# 基础

## 下载&安装&配置

Python下载官网：<https://www.python.org/downloads/windows/>

下载好后安装，环境变量加入path。使用VS code进行编写Python代码。vscode需安装插件：Python

## 注释

* 单行注释：以#开头，如：# 世界，你好
* 多行注释：使用'''或"""包裹起来，如：

'''

世界，

你好

'''

## 数据类型

Python变量名包含数字，下划线，字母，并且区分大小写，无需定义变量类型。

|  |  |
| --- | --- |
| 数字number | 16进制加**0x**前缀，10进制照样写  科学计数法：表示为1.23e9【e配合数字用】  int 有符号整形  ~~long 长整型，加l/L后缀（python 2.x可用）~~  float 浮点型  complex 复数，a±bj或complex(a,b)表示 |
| 字符串string | 用单引号和双引号表示字符串，用\转义 |
| 布尔值bool | True、False |
| 列表List |  |
| 元组Tuple |  |
| 集合Set | 集合是不可重复的数据集 |
| 字典Dictionary |  |

|  |  |  |
| --- | --- | --- |
| 功能 | 方法/属性 | 说明 |
| 删除变量 | del 变量1~n | 删除变量，后可跟多个变量，用“，”分割。  python的内存回收机制支持在变量=None的前提下自动回收内存 |
| 查看类型 | type(变量名) | 返回变量类型  字符串（str），整型（int），浮点数（float），  布尔（bool），列表（list），元组（tuple），字典（dict）， |
| 验证 | Obj1 is Obj2 | 判断两个变量是否指向同一内存地址，也可以用于判断数据类型：type(num\_6) is int，返回布尔值 |
| 浅复制 | obj.copy() | 可用于列表，字典。（浅复制：不会对深部进行复制，会使用原来的指针） |
| 迭代器 | iter(obj) | 支持迭代：字符串，列表，元组，集合，字典；字典仅能迭代健值 |
| 全局声明 | global obj | 将函数或是循环或是判断中的变量声明是全局变量，前提是全局存在此变量。 |

## 数字

数字运算

python不支持x++ 和x--的语法；不同类型数字可以相互比较。

|  |  |  |
| --- | --- | --- |
| 功能 | 方法/属性 | 说明 |
| 除法取整 | x//y | 取被除后的整数部分 |
| 除法取余 | x%y | 取被除后的余数部分 |
| 次方 | x\*\*y | x的y次方 |

类型转换

|  |  |  |
| --- | --- | --- |
| 功能 | 方法/属性 | 说明 |
| x转int | int(x) | x类型：float，bool，string |
| x转float | float(x) | x类型：int，bool，string |
| x转复数 | complex(x) ,complex(x,y) | x为实数部分，y为虚数部分 |

数学函数（仅列举常用型）

|  |  |  |
| --- | --- | --- |
| 功能 | 方法/属性 | 说明 |
| 绝对值 | abs(x) | 返回x绝对值 |
| 绝对值 | math.fabs(x) | 返回x绝对值，math包支持 |
| 上入整数 | math.ceil(x) | 返回x的上入整数，math包支持 |
| 下入整数 | math.floor(x) | 返回x的下入整数，math包支持 |
| 最大值 | max(x1,x2,...,xn) | 返回x1~xn中最大值，参数可以是列表 |
| 最小值 | min()x1,x2,...,xn) | 返回x1~xn中最小值，参数可以是列表 |
| 四舍五入 | round(x,n) | x保留n位小数的四舍五入，n不写表示保留整数， |

## 运算符

注意：操作数中含float类型的变量，得数为float。

运算优先级

()

\*\*

% // / \*

+ -

< < >= <= == !=

and or not

=

Python支持多个变量同时赋值：a,b,c = 1,2,3

Python三元运算格式res = a if a > b else b

|  |  |  |
| --- | --- | --- |
| 功能 | 方法/属性 | 说明 |
| 加 | + |  |
| 减 | - |  |
| 乘 | \* |  |
| 除 | / |  |
| 取余 | % |  |
| 整除 | // | x//y表示x÷y=z……k取z |
| 乘方 | \*\* | X\*\*Y表示 |
| 判断 | == is | is是判断地址，==是判断值 |
| 或 | or |  |
| 与 | and |  |
| 非 | not |  |

