2019刷题

Day7

1.来排个序

你的任务是对给定的字符串进行排序。字符串中的每个单词都包含一个数字。此数字是单词在结果中应具有的位置。注意：数字可以是1到9.因此1将是第一个单词（不是0）。

如果输入字符串为空，则返回空字符串。输入String中的单词只包含有效的连续数字。

例子:

```
"is2 Thi1s T4est 3a"  -->  "Thi1s is2 3a T4est"
"4of Fo1r pe6ople g3ood th5e the2"  -->  "Fo1r the2 g3ood 4of th5e pe6ople"
""  -->  ""
```

测试用例:

```
Test.assert_equals(order("is2 Thi1s T4est 3a"), "Thi1s is2 3a T4est")
Test.assert_equals(order("4of Fo1r pe6ople g3ood th5e the2"), "Fo1r the2 g3ood 4of th5e pe6ople")
Test.assert_equals(order(""), "")
```

1.1 菜鸟解法

```
def order(text):
    if len(text)==0:return ""
    arr=text.split()
    arr_index=[re.sub('[^\d+]','',each) for each in arr]
    new_arr=[each[1] for each in sorted(zip(arr_index,arr),key=lambda x:x[0])]
    return (' '.join(new_arr))
```

1.2 高手解法1

```
def order(words):
  return ' '.join(sorted(words.split(), key=lambda w:sorted(w)))
```

点评这个非常巧妙，利用数字和字母组合的字符串，有一个技巧数字总是排序在字母前面。

例如:

```
s='1ad9'
print (sorted(s))
>>['1', '9', 'a', 'd']
为啥数字排序排在字母前面，因为1和字母的ascii的码 ，数字的小，字母的打
print ({c:ord(c) for c in s})
>>{'1': 49, 'a': 97, 'd': 100, '9': 57}
```

1.3 高手解法2

```
def extract_number(word):
    for l in word:
        if l.isdigit(): return int(l)
    return None

def order(sentence):
    return ' '.join(sorted(sentence.split(), key=extract_number))
```

点评这个毕竟通俗一点，把key里面原来用lambda写的改成了extract_number函数

1.4 答案

https://www.codewars.com/kata/your-order-please

2.点赞

你可能知道Facebook和其他网页上的“点赞”系统。人们可以“喜欢”博客文章，图片或其他项目。我们想要创建应该在这样的项目旁边显示的文本。

实现一个函数`likes :: [String] -> String`，它必须包含输入数组，包含喜欢项目的人的名字。它必须返回显示文本，如示例所示：

例如:

```
likes [] // must be "no one likes this"
likes ["Peter"] // must be "Peter likes this"
likes ["Jacob", "Alex"] // must be "Jacob and Alex like this"
likes ["Max", "John", "Mark"] // must be "Max, John and Mark like this"
likes ["Alex", "Jacob", "Mark", "Max"] // must be "Alex, Jacob and 2 others
like this"
```

对于4个或更多名称，数字`and 2 others`只会增加。

2.1 菜鸟解法

```
def likes(names):
    if len(names) == 0:
        return "no one likes this"
    elif len(names) == 1:
        return "%s likes this" % names[0]
    elif len(names) == 2:
        return "%s and %s like this" % (names[0], names[1])
    elif len(names) == 3:
        return "%s, %s and %s like this" % (names[0], names[1], names[2])
    else:
        return "%s, %s and %s others like this" % (names[0], names[1],
len(names)-2)
```

2.2 高手解法1

```
def likes(names):
    n = len(names)
    return {
        0: 'no one likes this',
        1: '{} likes this',
        2: '{} and {} like this',
        3: '{}, {} and {} like this',
        4: '{}, {} and {others} others like this'
    }[min(4, n)].format(*names[:3], others=n-2)
```

2.3 高手解法2

```
def likes(names):
    formats = {
            0: "no one likes this",
            1: "{} likes this",
            2: "{} and {} like this",
            3: "{}, {} and {} like this",
            4: "{}, {} and {others} others like this"
        }
    n = len(names)
    return formats[min(n,4)].format(*names, others=n-2)
```

点评上面这些解法很巧妙:

1.首先利用字典来返回字符串结构，非常巧妙。

返回value,返回各种字符串

2.字典的遍历利用min(4,n)

