地空数算编程基础交流

By 易超

2016.03.13

Contents ● Python基础交流

1 Built-in Types Review

2 Some Built-in Functions

3 Examples

1

Built-in Types Review

list、tuple、range、str、dict

Common Sequence Operations

Operation	Result
x in s	True if an item of *s* is equal to *x*, else False
x not in s	False if an item of *s* is equal to *x*, else True
s + t	the concatenation of *s* and *t*
s * n or n * s	equivalent to adding *s* to itself *n* times
s[i]	*i*th item of *s*, origin 0
s[i:j]	slice of *s* from *i* to *j*
s[i:j:k]	slice of *s* from *i* to *j* with step *k*
len(s)	length of *s*
min(s)	smallest item of *s*
max(s)	largest item of *s*
s.index(x[, i[, j]])	index of the first occurrence of *x* in *s* (at or after index *i* and before index *j*)
s.count(x)	total number of occurrences of *x* in *s*

Mutable Sequence Operations

Operation	Result
s[i] = x	item *i* of *s* is replaced by *x*
s[i:j] = t	slice of *s* from *i* to *j* is replaced by the contents of the iterable *t*
del s[i:j]	same as s[i:j] = []
s[i:j:k] = t	the elements of s[i:j:k] are replaced by those of *t*
del s[i:j:k]	removes the elements of s[i:j:k] from the list
s.append(x)	appends *x* to the end of the sequence (same ass[len(s):len(s)] = [x])
s.clear()	removes all items from s (same as del s[:])
s.copy()	creates a shallow copy of s (same as s[:])
s.extend(t) or s += t	extends *s* with the contents of *t* (for the most part the same as s[len(s):len(s)] = t)
s *= n	updates *s* with its contents repeated *n* times
s.insert(i, x)	inserts *x* into *s* at the index given by *i* (same ass[i:i] = [x])
s.pop([i])	retrieves the item at *i* and also removes it from *s*
s.remove(x)	remove the first item from *s* where s[i] == x
s.reverse()	reverses the items of *s* in place

Lists

Lists may be constructed in several ways:

- Using a pair of square brackets to denote the empty list: []
- Using square brackets, separating items with commas: [a], [a, b, c]
- Using a list comprehension: [x for x in iterable]
- Using the type constructor: list() or list(iterable)

Lists implement all of the common and mutable sequence operations. Lists also provide the following additional method:

- sort(*, key=None, reverse=None)

Tuples

Tuples may be constructed in a number of ways:

- Using a pair of parentheses to denote the empty tuple: ()
- Using a trailing comma for a singleton tuple: a, or (a,)
- Separating items with commas: a, b, c or (a, b, c)
- Using the tuple() built-in: tuple() or tuple(iterable)

Tuples implement all of the common sequence operations.

Ranges

The range type represents an immutable sequence of numbers.

- class range(stop)
- class range(start, stop[, step])

Ranges

```
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(1, 11))
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> list(range(0, 30, 5))
[0, 5, 10, 15, 20, 25]
>>> list(range(0, 10, 3))
[0, 3, 6, 9]
>>> list(range(0, -10, -1))
[0, -1, -2, -3, -4, -5, -6, -7, -8, -9]
>>> list(range(0))
[]
>>> list(range(1, 0))
[]
```

Ranges

```
>>> r = range(0, 20, 2)
>>> r
range(0, 20, 2)
>>> 11 in r
False
>>> 10 in r
True
>>> r.index(10)
5
>>> r[5]
10
>>> r[:5]
range(0, 10, 2)
>>> r[-1]
18
```

Str

String literals are written in a variety of ways:

- Single quotes: 'allows embedded "double" quotes'
- Double quotes: "allows embedded 'single' quotes".
- Triple quoted: '''Three single quotes''', """Three double quotes"""

String Methods

- str.capitalize()
  Return a copy of the string with its first character capitalized and the rest lowercased.
- str.endswith(suffix[, start[, end]])
  Return True if the string ends with the specified suffix, otherwise return False. suffix can also be a tuple of suffixes to look for. With optional start, test beginning at that position. With optional end, stop comparing at that position.
- str.find(sub[, start[, end]])
  Return the lowest index in the string where substring sub is found within the slice s[start:end]. Optional arguments start and end are interpreted as in slice notation. Return -1 if sub is not found.

