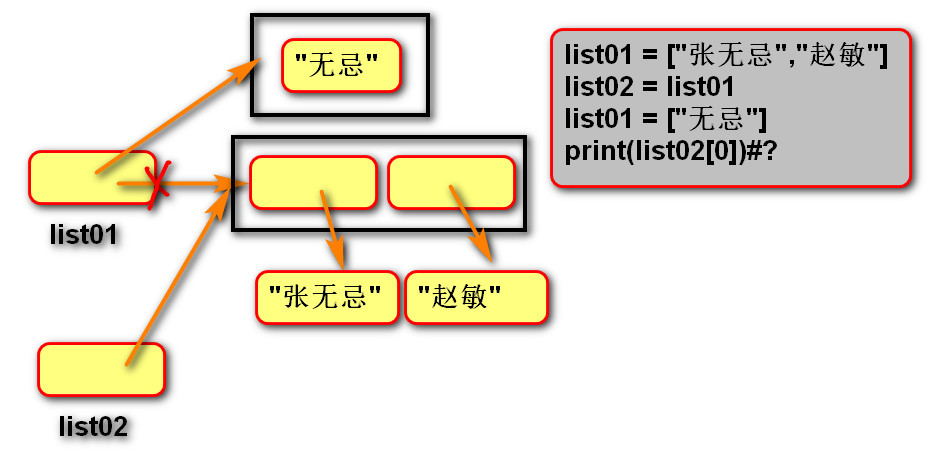
list02 = list01

# 修改的是list01变量

list01 = ["无忌"]

print(list02[0])#张无忌



list01 = [800,1000]

# 通过切片获取元素，会创建新列表.

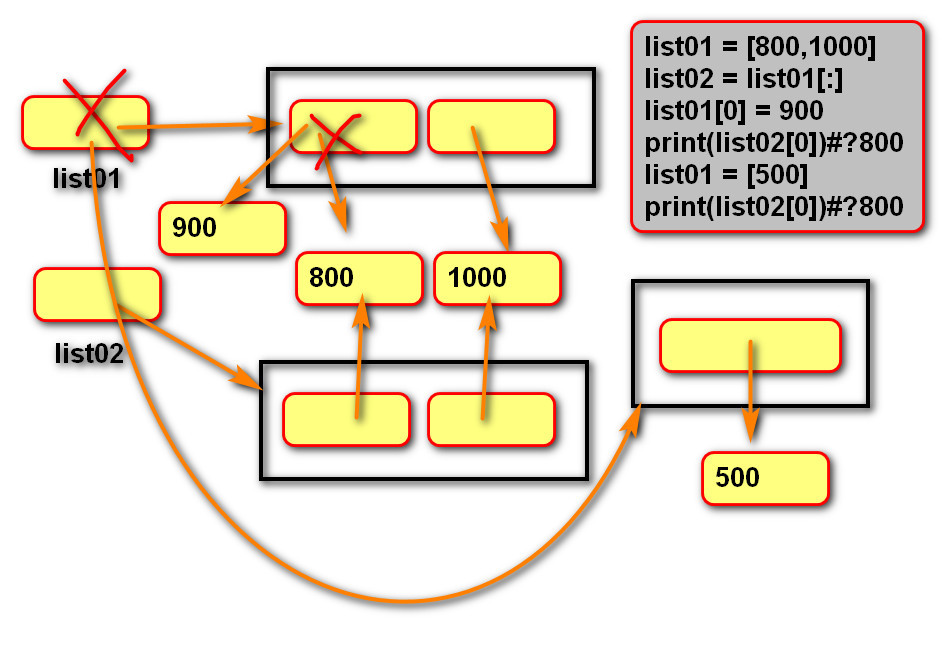
list02 = list01[:]

list01[0] = 900

print(list02[0])#?800

list01 = [500]

print(list02[0])#?800

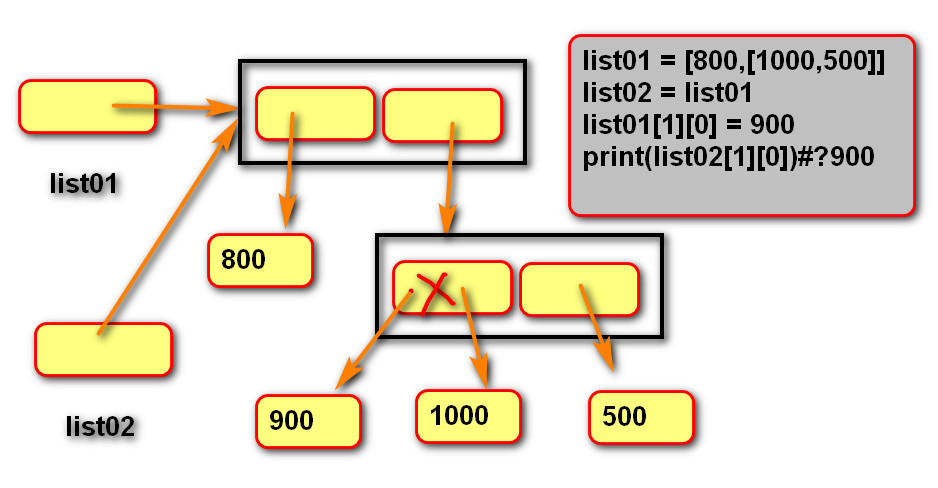
# 列表套列表

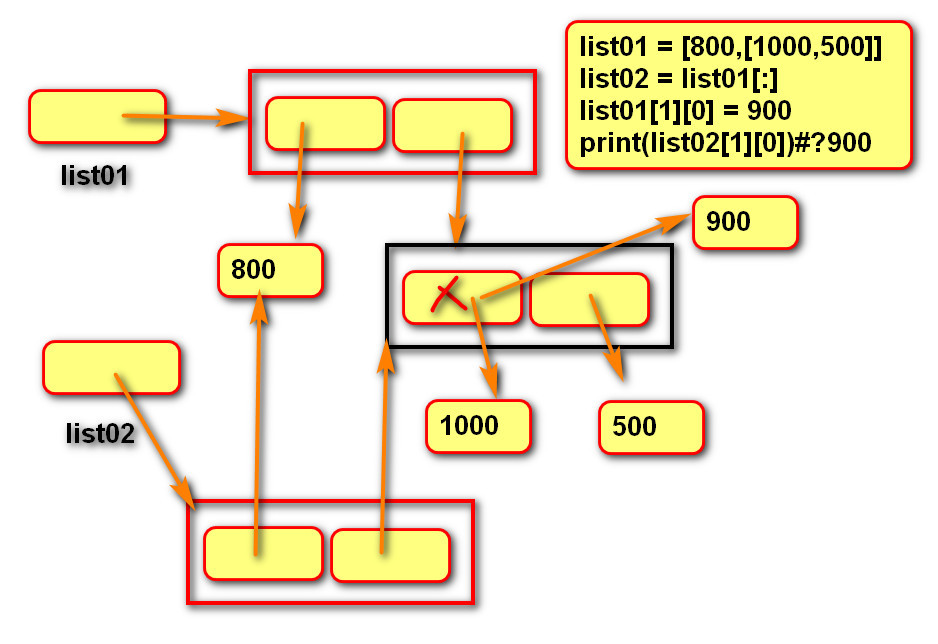
list01 = [800,[1000,500]]

list02 = list01

list01[1][0] = 900

print(list02[1][0])#?900





exercise04:

"""

练习１:

将列表[54,25,12,42,35,17]中，

大于30的数字存入另外一个列表.

并画出内存图.

练习２：

在控制台中录入５个数字，

打印最大值（不适用max）.

"""

# 练习１：

list01 = [54, 25, 12, 42, 35, 17]

list02 = []

for item in list01:

if item > 30:

list02.append(item)

print(list02)

# 练习2:

# 假设的最大值

max\_value = 0

for item in range(5):

number = int(input("请输入第%d个数字:" % (item + 1)))

if max\_value < number:

max\_value = number

print(max\_value)

exercise05:

# 练习３:

# 在列表中[54, 25, 12, 42, 35, 17]，选出最大值(不使用max).

list01 = [54, 25, 12, 42, 100, 17]

# 假设第一个是最大的

max\_value = list01[0]

# 与后面（从第二个开始）元素进行比较

# 1 2 3 4 5

for i in range(1, len(list01)):

if max\_value < list01[i]:

# 如果发现更大的，则替换假设的.

max\_value = list01[i]

print(max\_value)

exercise06:

# 练习４:在列表中[9, 25, 12, 8]，删除大于10的数字.

# 17:10

list01 = [9, 25, 12, 8]

# for item in list01:

# if item > 10:

# list01.remove(item)

#3 2 1 0

#-1 -2 -3 -4

for i in range(len(list01)-1,-1,-1):

if list01[i] > 10:

list01.remove(list01[i])

print(list01)

### 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。

Demo02:

list01 = [800,[1000,500]]

# 浅拷贝

# list02 = list01[:]

list02 = list01.copy()

list01[1][0] = 900

print(list02[1][0])#?900

import copy

list01 = [800,[1000,500]]

