# 十六进制数字的输出

#int(x,base),其中base表示进制，可以是2，8，10，16等 ，int(x,base)函数的作用是将小数x按照base进制取整，即去除小数后的数字，只保留整数部分

## int函数的描述

int（）函数用于将一个字符串或数字转换为整型。

int（）的语法：

[复制代码](https://www.nowcoder.com/practice/349fec692757406e92e9a5ebd9f97580?tpId=314&tqId=2365044&ru=/exam/company&qru=/ta/python/question-ranking&sourceUrl=/exam/company)

|  |  |
| --- | --- |
| 1 | **class** **int**(x, base=10) |

参数：

x--字符串或数字

base--进制数，默认十进制

返回值：

返回整型值

## 十六进制转十进制

例题：

描述：计算的世界，除了二进制与十进制，使用最多的就是十六进制了，现在使用input读入一个十六进制的数字，输出它的十进制数字是多少？

输入描述：input读入一个十六进制数字，按照字符串的形式包括数字0-9、字母A-F。

输出描述：输出读入数字的十进制大小。

题解：

num = input()

print(int(num,16))

注意：int(num,16)只能用于将十六进制数字转换成十进制数组

其中num表示一个十六进制的数字或字符串。要想实现十进制转换成十六进制，见下面。

## 十进制转十六进制

使用hex函数才能将十进制转换成十六进制

例题：

牛妹刚学习进制转换，对这方面掌控还不太熟练，她想请你帮她写一个十进制到十六进制的进制转换器，你能使用hex函数帮助她完成这个任务吗？

输入描述：

输入一个正整数。

输出描述：

输出该正整数的十六进制。

题解：

num = int(input())

print(hex(num))

与int（num，16）不同的是，hex（num）中的num必须为整型

# 格式化输出之大小写和首字母大写

python中针对大小写和首字母大写提供了专门的方法

## 方法描述

upper() # 把所有字符中的小写字母转换成大写字母

lower() # 把所有字符中的大写字母转换成小写字母

capitalize() # 把第一个字母转化为大写字母，其余小写

title() # 把每个单词的第一个字母转化为大写，其余小写

语法格式，比如：name1.title() # 把变量name1值中每个单词的第一个字母转化为大写，其余小写

## 例题：

描述：牛牛、牛妹和牛可乐都是Nowcoder的用户，某天Nowcoder的管理员希望将他们的用户名以某种格式进行显示，现在给定他们三个当中的某一个名字name，请分别按全小写、全大写和首字母大写的方式对name进行格式化输出（注：每种格式独占一行）。

输入描述：一行一个字符串表示名字。

输出描述：请分别按全小写、全大写和首字母大写的方式对name进行格式化输出（注：每种格式独占一行）。

示例：

输入 niuNiu

输出 niuniu

NIUNIU

Niuniu

题解：

name = input()

print(name.lower())

print(name.upper())

print(name.title())

# 格式化输出之删除语法

## 删除语法描述：

1. lstrip() : 删除字符串左边的空格和指定字符串；

2. rstrip() : 删除字符串末尾的空格；

3. strip([chars]) : 在字符串上执行 lstrip() 和 rstrip()；

4. replace(old， new, max) : 把字符串中的 old 替换成 new, 如果 max 指定, 则替换不超过 max 次。

## 例题：

描述：牛牛、牛妹和牛可乐都是Nowcoder的用户，某天Nowcoder的管理员由于某种错误的操作导致他们的用户名的左右两边增加了一些多余的空白符（如空格或'\t'等），现在给定他们三个当中的某一个名字name，请输出name去掉两边的空白符后的原本的内容。

输入描述：一行一个字符串表示名字name（注：name两边带有一些多余的空白符）。

输出描述：一行输出name去掉两边的空白符后的原本的内容。

示例：

输入： Niu Ke Le

输出：Niu Ke Le

题解：

name = input()

print(name.strip())

# split方法--针对字符串的方法

## split函数简介

Python中split()函数，具体作用如下：

拆分字符串。通过指定分隔符对字符串进行切片，并返回分割后的字符串列表（list）；

## 语法

split() 方法语法：

str.split(str="",num=string.count(str))[n]

参数说明：

str:表示为分隔符，默认为空格，但是不能为空(’’)。若字符串中没有分隔符，则把整个字符串作为列表的一个元素

num:表示分割次数。如果存在参数num，则仅分隔成 num+1 个子字符串，并且每一个子字符串可以赋给新的变量。默认为 -1, 即分隔所有。

[n]:表示选取第n个分片

注意：当使用空格作为分隔符时，对于中间为空的项会自动忽略

## 分离字符串

1、以’ . '为分隔符

string = “www.gziscas.com.cn”

print(string.split('.'))

