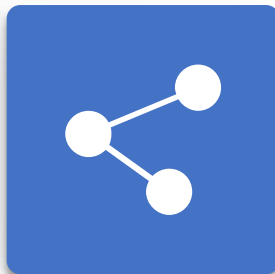


# Python

周嵩林

基本语法、函数





1

# 代码格式



1. 以缩进控制代码结构，无需大括号

2. 每句一行，行末无需分号

#写了也不报错

3. 跨行语句行末用反斜杠 \ 说明

#某些情况不需要

4. 空语句使用 `pass` 占一行

#否则会报错

5. 单行注释以 `#` 开头，多行注释两端三个 `"""` 或 `'''`

6. 大小写敏感，注意命名规范



# 2

## 数据与运算符



## 基本数据类型：

数

1

1.23

-5

字符串

'abc'

"A\nbc"

直接赋值给变量，无需声明类型

a = 1

b = "abc"



## 整数 (int)

-12345

#有符号

123631215621123321156521271321592

#不会溢出

0xacfd4

#16进制

0o14365

#8进制

0b1011010

#2进制



## 浮点数 (float)

1.234

1.234e18

#科学计数

4.2 + 2.1 = 6.3000000000000001

#精度有限

## 复数 (complex)

1+2j

complex (1,2)

# 1+2j

complex (1)

# 1+0j



## 布尔 (bool)

True

False

以下值进行bool转换时为False:

0, 0.0, 0j

# 值为0

"" , [], {}, (), set()

# 内容为空

其他值均为True

# bool( 'False' ) = True





## 算术运算符

$3*2+5*4$

# 26 优先级符合常识

$5**3$

#125 \*\* 幂运算

$9\%4$

#1 % 取模

$9/4$

#2.25 **python3**默认非整除, **python2**为整除

$9//4$

#2 // 整除



## 比较运算符

`==` `>` `>=` `!=`

# 与c相同，老版本有 `<>`不等于

`2 == 1.0 + 1.0`

# True 整数与浮点可比较

`6.3 == 4.2 + 2.1`

# False 注意浮点型误差

`abs(4.2 + 2.1 - 6.3) < 1e-10`

# True



## 逻辑运算符

and or not

# 注意不要用&&, ||, !

a or b 等价于

if a :

return a

else :

return b

a and b 等价于

if not a :

return a

else :

return b

例如:

2 or 3 = 2

2 and 3 = 3

3 and 0 = 0

# 不是True或False



## 位运算符

& | ^ ~ << >>

# 位运算需要数据类型相同

1.0 & 1.0

# TypeError 只能对整数进行

## 赋值运算符

+= /= \*\*= %=

# 没有++, --

## 条件运算符

b if a else c

# a ? b : c



## 成员运算符

in not in

1 in [1,2,3] 1.0 in [1,2,3]

# True True 涉及后续内容, 列表

## 身份运算符

is is not

1 == 1.0 1 is 1.0

# True False

1 is 1 [1,2,3] is [1,2,3]

# True False 2个相等的列表并不相同



高

低

运算符	描述
**	幂运算
+X, -X, ~X	正, 负, 按位取反
*, /, //, %, @	乘除取模, 装饰器
+, -	加, 减
<<, >>	左右移位
&	按位与
^	按位异或
	按位或
in, not in, is, is not	成员、身份运算
not x	非
and	与
or	或
if-else	条件运算符
lambda	lambda运算符

不本质

无需记忆

使用 ( ) 控制



## 字符串 (str)

```
'ABC' = "ABC"
```

# 单双引号等价

```
'''Hello  
World'''
```

# 三个单引号的字符串可换行，可用于多行注释



```
'AB' + 'c'
```

```
# 拼接 'ABc'
```

```
'AB'*3
```

```
# 复制 'ABABAB'
```

```
len('AB')
```

```
# 长度 2
```

```
'l' in 'Hello, world'
```

```
# 判断子串 True
```

```
'Hello, world'.count('l')
```

```
# 计数 3
```

```
'Hello, world'.find('lo')
```

```
# 查找下标 3
```

```
'Hello, world'.replace('llo','he')
```

```
# 替换 'Hehe,world'
```

使用 `help(str)` 运行可查看所有str类型内建函数





```
a = 'H e l l o , _ w o r l d '  
      0  1  2  3  4  5  6  7  8  9 10 11  
    -12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1
```

a[1]        # 'e'

a[-2]        # 'l'

a[3:8]        # 'lo, w'

a[-5:]        # 'world'

a[:4]        # 'Hell'

a[3:-3]        # 'lo, wo'



%d

整数

%f

浮点数

%s

字符串

%x

十六进制

```
'The answer is %d' %2
```

```
# 'The answer is 2'
```

```
'%d,%d' %(1,2)
```

```
# 使用元组, 不可用列表
```

```
'%(ABC)d, %(ghj)d' %{'ABC':1,'ghj':2} # 使用字典
```

```
print('AB\nCD')
```

```
# AB
```

```
CD
```

```
print(r'AB\nCD')
```

```
# AB\nCD 使用r取消转义功能
```



