九章算法

第二章 Python语法基础 II

九章算法-Justin

United States

林平之 老师

Scan the QR code to add me on WeChat

对课程有疑问？购买时遇到问题？获取更多优惠信息？扫一扫，在线咨询

微信: ninechapter

知乎专栏: http://zhuanlan.zhihu.com/jiuzhang

微博: http://www.weibo.com/ninechapter

官网: www.jiuzhang.com

Copyright © www.jiuzhang.com

版权声明

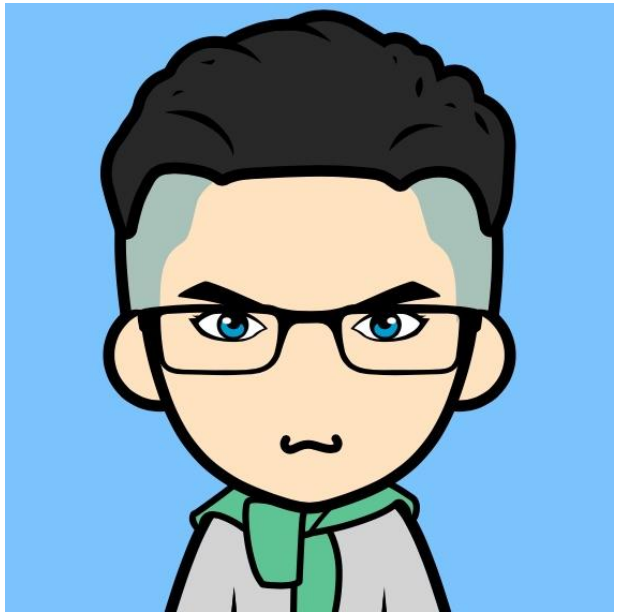
九章课程不提供视频，也严禁录制视频的侵权行为

否则将追求法律责任和经济赔偿

请不要缺课

Copyright © www.jiuzhang.com

自我介绍



林老师

全国算法竞赛一等奖
国内TOP2名校毕业
参加国家信息学竞赛NOI
前FLAG工程师
拥有丰富的面试经验

Copyright © www.jiuzhang.com

本节重点

- Python基础类型之 Dict & Set
- Python中如何使用排序方法
- lambda, Map, Filter, Reduce函数的使用
- Python的生成器与迭代器

Copyright © www.jiuzhang.com

Python的基础类型Dict

Copyright © www.jiuzhang.com

Python的基础类型Dict

- dict全称dictionary，其他语言中也称为map或者hash，使用键-值（key-value）存储

- 存储映射关系
-

```
names = ['Alice', 'Bob', 'linpz']
scores = [99, 97, 98]

d = {'Alice': 99, 'Bob': 97, 'linpz': 98}
```

Copyright © www.jiuzhang.com

Python的基础类型Dict

- 提问：dict为什么那么快？

- key-value存储方式：Insert放进去的时候，根据key计算value的存放位置，这样，取的时候再根据key就可以得到value存放的位置
-

```
names = dict()  # names = {}
names['ta'] = '96'
print(names['ta'])
```

Copyright © www.jiuzhang.com

- 访问不存在的key时会出现错误

```
print(names['linghc'])

KeyError: 'linghc'
```

- 判断key是否存在的两种方是:

```
'ta' in names

names.get('linpz')  # return None if key doesn't exist
```

Copyright © www.jiuzhang.com

Python的基础类型Dict

从Dict中获取key对应的值，当key不存在的时候范围一个default的值

```
# P:
linpz = {'yanzhi': 5, 'name': 8}
linpz['yanzhi'] = linpz.get('yanzhi', 0) + 1

# linpz = {'yanzhi': 6, 'nice': 8}

#NP:
linpz = {'yanzhi': 5, 'name': 8}
if 'yanzhi' in linpz:
    linpz['yanzhi'] += 1
else:
    linpz['yanzhi'] = 1
# linpz = {'yanzhi': 6, 'nice': 8}
```

dict的get(key,default)方法用于获取字典中key的值，若不存在该key，则默认值default

Copyright © www.jiuzhang.com

OrderedDict的Key会按照插入的顺序排列，不是Key本身排序

```
from collections import OrderedDict

d = OrderedDict()
```

普通的dict可以按key排序后放入OrderedDict来维持key的顺序

Copyright © www.jiuzhang.com

Python的基础类型Dict

对两组List配对组成Dict
keys = ['name', 'yanzhi', 'nice']
values = ['linpz', 5, 8]