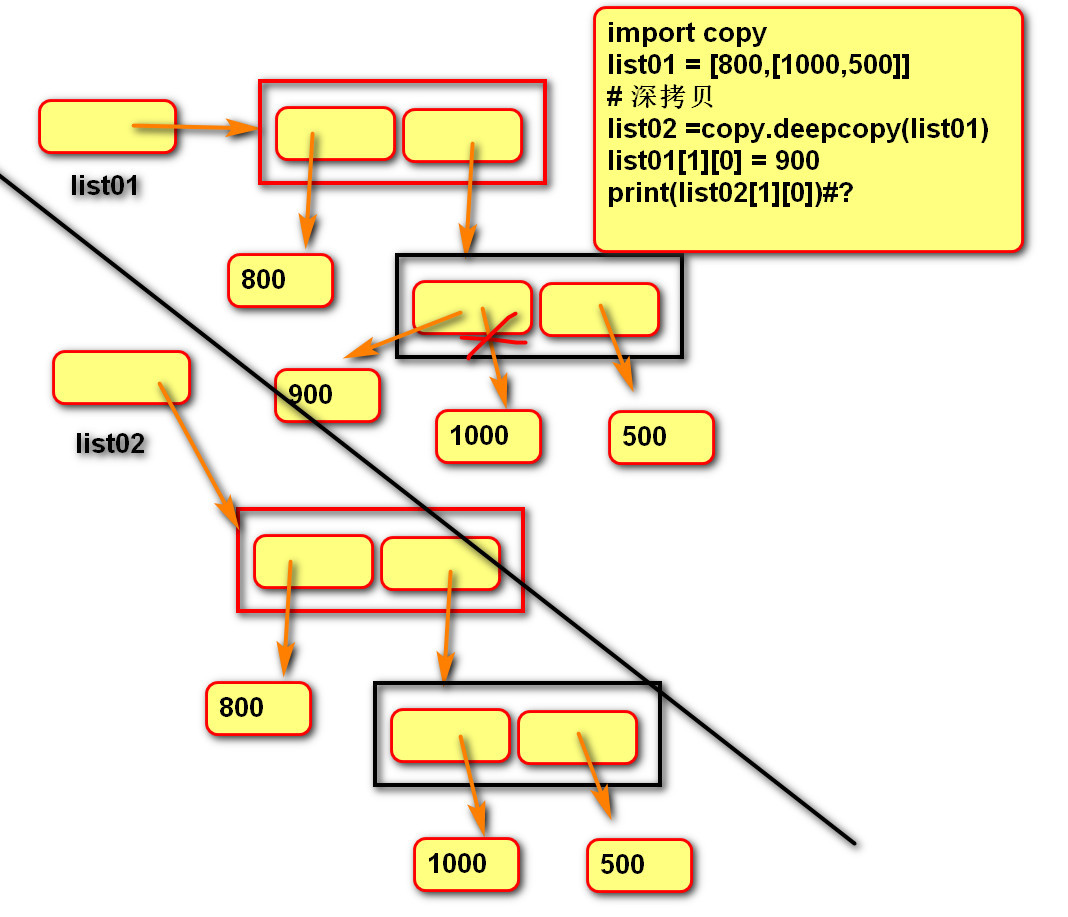
# 深拷贝

list02 =copy.deepcopy(list01)

list01[1][0] = 900

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

内存图：



### 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)

"""

list-->str

"""

# 需求：根据ｘｘ逻辑，拼接一个字符串.

# "0123456789"

# 缺点：每次循环形成（+=）一个新的字符串对象,替换变量引用result。

# result = ""

# for item in range(10):

# #""

# #"0"

# #"01"

# #"012"

# result = result + str(item)

# 优点：每次循环只向列表添加字符串，没有创建列表对象。

list\_temp = []

for item in range(10):

list\_temp.append(str(item))

# join : list --> str

result = " ".join(list\_temp)

print(type(result))

print(result)

# 练习:在控制台中循环输入字符串,如果输入空则停止。

# 最后打印所有内容（拼接后的字符串）.

list\_result = []

while True:

str\_input = input("请输入：")

if str\_input == "":

break

list\_result.append(str\_input)

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

print(str\_result)

将一个字符串拆分为多个。

列表 = “a-b-c-d”.split(“分隔符”)

"""

str --> list

"""

str01 = "张无忌-赵敏-周芷若"

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

print(list\_result)

#　练习:英文单词翻转

# "How are you" -->"you are How"

str01 = "How are you"

list\_temp = str01.split(" ")

str\_result = " ".join(list\_temp[::-1])

print(str\_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

"""

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

list02 = ["A", "B", "C"]

list03 = []

for r in list01:

for c in list02:

list03.append(r + c)

print(list03)

list04 = [r + c for r in list01 for c in list02]

print(list04)

# 练习:列表的全排列

# [“香蕉”,"苹果","哈密瓜"]

# [“可乐”,"牛奶"]

list01 = ["香蕉","苹果","哈密瓜"]

list02 = ["可乐","牛奶"]

list03 = []

for r in list01:

for c in list02:

list03.append(r+c)

list04 = [r+c for r in list01 for c in list02]

print(list03)

print(list03)

# 将list01中所有元素,增加１以后存入list02中.

list01 = [5, 56, 6, 7, 7, 8, 19]

# list02 = []

# for item in list01:

# list02.append(item + 1)

list02 = [item + 1 for item in list01]

print(list02)

# 将list01中大于１０元素,增加１以后存入list02中.

# list02 = []

# for item in list01:

# if item >10:

# list02.append(item + 1)

list02 = [item + 1 for item in list01 if item > 10]

# 练习:使用range生成1--10之间的数字,将数字的平方存入list01中

# 将list01中所有奇数存入list02

# 将list01中所有偶数存入list03

# 将list01中所有偶数大于5的数字增加1后存入list04

# list01 = []

# for item in range(1, 11):

# list01.append(item \*\* 2)

list01 = [item \*\* 2 for item in range(1, 11)]

# list02 = []

# for item in list01:

# if item % 2 == 1:

# list02.append(item)

list02 = [item for item in list01 if item % 2 == 1]

list03 = [item for item in list01 if item % 2 == 0]

# list04 = []

# for item in list01:

# if item % 2 == 0 and item > 5:

# list04.append(item + 1)

list04 = [item + 1 for item in list01 if item % 2 == 0 and item > 5]

print(list04)

## 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。

### 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 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):

元祖名[索引名]就是元素

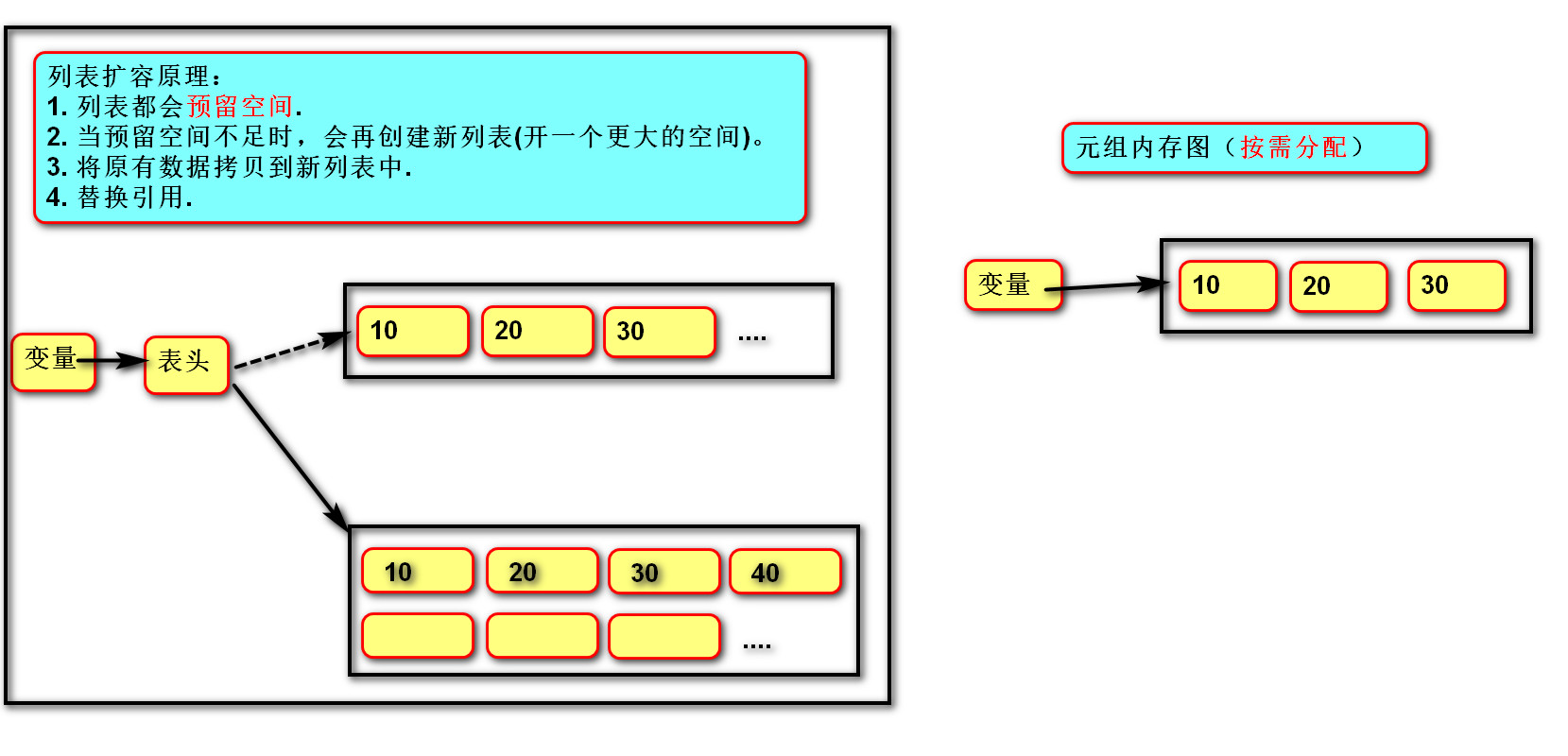
### 作用

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)