输出：

['www', 'gziscas', 'com', 'cn']

2、分割两次

print(string.split('.',2))

输出：

['www', 'gziscas', 'com.cn']

3、分割两次，并取序列为1的项

print(string.split('.',2)[1])

输出：

gziscas

4、分割两次，并把分割后的3个部分保存到3个变量

u1, u2, u3 =string.split('.',2)

print(u1) #输出：www

print(u2) #输出：gziscas

print(u3) #输出：com.cn

输出：

www

gziscas

com.cn

四、实例

str="hello boy<[www.baidu.com]>byebye"

print(str.split("[")[1].split("]")[0])

输出：

www.baidu.com

分析：

先执行 str.split(“[”)[1]，结果为：www.baidu.com]>byebye

再执行：“www.baidu.com]>byebye”.split(“]”)[0]，结果为：“www.baidu.com”

最后print输出结果：www.baidu.com

# insert方法--针对列表的方法

## 描述

insert()函数用于将指定对象插入列表的指定位置。

## 语法

inser()方法语法：

list.insert(index, obj)

参数：

index -- 对象obj需要插入的索引位置。

obj -- 要插入列表中的对象。

返回值：

该方法没有返回值，但会在列表指定位置插入对象。

## 实例

以下实例展示了insert()函数的使用方法：

aList = [123, 'xyz', 'zara', 'abc']

aList.insert(3, 2009)

print "Final List : ", aList

以上实例输出结果如下：

Final List : [123, 'xyz', 'zara', 2009, 'abc']

# sort方法和sorted函数

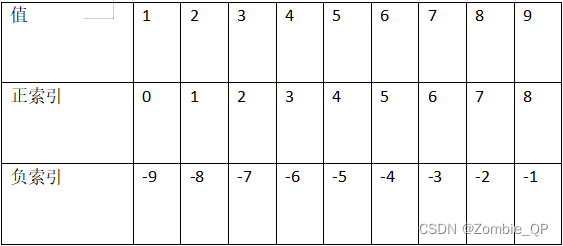
具体可见--https://zhuanlan.zhihu.com/p/299436819（非常详细）

# 切片

在Python中，切片(slice)是对序列型对象(如list, string, tuple)的一种高级索引方法。 普通索引只取出序列中 一个下标 对应的元素，而切片取出序列中 一个范围 对应的元素，这里的范围不是狭义上的连续片段。通俗一点就是在一定范围里面.用刀切出一部分,达到自己需要的一部分.

## 切片的索引方式

以 a = [ 1,2,3,4,5,6,7,8,9] 为例,正索引和负索引



## 索引的一般方式

一个完整的切片是包含三个参数和两个冒号" : " ,用于分隔三个参数(start\_index、end\_index、step)。当只有一个“:”时，默认第三个参数step=1；当一个“:”也没有时，start\_index=end\_index，表示切取start\_index指定的那个元素。

切片操作的基本表达式:object[start:end:step]

start:切片的起始位置,如果没有值的话从头开始.

end:切片的结束位置,但不包含end(前闭后开),如果没有值的话表示切割到结束为止.

step:步长,默认取值为1,如果步长为正数的情况表示从左往右,反正若为负数则表示从右往左. step的正负决定切的方向,这一点需要尤为注意!!!

## 切片操作的详细切法

1. 切割单个值

>> a = [1,2,3,4,5,6]

>>> a[0] ##单个数,代表位数,第0位就是a里面的第一位

1

>>> a[5]##a里面的第5位,注意要从0作为第一位开始数

6

1. 切割完整对象

>>> b=[6,3,6,7,8,2,5,4]

>>> b[:] ##单独一个冒号,代表从头取到尾,步长默认为1

[6, 3, 6, 7, 8, 2, 5, 4]

>>> b[::]##单独两个冒号一样代表从头取到尾,步长默认为1

[6, 3, 6, 7, 8, 2, 5, 4]

>>> b[::-1]##注意,两个冒号后面是步长,步长为1,故应从右往左取

[4, 5, 2, 8, 7, 6, 3, 6]

1. 当start和end全部为正数时

>>> a

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> a[1:6] ##默认步长为1,从左往右,注意前闭后开

[1, 2, 3, 4, 5]

>>> a[1:6:-1]

[] ## 当取值的大小方向与步长的方向冲突时,返回值是空.

