# 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：

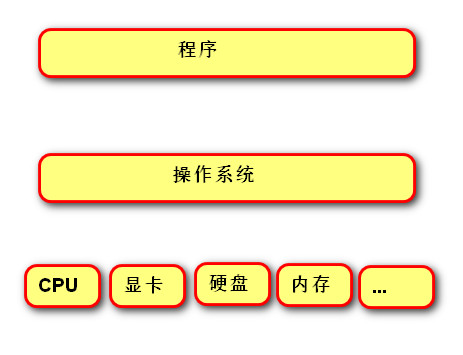
主板--计算机的主要电路系统。

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（能看到源码）、跨平台（在各种操作系统都能用）、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

## 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

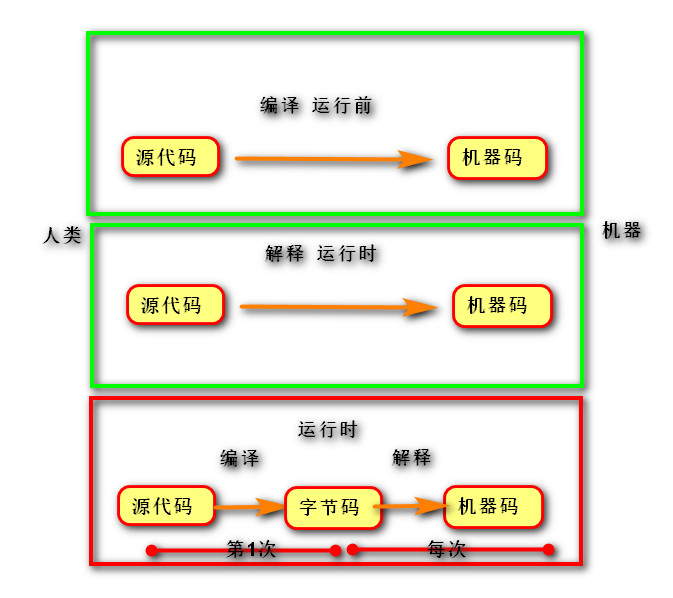
-- 优点：开发效率高，可以跨平台；

-- 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———|



## 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

# 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)

## 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

# 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。

## 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。

# 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。

# 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)

### 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

*bool  
运算符  
 比较运算符 > < >= <= 等于== 不等于!=  
 　　　结果是：ｂｏｏｌ类型  
 逻辑运算符：判断两个ｂｏｏｌ值关系  
 　　　　　　与　　或　　　非*

# 　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)

## 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))

## 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

### 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

### 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

# 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)

#### 非 not

表示取反

例如：

not True # 返回False

not False # 返回True

#### 短路运算

一但结果确定，后面的语句将不再执行。

### 身份运算符

语法:

x is y

x is not y

a = 800  
b = 1000  
# id函数，可以获取变量存储的对象地址。  
print(id(a)) #4505264  
print(id(b)) #13524208  
# flase  
print(a **is** b) # is 的本质就是通过ｉｄ函数进行判断的 #False  
  
c = a  
print(id(c)) #4505264  
print(c **is** a) #True  
  
d = 1000  
print(d **is** b) #True

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

### 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 = 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。

### if 语句的真值表达式

if 100:

print("真值")

等同于

if bool(100):

print("真值")

*真值表达式  
 if 数据:  
 语句  
 本质就是使用bool函数操作数据*

# 1. 真值表达式  
**if** "a":  
 # if bool("a"): #a只要不是0，返回的都是真值

print("真值")  
  
str\_input = input("请输入：")  
**if** str\_input:  
 print("输入的字符串不是空的")

### 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。

# 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)

## 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

# 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

# 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)

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 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。

### 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

# 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)

#　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)

## 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。

# 容器类型

## 通用操作

### 数学运算符

str01 = "悟空"  
str02 = "八戒"

1. +：用于拼接两个容器

str03 = str01 + str02

1. +=：用原容器与右侧容器拼接,并重新绑定变量

str01 += str02

1. \*：重复生成容器元素

print(str02 \* 3) #八戒八戒八戒

str02 \*=3  
print(str02) ) #八戒八戒八戒

1. \*=：用原容器生成重复元素, 并重新绑定变量
2. < <= > >= == !=：依次比较两个容器中元素,一但不同则返回比较结果。

### 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。

### 索引index

1. 作用：访问容器元素
2. 语法：容器[整数]

# 索引  
message = "我叫齐天大圣"  
# 获取正数第4个字  
print(message[3]) #天

# 获取最后一个字  
print(message[-1])

1. 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)。

### 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))]

# 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

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

### 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。

# 字　－－> 数  
num01 = ord("a")  
print(num01) #97  
  
# 数　－－> 字  
str01 = chr(97)  
print(str01) #a

### 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

#### 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”

#### 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

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) #在列表最后面添加

列表.insert(索引，元素) #在列表指定位置添加

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

修改元素：

列表名[0] = “ a” #修改第一个值为a

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

for 索引名 in range(0,len(列表名),1): #把列表的第一个到最后一个拿出来

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]

