2019刷题

Day1

1.有序字符数

计算每个字符的出现次数，并按照出现的顺序将其作为元组列表返回。

例如给你一个字符串"abracadabra"，统计里面的字符按照下面的格式输出:

```
ordered_count("abracadabra") == [('a', 5), ('b', 2), ('r', 2), ('c', 1), ('d', 1)]
```

测试用例:

```
test.describe("Basic Tests")

tests = (
    ('abracadabra', [('a', 5), ('b', 2), ('r', 2), ('c', 1), ('d', 1)]),
    ('Code Wars', [('C', 1), ('o', 1), ('d', 1), ('e', 1), (' ', 1), ('W', 1), ('a', 1), ('r', 1), ('s', 1)])
)

for t in tests:
    inp, exp = t
    test.assert_equals(ordered_count(inp), exp)
```

先不要看答案,自己动手思考几分钟....

1.1 高手解法1

```
from collections import OrderedDict, Counter

class OrderedCounter(Counter, OrderedDict):
    pass

def ordered_count(seq):
    m=OrderedCounter(seq)
    return list(OrderedCounter(seq).items())
```

1.2 高手解法2

```
from collections import Counter

def ordered_count(input):
    return list(Counter(input).items())
```

1.3 菜鸟解法

```
from collections import OrderedDict

def ordered_count(input):
  counts = OrderedDict()
  for char in input:
    counts[char] = counts.get(char, 0) + 1

  return list(counts.items())
```

1.4点评

这道题因为要考虑有序，就是按照字符串里面的字符的顺序，同时要统计字符出现的次数。第一种解法非常巧妙，里面两个类的继承，同时继承了collection 模块里面的OrderedDict和Counter两个子类。很妙！

1.5 答案

Link:https://www.codewars.com/kata/ordered-count-of-characters/train/python

2.缩写双字名称

编写一个函数将名称转换为首字母。这个kata严格地用两个词，它们之间有一个空格。

输出应该是两个大写字母，并用点分隔它们。

它应该如下所示：

```
Sam Harris` => `S.H
Patrick Feeney` => `P.F
```

测试用例:

```
def abbrevName(name):
    #your code

Test.assert_equals(abbrevName("Sam Harris"), "S.H");
Test.assert_equals(abbrevName("Patrick Feenan"), "P.F");
Test.assert_equals(abbrevName("Evan Cole"), "E.C");
Test.assert_equals(abbrevName("P Favuzzi"), "P.F");
Test.assert_equals(abbrevName("David Mendieta"), "D.M");
```


先不要看答案,自己动手思考几分钟....

2.1 高手解法

```
def abbrevName(name):
    return '.'.join(w[0] for w in name.split()).upper()
```

2.2 高手解法


```
abbrevName = lambda name: ".".join(e[0].upper() for e in name.split())
```

2.3点评

几种高手的解法都是利用类似推导列表进行快速处理，然后用字符串自带的upper()函数进行全大写。
建议要熟悉常用的字符串的内置函数，比如:

```
''.lower()
''.upper()
''.count()
''.startswith()
''.capitalize()
```

2.4答案

https://www.codewars.com/kata/abbreviate-a-two-word-name/train/python

Day2

1.求表达式中正数的和

你得到一组数字，返回所有正数的总和。

示例`[1,-4,7,12]`=>`1 + 7 + 12 = 20`

注意：如果没有要求的总和，则默认值为`0`。

测试用例:

```
Test.describe("positive_sum")

Test.it("works for some examples")
Test.assert_equals(positive_sum([1,2,3,4,5]),15)
Test.assert_equals(positive_sum([1,-2,3,4,5]),13)
Test.assert_equals(positive_sum([-1,2,3,4,-5]),9)

Test.it("returns 0 when array is empty")
```


先不要看答案,自己动手思考几分钟....

1.1 高手解法

```
def positive_sum(nums):
    return sum([n for n in nums if n >0])
```

1.2点评

大部分都是利用sum这用的内置函数来求和。建议了解sum(),min(),max()这些基本的内置函数!