>>> a[6:1:-1]

[6, 5, 4, 3, 2]

>>> a[:6] ## 没有start代表从头开始取

[0, 1, 2, 3, 4, 5]

>>> a[:6:-1]

[9, 8, 7]

>>> a[6:]

[6, 7, 8, 9]

>>> a[6::-1]

[6, 5, 4, 3, 2, 1, 0]

1. 当start和end全部取负数的时

>>> a

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> a[:-6] ## -6意思是从右往左数的第六位,故第六位是4,然后默认步长为1(从右往左),star未写,故从头开始取到4

[0, 1, 2, 3]

>>> a[-1:-6]

[]

>>> a[-1:-6:-1]

[9, 8, 7, 6, 5]

>>> a[-6:-1] ## 这个是从-6取到-1,步长为1,意思是从右往左数第6位到从右往左的第一位

[4, 5, 6, 7, 8]

>>> a[:-6:-1] ## 这个是从0取到-6,步长为-1,因为开头是冒号,故起点被隐藏了

[9, 8, 7, 6, 5]

>>> a[-6:]

[4, 5, 6, 7, 8, 9]

>>> a[-6::-1] ## 注意这个不等于[-6:-1],区别是这里是::(两个冒号),两个冒号后连接的是步长

[4, 3, 2, 1, 0]

1. start和end是正负混合情况

>>> a

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> a[1:-6]

[1, 2, 3]

>>> a[1:-6:-1]

[]

>>> a[-1:6]

[]

>>> a[-1:6:-1]

[9, 8, 7]

1. 连续的切片操作

>>> a

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> a[:8][2:5][-1:] ## [:8]就是0取到8,在从其中取2到5,最后取-1位

[4]

a[:8] ---- [0,1,2,3,4,5,6,7]

[0,1,2,3,4,5,6,7][2:5]----[2,3,4]

[2,3,4][-1:] ----[4]

1. 切片中的三个参数为表达式

>>> a

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> a[1+2:2\*3:7%2] ## 思路一样,运算出来,继续切

[3, 4, 5]

1. 切片可以操作其他的对象

>>> t = (1,2,3,4,5)

>>> t[1:3]

(2, 3)

>>> s = "ACDRF" ##切片在字母上也是可以使用的,所以说切片很强大

>>> s[1:3]

'CD'

>>> (0, 1, 2, 3, 4, 5)[:3]#元组的切片操作

>>> (0, 1, 2)

>>> for i in range(0,100):

... print(i)

...

>>> for i in range(0,100)[2::3][-10:]: ## 意思是从第二位开始取,步长为3,[-10]则是从倒数10位开始取,意思是去末尾10位.

... print(i)

# Python内置函数/方法详解—元组tuple

元组是有序且不可更改的集合。在Python中，元组使用圆括号 （）编写的。

## 1、创建元组

元组的创建很简单，使用圆括号（）直接创建或者使用 tuple（）函数创建，只需要在圆括号中添加元素，并使用逗号隔开即可。

### 1.1 使用 () 创建元组

通过 () 创建元组后，使用 = 将它赋值给变量，格式如下：

tuple\_name = (element\_1, element\_2, ..., element\_n)

其中，tuple\_name 表示变量名，element\_1 ~ element\_n 表示元组内的元素。

### 1.2 使用 tuple()函数 创建元组

我们除了可以使用 () 创建元组，还可以使用 tuple() 函数创建元组，但 tuple() 偏向于将某个类型转换为元组，具体用法见下述：

cre\_tuple = tuple(('a', 'b', 'c')) # 注意双括号

print(cre\_tuple)

print(type(cre\_tuple))

('a', 'b', 'c')

<class 'tuple'>

### 1.3 元组(单个元素)

当元组中只包含一个元素时，需要在元素后面添加逗号, ，否则括号会被当作运算符使用：

tuple\_1 = (20)

print(type(tuple\_1)) # <class 'int'>

tuple\_2 = (20,)

print(type(tuple\_2)) # <class 'tuple'>

当元组中只含一个元素未添加逗号时，使用 type() 打印其类型时，会输出 <class 'int'>。

当元组中只含一个元素并添加逗号时，使用type() 打印其类型时，会输出 <class 'tuple'>。

### 1.4 元组 VS 列表

在Python中，元组与列表相似，不同之处在于元组的元素不能修改，而列表的元素可以修改。

元组使用 小括号()，列表使用 中括号[]。

## 2、访问元组

和列表一样，我们既可以使用下标索引访问元组中的某个元素（得到一个元素的值），也可以使用切片访问元组中的一组元素（得到是子元组）。

### 2.1 下标索引访问

下标索引访问元组分为两大类，即正向索引和反向索引，格式为 tuple\_name[i] ，其中，tuple\_name 表示元组名，i表示索引值，i可以是正数（正向索引）也可以是负数（反向索引）。