String Methods

- str.index(sub[, start[, end]])
  Like find(), but raise ValueError when the substring is not found.
- str.join(iterable)
  Return a string which is the concatenation of the strings in the iterable iterable. A TypeError will be raised if there are any non-string values in iterable, including bytes objects. The separator between elements is the string providing this method.
- str.startswith(prefix[, start[, end]])
  Return True if string starts with the prefix, otherwise return False. prefix can also be a tuple of prefixes to look for. With optional start, test string beginning at that position. With optional end, stop comparing string at that position.

String Methods

- str.split(sep=None, maxsplit=-1)

```
>>> '1,2,3'.split(',')
['1', '2', '3']
>>> '1,2,3'.split(',', maxsplit=1)
['1', '2,3']
>>> '1,2,,3,'.split(',')
['1', '2', '', '3', '']

>>> '1 2 3'.split()
['1', '2', '3']
>>> '1 2 3'.split(maxsplit=1)
['1', '2 3']
>>> '   1   2   3   '.split()
['1', '2', '3']
```

String Methods

- str.strip([chars])
  Return a copy of the string with the leading and trailing characters removed. The chars argument is a string specifying the set of characters to be removed. If omitted or None, the chars argument defaults to removing whitespace. The chars argument is not a prefix or suffix; rather, all combinations of its values are stripped:

```
>>> '   spacious   '.strip()
'spacious'
>>> 'www.example.com'.strip('cmowz.')
'example'
```

Dict

```
class dict(**kwarg)
class dict(mapping, **kwarg)
class dict(iterable, **kwarg)

>>> a = dict(one=1, two=2, three=3)
>>> b = {'one': 1, 'two': 2, 'three': 3}
>>> c = dict(zip(['one', 'two', 'three'], [1, 2, 3]))
>>> d = dict([('two', 2), ('one', 1), ('three', 3)])
>>> e = dict({'three': 3, 'one': 1, 'two': 2})
>>> a == b == c == d == e
True
```

Dict Operations

- len(d)
  Return the number of items in the dictionary d.
- d[key]
  Return the item of d with key key. Raises a KeyError if key is not in the map.
- d[key] = value
  Set d[key] to value.
- del d[key]
  Remove d[key] from d. Raises a KeyError if key is not in the map.
- key in d
  Return True if d has a key key, else False.
- key not in d
  Equivalent to not key in d.

Dict Operations

- clear()
  Remove all items from the dictionary.
- copy()
  Return a shallow copy of the dictionary.
- get(key[, default])
  Return the value for key if key is in the dictionary, else default. If default is not given, it defaults to None, so that this method never raises a KeyError.
- items()¶
  Return a new view of the dictionary’s items ((key, value) pairs). See the documentation of view objects.
- keys()
  Return a new view of the dictionary’s keys. See the documentation of view objects.

Dict Operations

- pop(key[, default])
  If key is in the dictionary, remove it and return its value, else return default. If default is not given and key is not in the dictionary, a KeyError is raised.
- popitem()
  Remove and return an arbitrary (key, value) pair from the dictionary.
- setdefault(key[, default])
  If key is in the dictionary, return its value. If not, insert key with a value of default and return default. default defaults to None.
- update([other])
  Update the dictionary with the key/value pairs from other, overwriting existing keys. Return None.
- values()
  Return a new view of the dictionary’s values. See the documentation of view objects.