3.format里面里面关键字定位,非常易扩展

4.*['aa'],星号和列表连用的做法很独特

2.4 答案

https://www.codewars.com/kata/who-likes-it/train/python

Day8

1.计算重复字母出现的次数

编写一个函数，该函数将返回在输入字符串中出现多次(**不同的不区分大小写的**)字母字符和数字的计数。可以假定输入字符串仅包含字母（大写和小写）和数字。

例如:

```
"abcde" -> 0 # no characters repeats more than once
"aabbcde" -> 2 # 'a' and 'b'
"aabBcde" -> 2 # 'a' occurs twice and 'b' twice (`b` and `B`)
"indivisibility" -> 1 # 'i' occurs six times
"Indivisibilities" -> 2 # 'i' occurs seven times and 's' occurs twice
"aA11" -> 2 # 'a' and '1'
"ABBA" -> 2 # 'A' and 'B' each occur twice
```

代码:

```
def duplicate_count(text):
    # Your code goes here
```

测试用例:

```
test.assert_equals(duplicate_count("abcde"), 0)
test.assert_equals(duplicate_count("abcdea"), 1)
test.assert_equals(duplicate_count("indivisibility"), 1)
```

1.高手解法1

```
def duplicate_count(s):
  return len([c for c in set(s.lower()) if s.lower().count(c)>1])
```

1.2高手解法2

```
from collections import Counter
def duplicate_count(text):
    return sum(1 for c, n in Counter(text.lower()).iteritems() if n > 1)
```

点评,这类题目多半都是利用推到列表进行

1.3答案

https://www.codewars.com/kata/counting-duplicates/train/python

2.把0挪到队尾

编写一个算法，该算法采用数组并将所有零移动到最后，保留其他元素的顺序。

例如：

move_zeros([false,1,0,1,2,0,1,3,"a"]) # returns[false,1,1,2,1,3,"a",0,0]

代码:

```
def move_zeros(array):
    #your code here
```

测试用例:

```
Test.describe("Basic tests")
Test.assert_equals(move_zeros([1,2,0,1,0,1,0,3,0,1]),[ 1, 2, 1, 1, 3, 1, 0, 0,
0, 0 ])
Test.assert_equals(move_zeros([9,0.0,0,9,1,2,0,1,0,1,0.0,3,0,1,9,0,0,0,0,9]),
[9,9,1,2,1,1,3,1,9,9,0,0,0,0,0,0,0,0,0,0])
Test.assert_equals(move_zeros(["a",0,0,"b","c","d",0,1,0,1,0,3,0,1,9,0,0,0,0,9])
,["a","b","c","d",1,1,3,1,9,9,0,0,0,0,0,0,0,0,0,0])
Test.assert_equals(move_zeros(["a",0,0,"b",None,"c","d",0,1,False,0,1,0,3,
[],0,1,9,0,0,{},0,0,9]),["a","b",None,"c","d",1,False,1,3,[],1,9,
{},9,0,0,0,0,0,0,0,0,0,0])
Test.assert_equals(move_zeros([0,1,None,2,False,1,0]),[1,None,2,False,1,0,0])
Test.assert_equals(move_zeros(["a","b"]),["a","b"])
Test.assert_equals(move_zeros(["a"]),["a"])
Test.assert_equals(move_zeros([0,0]),[0,0])
Test.assert_equals(move_zeros([0]),[0])
Test.assert_equals(move_zeros([False]),[False])
Test.assert_equals(move_zeros([]),[])
```

2.1菜鸟解法

```
def move_zeros(chars):
    head=[]
    tail=[]
    for n in chars:
        if n==0 and len(str(n))<5:
            tail.append(n)
        else:
            head.append(n)

    head.extend(tail)
    return head
```

这道题目的难度在于False==0, 0.0 也是==0,菜鸟的解法只是利用str(n)然后取长度来过滤掉Flase 这样的情况

2.2高手解法1

```
def move_zeros(array):
    return sorted(array, key=lambda x: x==0 and type(x) is not bool)
```

2.3高手解法2

```
def move_zeros(arr):
    l = [i for i in arr if isinstance(i, bool) or i!=0]
    return l+[0]*(len(arr)-len(l))
```

2.3高手解法3

```
def move_zeros(array):
    return sorted(array, key=lambda x: x == 0 and x is not False)
```