可以得知，tuple\_name[0]表示元组内第一个元素，tuple\_name[-1]则表示元组内最后一个元素。

tuple\_name = ('wzq', 'lgl', 'gz', 'whl', 'sj', 'hxw')

print(tuple\_name[0])

print(tuple\_name[-1])

wzq

hxw

正向索引：从第一个（下标0）开始、第二个（下标1）…

反向索引：从倒数第一个（下标-1）、倒数第二个（下标-2）…

### 2.2 切片访问

如若对上方描述不太理解，可参考下表：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 元组值 | wzq | lgl | gz | whl | sj | hxw |
| 正向索引 | 0 | 1 | 2 | 3 | 4 | 5 |
| 反向索引 | -6 | -5 | -4 | -3 | -2 | -1 |

使用切片访问元组的格式为 tuple\_name[strat : end : step] ，其中，start 表示起始索引，end 表示结束索引，step 表示步长。

tuple\_name = ('wzq', 'lgl', 'gz', 'whl', 'sj', 'hxw')

print(tuple\_name[1:5:2])

print(tuple\_name[-6:-1:3])

('lgl', 'whl')

('wzq', 'whl')

在使用切片访问元组元素时，tuple\_name[strat : end : step]，[start：end] 是左闭右开区间，即访问不了 end 代表的元素。

### 2.3 for 循环遍历元组

可以使用 for 循环遍历元组中的项目：

fruit\_tuple = ('apple', 'pear', 'cherry')

for i in fruit\_tuple:

print(i)

apple

pear

cherry

### 2.4 检查项目是否存在

要确定元组中是否存在指定的项，我们可以使用 in 关键字：

# 检查元组中是否存在'apple'

fruit\_tuple = ('apple', 'pear', 'cherry')

print('apple' in fruit\_tuple)

True

使用 in 关键字检查元组中是否存在指定项时，如果存在，则返回 True ；反之，则返回 False 。

### 2.5 更改元组值

创建元组后，我们将无法更改其值。元组是不可变的，或者也被称为恒定的。

但是有一种解决方法：可以先将元组转换为列表，更改列表值，然后再将其转换回元组。

fruit\_tuple = ('apple', 'pear', 'cherry')

fruit\_list = list(fruit\_tuple)

fruit\_list[2] = 'banana'

fruit\_tuple = tuple(fruit\_list)

print(fruit\_tuple)

('apple', 'pear', 'banana')

### 2.6 元组连接(合并)/复制

与字符串一样，元组之间可以使用 + 号和 \* 号实现元组的连接和复制，这就意味着它们可以生成一个新的元组。

#### 1、+连接

x = (1, 2, 3)

y = (4, 5, 6)

print(x + y)

(1, 2, 3, 4, 5, 6)

#### 2、\*复制

x = ('Hello',)

print(x \* 5)

('Hello', 'Hello', 'Hello', 'Hello', 'Hello')

### 2.7 元组不可变性

所谓元组的不可变性指的是元组所指向的内存中的内容不可变。

tup = ('A', 'B', 'C')

# tup[0] = 'a'

不支持修改元素，会报错 TypeError: 'tuple' object does not support item assignment 。

tup = ('A', 'B', 'C')

print(id(tup))

# 查看内存地址

tup = (1, 2, 3)

print(id(tup))

2671405067264

2671405072256

从以上实例可以看出，重新赋值的元组 tup ，绑定到了新的对象了，不是修改了原来的对象。

## 3、内置函数

### 3.1 打印输出 print()

#### print()函数

print() 函数的功能我们已经非常熟悉了，就是打印输出。

my\_tuple = ('name', True, 21)

print(my\_tuple)

('name', True, 21)

### 3.2 确定元组项目 len()

#### len()函数

当我们要确定一个元组有多少项目（元素）时，可以使用len()函数。

fruit\_tuple = ('apple', 'banana', 'cherry')

print(len(fruit\_tuple))

3

### 3.3 返回变量类型 type()

#### type()函数

使用 type() 函数可以确定变量是什么类型（字符串、列表、元组、字典或集合）。

当对tup\_1 = (20)使用 type() 确定变量类型时，会返回<class 'int'>，表明这是一个整型。所以当一个元组只含有一个元素时，一定要加上逗号,。

当对tup\_1 = (20,)使用 type() 确定变量类型时，会返回<class 'tuple'>，表明这是一个元组。

### 3.4 转换为元组 tuple()

#### tuple()函数

tuple() 函数的功能是，将其他类型转换为元组类型，详细用法如下：

将字符串转换为元组

str1 = 'Hello Python'

print(tuple(str1))

