一、注释

单行注释：#

多行注释：””””””或者’’’’’’

二、数据类型

数据类型可以分数字型和非数字型

①、数字型

整型（int）

浮点型（float）

布尔型（bool）

真True【非0数】--不是零则为真

假false【0】

复数型

主要用于科学计算，例如：平面场问题、波动问题、电感电容等

②、非数字型

字符串

列表

元组

字典

三、查看变量类型

type函数可以查看一个变量的类型

四、不同类型之间的计算

数字型变量之间可以直接计算

如果变量是布尔型（bool），在计算时

True对应的数字为1

False对应的数字为0

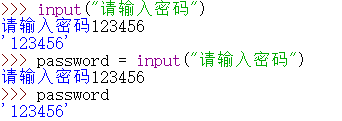
五、input函数的使用

input函数可以实现键盘输入

用户输入的任何内容都认为是一个字符串

input([prompt])

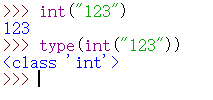
prompt：提示信息



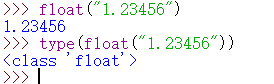
六、类型转换函数

|  |  |
| --- | --- |
| 函数 | 说明 |
| int(x) | 将x转换为一个整数 |
| float(x) | 将x转换到一个浮点数 |

①、int()函数



②、float()函数



七、变量的格式化输出

如果希望输出文字信息的同时，一起输出数据，就可以使用格式化操作符

%被称为格式化操作符，专门用于处理字符串中的格式

包含%的字符串，被称为格式化字符串

%和不同的字符连用，不同类型的数据需要使用不同的格式化字符

|  |  |
| --- | --- |
| 格式化字符 | 含义 |
| %s | 字符串 |
| %d | 有符号十进制整数，%06d表示输出的整数显示位数，不足的地方用0补全 |
| %f | 浮点数，%.02f表示小数点后只显示两位 |
| %% | 输出% |

八、变量的命名

①、标识符

标识符就是定义的变量名、函数名

·标识符可以由字母、下划线和数字组成

·不能以数字开头

·不能与关键字重名

②、关键字

关键字就是在Python内部已经使用的标识符

关键字具有特殊的功能和含义

不允许定义和关键字相同的名字的标识符

九、变量的命名规则

在Python中的标识符是区分大小写的

·在定义变量时，“=”的左右应该各保留一个空格

变量名是由两个或多个单词组成时，应该：

·每个单词都使用“小写”字母

·单词与单词之间使用“\_下划线”连接

驼峰命名

当变量名是由两个或多个单词组成时，可以使用驼峰命名法

小驼峰命名法

·第一个单词以小写字母开始，后续单词的首字母大写

·例如：firstName、lastName

大驼峰命名法

·每一个单词的首字母都采用大写字母

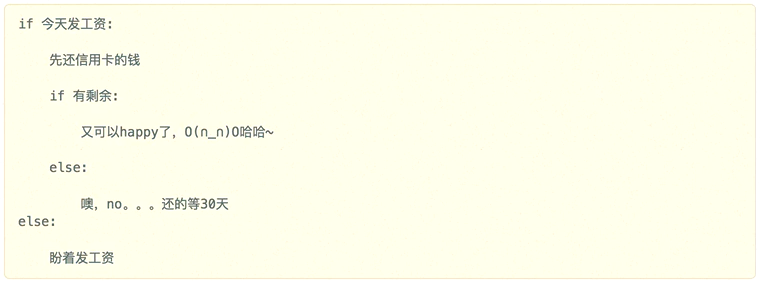
·例如：FirstName、LastName

十、判断语句（if语句）

判断定义

·如果条件满足，才能做某件事情

·如果条件不满足，就做另外一件十强，或者什么都不做



①、if语句就是用来进行判断的，格式如下：

if 要判断的条件:

条件成立时，要做的事情

……

注意：在Python开发中，Tab和空格不要混用

②、else处理条件不满足的情况，格式如下：

if 要判断的条件:

条件成立时，要做的事情

……

else:

条件不成立时，要做的事情

……

③、elif可以再增加一些条件，条件不同，需要执行的代码不同时，可以使用elif

格式如下：

if 条件1:

条件1满足执行的代码

……

elif 条件2:

条件2满足时，执行的代码

……

elif 条件3:

条件3满足时，执行的代码

……

else:

以上条件都不满足时，执行的代码

……

elif应用场景：同时判断多个条件，所有条件是平级的

④、if的嵌套

if的嵌套应该场景：在之前条件满足的前提下，在增加额外的判断

if的嵌套的语法格式：除了缩进之外和之前没有区别

if 条件1:

条件1满足执行的代码

……

if 条件1基础上的条件2:

条件2满足时，执行的代码

……

# 条件2不满足的处理

else:

条件2不满足时，执行的代码

# 条件1不满足的处理

else:

条件1不满足时，执行的代码

……

十一、运算符

①、比较运算符

|  |  |
| --- | --- |
| 运算符 | 描述 |
| == | 检查两个操作数的值是否相等，如果是，则条件成立返回True |
| != | 检查两个操作数的值是否不等于，如果是，则条件成立返回True |
| > | 检查左操作数的值是否大于右操作数的值，如果是，则返回True |
| < | 检查左操作数的值是否小于右操作数的值，如果是，则返回True |
| >= | 检查左操作数的值是否大于等于右操作数的值，如果是，则返回True |
| <= | 检查左操作数的值是否小于等于右操作数的值，如果是，则返回True |

②、逻辑运算符

Python中的逻辑运算符包括：

·与（and）

条件1 and 条件2

两个条件同时满足，返回True

只要有一个不满足，返回False

·或（or）

条件1 and 条件2

两个条件只要有一个满足，返回Ture

两个条件都不满足，返回False

·非（not）

not 条件

当条件成立时，返回False

当条件不成立时，返回True

③、赋值运算符

在Python中，使用“=”可以给变量赋值

注意：赋值运算符中间不能使用空格

|  |  |  |
| --- | --- | --- |
| 运算符 | 描述 | 实例 |
| = | 简单的赋值运算符 | c=a+b将a+b的运算结果赋值为c |
| += | 加法赋值运算符 | c+=a等效于c=c+a |
| -= | 减法赋值运算符 | c-=a等效于c=c-a |
| \*= | 乘法赋值运算符 | c\*=a等效于c=c\*a |
| /= | 除法赋值运算符 | c/=a等效于c=c/a |
| //= | 取整除赋值运算符 | c//=a等效于c=c//a |
| %= | 取模(余数)赋值运算符 | c%=a等效于c=c%a |
| \*\*= | 幂赋值运算符 | c\*\*=a等效于c=c\*\*a |

④、算术运算符

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 运算符 | 表达式 | 结果 | 描述 | 支持 |
| + | [1,2]+[3,4] | [1,2,3,4] | 合并 | 字符串、列表、元组 |
| \* | [“Hi”]\*4 | [“Hi”, “Hi”, “Hi” “Hi”] | 重复 | 字符串、列表、元组 |
| > >= == < <= | (1,2,3)<(2,2,3) | True | 元素比较 | 字符串、列表、元组 |

⑤、成员运算符

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 运算符 | 表达式 | 结果 | 描述 | 支持 |
| in | 3 in (1,2,3) | True | 元素是否存在 | 字符串、列表、元组、字典 |
| not in | 4 not in (1,2,3) | True | 元素是否不存在 | 字符串、列表、元组、字典 |

十二、随机数的处理

要使用随机数，首先需要导入随机数的模块

import random

导入模块后，可以直接在模块名称后面打个“.”后会提示该模块的函数

例如：

random.randint(12,20) # 生成的随机数在12到20之间

random.randint(20,20) # 结果永远是20

random.randint(20,10) # 该语句是错误的，下限必须小于上限

十三、循环

①、while循环基本使用

循环的作用就是让指定的代码重复执行

基本语法：

初始条件设置—通常是重复执行的 计算器

while 条件(判断 计算器 是否达到 目标次数):

条件满足时，做的事情1

条件满足时，做的事情2

条件满足时，做的事情3

……

处理条件（计算器+1）

注意：循环结束后，之前定义的计数器条件的数值是依旧存在的

②、for循环

可以遍历任何序列的项目

for 变量 in 序列:

…

③、完整的for循环

for 变量in 集合:

循环体代码

else:

没有通过break退出循环，循环结束后，会执行的代码

十四、break和continue

break和continue是专门在循环中使用的关键字

·break：某个条件满足时，退出循环，不在执行后续重复的代码

·continue：某个条件满足时，不执行后续重复的代码

注意：break和continue只针对当前所在循环有效

十五、循环嵌套

while嵌套就是：while里面还有while

while 条件1:

条件满足时，做的事情1

条件满足时，做的事情2

条件满足时，做的事情3

……

while 条件2:

条件满足时，做的事情1

条件满足时，做的事情2

条件满足时，做的事情3

……

处理条件2

处理条件1

十六、字符串中的转义字符

\t 在控制台输出一个制表符，协助在输出文本时垂直方向保持对齐

\n在控制台输出一个换行符

|  |  |
| --- | --- |
| 转义字符 | 描述 |
| \\ | 反斜杠符号 |
| \’ | 单引号 |
| \” | 双引号 |
| \n | 换行 |
| \t | 横向制表符 |
| \r | 回车 |