内存图：



笔记：

"""

元组

基础操作

"""

# 1. 创建元组

# 空

tuple01 = ()

# 　具有默认值

tuple01 = (1, 2, 3)

# 列表　--> 元组

tuple01 = tuple(["a", "b"])

print(tuple01)

# 元组　--> 列表

list01 = list(tuple01)

print(list01)

# 如果元组只有一个元素

# tuple02 = (100)

# print(type(tuple02))# int

tuple02 = (100,)

print(type(tuple02)) # int

# 不能变化

# tuple02[0] = 10

# 2. 获取元素（索引 切片）

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

e01 = tuple03[1]

print(type(e01)) # str

e02 = tuple03[-2:]

print(type(e02)) # tuple

# 可以直接将元组赋值给多个变量

tuper04 = (100, 200)

a, b = tuper04

print(a)

print(b)

# 3. 遍历元素

# 正向

for item in tuper04:

print(item)

# 反向

# 1 0

for i in range(len(tuper04) - 1, -1, -1):

print(tuper04[i])

exercise02:

"""

练习:借助元组完成下列功能.

"""

# month = int(input("请输入月份："))

#

# if month < 1 or month > 12:

# print("输入有误")

# elif month == 2:

# print("２８天")

# elif month == 4 or month == 6 or month == 9\

# or month == 11:

# print("３０天")

# else:

# print("３１天")

# 方式１：

# month = int(input("请输入月份："))

#

# if month < 1 or month > 12:

# print("输入有误")

# elif month == 2:

# print("２８天")

# elif month in (4,6,9,11):

# print("３０天")

# else:

# print("３１天")

# 方式2:

month = int(input("请输入月份："))

if month < 1 or month > 12:

print("输入有误")

else:

# 将每月天数存入元组

day\_of\_month = (31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31)

print(day\_of\_month[month – 1])

exercise03:

"""

练习:在控制台中录入日期(月日)，计算这是这一年的第几天.

例如：３月５日

1月天数 + 2月天数 + 5

5月8日

1月天数 + 2月天数 +3月天数 + 4月天数+ 8

"""

day\_of\_month = (31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31)

month = int(input("请输入月份"))

day = int(input("请输入日："))

# 方法一：

# # 累加前几个月天数

total\_day = 0

for i in range(month - 1):

total\_day += day\_of\_month[i]

# 累加当月天数

total\_day += day

print("是这年的第%d天." % total\_day)

# 方法二:

# 累加前几个月天数

total\_day = sum(day\_of\_month[:month - 1])

total\_day += day

print("是这年的第%d天." % total\_day)

## 字典 dict

### 定义

1. 由一系列键值对组成的可变映射容器。
2. 映射：一对一的对应关系，且每条记录无序。
3. 键必须惟一且不可变(字符串/数字/元组)，值没有限制。

### 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]

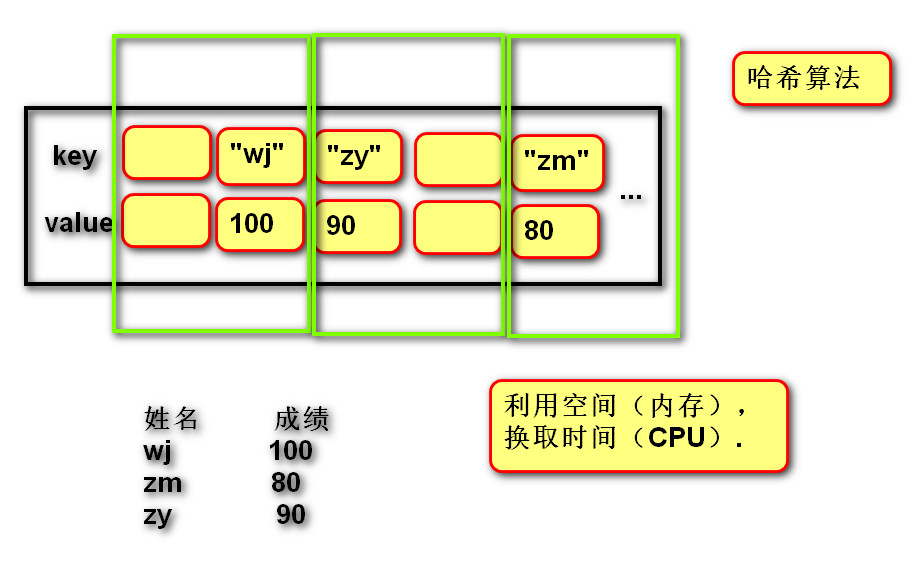
for 键名,值名 in 字典名.items():

语句

1. 删除元素：

del 字典名[键]

内存图：



笔记：

"""

字典

"""

# 1. 创建

# 空

dict01 = {}

dict01 = dict()

# 默认值

dict01 = {"wj":100,"zm":80,"zr":90}

dict01 = dict([("a","b"),("c","d")])

print(dict01)

# 2.　查找元素(根据ｋｅｙ查找ｖａｌｕｅ)

print(dict01["a"])

# 如果ｋｅｙ不存在，查找时会错误.

if "qtx" in dict01:# 如果存在key

print(dict01["qtx"])

# 3.　修改元素(之前存在ｋｅｙ)

dict01["a"] = "BB"

# 4. 添加(之前不存在ｋｅｙ)

dict01["e"] = "f"

# 5. 删除

del dict01["a"]

print(dict01)

# 6. 遍历（获取字典中所有元素）

# 遍历字典，获取key

for key in dict01:

print(key)

print(dict01[key])

# 遍历字典，获取value

for value in dict01.values():

print(value)

# 遍历字典，获取键值对key value(元组).

# for item in dict01.items():

# print(item[0])

# print(item[1])

for k,v in dict01.items():

print(k)

print(v)

exercise04:

练习1:在控制台中循环录入商品信息(名称,单价).

　如果名称输入空字符,则停止录入.

将所有信息逐行打印出来.

dict\_commodity\_info = {}

while True:

name = input("请输入商品名称：")

if name == "":

break

price = int(input("请输入商品单价："))

dict\_commodity\_info[name] = price

for key,value in dict\_commodity\_info.items():

print("%s商品单价是%d"%(key,value))

exercise05:

# 练习2: 在控制台中循环录入学生信息(姓名,年龄,成绩,性别).

# 　　　如果名称输入空字符, 则停止录入.

# 将所有信息逐行打印出来.

"""

# 字典内嵌列表:

{

"张无忌":[28,100,"男"],

}

"""

dict\_student\_info = {}

while True:

name = input("请输入姓名：")

if name == "":

break

age = int(input("请输入年龄："))

score = int(input("请输入成绩："))

sex = input("请输入性别：")

dict\_student\_info[name] = [age, score, sex]

# 打印所有学生信息

for name,list\_info in dict\_student\_info.items():

print("%s的年龄是%d,成绩是%d,性别是%s"%(name,list\_info[0],list\_info[1],list\_info[2]))

exercise06:

"""

# 字典内嵌字典:

{

"张无忌":{"age":28,"score":100,"sex":"男"},

}

"""

dict\_student\_info = {}

while True:

name = input("请输入姓名：")

if name == "":

break

age = int(input("请输入年龄："))

score = int(input("请输入成绩："))

sex = input("请输入性别：")

dict\_student\_info[name] = {"age": age, "score": score, "sex": sex}

for name, dict\_info in dict\_student\_info.items():

print("%s的年龄是%d,成绩是%d,性别是%s" %

(name, dict\_info["age"],

dict\_info["score"], dict\_info["sex"]))

exercise07:

"""

# 列表内嵌字典:

[

{"name":"张无忌","age":28,"score":100,"sex":"男"},

]

"""

list\_student\_info = []

while True:

name = input("请输入姓名：")

if name == "":

break

age = int(input("请输入年龄："))

score = int(input("请输入成绩："))

sex = input("请输入性别：")

dict\_info = {"name": name, "age": age, "score": score, "sex": sex}

list\_student\_info.append(dict\_info)

for dict\_info in list\_student\_info:

print("%s的年龄是%d,成绩是%d,性别是%s" % (dict\_info["name"], dict\_info["age"], dict\_info["score"], dict\_info["sex"]))

# 获取第一个学生信息

dict\_info = list\_student\_info[0]

print("第一个录入的是：%s,年龄是%d,成绩是%d,性别是%s" % (dict\_info["name"], dict\_info["age"], dict\_info["score"], dict\_info["sex"]))

exercise08:

"""

总结：

　　存储多个学生信息(姓名,年龄,成绩,性别)的多种方式

1. exercise05字典内嵌列表:

{

"张无忌":[28,100,"男"],

}

2. exercise06字典内嵌字典:

{

"张无忌":{"age":28,"score":100,"sex":"男"},

}

3. exercise07列表内嵌字典:

[

{"name":"张无忌","age":28,"score":100,"sex":"男"},

]

4. 列表内嵌列表

[

["张无忌",28,100男],

]

选择策略：根据具体需求，结合优缺点，综合考虑(两害相权取其轻).

字典：

优点：根据键获取值，读取速度快；

　　　代码可读性相对列表更高(根据键获取与根据索引获取).