3

# 内置容器



## 列表 (list)

```
a=[]    b=[1,2]    c=['bc',3.5]
```

# 列表元素可为任何类型 (c++, vector)

可进行类似字符串类型的操作

```
d = b+c
```

# 拼接 [1,2,'bc',3.5]

```
len(d)
```

# 长度 4

```
d[2] = 3
```

# 访问并修改列表元素 d = [1,2,3,3.5]

```
d[2:4]
```

# 截取 [2,'bc']



```
a = [1,2,4,8]
```

```
a.append(0)           # 末尾追加元素 a = [1,2,4,8,0]
```

```
a.insert(3,9)         # 插入元素 a = [1,2,4,9,8,0]
```

```
del a[:2]             # 按下标删除 a = [4,9,8,0]
```

```
a.remove(8)           # 删除元素 a = [4,9,0]
```

```
a.index(4)            # 获取下标 2
```

```
a.clear()             # 清空 a = []
```

同样可使用 `help(list)` 获取所有列表内建函数



## 元组 (tuple)

`()`    `(1, 2)`    `('bc', 3.5, 5)`    # 元组与列表类似

`a = (1, 2)`

`a[1] = 3`

# Error 元组一旦定义不可修改

`a = (1)`

# 1 元组定义与小括号发生歧义

`a = (1,)`

# (1,) 单元素元组定义方式

元组有列表所有不修改元素的内置函数



## 字典 (dict)

```
{ }           { 'a':1, 'b':2 }
```

# 键值对应 (c++, map)

```
{ 'a':1, 2:[1,2,3], None: 'NULL' }
```

# 键与值都可以为任何类型, 包括None

```
{ 'a':1, 'a':[1,2,3] }
```

# Error 键不可重复

字典所有内建函数同样推荐使用 `help(dict)` 自主学习

字典使用hash实现, 所有键值对无序 (hash函数未知), 查找插入耗时短, 空间开销大



```
D = {'a':1,3:'cde'}
```

```
D['a'] = 2
```

# 使用键访问并修改值

```
D['b'] = 4
```

# 赋值语句未找到键则添加键值对

```
del D['c']
```

# 删除键值对

```
D.keys()
```

# 返回所有键，是dict\_keys类的对象  
dict\_keys(['a', 3])

```
D.values()
```

# 返回所有值，是dict\_values类的对象  
dict\_values(['a', 3])

```
D.items()
```

# 返回所有键值对，是dict\_items类的对象  
dict\_items([('a', 1), (3, 'cde')])





## 集合 (set)

```
a = {1, 2}    b = {2, 'c' }
```

# 没有 '值' 只有 '键' 的字典

```
{ }    set()
```

# 前者为空字典, 后者为空集合

```
c = a.add(3)
```

# 添加元素 c = { 1, 2, 3 }

```
c = a.union(b)
```

# 取并集 c = { 1, 2, 'c' }



```
a = [ 'x' , 'y' ]   b = [1,2]
```

```
list:      [ 2**x for x in b ]           # 从1个容器生成另一个
```

```
          [ x for x in range(100) ]      # range为常用迭代器, 从0到99迭代
```

```
dict:      { x:y for x in a for y in b }  # 错误, 会生成{ 'x' : 2, 'y' : 2}, 所有值相同
```

```
          dict( zip(a,b) )               # 成功生成{ 'x' : 1, 'y' : 2}, zip函数自学
```

set, tuple与list相似



# 4

## 控制语句



```
if <条件1>:
```

```
...
```

```
elif <条件2>:
```

```
...
```

```
else:
```

```
...
```

例:

```
if a > 0:
```

```
    a = 1/a
```

```
elif a < 0:
```

```
    a = -1/a
```

```
else:
```

```
    a = None
```

**for <变量> in <迭代器>:**

...

例: `for i in [1,2,3]:`

`res += i`

`for i in range(100):`

`res += i`

`for i,j in dict.items():`

`res1 += i`

`res2 += j`

**while <判断语句>:**

...

`i = 1`

`while(i < 100):`

`res += i`

`i += 1`



5

# 输入输出



`print()` # 输出括号中的对象，自动换行，用 ‘,’ 分隔

`input()` # 读入一行，不包括换行符，返回str

`print( 1, 2, 3)` # 1 2 3 (换行)

`print( “我不想换行怎么办? ” , end=’ ’ )` # 令end为空字符，不换行

`print( “end还可以用别的” , end=’ \t’ )` # 自动输出制表符



6

# 作业1





1. 实现功能：输入任意长字符串，将其按照空格切分，输出不同子串及其出现次数  
思考时间复杂度低于 $O(n^2)$ （少于两重循环）的方法  
提示：help(str.split)

2. 选做：

```
a = [ 'alice' , 'bob' , 'caren' , 'daniel' ]  
print(_____)
```

补全该行代码

使输出为： [ 'Alice' , 'Bob' , 'Caren' , 'Daniel' ]

说明：本题有一定难度，使用map函数可以实现，但仍需上网搜索其他相关知识点。  
目的在于培养同学们自主查找资料的能力，实在难以实现，可以退而求其次，  
添加尽量少的其他代码完成要求。

提示：使用help学习str.join chr ord enumerate等函数

必做小学期结束前，选做暑假结束前

提交 姓名-x-y.py 文件（或打包所有作业一起提交）到 [zhousl16@mails.tsinghua.edu.cn](mailto:zhousl16@mails.tsinghua.edu.cn)