十七、函数介绍

函数，就是把具有独立功能的代码块组织为一个小模块，在需要的时候调用

函数的使用包含两个步骤

①、定义函数—封装独立的功能

②、调用函数—享受封装的成果

函数的作用：在开发时，使用函数可以提高编写的效率以及代码的重用

十八、函数的定义

定义函数的格式如下：

def 函数名():

函数封装的代码

……

def是英文define的缩写

函数名应该能够表达函数封装代码的功能，方便后续的调用

函数名的命名规则应该符合标识符的命名规则

·可以由字母、下划线和数字组成

·不能以数字开头

·不能与关键字重名

十九、函数调用

通过函数名()即可完成对函数的调用

二十、函数参数的使用

在函数名后面的小括号内部填写参数

多个参数之间使用，分隔

二十一、形参和实参

形参：定义函数时，小括号的参数，是用来接收参数，在函数内部作为变量

实参：调用函数时，小括号的参数，是用来把数据传递到函数内部用的

二十二、函数的返回值

返回值是函数完成工作后，最后给调用者的一个结果

在函数中使用return关键字可以返回结果

调用函数乙方，可以使用白能量来接收函数的返回结果

注意：return表示返回，后续的代码都不会被执行

二十三、函数的嵌套调用

一个函数里面又调用另一个函数，就是函数嵌套调用

如果函数test2中，调用了另一个函数test1

·那么执行到test1函数时，会先把函数test1执行完

·才会回到test2中调用函数test1的位置，继续执行后续的代码

二十四、模块概念

模块是Python程序架构的一个核心概念

模块就像工具包，要想使用这个工具包的工具，就需要导入(import)这个模块

每一个以扩展名py结尾的python源代码文件都是一个模块

在模块中定义的全局变量、函数都是模块能够提供给外界直接使用的工具

二十五、模块体验

可以在一个Python文件中定义变量或者函数

然后在另外一个文件中使用import导入这个模块

导入之后，就可以使用“模块名.变量”/“模块名.函数”的方式使用函数和变量

二十六、模块

模块名也是一个标识符

标识符可以由字母、下划线和数字组成

不能以数字开头

不能与关键字重名

注意：如果在给Python文件起名时，以数字开头是无法在Pycharm中导入模块

二十七、列表

list（列表）是Python中使用最频繁的数据类型，在其他语言叫数组

专门用于存储一串信息

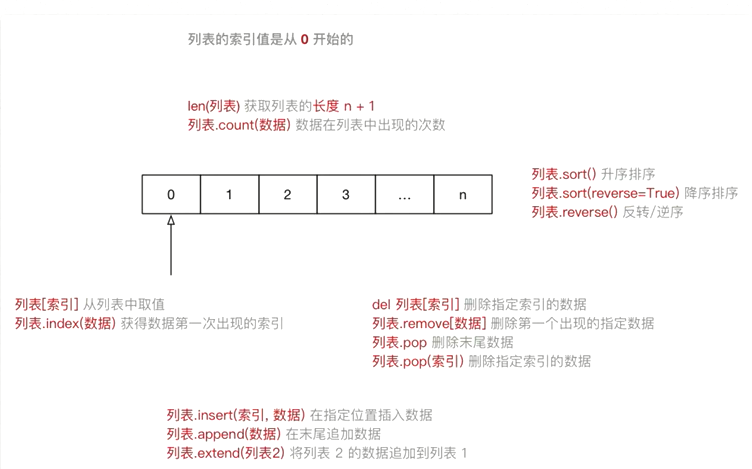
列表用[]定义，数据之间使用，分隔

列表的索引从0开始

索引就是数据在列表中的位置编号，索引也可以称为下标

注意：从列表中取值时，如果超出索引范围，程序会报错

name\_list = [“zhangsan”,”list”,”wangwu”]