## 字符串

字符串的位置，从左向右从0开始，从右向左从-1开始。

|  |  |  |
| --- | --- | --- |
| 功能 | 方法/属性 | 说明 |
| 格式化 | format() | "{0}今年{1}岁了".format(name,age)//{x}中指的是format中的参数下标  或者  "{name}今年{age}岁了" name和age是定义的变量 |
|  | '%5.3s morning'.% ('good') 5：代表占5位不足的就以空格填充，3代表截取填充字符串的位数如例子，截取goo；此方法支持%d %s；**先截取，后填充** |
| 转大写 | str.upper()  str.title() | 不改变原字符串，返回大写结果  title()的意思是，将空格后的字母转为大写，如i am good转为I Am Good |
| 转小写 | str.lower() | 不改变原字符串，返回小写结果 |
| 删首空白 | str.rstrip() | 不改变原字符串，返回结果 |
| 删尾空白 | str.lstrip() | 不改变原字符串，返回结果 |
| 获取字符 | str[index] | 获取下标=index的字符，不可使用此方法更改字符串中的值 |
| 重复拼接 | str \* n | 重复字符串n次 |
| 拼接 | str1+str2 | 拼接两个字符串，字符串只能与字符串拼接 |
| 截取 | str[start:end:step]  str[start:end] | 截取字符串  start开始位置，end结束位置  step步长：按步长将字符串截成多段，取每段第一位，步长为负代表从右取 |
| 长度 | len(str) | 字符串长度，返回int |
| 包含 | Str1 in Str2 | Str2中是否包含Str1字符串；返回Boolean；也可以用not in |
| 替换 | str.replace(str1,str2) | 将str中的str1替换为str2返回新字符串，并不改变原字符串 |
| 查找 | str.find(str1) | 查str中第一个匹配str1的位置，没有返回-1 |
| 计数 | str.count(str1) | 查找str1在str中出现的次数 |
| 分割 | str.split(",") | 按照某个字符分割字符串，返回list |

## 列表

python列表与数组类似；可以同时存储不同的数据类型

|  |  |  |
| --- | --- | --- |
| 功能 | 方法/属性 | 说明 |
| 获取元素 | list [index] | index可以是正数，代表以0开始从左向右取；也可以是负数，代表以1开始从右向左取第index位。 |
| 元素数量 | len(list) | 返回数组元素个数（整数型） |
| 截取列表 | list[start:end:step] | 从start位开始到end位截取数组（不包括end位），start为正从左计数，反之亦然，start和end不写代表从头开始/到尾部结束。step代表步长 |
| 批量替换 | list[start:end] = [……]  list[start:end:step] = [……] | 删除原列表中start~end位的数据用[……]中的数据替换，加入step时请注意[……]中元素个数必须与切片数一样 |
| 列表合并 | arr1+arr2+…+arrn | 合并多个列表，返回合并后的结果 |
| 重复列表 | list\*n | 将列表中的数据重复n次组成一个新列表并返回 |
| 拷贝列表 | list.copy() | 拷贝列表并返回，复制列表必须使用此方法，单纯地使用a=b的方式会有问题 |
| 颠倒列表 | list.reverse() | 颠倒列表，会改变原列表，不返回任何值 |
| 获取极值 | max(list)、min(list) | 获取列表中最大/最小的元素，返回最大/最小值 |
| 排序 | list.sort()/list.sort(reverse=True) | 列表升序排序。reverse=True代表降序排列。改变原列表且无返回值 |
| 长度 | len(list) | 列表长度 |
| 计数 | list.count(obj) | obj在list中出现的次数 |
| 转字符串 | str.join(list) | 将list转换为由str分割的字符串，如"-".join(arr) 就是1-2-3-4-5 |
| 添加 | list.append(obj)  list.insert(index,obj)  list.extend(list2) | list末尾添加ob  list指定下标位置插入obj  list末尾添加一个新列表arr2 |
| 删除 | list.pop(i)/list.pop()  del list[i]  list.remove(obj)  list.clear() | 删除list中下标i的元素，默认移除最后一位；返回被移除的元素  删除list中下标i的元素  删除list中指定的内容  清空list列表 |
| 元组  转列表 | list(tuple) | 元组转列表 |