('H', 'e', 'l', 'l', 'o', ' ', 'P', 'y', 't', 'h', 'o', 'n')

将列表转换为元组

list1 = ['Hello', 'Python']

print(tuple(list1))

('Hello', 'Python')

将字典转换为元组

dict1 = {'Hello': 'Python', 'name': 'pink'}

print(tuple(dict1))

('Hello', 'name')

将集合转换为元组

set1 = {'Hello', 'Python', 'name', 'pink'}

print(tuple(set1))

('Hello', 'name', 'Python', 'pink')

将区间转换为元组

range1 = range(1, 6)

print(tuple(range1))

(1, 2, 3, 4, 5)

### 3.5 元组元素最大/小值 max()、min()

#### max()函数和min()函数

max() 函数的作用是返回元组中元素最大值。min() 函数的作用是返回元组中元素最小值。

tup1 = (4, 6, 2, 0, -5)

print(max(tup1))

print(min(tup1))

tup2 = ('a', 'z', 'A', 'Z')

print(max(tup2))

print(min(tup2))

6

-5

z

A

### 3.6 删除元组 del

元组中的元素值是不允许删除的，但我们可以使用 del 函数来删除整个元组。

word\_tuple = ('a', 'b', 'c')

del word\_tuple

当我们使用 del 函数删除某元组后，再使用 print() 函数打印输出时，会报错NameError: name 'word\_tuple' is not defined,表明该元组未被定义。

## 4、内置方法

### 4.1 元素出现次数 count()

count() 方法返回指定值在元组中出现的次数。

语法：tuple.count(value)

实例：

返回值 5 在元组中出现的次数：

num\_tuple = (1, 3, 7, 8, 7, 5, 4, 6, 8, 5, 5)

print(num\_tuple.count(5))

1

### 4.2 元素位置 index()

index() 方法查找指定值的第一次出现。

语法：tuple.index(value)

实例：

num\_tuple = (1, 3, 7, 8, 7, 5, 4, 6, 8, 5, 5)

print(num\_tuple.index(5))

5

注意：当被搜索值在元组中存在多个，则返回第一次出现的下标。

num\_tuple = (1, 3, 7, 8, 7, 5, 4, 6, 8, 5, 5)

print(num\_tuple.index(999))

如果未找到该值，index() 方法将引发异常，ValueError: tuple.index(x): x not in tuple。

5、总结

函数 描述

print() 打印输出

len() 确定元组项目

type() 返回变量类型

tuple() 转换为元组

max() 返回元组元素最大值

min() 返回元组元素最小值

del 删除元组

方法 描述

count() 返回元组中指定值出现的次数

index() 在元组中搜索指定的值并返回它被找到的位置

# **遍历字典以及字典中的键和值**

## 1、遍历字典

### 1.1、访问某个键对应的值，直接将键名放在方括号内置于字典名后即可。

示例

dict\_1 = {"我":5,"爱":2,"你":1,"祖":13,"国":14}

print(dict\_1["我"],dict\_1["爱"],dict\_1["你"])

#5 2 1

### 1.2、直接进行for循环遍历，得到的是字典中的键。

示例：

dict\_1 = {"我":5,"爱":2,"你":1,"祖":13,"国":14}

for num in dict\_1:

print(num)

我

爱

你

祖

国

## 2、遍历字典的键值对

方法：结合for循环和dict.items()进行遍历

示例：

dict\_1 = {"我":5,"爱":2,"你":1,"祖":13,"国":14}

for k,v in dict\_1.items():

print("字典中的键值对为:(%s:%s)"%(k,v))

字典中的键值对为: (我:5)

字典中的键值对为: (爱:2)

字典中的键值对为: (你:1)

字典中的键值对为: (祖:13)

字典中的键值对为: (国:14)

## 3、遍历字典中的键

方法：结合for循环和dict.keys()

示例：

dict\_1 = {"我":5,"爱":2,"你":1,"祖":13,"国":14}

for k in dict\_1.keys():

#print("字典中的键为:%s"%k.title())

print("字典中的键为:%s"%k)

字典中的键为: 我

字典中的键为: 爱

字典中的键为: 你

字典中的键为: 祖

字典中的键为: 国

## 4、遍历字典中的值

方法：结合for循环和dict.values()

示例：

dict\_1 = {"我":5,"爱":2,"你":1,"祖":13,"国":14}

for v in dict\_1.values():

print("字典中的值为:%s"%v)