高手解法 是利用type和isinstane来做类型判断

2.4答案

https://www.codewars.com/kata/moving-zeros-to-the-end/train/python

Day9

1.创建一个电话号码

编写一个接受10个整数（0到9之间）数组的函数，它以电话号码的形式返回这些数字的字符串。

例如：

create_phone_number([1, 2, 3, 4, 5, 6, 7, 8, 9, 0]) # => returns "(123) 456-7890"

测试用例：

```
Test.describe("Basic tests")
Test.assert_equals(create_phone_number([1, 2, 3, 4, 5, 6, 7, 8, 9, 0]), "(123)
456-7890")
Test.assert_equals(create_phone_number([1, 1, 1, 1, 1, 1, 1, 1, 1, 1]), "(111)
111-1111")
Test.assert_equals(create_phone_number([1, 2, 3, 4, 5, 6, 7, 8, 9, 0]), "(123)
456-7890")
Test.assert_equals(create_phone_number([0, 2, 3, 0, 5, 6, 0, 8, 9, 0]), "(023)
056-0890")
Test.assert_equals(create_phone_number([0, 0, 0, 0, 0, 0, 0, 0, 0, 0]), "(000)
000-0000")
```

1.1菜鸟解法

```
def create_phone_number(nums):
    nums.insert(0, '(')
    nums.insert(4, ') ')
    nums.insert(8,'-')
    return ''.join(map(str,nums))
```

1.2高手解法

```
def create_phone_number(n):
  return "({}{}{}) {}{}{}-{}{}{}{}".format(*n)
```

1.3高手解法2

```
def create_phone_number(n):
    n = ''.join(map(str,n))
    return '(%s) %s-%s'%(n[:3], n[3:6], n[6:])
```

1.4答案

https://www.codewars.com/kata/create-phone-number/train/python

2.人性化的可读性时间

编写一个函数，它以非负整数（秒）作为输入，并以人类可读的格式返回时间（`HH:MM:SS`）

- `HH` =小时，填充到2位数，范围：00 - 99
- `MM` =分钟，填充到2位数，范围：00 - 59
- `SS` =秒，填充到2位数，范围：00 - 59

最长时间永远不会超过359999（`99:59:59`）

测试用例:

```
Test.assert_equals(make_readable(0), "00:00:00")
Test.assert_equals(make_readable(5), "00:00:05")
Test.assert_equals(make_readable(60), "00:01:00")
Test.assert_equals(make_readable(86399), "23:59:59")
Test.assert_equals(make_readable(359999), "99:59:59")
```

1.1高手解法1

```
def make_readable(s):
    return '{:02}:{:02}:{:02}'.format(s / 3600, s / 60 % 60, s % 60)
```

1.2高手解法2

```
def make_readable(seconds):
    h= seconds/60**2
    m= (seconds%60**2)/60
    s= (seconds%60**2%60)
    return "%02d:%02d:%02d" % (h, m, s)
```

1.3答案

https://www.codewars.com/kata/human-readable-time/train/python

Day10

1.反转一个数字

给定一个数字，写一个函数来输出其反向数字。（例如，给出123答案是321）

数字应该保留他们的标志; 即反转时负数仍应为负数。

比如:

```
 123 ->  321
-456 -> -654
1000 ->    1
```

测试用例:

```
Test.assert_equals(reverse_number(123), 321)
Test.assert_equals(reverse_number(-123), -321)
Test.assert_equals(reverse_number(1000), 1)
Test.assert_equals(reverse_number(4321234), 4321234)
Test.assert_equals(reverse_number(5), 5)
Test.assert_equals(reverse_number(0), 0)
Test.assert_equals(reverse_number(98989898), 89898989)
```

1.1 菜鸟解法

```python
def reverse_number(n):
    if n>0:
        num_chars = list(str(n))[::-1]
        return int(''.join(num_chars))
    elif n<0:
        num_chars = list(str(abs(n)))[::-1]
        return int('-'+''.join(num_chars))
    else:
        return n
```

1.2 高手解法1

```python
def reverseNumber(n):
    if n < 0: return -reverseNumber(-n)
    return int(str(n)[::-1])
```