{'name': 'linpz', 'yanzhi': 5, 'nice': 8}

```
keys = ['name', 'yanzhi', 'nice']
values = ['linpz', 5, 8]

# P
result = dict(zip(keys, values))
# {'name': 'linpz', 'yanzhi': 5, 'nice': 8}


# NP:
result = {}
for index, key in enumerate(keys):
    result[key] = values[index]
```

Copyright © www.jiuzhang.com

Python的基础类型Dict

- dict的特点，与list对比

Dict 特点

1、查找和插入的速度快，不会随着key的增加而变慢

2、占用大量的内存，内存浪费多

List 特点

1、查找和插入的时间随着元素的增加而增加，超多的元素就会巨慢

2、浪费的内存很少

Copyright © www.jiuzhang.com

Python的基础类型Set

Copyright © www.jiuzhang.com

Python的基础类型Set

- set和dict非常类似，但是只有key，没有value
- set就是key和value相等情况下的特殊dict
- set中没有重复的key
- ```
  s = set([1, 2, 3, 4])
  print(s)
  ```

Copyright © www.jiuzhang.com
  ```

- set添加元素 - add

```
s = set([])
s.add(5)
```

- set删除元素 - remove

```
s = set([1, 2, 3])
s.remove(2)
```

Copyright © www.jiuzhang.com

- 两个元素的交集

```
s1 & s2
```

- 两个元素的并

```
s1 | s2
```

Copyright © www.jiuzhang.com

Python的基础类型Set

- 判断元素是否在集合内

```
ele in set
```

- 对比两份代码的时间效率

```
import time

l = [i for i in range(100000)]

start = time.time()
for i in range(50000):
    if i in l:
        pass

print(time.time() - start)
```

```
import time

l = set([i for i in range(100000)])

start = time.time()
for i in range(50000):
    if i in l:
        pass

print(time.time() - start)
```

Copyright © www.jiuzhang.com

Sort 排序

Copyright © www.jiuzhang.com

- 对list进行排序
  - 利用list的成员函数sort()排序
  - 利用内置函数sorted()进行排序
  - 利用重写__lt__函数来覆盖object的比较方法

Copyright © www.jiuzhang.com

Python中的排序 - 对List进行排序


```
nums = [1, 2, 5, 4, 3]

nums.sort()

nums_copy = sorted(nums)
```

- sort() 对list进行排序,改变list的值
- sorted() 产生一个新的list，不改变原list的值

Copyright © www.jiuzhang.com

Python中的排序 - 对List进行排序

- 参数sort中参数key

```
nums = [(1, 2), (3, 4), (1, 6)]

nums.sort(key=lambda x:x[1])

print(nums)
```

Copyright © www.jiuzhang.com

Python中的排序 - 对List进行排序


```
from functools import cmp_to_key

def cmp(a, b):
    return a - b

nums = [1, 2, 5, 3, 4]
nums.sort(key=cmp_to_key(cmp))
print(nums)
```

- Python3中sort中的参数不再有cmp
- In Py3, the cmp parameter was removed entirely (as part of a larger effort to simplify and unify the language, eliminating the conflict between rich comparisons and the __cmp__() magic method

Copyright © www.jiuzhang.com

Python中的排序 - 对对象进行排序，重写lt方法

九章算法

```
class Point(object):

    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __lt__(self, other):
        if self.x != other.x:
            return self.x < other.x
        return self.y < other.y

points = [Point(3, 10), Point(4, 2), Point(3, -1), Point(1, -1), Point(4, 10)]

points.sort()

for p in points:
    print(p.x, p.y)
```

Copyright © www.jiuzhang.com

Map，Filter与Reduce

Copyright © www.jiuzhang.com

Python - 匿名函数lambda


```
f = lambda x: 2 ** x
print(f(3))
```

1. 使用lambda的地方都可以使用正常的函数
2. 更像一个表达式
3. lambda函数是否可以有多个参数？

Copyright © www.jiuzhang.com

Python - filter函数


```
a = filter(lambda x: x % 2 == 1, [1, 2, 3, 4, 5])
print(a)
print(list(a))
```

filter过滤出满足表达式的元素

Copyright © www.jiuzhang.com

Python - map函数


```
b = map(lambda x: 2 ** x, [1, 2, 3, 4, 5])
print(b)
print(list(b))
```

对每一个元素做变换，从x映射成为2 ** x

Copyright © www.jiuzhang.com