## 元组

元组（Tuple）不可变化的有序容器；不能增删改元素。

定义：arr = (1,2,3,4,5) 或 arr = (1,)一个数也要打逗号，但这样的元组没有太多存在的必要

|  |  |  |
| --- | --- | --- |
| 功能 | 方法/属性 | 说明 |
| 获取 | tuple[i] | i为正获取从左开始的第i个元素，为负获取从又开始的第i个元素 |
| 截取 | tuple[start,end,step] | 从start开始截取到end位，返回新tuple，并不改变原tuple。step：步长 |
| 组合 | tuple1+tuple2 | Tuple无法更改，tuple合并后不能赋值给原来的tuple |
| 删除 | del tuple | 删除后此元组无法再次使用 |
| 验证 | x in tuple | Tuple中是否包含x |
| 重复 | tuple\*n | tuple重复n次 |
| 极值 | max(tuple)min(tuple) | 返回tuple中的极值 |
| 长度 | len(tuple) | 返回tuple长度 |
| 转化 | tuple(iterator) | 将可迭代的数据转换为元组 |

## 集合

元素不可重复的可变且无序容器

定义：set\_demo = {1,2,3,4,5,6} ，若创建空集合，使用set\_demo = set()

|  |  |  |
| --- | --- | --- |
| 功能 | 方法/属性 | 说明 |
| 增 | set.add(item)  set.update(set2) | 将item加入set中，若重复则不添加  将集合set2加入到集合set中，也可以传递字典，元组，列表，字典只会传入键 |
| 删 | set.pop()  set.remove(item)  set.clear() | 随机删除集合中一个元素  删除指定元素  清空集合 |
| 验证 | item in set | 判断set中是否包含item |
| 长度 | len(set) | 返回set长度 |
| 浅复制 | set.copy() |  |

## 字典

字典dictionary是一种存储key-value元素的可被改变的有序容器，支持层层嵌套，通常用来存储对象。字典中的键必须唯一且不能重复和不可变。如下定义：

d = {"北京":22,"南京":23,"上海":24}

|  |  |  |
| --- | --- | --- |
| 功能 | 方法/属性 | 说明 |
| 增 | dict[key] = value、dict.update({key:value}) |  |
| 删 | dict.pop(key)、d.pop(key,-1)  dict.popitem()#随机删除，返回被删元素 | 若key存在则删除后返回value值，若不存在返回-1， |
| 清空 | dict.clear() | 清空字典 |
| 查 | dict.get(key)、dict.get(key,9) | 若key存在则获取value值，若不存在返回9 |
| 改 | dict[key] = value  d.setdefault(key,val) | 若key存在，修改value，不存在则报错  若key存在，返回value且不做改变；若不存在则加入并返回val值 |
| 长度 | len(dict) | 获取字典长度 |
| 验证 | key in dict | 判断key值在dict中是否存在 |
| 转换 | str(dict) | 将字典转换成字符串类型 |
| 获取key | dict.keys() | 返回([……])格式的字典的全部keys |
| 获取value | dict.values() | 返回value，格式同上 |
| 获取item | dict.item() | 返回item，格式同上 |

### 字典的遍历

for item in dictionary循环遍历获取的item是key值，但也可以使用for key,value in dictionary.items()，获取元素，key就是key值，value就是value值。

d.key()：获取字典key值集合，返回列表

d.value()：获取字典value值集合，返回列表

## 判断/循环

|  |  |  |
| --- | --- | --- |
| if条件A:  …  elif 条件B:  …  else:  … | while 条件:  if条件A:  ……  continue  else: break  以及  while 条件:  ……  else:  …… # while执行后所执行的语句 | for i in range(10):  ……  for i in range(10):  ……  else:  …… |