1.3 高手解法2

```python
def reverseNumber(n):
    return int(str(abs(n))[::-1]) * (-1 if n < 0 else 1)
```

https://www.codewars.com/kata/reverse-a-number/train/python

2.检查ip

编写一种算法，以十进制格式识别有效的IPv4地址。如果IP由四个八位字节组成，其值在`0`和之间`255`，则应视为有效。该函数的输入保证是单个字符串。例子：有效输入

```
1.2.3.4
123.45.67.89
```

输入无效：

```
1.2.3
1.2.3.4.5
123.456.78.90
123.045.067.089
```

请注意，前导零（例如`01.02.03.04`）被视为无效。测试用例:

```
Test.assert_equals(is_valid_IP('12.255.56.1'),     True)
Test.assert_equals(is_valid_IP(''),                False)
Test.assert_equals(is_valid_IP('abc.def.ghi.jkl'), False)
Test.assert_equals(is_valid_IP('123.456.789.0'),   False)
Test.assert_equals(is_valid_IP('12.34.56'),        False)
Test.assert_equals(is_valid_IP('12.34.56 .1'),     False)
Test.assert_equals(is_valid_IP('12.34.56.-1'),     False)
Test.assert_equals(is_valid_IP('123.045.067.089'), False)
Test.assert_equals(is_valid_IP('127.1.1.0'),        True)
Test.assert_equals(is_valid_IP('0.0.0.0'),          True)
Test.assert_equals(is_valid_IP('0.34.82.53'),       True)
Test.assert_equals(is_valid_IP('192.168.1.300'),   False)
```

1.1菜鸟解法1

```
import string
def verify_each(ip_addr):
    if len(ip_addr)>1 and ip_addr.startswith('0'):return  False
    for c in ip_addr:
        if c in string.digits:
            continue
        elif c in string.ascii_letters:
            return False
        else:
            return False

    else:
        return True if int(ip_addr)<=255 and int(ip_addr)>=0 else False

def is_valid_IP(ip_str):
    ip_list=ip_str.split('.')
    if len(ip_list)!=4:
        return False
    res=[verify_each(each) for each in ip_list]
    return True if all(res) else False
```

1.2 高手解法1:

```
def is_valid_IP(strng):
  lst = strng.split('.')
  passed = 0
  for sect in lst:
    if sect.isdigit():
      if sect[0] != '0':
        if 0 < int(sect) <= 255:
          passed += 1
  return passed == 4
```

点评 高手解法还是很巧妙的，利用字符串的内置函数isdigit(),来判断这个字符串是不是全数字。更巧妙的是，用passed的len来记录IP长度，最后 return passed==4 一剑封喉。这个确实可以借鉴,如果有搜索判断的类似的函数!

1.3 高手解法2:

```
import re
def is_valid_IP(strng):
    return bool(re.match(r'^((\d{1,2}|1\d{2}|2[0-4]\d|25[0-5])(\.(?!$)|$)){4}(?
=$)',strng))
```

点评 字符串的过滤，匹配都离不开正则，学好正则，学会正则有的时候可以借巧力，提高效率。

1.4 高手解法3:

```
import socket

def is_valid_IP(addr):
    try:
        socket.inet_pton(socket.AF_INET, addr)
        return True
    except socket.error:
        return False
```

点评 直接用socket模块里面的inet_pton来检查字符串能否转换成IP地址，然后用异常进行捕获，这招确实省事，高招!

1.5答案

https://www.codewars.com/kata/ip-validation

Day11

1.用函数计算

这次我们想用函数编写计算并得到结果。我们来看看一些例子：

例如：

```
seven(times(five())); // must return 35
four(plus(nine())); // must return 13
eight(minus(three())); // must return 5
six(dividedBy(two())); // must return 3
```

测试用例：

```
Test.describe('Basic Tests')
Test.assert_equals(seven(times(five())), 35)
Test.assert_equals(four(plus(nine())), 13)
Test.assert_equals(eight(minus(three())), 5)
Test.assert_equals(six(divided_by(two())), 3)
```