缺点：内存占用大；

　　　获取值只能根据键,不灵活.

列表：

优点：根据索引/切片，获取元素更灵活.

相比字典占内存更小。

缺点：通过索引获取，如果信息较多，可读性不高.

"""

"""

练习:在控制台中录入多个人的多个喜好,输入空字符停止.

例如:请输入姓名：

请输入第1个喜好：

请输入第2个喜好：

...

请输入姓名：

...

最后在控制台打印所有人的所有喜好.

[

{“无忌”:[赵敏,周芷若,小赵]}

]

list\_person\_bobby = []

while True:

name = input("请输入姓名：")

if name == "":

break

dict\_person = {name:[]}

list\_person\_bobby.append(dict\_person)

while True:

bobby = input("请输入喜好：")

if bobby == "":

break

dict\_person[name].append(bobby)

{

“无忌”:[赵敏,周芷若,小赵]

}

"""

dict\_person\_bobby = {}

while True:

name = input("请输入姓名：")

if name == "":

break

dict\_person\_bobby[name] = []

while True:

bobby = input("请输入喜好：")

if bobby == "":

break

dict\_person\_bobby[name].append(bobby)

for name, list\_bobby in dict\_person\_bobby.items():

print("%s喜欢：" % name)

for item in list\_bobby:

print(item)

课后作业：计算列表中最小值(不使用min)

list01 = [43, 54, 5]

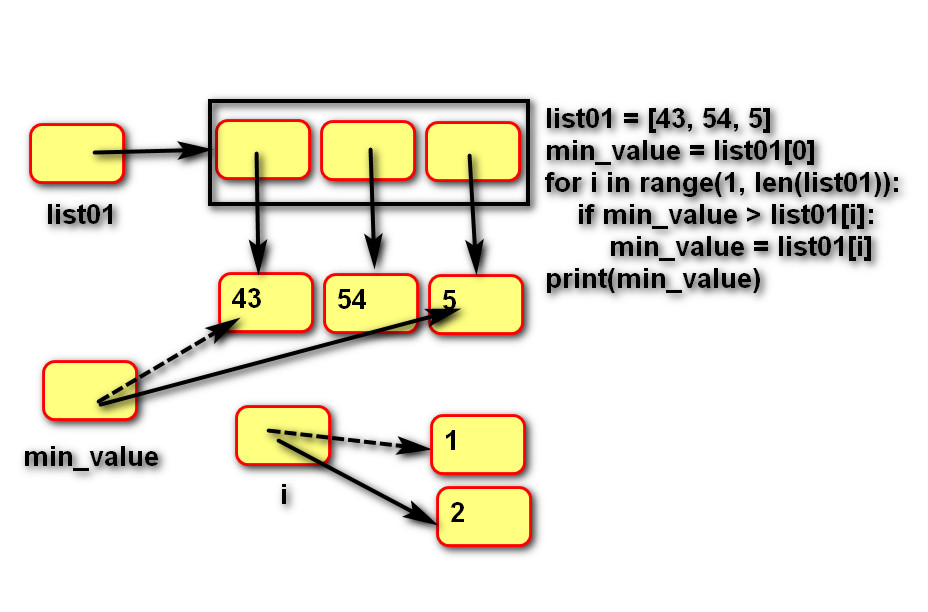
min\_value = list01[0]

for i in range(1, len(list01)):

if min\_value > list01[i]:

min\_value = list01[i]

print(min\_value)



课后作业：

"""

(扩展)计算一个字符串中的字符以及出现的次数.

# 思想：

# 逐一判断字符出现的次数.

# 如果统计过则增加１，如果没统计过则等于１.

abcdefce

a 1

b 1

c 2

d 1

e 2

f 1

"""

dict\_result = {}

str\_target = "abcdefce"

for item in str\_target:

if item not in dict\_result:

dict\_result[item] = 1

else:

dict\_result[item] += 1

print(dict\_result)

### 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

"""

# 1 2 3 4 ... 10 -> 平方

dict01 = {}

for item in range(1, 11):

dict01[item] = item \*\* 2

print(dict01)

# 推导式:

dict02 = {item: item \*\* 2

for item in range(1, 11)}

print(dict02)

# 只记录大于５的数字

dict01 = {}

for item in range(1, 11):

if item >5:

dict01[item] = item \*\* 2

print(dict01)

dict02 = {item: item \*\* 2

for item in range(1, 11) if item >5}

print(dict02)

exercise01:

# 练习1:["无忌","赵敏","周芷若"]

# # ->{"无忌":2,"赵敏":2,"周芷若":3}

list01 = ["无忌", "赵敏", "周芷若"]

dict01 = {}

for item in list01:

dict01[item] = len(item)

dict02 = {item: len(item) for item in list01}

print(dict01)

print(dict02)

exercise02:

# 练习2:["无忌","赵敏","周芷若"] [101,102,103]

# {"无忌":101,"赵敏":102,"周芷若":103}

# 10:18

list01 = ["无忌", "赵敏", "周芷若"]

list02 = [101, 101, 103]

dict01 = {}

# 通过索引同时在多个列表中获取元素

for i in range(len(list01)):

# key = list01[i]

# value = list02[i]

# dict01[key] = value

dict01[list01[i]] = list02[i]

print(dict01)

# 需求：字典如何根据value查找key

# 解决方案１:键值互换

dict02 = {value: key for key, value in dict01.items()}

print(dict02)

print(dict02[101])

# 缺点:如果key重复,交换或则丢失数据。

# 如果需要保持所有数据

# [(k,v),]

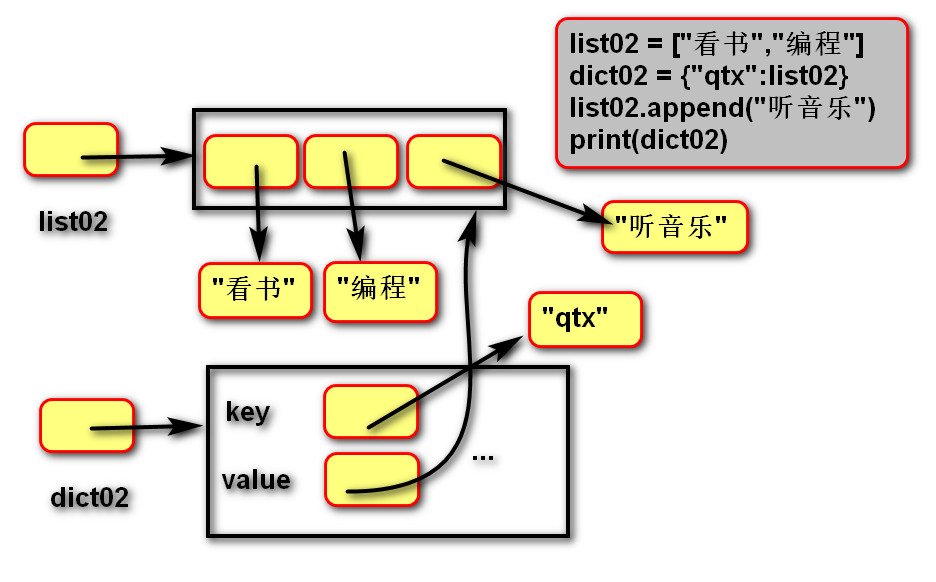
list02 = [(value, key) for key, value in dict01.items()]

print(list02)

### 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。

字典内嵌列表内存图：



## 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映射容器。
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)

笔记：

# 1. 创建集合

set01 = set()

# set --> str

set01 = set("abcac")

list01 = list(set01)

str01 = "".join(list01)

print(str01) # "bca"

# 创建具有默认值的集合

set02 = {"a", "b", "a"}

# 2. 添加元素

set02.add("qtx")

# 3. 删除元素

set02.remove("a")

# 4. 获取所有元素

for item in set02:

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 <= >=

笔记：

set01 = {1, 2, 3}

set02 = {2, 3, 4}

# 交集

print(set01 & set02) # {2,3}

# 并集

print(set01 | set02) # {1, 2, 3, 4}

# 补集

print(set01 ^ set02) # {1, 4}

print(set01 - set02) # {1}

print(set02 - set01) # {4}

# 子集

set03 = {1, 2}

print(set03 < set01)

# 超集

print(set01 > set03)

### 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

"""

set01 = frozenset([1,2,3,3,5])

list02 = list(set01)

print(set01)

print(list02)

补充：for嵌套

demo04:

# 外层循环控制行

for r in range(3):# 0 1 2

# 内层循环控制列

for c in range(4):# 0123 0123 0123

print("\*",end = " ")

print()

"""

\*#\*#\*#

\*#\*#\*#

\*#\*#\*#

\*#\*#\*#

"""

for r in range(4):

for c in range(6):

if c % 2 ==0:

print("\*", end = " ")

else:

print("#", end = " ")

print()

""" 外层4　　　　内层

\* 0

\*\* 01

\*\*\* 012

\*\*\*\* 0123

"""

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

#

for c in range(r+1):#1 2 3 4

print("\*",end = "")

print()

exercise05:

# 列表排序(升序小　--> 大)

# [3,80,45,5,7,1]

# 目标:列表中所有元素两两比较

# 思想:

# 　取出第一个元素,与后面元素进行比较.

# 　取出第二个元素,与后面元素进行比较.

# 　取出第三个元素,与后面元素进行比较.

# ...

# 取出倒数第二个元素,与后面元素进行比较.

# 如果取出的元素大于(>)后面的元素,

# 则交换

list01 = [3, 80, 45, 5, 7, 1]

# 取出第一个元素,与后面元素进行比较

# list01[0] list01[1]

# list01[0] list01[2]

# list01[0] list01[3]

# for c in range(1,len(list01)):

# # list01[0] list01[c]

# pass

# 取出第二个元素,与后面元素进行比较

# for c in range(2,len(list01)):

# # list01[1] list01[c]

# pass

# 取出第三个元素,与后面元素进行比较

# for c in range(3,len(list01)):

# # list01[2] list01[c]

# pass

# 取数据

for r in range(len(list01) - 1):

# 作比较

for c in range(r + 1, len(list01)):

# list01[2] list01[c]

if list01[r] > list01[c]:

list01[r], list01[c] = list01[c], list01[r]

print(list01)

exercise06:

"""

判断列表中元素是否具有相同的[3,80,45,5,80,1]

思路：所有元素俩俩比较,发现相同的则打印结果

　　　所有元素比较结束，都没有发现相同项，则打印结果.