字典中的值为: 5

字典中的值为: 2

字典中的值为: 1

字典中的值为: 13

字典中的值为: 14

注意：

关于字典的创建、修改、删除、遍历、查找，见

<https://blog.csdn.net/cadi2011/article/details/85857917>

# zip函数的用法

## 介绍

zip()是Python的一个内建函数，让我们直接观察一下它的运行结果：

>>> a = ['a', 'b', 'c', 'd']

>>> b = ['1', '2', '3', '4']

>>> list(zip(a, b))

[('a', '1'), ('b', '2'), ('c', '3'), ('d', '4')]

很明显，对于我们的两个list，a和b，list(zip(a, b))生成了一个列表。在这个列表中，每个元素是一个tuple；对于第i个元组，它其中的内容是(a[i-1], b[i-1])。这样的操作，与压缩软件的“压缩”十分接近。如果我们继续在zip()中加入更多的参数，比如zip(a, b, c, d)，那么在将它转换成list之后，结果当然就是[(a[0], b[0], c[0], d[0]), (a[1], b[1], c[1], d[1]), ..., (a[n-1], b[n-1], c[n-1], d[n-1])]。

事实上，在 Python 3 中，为了节省空间，zip()返回的是一个tuple的迭代器，这也是我们为什么要调用list()将它强制转换成list的原因。不过，Python 2中，它直接返回的就是一个列表了。

看了上面这些介绍，你也许对zip()这个函数有了稍微初步的认识。但是，你有没有意识到，什么地方可能存在问题呢？

这个问题就是：如果我们传入zip()中的几个参数不等长，会有什么结果呢？zip()很灵活，如果几个参数不等长，那么它会取最短的那个参数的长度，然后再进行处理。至于那些超出最短长度的成员，就只好被舍弃掉了。

那么，如果我们想从得到的“压缩”后的结果，还原成“压缩”前的几个列表，应该怎么做呢？

对于zip(args)这个函数，Python还提供了一种逆操作。例如，我们有

result = zip(a, b)

那么，只要调用

origin = zip(\*result) #前面加\*号，事实上\*号也是一个特殊的运算符，叫解包运算符

就可以得到原来的a和b了。利用这个特性，可以用一种特殊的方法处理一些问题，我们待会说。

## 用法

### 1.同时遍历多个字典

在 Python 3.6+ 中，字典成为了一种有顺序的集合。利用这个特性和zip，我们可以同时遍历多个字典：

>>> dict\_one = {'name': 'John', 'last\_name': 'Doe', 'job': 'Python Consultant'}

>>> dict\_two = {'name': 'Jane', 'last\_name': 'Doe', 'job': 'Community Manager'}

>>> for (k1, v1), (k2, v2) in zip(dict\_one.items(), dict\_two.items()):

... print(k1, '->', v1)

... print(k2, '->', v2)

...

name -> John

name -> Jane

last\_name -> Doe

last\_name -> Doe

job -> Python Consultant

job -> Community Manager

### 2.对多个元素同时进行排序

考虑一个场景：你正在处理一些学生的成绩，有这样两个列表：

names = ['John', 'Amy', 'Jack']

scores = [98, 100, 85] # 分数和名字是一一对应的

如果你想对它们进行排序，又不想破坏对应关系的话，就可以这样：

data = list(zip(names, scores))

data.sort()

结果是：

[('Amy', 100), ('Jack', 85), ('John', 98)]

如果要先对分数进行排序：

data2 = list(zip(scores, names))

data2.sort()

结果是：

[(85, 'Jack'), (98, 'John'), (100, 'Amy')]

### 3.将数据成对进行计算

假设你有这样一个表格：

|  |  |  |  |
| --- | --- | --- | --- |
|  | 一月 | 二月 | 三月 |
| 销售额 | 52,000.00 | 51,000.00 | 48,000.00 |
| 成本 | 46,800.00 | 45,900.00 | 43,200.00 |

利润 = 销售额 - 成本，如果要求每月的利润的话，就可以：

total\_sales = [52000.00, 51000.00, 48000.00]

prod\_cost = [46800.00, 45900.00, 43200.00]

for sales, costs in zip(total\_sales, prod\_cost):

profit = sales - costs

print(f'Total profit: {profit}')

结果是：

Total profit: 5200.0

Total profit: 5100.0

Total profit: 4800.0

### 4.构建字典

回到上面处理学生成绩的场景：如果你想将两个列表合并起来，得到一个姓名-分数的字典，就可以：

stu = dict(zip(names, scores))