1.1高手解法1

```
def zero(f = None): return 0 if not f else f(0)
def one(f = None): return 1 if not f else f(1)
def two(f = None): return 2 if not f else f(2)
def three(f = None): return 3 if not f else f(3)
def four(f = None): return 4 if not f else f(4)
def five(f = None): return 5 if not f else f(5)
def six(f = None): return 6 if not f else f(6)
def seven(f = None): return 7 if not f else f(7)
def eight(f = None): return 8 if not f else f(8)
def nine(f = None): return 9 if not f else f(9)

def plus(y): return lambda x: x+y
def minus(y): return lambda x: x-y
def times(y): return lambda  x: x*y
def divided_by(y): return lambda  x: x/y
```

点评 能用lambda来设计嵌套函数，都是高手，这解法非常巧妙。比如：def plus(y): return lambda x: x+y

正常的都是函数返回值，它这个是通过函数返回一个函数地址(用匿名函数生成的)

1.2高手解法2

```
def zero(arg=""):  return eval("0" + arg)
def one(arg=""):   return eval("1" + arg)
def two(arg=""):   return "2" + arg
def three(arg=""): return eval("3" + arg)
def four(arg=""):  return eval("4" + arg)
def five(arg=""):  return eval("5" + arg)
def six(arg=""):   return eval("6" + arg)
def seven(arg=""): return eval("7" + arg)
def eight(arg=""): return eval("8" + arg)
def nine(arg=""):  return eval("9" + arg)

def plus(n):       return "+%s" % n
def minus(n):      return "-%s" % n
def times(n):      return "*%s" % n
def divided_by(n): return "/%s" % n
```

点评 先构造表达式，然后利用eval来求值。确实很简洁！

1.3 答案

https://www.codewars.com/kata/calculating-with-functions/train/python

Day12

1.电话目录

约翰将他的旧个人电话簿备份为文本文件。在文件中的每一行，他能找到的电话号码（如格式化`+X-abc-def-ghij`其中X代表一个或两个数字），与相应的名称`<`，并`>`和地址。

不幸的是，一切都是混合的，事情并不总是在同一个顺序; 线条的一部分混杂着非字母数字字符（电话号码和姓名除外）。

John的电话簿行示例：

```
"/+1-541-754-3010 156 Alphand_St. <J Steeve>\n"
" 133, Green, Rd. <E Kustur> NY-56423 ;+1-541-914-3010!\n"
"<Anastasia> +48-421-674-8974 Via Quirinal Roma\n"
```

你能帮助约翰做一个程序吗，根据他的电话簿和电话号码的行，返回一个这个数字的字符串：`"Phone => num, Name => name, Address => adress"`

例子：

```
s = "/+1-541-754-3010 156 Alphand_St. <J Steeve>\n 133, Green, Rd. <E Kustur>
NY-56423 ;+1-541-914-3010!\n"

phone(s, "1-541-754-3010") should return "Phone => 1-541-754-3010, Name => J
Steeve, Address => 156 Alphand St."
```

测试用例：

```
dr = ("/+1-541-754-3010 156 Alphand_St. <J Steeve>\n 133, Green, Rd. <E Kustur>
NY-56423 ;+1-541-914-3010;\n"
"+1-541-984-3012 <P Reed> /PO Box 530; Pollocksville, NC-28573\n :+1-321-512-
2222 <Paul Dive> Sequoia Alley PQ-67209\n"
"+1-741-984-3090 <Peter Reedgrave> _Chicago\n :+1-921-333-2222 <Anna Stevens>
Haramburu_Street AA-67209\n"
"+1-111-544-8973 <Peter Pan> LA\n +1-921-512-2222 <Wilfrid Stevens> Wild Street
AA-67209\n"
"<Peter Gone> LA ?+1-121-544-8974 \n <R Steell> Quora Street AB-47209 +1-481-
512-2222!\n"
"<Arthur Clarke> San Antonio $+1-121-504-8974 TT-45120\n <Ray Chandler> Teliman
Pk. !+1-681-512-2222! AB-47209,\n"
"<Sophia Loren> +1-421-674-8974 Bern TP-46017\n <Peter O'Brien> High Street +1-
908-512-2222; CC-47209\n"
"<Anastasia> +48-421-674-8974 Via Quirinal Roma\n <P Salinger> Main Street, +1-
098-512-2222, Denver\n"
"<C Powel> *+19-421-674-8974 Chateau des Fosses Strasbourg F-68000\n <Bernard
Deltheil> +1-498-512-2222; Mount Av.  Eldorado\n"
"+1-099-500-8000 <Peter Crush> Labrador Bd.\n +1-931-512-4855 <William Saurin>
Bison Street CQ-23071\n"
"<P Salinge> Main Street, +1-098-512-2222, Denve\n")


Test.describe("phone")
Test.it("Basic tests")
testing(phone(dr, "48-421-674-8974"), "Phone => 48-421-674-8974, Name =>
Anastasia, Address => Via Quirinal Roma")
testing(phone(dr, "1-921-512-2222"), "Phone => 1-921-512-2222, Name => Wilfrid
Stevens, Address => Wild Street AA-67209")
testing(phone(dr, "1-908-512-2222"), "Phone => 1-908-512-2222, Name => Peter
O'Brien, Address => High Street CC-47209")
testing(phone(dr, "1-541-754-3010"), "Phone => 1-541-754-3010, Name => J Steeve,
Address => 156 Alphand St.")
testing(phone(dr, "1-121-504-8974"), "Phone => 1-121-504-8974, Name => Arthur
Clarke, Address => San Antonio TT-45120")
testing(phone(dr, "1-498-512-2222"), "Phone => 1-498-512-2222, Name => Bernard
Deltheil, Address => Mount Av. Eldorado")
testing(phone(dr, "1-098-512-2222"), "Error => Too many people: 1-098-512-2222")
testing(phone(dr, "5-555-555-5555"), "Error => Not found: 5-555-555-5555")
```