"""

list01 = [3, 81, 3, 5, 81, 5]

# 取出第一个，与后面比较

# list01[0] list01[1]

# list01[0] list01[2]

# list01[0] list01[3]

# for c in range(1,len(list01)):

# # list01[0] list01[c]

# pass

# for c in range(2,len(list01)):

# # list01[1] list01[c]

# pass

# for c in range(3,len(list01)):

# # list01[2] list01[c]

# pass

# 结果：假设没有相同项

result = False

for r in range(0, len(list01) - 1):

for c in range(r + 1, len(list01)):

if list01[r] == list01[c]:

print("具有相同项")

result = True

break # 退出循环

if result:

break

if result == False:

print("没有相同项")

exercise07:

# 练习1:打印第二行第三个元素 15:56

# 练习2:打印第三行每个元素

# 练习3:打印第一列每个元素

list01 = [

[1, 2, 3, 4],

[5, 6, 7, 8],

[9, 10, 11, 12],

[13, 14, 15, 16],

]

print(list01[1][2])

for item in list01[2]:

print(item)

# 00 10 20 30 [行索引][列索引]

for r in range(len(list01)):

print(list01[r][0])

exercise08:

# 练习:矩阵转置 将二维列表的列，变成行，形成一个新列表.

# 第一列变成第一行

# 第二列变成第二行

# 第三列变成第三行

list01 = [

[1,2,3,4],

[5,6,7,8],

[9,10,11,12],

[13,14,15,16],

]

# result = []

# for c in range(4):

# line = []

# result.append(line)

# #00 10 20 30

# for r in range(4):

# line.append(list01[r][c])

result = []

for c in range(len(list01[0])):

result.append([])

for r in range(len(list01)):

result[c].append(list01[r][c])

print(result)

"""

4. (扩展)方阵转置.（不用做成函数）

提示：详见图片.

"""

list01 = [

[1, 2, 3, 4],

[5, 6, 7, 8],

[9, 10, 11, 12],

[13, 14, 15, 16],

]

"""

#list01[1][0] <->list01[0][1]

#list01[2][0] <->list01[0][2]

#list01[3][0] <->list01[0][3]

for r in range(1,4):

#list01[r][0]<->list01[0][r]

pass

#list01[2][1] <->list01[1][2]

#list01[3][1] <->list01[1][3]

for r in range(2,4):#2 3

# list01[r][1] <->list01[1][r]

pass

#list01[3][2] <->list01[2][3]

for r in range(3,4):

# list01[r][2] <->list01[2][r]

pass

for c in range(1,4):#1 2 3

for r in range(c,4):

list01[r][c-1],list01[c-1][r]=list01[c-1][r],list01[r][c-1]

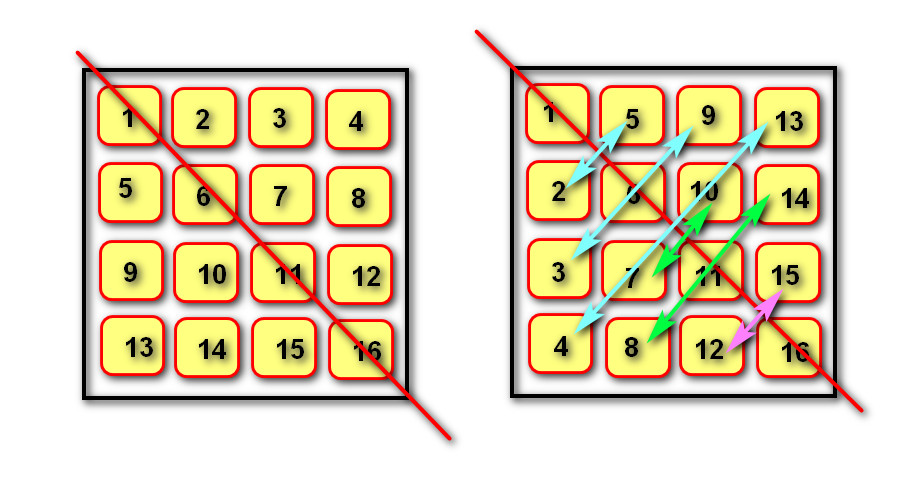
"""

for c in range(1, len(list01)): # 1 2 3

for r in range(c, len(list01)):

list01[r][c - 1], list01[c - 1][r] = list01[c - 1][r], list01[r][c - 1]

print(list01)



# 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。

"""

自定义函数

"""

"""

print("直拳")

print("摆拳")

print("肘击")

print("临门一脚")

#...............

print("直拳")

print("摆拳")

print("肘击")

print("临门一脚")

#...............

print("直拳")

print("摆拳")

print("肘击")

print("临门一脚")

"""

# 定义（做功能）函数

def attack():

"""

单次攻击

"""

print("临门一脚")

print("直拳")

print("摆拳")

print("肘击")

# 形式参数

def attack\_repeat(count):

"""

重复攻击

:param count: 攻击次数,int类型

"""

for i in range(count):

print("临门一脚")

print("直拳")

print("摆拳")

print("肘击")

# 调用函数

attack()

# ...............

# 调用函数

attack()

# ...............

# 调用函数

attack()

print("--------------")

# 调用函数

# 实际参数

attack\_repeat(2)

exercise1:

# 练习:将下列代码，定义到函数中，再调用一次.

def print\_rectangle(r\_count,c\_count,char):

"""

打印矩形

:param r\_count: 行数

:param c\_count: 列数

:param char: 填充的字符

"""

for r in range(r\_count):

# 内层循环控制列

for c in range(c\_count):

print(char, end=" ")

print()

print\_rectangle(5,2,"#")

exercise02:

# 练习:定义在控制台中打印一维列表的函数.

# 例如：[1,2,3]-->1 2 3 每个元素一行

def print\_list(list\_target):

"""

打印列表

:param list\_target: 目标列表

"""

for item in list\_target:

print(item)

list01 = [1, 2, 3]

list02 = ["a", True, 1.5, 10]

print\_list(list02)

"""

3. 定义在控制台中打印二维列表的函数

[

[1,2,3,44],

[4,5,5,5,65,6,87],

[7,5]

]

1 2 3 44

4 5 5 5 65 6 87

7 5

"""

def print\_double\_list(double\_list):

"""

打印二维列表

:param double\_list: 需要打印的二维列表

"""

for line in double\_list:

for item in line:

print(item, end=" ")

print()

list01 = [

[1, 2, 3, 44],

[4, 5, 5, 5, 65, 6, 87],

[7, 5]

]

print\_double\_list(list01)

## 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。

"""

函数返回值 语法

"""

# 参数：调用者传递给定义者的信息

# 返回值：定义者传递给调用者的结果

def fun01(a):

print("fun01执行喽")

# 作用：1. 返回结果 2.退出方法

return 20

print("fun01又执行喽")

# F8 逐过程　（调试时跳过方法）

# F7 逐语句 （调试时进入方法）

re = fun01(10)

print(re)

# 无返回值函数

def fun02(a):

print("fun01执行喽")

# return None

re = fun02(100)

print(re)

"""

函数返回值　应用

"""

# 设计思想：分而治之

# 干一件事

# 需求：定义两个数字相加的函数

# def add():

# 1. 获取数据

# number01 = int(input("请输入第一个数字："))

# number02 = int(input("请输入第二个数字："))

# 2. 逻辑计算

# result = number01 + number02

# 3. 显示结果

# print(result)

#

# add()

def add(number01, number02):

# 逻辑处理

return number01 + number02

# 调用者提供数据

number01 = int(input("请输入第一个数字："))

number02 = int(input("请输入第二个数字："))

result = add(number01, number02)

# 调用者负责显示结果

print("结果是:" + str(result))

"""

练习1:定义计算四位整数，每位相加和的函数.

测试："1234" "5428"

"""

def each\_unit\_sum(number):

"""

计算整数的每位相加和

:param number: 四位整数

:return: 相加的结果

"""

result = number % 10

result += number // 10 % 10

result += number // 100 % 10

result += number // 1000

return result

# 测试

re01 = each\_unit\_sum(1234)

print(re01)

re01 = each\_unit\_sum(4875)

print(re01)

# 练习2:定义根据两,计算几斤零几两的函数

# weight\_liang = int(input("请输入两："))

# jin = weight\_liang // 16

# liang = weight\_liang % 16

# print(str(jin) + "斤零" + str(liang) + "两")

def get\_weight\_for\_jin(liang\_weight):

"""

根据两,计算几斤零几两.