2.拼接字符

根据下面的表达式设计一个函数:

accum("abcd") -> "A-Bb-Ccc-Dddd" accum("RqaEzty") -> "R-Qq-Aaa-Eeee-Zzzzz-Ttttt-Yyyyyyy" accum("cwAt") -> "C-Ww-Aaa-Tttt"

```
def accum(s):
    # your code
```

测试用例:

```
Test.describe("accum")
Test.it("Basic tests")
Test.assert_equals(accum("ZpglnRxqenU"), "Z-Pp-Ggg-Llll-Nnnnn-Rrrrrr-Xxxxxxx-Qqqqqqqq-Eeeeeeeee-Nnnnnnnnnn-Uuuuuuuuuuu")
Test.assert_equals(accum("NyffsGeyylB"), "N-Yy-Fff-Ffff-Sssss-Ggggggg-Eeeeeee-Yyyyyyyy-Yyyyyyyyy-Lllllllll-Bbbbbbbbbb")
Test.assert_equals(accum("MjtkuBovqrU"), "M-Jj-Ttt-Kkkk-Uuuuu-Bbbbbb-Ooooooo-Vvvvvvvv-Qqqqqqqqq-Rrrrrrrrr-Uuuuuuuuuu")
Test.assert_equals(accum("EvidjUnokmM"), "E-Vv-Iii-Dddd-Jjjjj-Uuuuuu-Nnnnnnn-Oooooooo-Kkkkkkkkk-Mmmmmmmmmm-Mmmmmmmmmmm")
Test.assert_equals(accum("HbideVbxncC"), "H-Bb-Iii-Dddd-Eeeee-Vvvvvv-Bbbbbbb-Xxxxxxxx-Nnnnnnnnn-Ccccccccc-Ccccccccccc")
```

先不要看答案,自己动手思考几分钟....

2.1高手解法

```
def accum(words):
    res=[c*(index+1) for index,c in enumerate(words)]
    return '-'.join(map(lambda x:x.capitalize(),res))
```

2.2高手解法

```
def accum(s):
    return '-'.join(c.upper() + c.lower() * i for i, c in enumerate(s))
```

2.3点评

这道题目需要跟字符在字符串中序列号有关，高手的解法几乎千篇一律的都用到了enumerate 这个特性，这个是Python里面非常强的技巧，一定要掌握。

2.4答案

https://www.codewars.com/kata/mumbling

Day3

1.判断奇数还是偶数

创建一个函数，它以整数作为参数，对偶数返回“偶数”，对奇数返回“奇数”。

代码:

```
def even_or_odd(number):
        #your code
```

测试用例

```
test.expect(even_or_odd(2) == "Even")
test.expect(even_or_odd(0) == "Even")
test.expect(even_or_odd(7) == "Odd")
test.expect(even_or_odd(1) == "Odd")
test.expect(even_or_odd(-1) == "Odd")
```

先不要看答案,自己动手思考几分钟....

1.1高手解法

```
def even_or_odd(number):
  return ["Even", "Odd"][number % 2]
```

1.2高手解法

```
def even_or_odd(number):
  return 'Even' if number % 2 == 0 else 'Odd'
```


1.3点评

上面两种解法，尤其是第一种解法很巧妙，利用一个列表放置好现成的结果，用num%==2进行列表切片选择。非常巧妙，大部分都是想到第二种解法。第二种解法用一行代码来代替if /else这种的写法非常老道，尤其是在很多源码库里面非常常见，建议熟练掌握。

1.4答案

https://www.codewars.com/kata/even-or-odd/train/python

2.检查信用卡

给定一个信用卡号码，我们可以通过一些基本知识来确定发行人/供应商是谁。

完成`get_issuer()`将使用下面显示的值的功能来确定给定卡号的发卡机构。如果数字不匹配，则该函数应返回该字符串`Unknown`。

Card Type	Begins With	Number Length
AMEX	34 or 37	15
Discover	6011	16
Mastercard	51, 52, 53, 54 or 55	16
VISA	4	13 or 16

测试用例:

```
get_issuer(4111111111111111) == "VISA"
get_issuer(4111111111111) == "VISA"
get_issuer(4012888888881881) == "VISA"
get_issuer(378282246310005) == "AMEX"
get_issuer(6011111111111117) == "Discover"
get_issuer(5105105105105100) == "Mastercard"
get_issuer(5105105105105106) == "Mastercard"
get_issuer(9111111111111111) == "Unknown"
```

先不要看答案,自己动手思考几分钟....