Contents ● Python基础交流

1 Built-in Types Review

2 Some Built-in Functions

3 Examples

2

Some Built-in Functions

eval()、exec()、sorted()、zip()

eval() and exec()

```
>>> a = "[[1,2], [3,4], [5,6], [7,8], [9,0]]"
>>> b = eval(a)
>>> b
[[1, 2], [3, 4], [5, 6], [7, 8], [9, 0]]
>>> type(b)
<class 'list'>
>>> a = "{1: 'a', 2: 'b'}"
>>> b = eval(a)
>>> b
{1: 'a', 2: 'b'}
>>> type(b)
<class 'dict'>
>>> a = "([1,2], [3,4], [5,6], [7,8], (9,0))"
>>> b = eval(a)
>>> b
([1, 2], [3, 4], [5, 6], [7, 8], (9, 0))
>>> type(b)
<class 'tuple'>
```

eval() and exec()

```
>>> exec("print(\"hello, world\")")
hello, world
>>> a = 1
>>> exec("a = 2")
>>> a
2
>>> scope = {}
>>> exec("a = 4", scope)
>>> a
2
>>> scope['a']
4
>>> result = eval('2+3')
>>> result
5
```

```
>>> scope={}
>>> scope['a'] = 3
>>> scope['b'] = 4
>>> result = eval('a+b',scope)
>>> result
7
```

sorted()

```
>>> sorted([2,1,4,5,3])
[1, 2, 3, 4, 5]
>>> L = [('b',2),('a',1),('c',3),('d',4)]
>>> sorted(L, key=lambda x:x[1])
[('a', 1), ('b', 2), ('c', 3), ('d', 4)]
>>> sorted([5, 2, 3, 1, 4],
reverse=True)
[5, 4, 3, 2, 1]
>>> sorted([5, 2, 3, 1, 4],
reverse=False)
[1, 2, 3, 4, 5]
```

zip()

```
>>> x = [1, 2, 3]
>>> y = [4, 5, 6]
>>> z = [7, 8, 9]
>>> xyz = zip(x, y, z)
>>> list(xyz)
[(1, 4, 7), (2, 5, 8), (3, 6, 9)]
>>> dict(zip(x,y))
{1: 4, 2: 5, 3: 6}
```

Contents ● Python基础交流

1 Built-in Types Review

2 Some Built-in Functions

3 Examples

3

Examples

—— 三角形面积 ——

以((x1,y1),(x2,y2),(x3,y3))的格式输入三角形的三个顶点，输出三角形的面积
其中三角形的面积公式为S=√[p(p-a)(p-b)(p-c)]，（p=(a+b+c)/2）

Geometry类

写三个类Geometry、Triangle、Quadrange，分别代表几何类型、三角形、四边形，其中后面两个类继承第一个类，并且三角形和四边形都能获得边数和面积

```
class Geometry:
    def __init__(self,points):
        self.points=points
    def getEdgeNum(self):
        return len(self.points)

class Triangle(Geometry):
    def getArea(self):

class Quadrange(Geometry):
    def getArea(self):
```

—— 统计字符——

```
#输入字符串(输入0结束)，统计每个字符出现的次数，并且以列表形式排序
#输出（按字符出现的次数从小到大排序，在次数相同的情况下按照字符的
#ASCII码从小到大排序）
while 1:
    s=input("Input a string(0 for exit):")
    if s=="0":
        break
    dict={}
    for c in s:
        if c in dict:
            dict[c]+=1
        else:
            dict[c]=1
    print(sorted(dict.items(),key=lambda d:(d[1],d[0]),reverse=False))
```

——计算排列种数——

```
#男女排成一条直线，其中男和男不能相邻，输入人数n，输出有多少种排列方式
#如n为3，排列的可能有 男女男、女女女、男女女、女男女、女女男 共5种可能
while 1:
    n=input("Input n:")
    if n=="-1":
        break
    n=int(n)
    if n<1:
        print("n is less than 1")
        continue
    array=[(0,0),(1,1)]
    for i in range(n-1):
        array.append((array[-1][1],array[-1][0]+array[-1][1]))
    print(array[-1][0]+array[-1][1])
```

References

https://docs.python.org/3.4/

谢谢！