#删除多个元素的时候，得从后面开始删,但是别用-1，-2，-3这种方法（会报错），要采用2，1，0这种方法。

### 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。

### 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) #列表转字符串

将一个字符串拆分为多个。

列表 = “a-b-c-d”.split(“分隔符”)

### 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]

# 实例：  
# 将1970年到2050年中的闰年，存入列表。

# 方法一  
list\_runnian = []  
**for** item **in** range(1970,2051):  
 **if** item % 4 == 0 **and** item % 100 != 0 **or** item % 400 == 0:  
 list\_runnian.append(item)  
print(list\_runnian)  
# 方法二  
# 使用 列表推导式 完成  
list\_runnian = [item **for** item **in** range(1970,2051) **if** item % 4 == 0 **and** item % 100 != 0 **or** item % 400 == 0]  
print(list\_runnian)

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  
 练习:exercise09.py  
 16:55  
"""*# 两个元素两两拼接  
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] #列表推导式嵌套  
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)

## 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)

## 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. 创建  
# 空  
dict01 = {}  
dict01 = dict()  
# 默认值  
dict01 = {"wj":100,"zm":80,"zr":90}  
dict01 = dict([("a","b"),("c","d")])  
print(dict01)

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。

# 修改元素(之前存在ｋｅｙ)  
dict01["a"] = "BB"

# 添加(之前不存在ｋｅｙ)  
dict01["e"] = "f"

扩展：

list02 = ["看书", "编程", "美食"]  
dict02 = {"qtx": list02}  
list02.append("听音乐")  
print(dict02) #{'qtx': ['看书', '编程', '美食', '听音乐']}

1. 获取元素：

变量 = 字典名[键] # 没有键则错误

# 　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"])

1. 遍历字典：

for 键名 in 字典名: #得到的是键

Print(键名)#得到的是键

Print(字典名[键名]) #得到的是该键对应的值

for 值名 in 字典名.values() #得到的是值

for 键名,值名 in 字典名.items(): #得到的是键和值

Print(键名)

Print(值名)

# 遍历（获取字典中所有元素）  
# 遍历字典，获取key  
**for** key **in** dict01: #获取键  
 print(key) #打印键  
 print(dict01[key]) #打印键对应的值  
# 遍历字典，获取value（遍历字典直接获取-->值）  
**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]) #单独获取键  
# print(item[1]) #单独获取值  
**for** k,v **in** dict01.items():  
 print(k) #单独获取键  
 print(v) #单独获取值

1. 删除元素：

del 字典名[键]

# 删除  
**del** dict01["a"]

### 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:exercise01.py  
 练习:exercise02.py  
"""*# 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)

# 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)

# 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, 102, 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。

## 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. 创建集合  
set01 = set()  
# set --> str #set直接转str会带着括号，所以一般先转list再转str  
set01 = set("abcac") #集合会自动去掉重复的  
list01 = list(set01)  
str01 = "".join(list01)  
print(str01) # "bca"

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)

# 创建具有默认值的集合  
set02 = {"a", "b", "a"}

1. 添加元素：

集合名.add(元素)

set02.add("qtx")

1. 删除元素：

集合名.discard(元素)

set02.remove("a")

1. 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 <= >=

### 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)

set01 = frozenset([1,2,3,3,5])  
list02 = list(set01)  
print(set01)  
print(list02)

#### 运算

等同于set

# 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）。

## 自定义函数

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。

## 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　（调试时按F8调试会跳过方法（上面的定义函数））  
# F7 逐语句 （调试时按F7调试会进入方法（上面的定义函数））  
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)  
  
print("--------------------下一个demo--------------")  
"""  
 函数返回值　应用  
 10:35  
"""  
# 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()  
#以后少用上面的这个方法，代码没写活  
**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))

函数体没有return，相当于返回None。

## 可变／不可变类型在传参时的区别

可参考.py文件：路径：第一阶段\02. PythonBase02\day04\demo03.py

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。

## 函数参数

### 实参传递方式argument

**def fun01**(a, b, c, d):  
 print(a)  
 print(b)  
 print(c)  
 print(d)

#### 位置传参

定义：实参与形参的位置依次对应。

位置实参：实参与形参根据位置进行依次对应.  
fun01(1, 2, 3, 4)

##### 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。

# 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) # a b c d 四行

#### 关键字传参

定义：实参根据形参的名字进行对应。

# 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) #4 1 3 2 四行

##### 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。

# 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。

# 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) #None 2 3 0 四行

#### 位置形参

语法：

def 函数名(形参名1, 形参名2, ...):

函数体

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

##### 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

# 作用：让实参个数无限  
**def fun03**(\*args):  
 print(args)  
  
  
# fun03()# ()  
# fun03(1)# (1,)  
# fun03(1,"2")# (1, '2')

#### 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

# 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)#1 2 如果写fun05（1，2）会报错

##### 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

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)

#### 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

# 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。

## 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