1.1菜鸟解法

```
mappings=[
    {'name':'AMEX','Begins':[34,37],"Number Length":15},
    {'name': 'Discover', 'Begins': [6011], "Number Length": 16},
    {'name': 'Mastercard', 'Begins': [51, 52, 53, 54 ,55], "Number Length": 16},
    {'name': 'VISA', 'Begins': [4], "Number Length": [13,16]}
]
def get_issuer(num):
    for mapping in mappings:
        flags=[str(num).startswith(str(x)) for x in mapping['Begins']]
        if any(flags):
            return mapping['name']
    else:
        return 'Unknown'
```


菜鸟解法就是比较平铺直叙，数据结构上直接就用列表和字典构建，然后遍历列表进行匹配即可！

1.2高手解法1

```
ISSUERS = {
    ((34, 37), (15,)): 'AMEX',
    ((6011,), (16,)): 'Discover',
    ((51, 52, 53, 54, 55), (16,)): 'Mastercard',
    ((4,), (13, 16)): 'VISA',
}

def get_issuer(number):
    return next((issuer for (starts, lengths), issuer in ISSUERS.items()
        if str(number).startswith(tuple(map(str, starts))) and len(str(number))
in lengths), 'Unknown')
```

点评这种解法非常巧妙，函数体内一行搞定，数据结构设计的时候就很妙，跟上面的不太一样。

第一，利用字典数据结构，字典是hash结构，遍历速度快，比列表要快n倍，性能上高。

第二，字典的Key设计很讲究，整体的思路也是遍历字典，但是字典的Key使用元组包裹，想到用元组作为key，这样的写法都是老手所为！

也就是信用卡的开头数组，但是字符串匹配的开头的时候如果用startswith函数，一定要是字符串类型，所以又有的map函数把数字转换为字符。

第三，利用next函数，一般next函数都是跟生成器相关的，就是不断的循环可迭代的对象。这里用next来迭代整个字典，如果遍历之后找不到，显示Unknow. 这样的写法极其老道，可能写成下面的做法，更适合理解:

```
def get_issuer(number):
    for (starts,lenghs),issuer in ISSUERS.items():
        if str(number).startswith(tuple(map(str,starts))) and len(str(number))
in lenghs:
            return issuer

    return 'Unknown'
```

1.3 高手解法2

```
def get_issuer(number):
    s = str(number)
    return ("AMEX"       if len(s)==15 and s[:2] in ("34","37") else
            "Discover"   if len(s)==16 and s.startswith("6011") else
            "Mastercard" if len(s)==16 and s[0]=="5" and s[1] in "12345" else
            "VISA"       if len(s) in [13,16] and s[0]=='4' else
            "Unknown")
```


点评，这return里面大作文章这样的用法也是比较老道的，而且用了5层的if else if else ,算是5目运算了!

大部分return里面用这样的写法:

```
return [{
    'aa':student.name
    'bb':student.age
} for student in students]
```

1.4高手解法4

```
from collections import namedtuple


Issuer = namedtuple('Issuer', 'name, begins, length')

def get_issuer(number):
    issuers = [
        Issuer('AMEX', [34, 37], [15]),
        Issuer('Discover', [6011], [16]),
```

```
        Issuer('Mastercard', [51, 52, 53, 54, 55], [16]),
        Issuer('VISA', [4], [13, 16]),
    ]
    number_str = str(number)
    for issuer in issuers:
        if any([number_str.startswith(str(n)) for n in issuer.begins]) and
len(number_str) in issuer.length:
            return issuer.name
    return 'Unknown'
```

点评，利用namedtuple这样的轻量级类完成是最好的，直观而且简洁！这种数据结构的设计非常不错。

把搜索的结果放在一个列表里面，结果是True/False ,巧妙的用了any来快速处理！省去了再次遍历的麻烦。

1.5 高手解法5

```
import re

def get_issuer(number):
    for n, r in ('AMEX', '3[47].{13}'), ('Discover', '6011.{12}'),
('Mastercard', '5[1-5].{14}'), ('VISA', '4(...){4,5}'):
        if re.fullmatch(r, str(number)):
            return n
    return 'Unknown'
```


用正则去匹配非常简洁，上面的匹配需要2个条件一个是startswith和len(num) in length,而正则只需要用[47].{13}这样几个字符串就描述了上面复杂的表达式，确实厉害！

1.5答案

https://www.codewars.com/kata/credit-card-issuer-checking

Day4

1.未使用的最小的ID

你需要管理大量数据，使用零基础和非负ID来使每个数据项都是唯一的！

因此，需要一个方法来计算下一个新数据项返回**最小的未使用ID**...