查找：

.index 找出某个值的索引，当数据不在列表中，则会报错

添加：

.append 在列表末尾添加新的对象

.insert 将对象插入到指定的列表位置

.extend 将另一个列表的数据在末尾的插入进去

删除：

.remove 用于移除列表中的某个值

.pop 用于移除列表中的一个元素（默认是最后一个）

.clear 用于清空列表

del关键字 删除列表元素，本质上是用来将一个变量从内存中删除

统计：

len函数 可以统计列表中元素的总数

.count 用于统计某个元素在列表中出现的次数

排序：

.reverse 用于反转整个列表

.sort 用于对列表进行排序（默认为升序），可将reverse=True进行降序

复制：

.copy 用于复制列表元素

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **分类** | **关键字/函数/方法** | **说明** |
| 1 | 增加 | 列表.insert(索引,数据) | 在在指定位置插入数据 |
|  |  | 列表.append(数据) | 在末尾追加数据 |
|  |  | 列表.extend(列表2) | 将列表2的数据追加到列表 |
| 2 | 修改 | 列表[索引] = 数据 | 修改指定索引的数据 |
| 3 | 删除 | del列表[索引] | 删除指定索引的数据 |
|  |  | 列表.remove[数据] | 删除第一个出现的指定数据 |
|  |  | 列表.pop | 删除末尾数据 |
|  |  | 列表.pop(索引) | 删除指定索引数据 |
|  |  | 列表.clear | 清空列表 |
| 4 | 统计 | len(列表) | 列表长度 |
|  |  | 列表.count(数据) | 数据在列表中出现的次数 |
| 5 | 排序 | 列表.sort() | 升序排序 |
|  |  | 列表.sort(reverse=Ture) | 降序排序 |
|  |  | 列表.reverse() | 逆序、反转 |

二十八、关键字、函数、方法的区别

关键字是python内置的，具有特殊意义的标识符

关键字后面不需要使用括号

函数封装了独立功能，可以直接调用

函数名(参数)

方法和函类似，同样封装了独立多的功能

需要通过对象来调用，标识针对这个对象要做的操作

对象.方法名(参数)

二十九、元组

Tuple(元组)与列表类似

不同之处是在元组中的元素不能修改

元组表示多个元素组成的序列

元组在Python开发中，有特定的应用场景

用于存储一串信息，数据之间用“，”分隔

元组用()定义

元组的索引从0开始

索引就是数据在元组中的位置编号

tuple\_name = (“张三”,18,1.75)

创建空元组

tuple\_name = ()

常用操作：

.index 取值和取索引

.count 统计计数

三十、元组和列表之间的转换

使用list函数将元组转换成列表

list(元组)

使用tuple函数将列表转换成元组

tuple(列表)

三十一、字典

dictionary(字典)，同样可以用来存储多个数据

通常用于存储描述一个物体的相关信息

和列表的区别：

·列表是有序的对象集合

·字典是无序的对象集合

字典用“{}”定义

字典使用键值对存储数据，键值对之间使用“,”分隔

·键key是索引

·值value是数据

·键和值之间使用:分隔

·键必须是唯一的

·值可以取任何数据类型，但键只能使用字符串、数字或元组





常用操作：

获取长度：

len函数 统计键值对数量.

获取键值对：

.keys 以列表的形式返回字典所有的键

.values 以列表返回字典中的所有值

.items 以列表的形式返回遍历的键值对

获取值：

.get 返回指定键的值

删除元素：

clear 删除字典内所有元素

.pop 删除字典指定的键

.popitem 返回并删除字典中最后一对键值，如果字典为空，就报出KeyError异常

创建键值对

.setdefault返回指定键的值，如果键不存在，则会创建该键，并将值设为默认值或指定的值

.update 合并字典，如果合并的字典中有存在的键值对，则会覆盖原有的键值对

,copy 复制字典元素

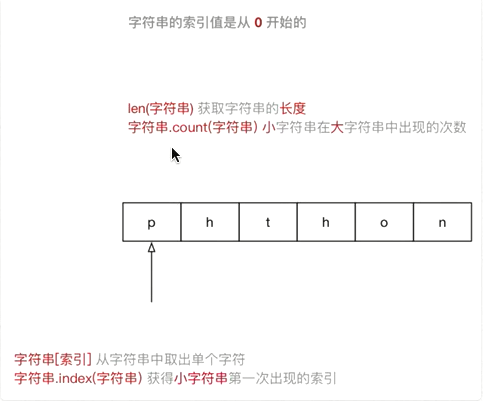
.fromkeys 用于创建一个新字典

三十二、字符串

字符串就是一串字符，表示文本的数据类型

可以使用双引号和单引号定义一个字符串

可以使用索引获取一个字符串中指定位置的字符，索引计数从0开始



常用操作：

查找：

len()：获取字符串的长度

.count：统计某一个字符串出现的次数

.index：某一个字符串出现的位置

判断类型：

|  |  |
| --- | --- |
| 方法 | 说明 |
| string.isspace | 如果string中只包含空格，则返回True |
| string.isalnum | 如果string至少有一个字符且所有字符都是字母和数字则返回True |
| string.isalpha | 如果string至少有一个字符且所有字符都是字母则返回True |
| string.isdecimal | 如果string只包含数字则返回True，全角数字 |
| string.isdigit | 如果string只包含数字则返回True，全角数字、(1)、\u00b2 |
| string.isnumeric | 如果string中只包含数字则返回True，全角数字，汉字数字 |
| string.istitle | 如果string是标题华的（每个单词的首字母大写）则返回True |
| string.islower | 如果string中包含至少一个区分大小写的中字符，并且所有这些（区分大小写的）字符都是小写，则返回True |
| string.isupper | 如果string中包含至少一个区分大小写的字符，并且所有这些（区分大小写的）字符都是大写，则返回True |