```
from functools import reduce
reduce(lambda x, y: y ** x, [1, 2, 3, 4])
```

不断的对这个list内的元素进行迭代计算，reduce有范围值

x, y = 1, 2　返回 2 ** 1 = 2

x, y = 2, 3　返回 3 ** 2 = 9

x, y = 9, 4　范围 4 ** 9 = 262144

Copyright © www.jiuzhang.com

Python的迭代器与生成器

Copyright © www.jiuzhang.com

Python中的迭代器与生成器

- 迭代是Python最强大的功能之一
- 迭代器是可以记住遍历的位置的对象。
- 代器对象从集合的第一个元素开始访问，直到所有的元素被访问完结束。*迭代器只能往前不会后退*。
- 迭代器有两个基本的方法：iter() 和 next()。

Copyright © www.jiuzhang.com

创建迭代器与使用迭代器

```
nums = [10, 200, 300, 50]
it = iter(nums)
print (next(it))
print (next(it))
```

任何列表，字符串，字典等类型都可以创建迭代器

Copyright © www.jiuzhang.com

可以使用for或者循环去遍历迭代器

```
nums = [10, 200, 300, 50]
it = iter(nums)
print (next(it))
print (next(it))


while True:
    try:
        print(next(it))
    except StopIteration:
        break
```

Copyright © www.jiuzhang.com

迭代器的本质是内部实现了__iter__和__next__的方法

例如lintcode上iterator相关的问题：

https://www.lintcode.com/problem/zigzag-iterator/description

Copyright © www.jiuzhang.com

Python中的迭代器与生成器

- 迭代器就是实现了工厂模式的对象
- 每次你询问要下一个值的时候给出返回
- 比如itertools函数返回的都是迭代器对象

Copyright © www.jiuzhang.com

例如:生成无限序列

```
from itertools import count
counter = count(start=5)

print(next(counter))
print(next(counter))
```

Copyright © www.jiuzhang.com

例如:有限序列中生成无限序列

```
from itertools import cycle
colors = cycle(['red', 'yellow', 'green'])
print(next(colors))
print(next(colors))
print(next(colors))
```

Copyright © www.jiuzhang.com

例如:从无限的序列中生成有限序列

```
from itertools import cycle, islice
colors = cycle(['red', 'yellow', 'green'])
print(next(colors))
print(next(colors))
print(next(colors))


content = islice(colors, 0, 10)
for x in content:
    print(x)
```

Copyright © www.jiuzhang.com

什么是生成器:

生成器是一种优雅的特殊迭代器

在 Python 中，使用了 yield 的函数被称为生成器（generator）

Copyright © www.jiuzhang.com

什么是生成器:

生成器是一种优雅的特殊迭代器

在 Python 中，使用了 yield 的函数被称为生成器（generator）

Copyright © www.jiuzhang.com

- 任何生成器是以一种懒加载的模式生成值
- 用生成器来实现斐波那契数列

```
from itertools import cycle, islice

def fib():
    a, b = 0, 1
    while True:
        yield b
        a, b = b, a + b

f = fib()
list(islice(f, 3, 10))
```

以上fib函数没有return关键字

Copyright © www.jiuzhang.com

- 生成器在Python中是一个非常强大的编程结构
- 更少地中间变量写流式代码

```
def something():
    result = []
    for ... in ...:
        result.append(x)
    return result
```

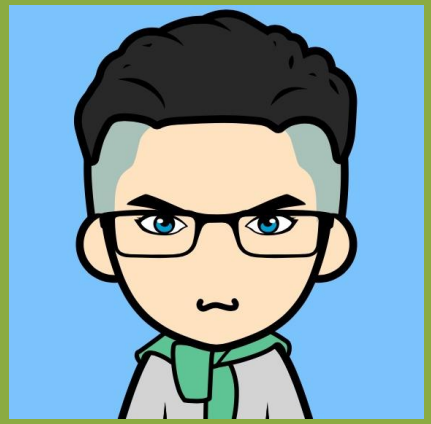
```
def iter_something():
    for ... in ...:
        yield x
```

Copyright © www.jiuzhang.com

扫描二维码关注微信/微博
获取最新面试题及权威解答

微信: ninechapter
微博: http://www.weibo.com/ninechapter
官网: www.jiuzhang.com

Copyright © www.jiuzhang.com



谢谢大家

Copyright © www.jiuzhang.com