注意：给定的已使用ID数组可能未排序。出于测试原因，可能存在重复的ID，但你无需查找或删除它们！

```
def next_id(arr):
    #your code here
```

测试用例:

```
Test.assert_equals(next_id([0,1,2,3,4,5,6,7,8,9,10]), 11)
Test.assert_equals(next_id([5,4,3,2,1]), 0)
Test.assert_equals(next_id([0,1,2,3,5]), 4)
Test.assert_equals(next_id([0,0,0,0,0,0]), 1)
Test.assert_equals(next_id([]), 0)
```

先不要看答案,自己动手思考几分钟....

1.1菜鸟解法

```
def next_id(array):
    if not array:
        return 0

    sorted_arr=set(sorted(array))
    array=set(range(0, max(array) + 1))
    gap=array-sorted_arr
    if gap:
        return min(gap)
    else:
        return 0 if 0 not in sorted_arr else max(sorted_arr)+1
```


1.2 高手解法1

```
def next_id(arr):
    t = 0
    while t in arr:
        t +=1
    return t
```


这个解法太巧妙了，有庖丁解牛的感觉！while的条件里面用t in arr,如果在里面就循环，直到最后一个列表最大元素，然后t+1返回，如果不在列表里面，直接返回最小的值！

1.3 高手解法2

```
def next_id(arr):
    return reduce(lambda acc, x: acc + 1 if x == acc else acc, sorted(arr), 0)
```

点评，三大高阶函数map,zip和reduce, 其中前两个比较通用，reduce 不常见，但是他可以返回列表里面的两个元素，有的时候会有妙用！注意的是Python3里面已经移除了reduce，需要用的话需要from functools import reduce

1.3 答案

https://www.codewars.com/kata/smallest-unused-id/train/python

2.获得中间字符

你会得到一个字符串,你需要写一个函数返回单词的中间字符。如果单词的长度为奇数，则返回中间字符。如果单词的长度是偶数，则返回中间2个字符。

例子:

```
Kata.getMiddle("test") should return "es"
Kata.getMiddle("testing") should return "t"
Kata.getMiddle("middle") should return "dd"
Kata.getMiddle("A") should return "A"
```

```
def get_middle(s):
    #your code here
```

测试用例:

```
Test.assert_equals(get_middle("test"),"es")
Test.assert_equals(get_middle("testing"),"t")
Test.assert_equals(get_middle("middle"),"dd")
Test.assert_equals(get_middle("A"),"A")
Test.assert_equals(get_middle("of"),"of")
```

先不要看答案,自己动手思考几分钟....

2.1 菜鸟解法

```
def getMiddle(words):
    if len(words)>2 and len(words)%2==0 :
        return words[int(len(words)/2)-1:int(len(words)/2+1)]
    if len(words)>2 and len(words)%2==1 :
        return words[int(len(words)/2)]
    elif len(words)==2:
        return words
    else:
        return words
```


2.2 高手解法

```
def get_middle(s):
   return s[(len(s)-1)/2:len(s)/2+1]
```


利用列表切片来解决，确实简洁而且巧妙！

2.3答案

https://www.codewars.com/kata/get-the-middle-character

Day5

1.统计元音字母

给一个字符串，统计里面的元音字母！我们给定的元音列表是:[a, e, i, o, u] ,输入的字符串只会是小写字母或者含有空格。

代码:

```
def getCount(inputStr):
    num_vowels = 0
    # your code here

    return num_vowels
```


测试用例:

```
test.assert_equals(getCount("abracadabra"), 5)
```

先不要看答案,自己动手思考几分钟....