:param liang\_weight:需要计算的两

:return: 元组 (斤,两)

"""

jin = liang\_weight // 16

liang = liang\_weight % 16

return (jin,liang)

re = get\_weight\_for\_jin(100)

print(str(re[0]) + "斤零" + str(re[1]) + "两")

# 练习:定义 根据成绩计算等级 的函数

# score = int(input("请输入成绩："))

# if score > 100 or score < 0:

# print("输入有误")

# elif 90 <= score:

# print("优秀")

# elif 80 <= score:

# print("良好")

# elif 60 <= score:

# print("及格")

# else:

# print("不及格")

# def get\_score\_level(score):

# if score > 100 or score < 0:

# return "输入有误"

# elif 90 <= score:

# return "优秀"

# elif 80 <= score:

# return "良好"

# elif 60 <= score:

# return "及格"

# else:

# return "不及格"

def get\_score\_level(score):

if score > 100 or score < 0:

return "输入有误"

if 90 <= score:

return "优秀"

if 80 <= score:

return "良好"

if 60 <= score:

return "及格"

return "不及格"

print(get\_score\_level(80))

# 练习:定义　判断列表中是否存在相同元素的　函数

# list01 = [3, 81, 3, 5, 81, 5]

# result = False

# for r in range(0, len(list01) - 1):

# for c in range(r + 1, len(list01)):

# if list01[r] == list01[c]:

# print("具有相同项")

# result = True

# break # 退出循环

# if result:

# break

# if result == False:

# print("没有相同项")

# def is\_repeating(list\_target):

# for r in range(0, len(list\_target) - 1):

# for c in range(r + 1, len(list\_target)):

# if list\_target[r] == list\_target[c]:

# return "具有相同项"

# return "没有相同项"

def is\_repeating(list\_target):

for r in range(0, len(list\_target) - 1):

for c in range(r + 1, len(list\_target)):

if list\_target[r] == list\_target[c]:

return True # 有重复

return False # 没有重复

print(is\_repeating([3, 8, 23, 5, 81, 1]))

# 定义函数,根据年月，计算有多少天。考虑闰年29天，平年28天

# month = int(input("请输入月份："))

# if month < 1 or month > 12:

# print("输入有误")

# elif month == 2:

# print("２８天")

# elif month in (4,6,9,11):

# print("３０天")

# else:

# print("３１天")

# def get\_day\_by\_month(year, month):

# if month < 1 or month > 12:

# return "输入有误"

# if month == 2:

# if year % 4 == 0 and year % 100 != 0 or year % 400 == 0:

# return "29天"

# else:

# return "28天"

# if month in (4, 6, 9, 11):

# return "３０天"

# return "３１天"

# 不建议方法的返回值类型可能是多种

# bool int

# def get\_day\_by\_month(year, month):

# if month < 1 or month > 12:

# return False

# if month == 2:

# if year % 4 == 0 and year % 100 != 0 or year % 400 == 0:

# return 29

# else:

# return 28

# if month in (4, 6, 9, 11):

# return 30

# return 31

# def get\_day\_by\_month(year, month):

# if month < 1 or month > 12:

# return 0

# if month == 2:

# if year % 4 == 0 and year % 100 != 0 or year % 400 == 0:

# return 29

# else:

# return 28

# if month in (4, 6, 9, 11):

# return 30

# return 31

def is\_leap\_year(year):

# if year % 4 == 0 and year % 100 != 0 or year % 400 == 0:

# return True

# else:

# return False

return year % 4 == 0 and year % 100 != 0 or year % 400 == 0

# def get\_day\_by\_month(year, month):

# if month < 1 or month > 12:

# return 0

# if month == 2:

# if is\_leap\_year(year):

# return 29

# else:

# return 28

# if month in (4, 6, 9, 11):

# return 30

# return 31

def get\_day\_by\_month(year, month):

if month < 1 or month > 12:

return 0

if month == 2:

return 29 if is\_leap\_year(year) else 28

if month in (4, 6, 9, 11):

return 30

return 31

print(get\_day\_by\_month(2019,5))

## 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。

"""

函数内存图

"""

# 在方法区中存储函数代码,不执行函数体

def fun01(a):

a = 100

num01 = 1

# 因为调用函数，所以开辟一块内存空间，叫做栈帧

# 用于存储在函数内部定义的变量(包含参数).

fun01(num01)

# 函数执行完毕后，栈帧立即释放(其中定义的变量也会销毁).

print(num01)#1



def fun02(a):

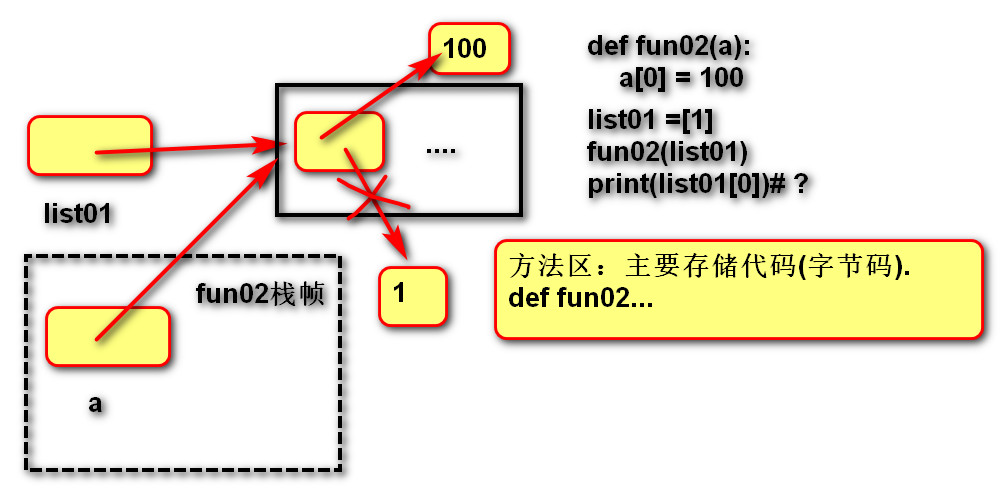
# 改变的是传入的可变对象

a[0] = 100

list01 = [1]

fun02(list01)

print(list01[0])# 100



def fun03(a):

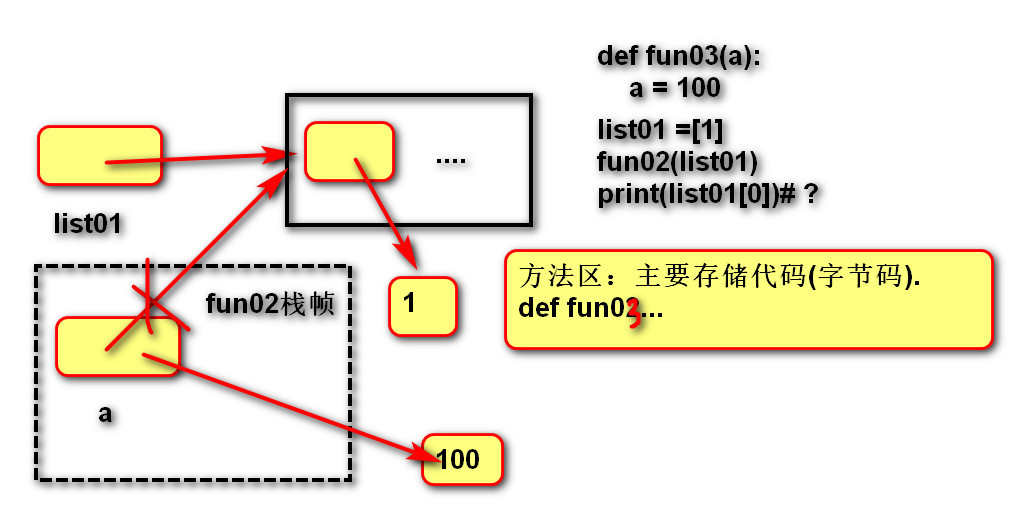
# 改变的是fun03栈帧中变量a的指向

a = 100

list01 = [1]

fun03(list01)

print(list01[0])# 1



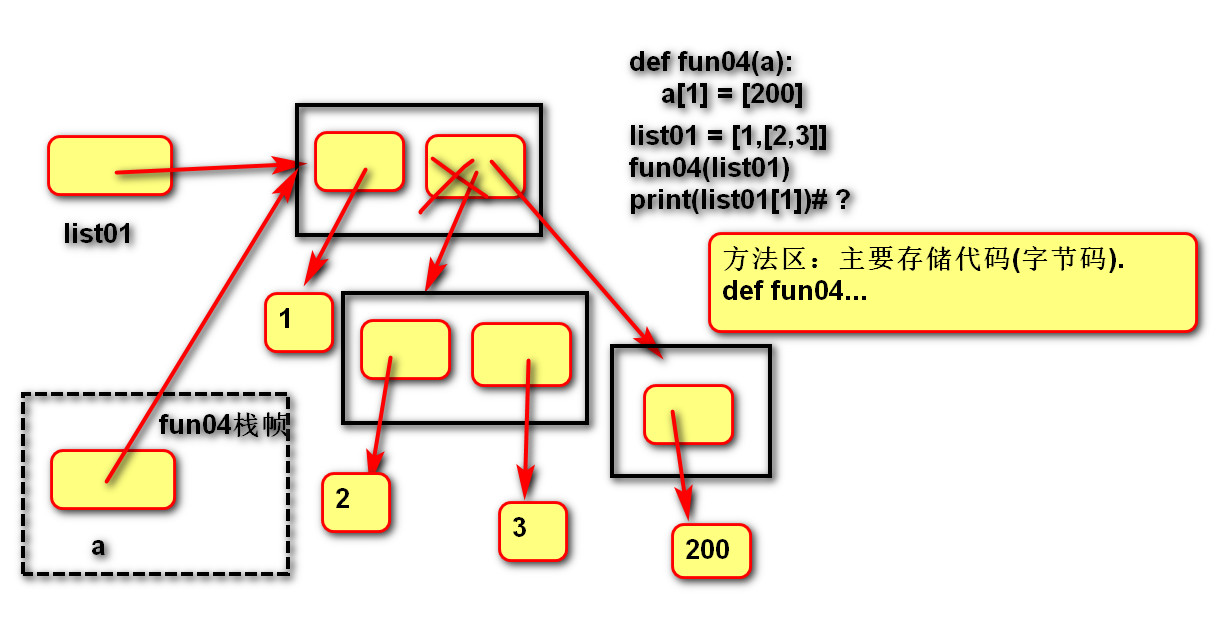
def fun04(a):

a[1] = [200]

list01 = [1,[2,3]]

fun04(list01)

print(list01[1])# [200]