1.高手解法1

```
import re
def phone(string, num):
    count = len(re.findall(r'\+'+num,string))
    if count > 1:return f"Error => Too many people: {num}"
    elif count == 0:return f"Error => Not found: {num}"
    for each in string.splitlines():
        number = re.search(r'\+(\d{1,2}-\d{3}-\d{3}-\d{4})',each).group(1)
        if number == num:
            name = re.search(r'<(.*?)>',each).group(1)
            address = re.sub(r' +'," ",re.sub(r'[^0-9A-Za-z\.-]',"
",each.replace(number,"").replace(name,""))).strip()
            return f"Phone => {num}, Name => {name}, Address => {address}"
```

2.高手解法2

```
import re
class Phonebook(object):
    def __init__(self):
        self.people = []

    def add_person(self, person):
        self.people.append(person)

    def __iter__(self):
        for elem in self.people:
            yield elem

    def find_phone(self, phone ):
        found = []
        for person in self.people:
            if str(person.phone) == str(phone):
                found.append(person)
        return found

class Person(object):
    def __init__(self, name, phone=None, address=None):
        self.name = name
        self.phone = phone
        self.address = address

    def add_phone(self, number):
        self.phone = number
```

```
    def add_address(self, address):
        self.address = address

    def show(self):
        print("Data:")
        s = 'name: %s \n' % self.name
        if self.phone is not None:
            s += 'general phone:   %s\n' % self.phone
        if self.address is not None:
            s += 'address address:  %s\n' % self.address
        print (s)

def phone(strng, num):
# Working with the given data:
    phonebook = Phonebook()
    datas = strng.split("\n")
    datas = [data for data in datas if data]

    for data in datas:
        tel = re.findall("([1-9]*[1-9]-[0-9]\d{2}-[0-9]\d{2}-[0-9]\d{3})", data)
[0]
        name = re.findall("(?<=\<)(.*?)(?=\>)", data)[0]
        address = data.replace(tel,"")
        address = address.replace("<" + name + ">", "")
        address = re.sub('[^A-Za-z0-9-" ."]+', ' ', address)
        address = " ".join(address.split())
    # Now with the data clean, add it to a person:
        person = Person(name=name, phone=tel, address=address)
    # And add the person to phonebook
        phonebook.add_person(person)
    # Find the person by the Phone:
    results = phonebook.find_phone(num)
    if len(results) == 1:
        return ("Phone => {}, Name => {}, Address =>
{}".format(results[0].phone, results[0].name, results[0].address))
    if len(results) > 1:
        return "Error => Too many people: {}".format(num)
    if len(results) == 0:
        return "Error => Not found: {}".format(num)
```

点评，两种解法都是用正则，第一种非常简洁，第二种用类去封装数据结构，扩张性会好很多！而且相对更清晰一点。

1.3 答案

https://www.codewars.com/kata/phone-directory