1.1 菜鸟解法

```
def getCount(chars):
    return len([ c for c in chars if c in ['a', 'e', 'i', 'o', 'u' ]])
```

1.2 高手解法1


```
def getCount(inputStr):
    return sum(1 for let in inputStr if let in "aeiouAEIOU")
```

1.3 高手解法2

```
def getCount(s):
    return sum(c in 'aeiou' for c in s)
```

点评，非常巧妙的利用了sum和生成器的结合。注意(1 for let in inputStr if let in "aeiouAEIOU")其实是一个生成器，然后sum来统计总和。

1.4 高手解法3


```
import re
def getCount(str):
    return len(re.findall('[aeiou]', str, re.IGNORECASE))
```

点评，正则做字符串匹配的时候总是很简洁

https://www.codewars.com/kata/vowel-count/train/python

2.反转字符串

编写一个函数，它接受一个或多个单词的字符串，其中里面含五个或更多字母单词必须要反转。传入的字符串只包含字母和空格。仅当存在多个单词时才会包含空格。

比如:

```
spinWords( "Hey fellow warriors" ) => returns "Hey wollef sroirraw"
spinWords( "This is a test") => returns "This is a test"
spinWords( "This is another test" )=> returns "This is rehtona test"
```

测试用例：

```
test.assert_equals(spin_words("Welcome"), "emocleW")
```

先不要看答案,自己动手思考几分钟....

2.1 高手解法

```
def spin_words(words):
    return ' '.join([w[::-1] if len(w)>=5 else w for w in words.split()])
```

2.2 高手解法

```
import re
def spin_words(sentence):
    return re.sub(r"\w{5,}", lambda w: w.group(0)[::-1], sentence)
```


点评，两种解法几乎思想都是一样的，第一种利用len(w)来判断，第二种\w{5,}来判断长度为5.二者各有千秋，正则短小精悍，威力更大!

2.3答案

https://www.codewars.com/kata/stop-gninnips-my-sdrow/train/python

Day6

1.找到大写字母

写一个函数capitals()给你一串字符串，找到里面的大写字母，并返回它们的index.

比如：

```
capitals('CodEWaRs')输出为 [0,3,4,6]
```

代码:

```
def capitals(word):
    #your code here
```

测试用例:

```
Test.assert_equals( capitals('CodEWaRs'), [0,3,4,6] )
```

先不要看答案,自己动手思考几分钟....

1.1 高手解法1

```
def capitals(words):
    return [index for index , w in enumerate(words) if w.isupper()]
```

1.2 高手解法2

```
def capitals(word):
    return [i for i in range(len(word)) if word[i].isupper()]
```


点评，两种解法都是用推导列表+if 进行判断，有一个地方有注意，有人觉得可以用words.index(c) 这个会有问题，比如'helloworld'.index('o'),在取第二个o的时候，返回的是第一个o的index.所以在这题里面如果这样写，就就对！

```
return [words.index(w) for w in words if w.isupper()]
```

1.3 答案

https://www.codewars.com/kata/find-the-capitals-1/train/python

2.递归数字总和

写一个函数叫digital_root,给定一个数字，递归遍历数字从个位，十位，百位...以此相加计算总和。则以这种方式继续减少，直到产生一位数字。这仅适用于自然数,比如:

```
digital_root(16)
```

```
=> 1 + 6
=> 7

digital_root(942)
=> 9 + 4 + 2
=> 15 ...
=> 1 + 5
=> 6

digital_root(132189)
=> 1 + 3 + 2 + 1 + 8 + 9
=> 24 ...
=> 2 + 4
=> 6

digital_root(493193)
=> 4 + 9 + 3 + 1 + 9 + 3
=> 29 ...
=> 2 + 9
=> 11 ...
=> 1 + 1
=> 2
```

先不要看答案,自己动手思考几分钟....

2.1菜鸟解法

```
def digital_root(num):
    if len(str(num))==1:
        print ('=>',num)
        return num
    else :
        arr=list(str(num))
        print ('=>','+'.join(arr))
        s=sum(map(int,arr))
        if len(str(s))>1:
            print ('=>',s,'...')
        return digital_root(s)
```

2.2 高手解法1

```
def digital_root(n):
    return n if n < 10 else digital_root(sum(map(int,str(n))))
```

2.3 高手解法2

```
def digital_root(n):
    while n>9:
        n=sum(map(int,str(n)))
    return n
```


点评，前两种解法都是利用递归，只是高手解法写法更简洁！其实大部分的递归都可以用while来改写，第三种解法显然更牛！

2.4 答案

https://www.codewars.com/kata/sum-of-digits-slash-digital-root