结果是：

{'John': 98, 'Amy': 100, 'Jack': 85}

### 5.秒杀某些字符串处理的题

还记得吗，我上面留了个坑，现在就是填坑的时候了。

利用zip()解压的特性，我们可以使用它秒杀某些字符串处理的题目。例如，力扣的最长公共前缀：

编写一个函数来查找字符串数组中的最长公共前缀。

如果不存在公共前缀，返回空字符串 “”。

示例 1:

输入: [“flower”,“flow”,“flight”]

输出: “fl”

示例 2:

输入: [“dog”,“racecar”,“car”]

输出: “”

解释: 输入不存在公共前缀。

说明:

所有输入只包含小写字母 a-z 。

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/longest-common-prefix

著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。

这道题，我一开始的思路是：

class Solution:

def longestCommonPrefix(self, strs: List[str]) -> str:

if len(strs) == 0:

return ""

target = strs[0] # 令第一个字符串作为基准

for i in strs[1:]: # 分别对每个字符串进行比较

target = target[:min(len(target), len(i))] # 最长公共前缀肯定<=最短字符串的长度

for j in range(len(target)): # 比较基准字符串和当前字符串的没个字幕

if target[j] != i[j]: # 如果出现不一样的字符，说明最长前缀肯定不包括当前字符

if j == 0:

return ""

else:

target = target[:j] # 当前字符以前的字符都有可能是公共前缀

break

return target

最后虽然过了，但是执行用时大概排在50%左右，不是很好的成绩。当我看到最短用时的范例的时候，有一点惊为天人的感觉：

class Solution:

def longestCommonPrefix(self, strs: List[str]) -> str:

result = ""

for temp in zip(\*strs):

if len(set(temp)) == 1:

result += temp[0]

else:

break

return result

这里就用到了解压的特性。上面已经说过，如果我们把元素压缩后，得到：

[('a', '1'), ('b', '2'), ('c', '3'), ('d', '4')]

那么还原回去就是：

['a', 'b', 'c', 'd']

['1', '2', '3', '4']

这个题目中给的字符串列表是这样的：

["flower","flow","flight"]

如果我们把它看作是压缩后的结果，也就是说，

(a[0], b[0], c[0], d[0], e[0], f[0]) == ('f', 'l', 'o', 'w', 'e', 'r')

(a[1], b[1], c[1], d[1]) == ('f', 'l', 'o', 'w')

(a[2], b[2], c[2], d[2], e[2], f[2]) == ('f', 'l', 'i', 'g', 'h', 't')

那么，如果我们将它们进行解压，得到的结果就应该是这样的：

a = ['f', 'f', 'f']

b = ['l', 'l', 'l']

c = ['o', 'o', 'i']

d = ['w', 'w', 'g']

# 没有e和f，因为压缩是按最短来的，解压也一样，上面只是为了方便说明

是不是一目了然了？另外，力扣的螺旋矩阵也可以用到zip()来简化代码，试试看！

# join函数的用法

## 1.join函数的语法及用法

### （1）语法：'sep'.join(sep\_object)

参数说明：

sep：分割符，可为“，、；”等。

sep\_object：分割对象，可为字符串、以及储存字符串的元组、列表、字典。

### （2）用法：连接任意数量的字符串（包括要连接的元素字符串、元组、列表、字典），用新的目标分隔符连接，返回新的字符串。

## 2.join函数的实例

### （1）对象为字符串

';'.join('abc')

#输出结果为：'a;b;c'

string1 = 'good idea'#定义字符串

' '.join(string1)

#输出结果：'g o o d i d e a'

### （2）对象为元组

tuple1 = ('a','b','c') #定义元组tuple1

'、'.join(tuple1)

# 输出结果为‘a、b、c’

tuple2 = ('hello','peace','world') #定义元组tuple2  
' '.join(tuple2)

# 输出结果为’hello peace world’

### （3）对象为列表

b = ['a','b','c'] #定义列表

'、'.join(b)

# 输出结果为‘a、b、c’

list1 = ['hello','peace','world'] #定义列表

' '.join(list1)

# 输出结果为’hello peace world’

### 对象为字典

c={'hello':1,'world':2}

';'.join(c)

#输出结果为’hello;world’

d = {'hello':'hi','world':'2'}

' '.join(d)

#输出结果为’hello world’

## 3.易错点及解决办法

### （1）问题：储存非字符串的元组、列表、字典等报错，比如元组储存数字进行连接

a = (1,2,3) #定义元组a

';'.join(a)

#报错:TypeError: sequence item 0: expected str instance, int found

### 解决办法：要将数字连接起来成为一个字符串，则结合for循环语句并将数字转为字符串再连接起来

a = (1,2,3) #定义元组a

';'.join(str(i) for i in a)