# 定义列表升序排列的函数

# list01= [43,4,5,6,7]

# for r in range(len(list01) - 1):

# for c in range(r + 1, len(list01)):

# if list01[r] > list01[c]:

# list01[r], list01[c] = list01[c], list01[r]

#

# print(list01)

def sort(list\_target):

# 满足以下两个条件，就无需通过返回值传递结果。

# 1.传入的是可变对象

# 2.函数体修改的是传入的对象

for r in range(len(list\_target) - 1):

for c in range(r + 1, len(list\_target)):

if list\_target[r] > list\_target[c]:

list\_target[r], list\_target[c] = list\_target[c], list\_target[r]

list01 = [43, 4, 5, 6, 7]

sort(list01)

print(list01)

# 练习：定义方阵转置函数

# list01 = [

# [1, 2, 3, 4],

# [5, 6, 7, 8],

# [9, 10, 11, 12],

# [13, 14, 15, 16],

# ]

# for c in range(1, len(list01)): # 1 2 3

# for r in range(c, len(list01)):

# list01[r][c - 1], list01[c - 1][r] = list01[c - 1][r], list01[r][c - 1]

# print(list01)

def square\_matrix\_transpose(sqr\_matrix):

"""

方阵转置

:param sqr\_matrix: 二维列表类型的方阵

"""

for c in range(1, len(sqr\_matrix)):

for r in range(c, len(sqr\_matrix)):

sqr\_matrix[r][c - 1], sqr\_matrix[c - 1][r] = sqr\_matrix[c - 1][r], sqr\_matrix[r][c - 1]

list01 = [

[1, 2, 3, 4],

[5, 6, 7, 8],

[9, 10, 11, 12],

[13, 14, 15, 16],

]

# 矩阵转置的转置等于原矩阵

square\_matrix\_transpose(list01)

print(list01)

square\_matrix\_transpose(list01)

print(list01)

## 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, d):

print(a)

print(b)

print(c)

print(d)

# 位置实参：实参与形参根据位置进行依次对应.

# fun01(1, 2, 3, 4)

# 关键字实参:实参与形参根据名称进行对应

# fun01(b=1, d=2, c=3, a=4)

# 序列实参:星号将序列拆分后按位置与形参进行对应

# 如果参数很多，可以存储在序列(字符串/列表/元组)中,

# 　　再通过\*拆分,直接传入函数.

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

# fun01(\*list01)

# 字典实参:双星号将字典拆分后按名称与形参进行对应

# 如果参数很多，可以存储在字典中,

# 　　 再通过\*\*拆分,传入函数.

dict01 = {"a": 1, "c": 3, "d": 4, "b": 2}

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：

收集多余的位置传参。

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

形式参数

"""

# 1. 缺省(默认)形参:如果实参不提供，可以使用默认值.

def fun01(a=None, b=0, c=0, d=0):

print(a)

print(b)

print(c)

print(d)

# 关键字实参 + 缺省形参:调用者可以随意传递参数.

# fun01(b=2, c=3)

# 2. 位置形参

def fun02(a, b, c, d):

print(a)

print(b)

print(c)

print(d)

# 3.星号元组形参: \* 将所有实参合并为一个元组

# 作用：让实参个数无限

def fun03(\*args):

print(args)

# fun03()# ()

# fun03(1)# (1,)

# fun03(1,"2")# (1, '2')

# 4.命名关键字形参:在星号元组形参以后的位置形参

# 目的：要求实参必须使用关键字实参.

def fun04(a, \*args, b):

print(a)

print(args)

print(b)

fun04(1, b=2)

fun04(1, 2, 3, 4, b=2)

def fun05(\*, a, b):

print(a)

print(b)

fun05(a=1, b=2)

# 5. 双星号字典形参：\*\*目的是将实参合并为字典.

# 实参可以传递数量无限的关键字实参.

def fun06(\*\*kwargs):

print(kwargs)

fun06(a=1, b=2)

# 作业:调用fun07。

def fun07(a, b, \*args, c, d, \*\*kwargs):

pass

# 练习:定义函数，根据小时，分钟，秒，计算总秒数.

# 要求：可以只计算小时-->秒

# 　　　可以只计算分钟-->秒

# 　　　可以只计算小时＋分钟-->秒

# 　　　可以只计算小时＋秒-->秒

def get\_total\_second(hour=0, minute=0, second=0):

return hour \* 3600 + minute \* 60 + second

# 小时，分钟，秒，

print(get\_total\_second(1, 1, 1))

# 小时，分钟

print(get\_total\_second(2, 3))

# 分钟，秒，

print(get\_total\_second(minute=2, second=3))

# 小时，

print(get\_total\_second(2))

# 分钟，

print(get\_total\_second(minute=2))

# 练习:定义函数，数值相加的函数.

def adds(\*args):

# result = 0

# for item in args:

# result += item

# return result

return sum(args)

print(adds(1, 22, 3, 3, 4, 5, 6))

print(adds(1, 4, 5, 6))

# 调用fun07。

# ｜位置｜｜星号元组｜｜命名关键字｜｜双星号字典｜

def fun07(a, b, \*args, c, d, \*\*kwargs):

print(a)

print(b)

print(args)

print(c)

print(d)

print(kwargs)

fun07(1, 2, 3, 4, 5, c=6, d=7, e=8, f=9)

# 位置实参无限　＋　关键字实参无限

def fun01(\*args, \*\*kwargs):

print(args)

print(kwargs)

fun01(1, 2, 3, a=4, c=5)

"""

4. 定义函数，计算指定范围内的素数

"""

# def get\_prime(begin,end):#5 30

# list\_result = []

# # 生成范围内的整数

# for number in range(begin,end):

# # 判断素数

# for item in range(2, number):

# if number % item == 0:

# break

# else:

# list\_result.append(number)

# return list\_result

def is\_prime(number):

"""

判断指定的数字是否为素数

:param

number: 指定的整数

:return: True

表示是素数，False表示不是素数.

"""

for item in range(2, number):

if number % item == 0:

return False

return True

# def get\_prime(begin, end): # 5 30

# \"""

# 获取范围内的素数

# :param begin: 开始值(包含)

# :param end: 结束值(不包含)

# :return: 所有素数的列表

# \"""

# list\_result = []

# for number in range(begin, end):

# if is\_prime(number):

# list\_result.append(number)

# return list\_result

def get\_prime(begin, end): # 5 30

""""

获取范围内的素数

:param

begin: 开始值(包含)

:param

end: 结束值(不包含)

:return: 所有素数的列表

"""

return [number for number in range(begin, end) if is\_prime(number)]

print(get\_prime(5, 30))

# 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。

# 全局变量

g01 = "ok"

# print(l01)

def fun01():

# 局部变量：在函数内部定义的变量

l01 = 100

# print(l01)

print(l01)

# 　在函数内部可以读取全局变量

# print(g01)

# 创建了一个局部变量g01，而不是修改全局变量

# g01 = "no"

# 定义全局变量g01

global g01

# 此时修改的是全局变量

g01 = "no"

print(g01)

# 定义全局变量g02

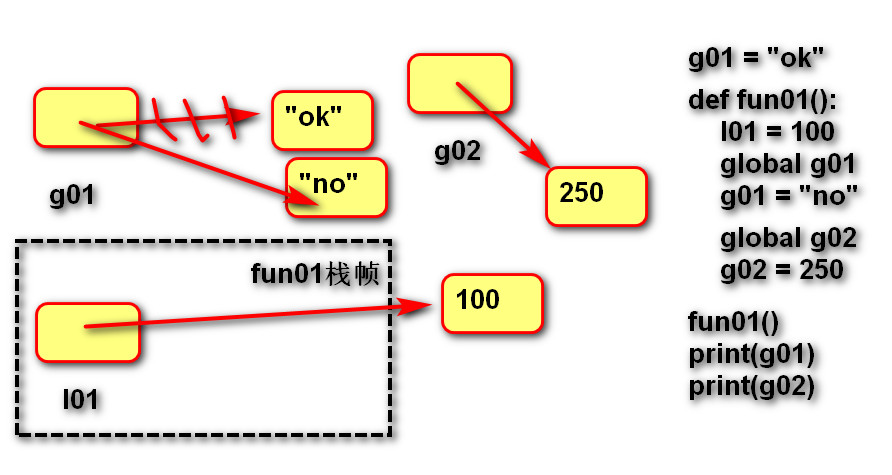
global g02

g02 = 250

fun01()

print(g01) # ?

print(g02)



# 练习:记录一个函数fun01的执行次数.

# 画出内存图

# def fun01():

# pass

#

# fun01()

# fun01()

# fun01()

# fun01()

# fun01()

# print("调用?次")

count = 0

def fun01():

global count

count += 1

fun01()

fun01()

fun01()

fun01()

fun01()

print("调用" + str(count) + "次")

## 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

"""

２０４８　游戏核心算法

"""

list\_merge = None

# 练习１：零元素移至末尾

# 　　[2,0,2,0] --> [2,2,0,0]

# [2,0,0,2] --> [2,2,0,0]

# [2,4,0,2] --> [2,4,2,0]

def zero\_to\_end():

"""

零元素移动到末尾.