查找和替换：

|  |  |
| --- | --- |
| 方法 | 说明 |
| string.startswith(str) | 检查字符串是否以str开头，则返回True |
| string.endswith(str) | 检查字符串是否以str结束，则返回True |
| string.find(str,start=0,  end=len(string)) | 检测str是否包含在string中，如果start和end指定范围，则检查是否包含在支付ing返回内，如果是返回开始的索引值，否则返回-1 |
| string.rfind(str,start=0,  end=len(string)) | 类似于find()函数，不过是从右边开始查找 |
| string.index(str,start=0,  end=len(string)) | 跟find()方法类似，只不过如果str不在string会报错 |
| string.rindex(str,start=0,  end=len(string)) | 类似于index()，不过是从右边开始 |
| string.replace(old\_str,new\_str,  num=string.count(old)) | 把string中的old\_str替换成new\_str，如果num指定，则替换不超过num次 |

大小写转换：

|  |  |
| --- | --- |
| 方法 | 说明 |
| string.capitalize | 把字符串的第一个字符大写 |
| string.title | 把字符串的每个单词首字母大写 |
| string.lower | 转换string中所有大写字符为小写 |
| string.upper | 转换string中的小写字母为大写 |
| string.swapcase | 翻转string中的大小写 |

文本对齐：

|  |  |
| --- | --- |
| 方法 | 说明 |
| string.ljust(width) | 返回一个原字符串左对齐，并使用空格填充至长度width的新字符串 |
| string.rjust(width) | 返回一个原字符串右对齐，并使用空格填充至长度width的新字符串 |
| string.center(width) | 返回一个原字符串居中，并使用空格填充至长度width的新字符串 |

去除空白字符：

|  |  |
| --- | --- |
| 方法 | 说明 |
| string.lstrip | 截掉string左边（开始）的空白字符 |
| string.rstrip | 截掉string右边（末尾）的空白字符 |
| string.strip | 截掉string左右两边的空白字符 |

拆分和连接：

|  |  |
| --- | --- |
| 方法 | 说明 |
| string.partition(str) | 把字符串string分成一个3元素的元组(str前面,str,str后面) |
| string.rpartition(str) | 类似于parttition()函数，不过是从右边开始查找 |
| string.split(str=””,num) | 以str为分隔符切片string，如果num有指定值，则仅分隔num+1个子字符串，str默认包含’\r’,’\t’,’\n’和空格 |
| string.splitlines() | 按照行(‘\r’,’\n’,’\r\n’)分隔，返回一个包含各行作为元素的列表 |
| string.join(seq) | 以string作为分隔符，将seq中所有的元素（的字符串表示）合并为一个新的字符串 |

三十三、字符串的切片

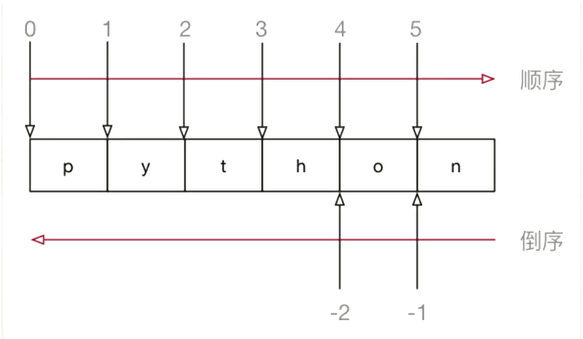
切片方法使用于字符串、列表、元组

·切片使用索引值来限定范围，从一个大的字符串中切除小的字符串

·列表和元祖都是有序的集合，都能够通过索引值获取到对应的数据

·字典是一个无序的集合，是使用键值对保存数据

字符串[开始索引:结束索引:步长]



|  |  |  |
| --- | --- | --- |
| 函数 | 描述 | 备注 |
| len(item) | 计算容器中元素个数 |  |
| del(item) | 删除变量 | del有两种方式 |
| max(item) | 返回容器中元素最大值 | 如果是字典，只针对key比较 |
| min(item) | 返回容器中元素最小值 | 如果是字典，只针对key比较 |