#输出结果为’1;2;3’

b = (186234,1385475,1235462)

';'.join(str(i) for i in b)

#输出结果为‘186234;1385475;1235462‘

#调用set函数去重处理

e = (1,2,3,2) #定义元组a

'、'.join(str(i) for i in set(e))

#输出结果为’1、2、3’

注意！join函数中分割的对象必须为字符串，否则会报错

# round函数

在Python中，round()函数是一个内置函数，用于将一个数字四舍五入为指定的小数位数或整数位数。

round()函数有两个参数：第一个参数是要四舍五入的数字，第二个参数（可选）是小数位数或整数位数，表示要保留的小数位数或整数位数，默认为0。下面分别介绍参数的用法：

四舍五入到整数位数：如果第二个参数为0，则round()函数将把数字四舍五入到最接近的整数。例如：

> round(2.7)

3

> round(2.4)

2

四舍五入到指定小数位数：如果第二个参数为正数，则round()函数将把数字四舍五入到指定的小数位数。例如：

>round(2.3456, 2)

2.35

>round(2.3456, 1)

2.3

在这个例子中，第一个例子将数字四舍五入到两位小数，而第二个例子将数字四舍五入到一位小数。

四舍五入到指定整数位数：如果第二个参数为负数，则round()函数将把数字四舍五入到指定的整数位数。例如：

> round(123456, -3)

123000

>round(123456, -4)

120000

在这个例子中，第一个例子将数字四舍五入到千位，而第二个例子将数字四舍五入到万位。

需要注意的是，round()函数的四舍五入规则遵循“银行家舍入法”，也就是说，如果要舍去的数字恰好是5，而其后面还有数字，则根据该数字的奇偶性来判断舍入结果。例如：

> round(2.5)

2

> round(3.5)

4

在这个例子中，第一个例子将2.5舍入为2，而第二个例子将3.5舍入为4

总之，round()函数可以帮助我们将数字四舍五入到指定的小数位数或整数位数，是一个非常常用的函数。

# replace函数

replace函数用于把字符串中的子字符串替换成指定字符串

语法：

str.replace(oldvalue, newvalue[, count])

提示Tips: 如果参数只指定了oldvalue以及newvalue，未指定count，则将替换所有出现的指定子字符串oldvalue

参数 ：

oldvalue：将被替换的子字符串(必需)

newvalue：用于替换oldvalue的子字符串(必需)

count：数值次数，指定要替换多少个oldvalue(可选)

返回值：

返回字符串中的oldvalue替换成newvalue后生成的新字符串，如果指定了第三个参数count，则替换不超过count次

例1: 把字符串words中的每个空格替换成"%20"

words = "We are from China."

res = words.replace(' ', '%20')

print(res)

# We%20are%20from%20China.

例2: 将txt字符串中所有的"is"替换成"was"

txt = "this is a test, this is a test."

res = txt.replace("is", "was")

print(res)

# thwas was a test, thwas was a test.

例3: 将txt字符串中前两次出现的"is"替换成"was"

txt = "this is a test, this is a test."

res = txt.replace("is", "was", 2)

print(res)

# thwas was a test, this is a test.

注意:replace函数不会改变原来字符串的内容

original\_str = 'My Name is Andy.'

print(original\_str.replace('is','IS'))

# My Name IS Andy.  
print(original\_str)

# My Name is Andy.

# set函数

查看python中set介绍，有如下介绍：

"""

set() -> new empty set object

set(iterable) -> new set object

Build an unordered collection of unique elements.

"""

由此可知：

set函数返回值：

一个 set object （集合对象，即 一个 “无序不重复元素集”，特点 ：无序，不重复）

参数：

空 或者 一个可迭代对象。

根据set对象的特性，探讨其用法与不同。

特性一：不重复

可以十分方便的实现一个去重的目的。

当然，我们也可以将set对象转化为其它的可迭代对象。

去重实例：

ls = [1, 2, 2, 3, 3, 3]

x = set(ls)

print(type(x), x, list(x), tuple(x))

结果：

<class 'set'> {1, 2, 3} [1, 2, 3] (1, 2, 3)

特性二：无序

注：对于无序的可迭代对象，不能通过索引的方式取值。

例：

ls = [1, 2, 2, 3, 3, 3]

x = set(ls)

print(x)

a = x[0]

print(a)

结果报错：

{1, 2, 3}

a = x[0]

TypeError: 'set' object is not subscriptable

tips：在集合set里查找元素效率很高，超过在列表里查找的效率，这是由于两种类的底层实现原理不同。