"""

# 思想：从后向前，如果发现零元素，删除并追加.

for i in range(-1, -len(list\_merge) - 1, -1):

if list\_merge[i] == 0:

del list\_merge[i]

list\_merge.append(0)

# 测试．．．

# zero\_to\_end()

# print(list\_merge)

# 练习2：将相同数字合并 14:38

# 　　[2,2,0,0] --> [4,0,0,0]

# [2,0,0,2] --> [4,0,0,0]

# [2,0,4,0] --> [2,4,0,0]

# [2,2,2,2] --> [4,4,0,0]

# [2,2,2,0] --> [4,2,0,0]

def merge():

"""

合并

"""

# 先将中间的零元素移到末尾

# 再合并相邻相同元素

zero\_to\_end()

for i in range(len(list\_merge) - 1):

if list\_merge[i] == list\_merge[i + 1]:

# 将后一个累加前一个之上

list\_merge[i] += list\_merge[i + 1]

del list\_merge[i + 1]

list\_merge.append(0)

# 测试...

# merge()

# print(list\_merge)

# 练习3:地图向左移动

map = [

[2, 0, 0, 2],

[4, 4, 2, 2],

[2, 4, 0, 4],

[0, 0, 2, 2],

]

def move\_left():

"""

向左移动

"""

# 思想:将二维列表中每行交给merge函数进行操作

for line in map:

global list\_merge

list\_merge = line

merge()

# move\_left()

# print(map)

def move\_right():

"""

向右移动

"""

# 思想:将二维列表中每行(从右向左)交给merge函数进行操作

for line in map:

global list\_merge

# 从右向左取出数据　形成　新列表

list\_merge = line[::-1]

merge()

# 从右向左接受　合并后的数据

line[::-1] = list\_merge

# move\_right()

# print(map)

# 练习4:向上移动　　向下移动

def move\_up():

square\_matrix\_transpose(map)

move\_left()

square\_matrix\_transpose(map)

def move\_down():

square\_matrix\_transpose(map)

move\_right()

square\_matrix\_transpose(map)

# 提示:利用方阵转置函数

def square\_matrix\_transpose(sqr\_matrix):

"""

方阵转置

:param sqr\_matrix: 二维列表类型的方阵

"""

for c in range(1, len(sqr\_matrix)):

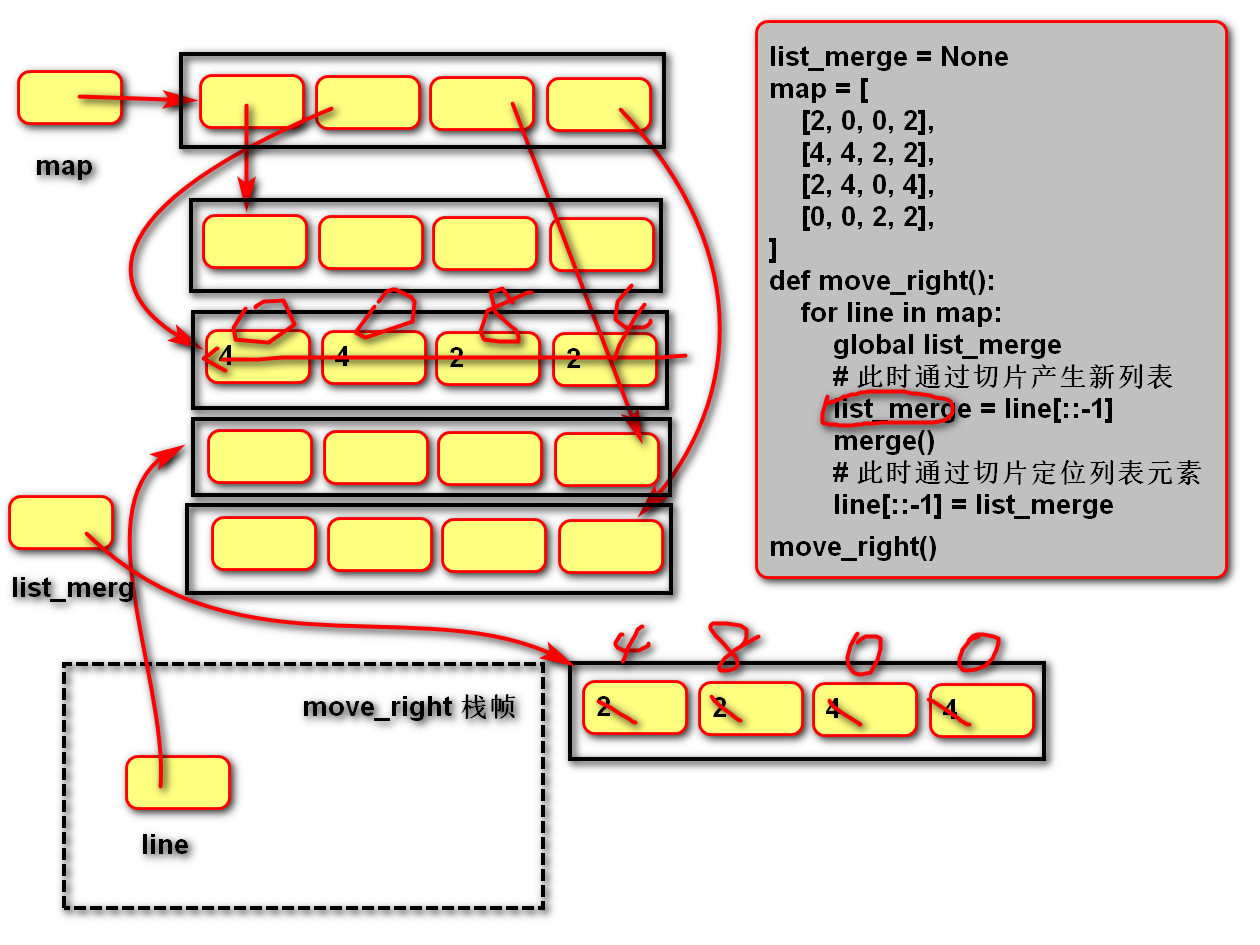
for r in range(c, len(sqr\_matrix)):

sqr\_matrix[r][c - 1], sqr\_matrix[c - 1][r] = sqr\_matrix[c - 1][r], sqr\_matrix[r][c - 1]

move\_down()

print(map)

向右移动内存图：



重构代码

dict\_commodity\_info = {

101: {"name": "屠龙刀", "price": 10000},

102: {"name": "倚天剑", "price": 10000},

103: {"name": "九阴白骨爪", "price": 8000},

104: {"name": "九阳神功", "price": 9000},

105: {"name": "降龙十八掌", "price": 8000},

106: {"name": "乾坤大挪移", "price": 10000}

}

list\_order = []

def select\_menu():

"""

选择菜单

"""

while True:

item = input("1键购买，2键结算。")

if item == "1":

buying()

elif item == "2":

settlement()

def settlement():

"""

结算

"""

print\_orders()

total\_price = calculate\_total\_price()

paying(total\_price)

def paying(total\_price):

"""

支付过程

:param total\_price: 需要支付的价格

"""

while True:

money = float(input("总价%d元，请输入金额：" % total\_price))

if money >= total\_price:

print("购买成功，找回：%d元。" % (money - total\_price))

list\_order.clear()

break

else:

print("金额不足.")

def calculate\_total\_price():

"""

计算总价格

"""

total\_price = 0

for order in list\_order:

commodity = dict\_commodity\_info[order["cid"]]

total\_price += commodity["price"] \* order["count"]

return total\_price

def print\_orders():

"""

打印订单

"""

for order in list\_order:

commodity = dict\_commodity\_info[order["cid"]]

print("商品：%s，单价：%d,数量:%d." % (commodity["name"], commodity["price"], order["count"]))

def buying():

"""

购买

"""

print\_commodity\_info()

create\_order()

print("添加到购物车。")

def create\_order():

"""

创建订单

"""

cid = input\_commodity\_id()

count = int(input("请输入购买数量："))

order = {"cid": cid, "count": count}

list\_order.append(order)

def input\_commodity\_id():

"""

获取商品订单

"""

while True:

cid = int(input("请输入商品编号："))

if cid in dict\_commodity\_info:

break

else:

print("该商品不存在")

return cid

def print\_commodity\_info():

"""

打印商品信息

"""

for key, value in dict\_commodity\_info.items():

print("编号：%d，名称：%s，单价：%d。" % (key, value["name"], value["price"]))

select\_menu()