## 类型转换

object只能是基础数据类型；int，string，float

|  |  |  |
| --- | --- | --- |
| 方法/属性 | 功能 | 说明 |
| int(x) | x转int |  |
| ~~long(x)~~ | ~~x转long~~ |  |
| float(x) | x转float |  |
| str(x) | x转string |  |

## 函数

def functionName(\*params):  
 # 函数主体

def是定义函数的关键字，若函数不确定参数数量，使用\*或\*\*。\*的情况下Params会被组成一个元组作为一个参数，\*\*的情况下params的写法有讲究，应该写成KeyValue形式的如fun(a=1, b=2)，最后params会被组成一个字典作为一个参数，处理数据前需从元组中拿取值，但如果参数是 (a,\*b,c)形式，那么调用时需这样写(val1,val2,val3,c = val4)。

方法注释这样写：

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

'''

用于打印abcd

:param a: 形参a

:param b: 形参b

:param c: 形参c

:param d: 形参d

:return:

'''

print(a, b, c, d)

值得注意的是：形参名和实参名最好定义得不一样；使用时，调用函数名即可

匿名函数简短写法：res = (lambda i: i \* 10)(10)，lambda是关键字，i是参数，(10)是实参。

## 类

class MyClass:

age = 0 #属性

def 方法名():

构造函数def \_\_init\_\_(self, num=0, num2=0):类中的构造函数写一个即可，多了报错。构造函数最好使用关键字法调用，如x = MyClass(num=1, num2=2)和x2 = MyClass(num=10)都行。其中self代表实体而非类。

**类的继承**

Python支持一个子类继承多个父类，重写父类方法，私有方法在方法名前加\_\_，如：def \_\_private(self)，公开方法则不用。私有属性变量名前加\_，如\_name。

class father:

i = 0

def \_\_init\_\_(self, num=0):

self.i = num

def f(self, str=''):

return 'hello world,' + str.upper()

class child(father,father2):

j = 0

def \_\_init\_\_(self, f\_num=0, c\_num=0):

father.\_\_init\_\_(self,num =f\_num) # 相当于super

self.j = c\_num

def f(self, str=''):

return 'hello,' + str.upper()

fa = father(num=1)

ch = child(f\_num=1, c\_num=2)

print(fa.i, fa.f('ok'))

print(ch.i, ch.j, ch.f('哈哈'))

**利用类装饰器优化代码**

class student:  
 def \_\_init\_\_(self, name='', age=0):  
 self.\_name = name  
 self.\_age = age  
  
 @property  
 def name(self):  
 return self.\_name  
  
 @name.setter  
 def name(self, name=''):  
 self.\_name = name  
  
 @property  
 def age(self):  
 return self.\_age  
  
 @age.setter  
 def age(self, age=0):  
 self.\_age = age

python有实例方法，类方法，静态方法装饰器。

|  |  |  |
| --- | --- | --- |
| 方法/属性 | 说明 | 功能 |
| @classmethod | 类方法 | 实体，类均可调 |
| @staticmethod | 静态方法 | 类和实例均可调 |
| 啥也不加 | 实体方法 | 对象可调，类不可调 |
| @?.setter | setter方法 | ?代表属性 |
| @property | 对象属性 | @property标识的“方法名”就是对象属性名；对象可调 |
| \_? | 私有属性 | 对象私有属性，类是否可调，视定义的位置情况而定。定义在外可调，定义为@property修饰下的不可调用，如上述例子 |

## 模块调用

模块调用，格式为from 包名 import 变量/方法。如果导入全部直接写 import 包名。另外可以使用as给导入的元素命名别名

# 转大写

def **to\_upper\_case**(string):

if type(string) is str:

string = string.upper()

return string

# 转小写

def **to\_lower\_case**(string):

if type(string) is str:

string = string.lower()

return string

**from stringTools import to\_lower\_case, to\_upper\_case**

string = 'aBcDeFgHiJkLmN'

string = **to\_upper\_case**(string)

print('打印结果\n'+string)

string = **to\_lower\_case**(string)

print('打印结果\n'+string)