Python 程序设计 实验 4：字符串与函数式编程

注意事项：

（1） 实验报告提交**截止日期：2022.05.04， 23:59pm**，迟交扣 20%，缺交 0 分。

（2） 实验报告内容包括：解决问题的思路与方法（如代码的解释）、遇到的问题以及收获（简单描述即可）、代码运行结果的展示。

（3） 实验报告提交方法：**blackboard。**

（4） 提交要求：实验报告+源代码，打包上传，命名：学号_姓名_实验报告_5。

（5） **禁止抄袭，一经发现 0 分处理（包括抄袭者和提供代码或实验报告者）!**

1. 删除字符：输入两个字符串 s 和 t，判断 s 能否在删除一些字符后得到 t。

2. 杨辉三角：编写函数 print_yanghui_triangle(N)，输出杨辉三角的前 N 行，要求使用 format 函数进行格式化输出。

例子：print_yanghui_triangle(10)，输出

```
                1
              1   1
            1   2   1
          1   3   3   1
        1   4   6   4   1
      1   5   10  10  5   1
    1   6   15  20  15  6   1
  1   7   21  35  35  21  7   1
1   8   28  56  70  56  28  8   1
1   9   36  84  126 126 84  36  9   1
```

3. 字符串推导式：我们之前学过列表推导式。例如，生成前 4 个奇数，

```
[2 * num - 1 for num in range(1,5)]   # 生成[1, 3, 5, 7]
```

仿照上面写法，使用推导式完成以下字符串操作:

1) ['apple', 'orange', 'pear'] -> ['A', 'O', 'P']

2) ['apple', 'orange', 'pear'] -> ['apple', 'pear']
3) ["TA_parth", "student_poohbear", "TA_michael", "TA_guido", "student_htiek"] -> ["parth", "michael", "guido"]
4) ['apple', 'orange', 'pear'] -> [('apple', 5), ('orange', 6), ('pear', 4)] #生成列表
5) ['apple', 'orange', 'pear'] -> {'apple': 5, 'orange': 6, 'pear': 4} #生成字典

4. 可读性等级

不同书的阅读群体不一样。例如，一本书中可能有许多很长的复杂单词；而较长的单词可能与较高的阅读水平有关。同样，较长的句子也可能与较高的阅读水平相关。研究者开发了许多可读性测试，给出了计算文本阅读水平的公式化过程。其中一个可读性测试是 Coleman Liau 指标：文本的 Coleman Liau 指标旨在划分理解文本所需的阅读水平等级。Coleman Liau 指标公式如下

```
index = 0.0588 * L - 0.296 * S - 15.8
```

其中，L 是文本中每 100 个单词的平均字母数，S 是文本中每 100 个单词的平均句子数。

考虑以下文本：

```
Congratulations! Today is your day. You're off to Great Places! You're off and away!
```

该文本有 65 个字母，4 个句子，14 个单词。文本中每 100 个单词的平均字母数是 L=65/14*100=464.29；文本中每 100 个单词的平均句子数是 S=4/14*100=28.57。代入 Coleman Liau 指标公式，并向最近的整数取整，我们得到可读性指数为 3 级。

我们将编写一个函数 readability，输入参数为字符串，返回可读性等级。

实现要求：

（1） 若计算结果小于 1，输出“Before Grade 1”；
（2） 若计算结果大于或等于 16，输出“Grade 16+”；
（3） 除（1）和（2）外，输出“Grade X”，X 为相应等级。

（4） 字母包括大写字母和小写字母（不考虑数字和标点符号）；

（5） 以空格分隔作为标准区分单词，如；

```
    It was a bright cold day in April, and the clocks were striking thirteen. Winston Smith, his chin nuzzled into his breast in an effort to escape the vile wind, slipped quickly through the glass doors of Victory Mansions, though not quickly enough to prevent a swirl of gritty dust from entering along with him.
    55 words
```

（6） 句号（.）、感叹号（!）或问号（?）表示句子的结尾。如

Mr. and Mrs. Dursley, of number four Privet Drive, were proud to say that they were perfectly normal, thank you very much.

3 sentences

（7） 可读性等级的参考例子见 readability.txt。

5. 函数的参数传递:定义一个简单的函数 sum 如下，

```
def sum(a, b, c):
    print("a=%d, b=%d, c=%d"%(a,b,c))
    print(a+b+c)
```

以下哪些语句是合法的，哪些是不合法的？分别输出什么？解释原因。

a) sum(*(1, 2, 3))
b) sum(1, *(2, 3))
c) sum(*(1,),b=2, 3)
d) sum(*(1,),b=2, c=3)
e) sum(*(1, 2),c=3)
f) sum(a=1, *(2, 3))
g) sum(b=1, *(2, 3))
h) sum(c=1, *(2, 3))

6. Lambda: 思考以下句子的输出，解释每句话意思，并通过实际测试验证自己想法。

(1) (lambda val: val % 2)(5)
(2) (lambda x, y: x ^ y)(3, 8)
(3) (lambda s: s.strip().lower()[1:4])(' PyTHon')

7. Map: 使用 map 语句将以下输入，分别转化为指定的输出。

(1) ['12', '-2', '0'] --> [12, -2, 0]
(2) ['hello', 'world'] --> [5, 5]
(3) ['hello', 'world']` --> ['olleh', 'dlrow']
(4) range(2, 6) --> [(2, 4, 8), (3, 9, 27), (4, 16, 64), (5, 25, 125)]
(5) zip(range(2, 5), range(3, 9, 2)) --> [6, 15, 28]

8. Filter: 使用 filter 语句将以下输入，分别转化为指定的输出。

(1) ['12', '-1', '0'] --> ['12"]

(2) ['hello', 'world'] --> ['world']

(3) ['technology', 'method', 'technique'] --> ['technology', 'technique']

(4) range(20) --> [0, 3, 5, 6, 9, 10, 12, 15, 18]

9. Reduce 1: 使用 reduce 语句将以下输入，分别转化为指定的输出。

(1) [23, 49, 6, 32] --> 49

(2) ['foo', 'bar','baz','quz'] --> 'foobarbazquz'

(3) [2,4,6] --> 48

10. Reduce 2: 使用 reduce 语句编写函数 lcm(*nums)，计算任意数量个正整数的最小公倍数，**要求只写一句 Python 语句** （提示：可使用 math 模块的 gcd 函数先求出最大公约数）。

例子：

```
lcm(3, 5)                    # 15
lcm(41, 106, 12)             # 26076
lcm(1, 2, 6, 24, 120, 720)   # 720
lcm(3)                       # 3
lcm()                        # 如果没有向函数提供数字，可以返回值 1。
```

11. Iterator: 运行以下代码，观察输出并解释输出的原因。

```
it = iter(range(100))
66 in it   # => True

print(next(it))   # => ??
print(33 in it)   # => ??
print(next(it))   # => ??
```

12. Generator:

编写一个生成器 generate_triangles()，连续地产生三角数 1，3，6，10，… 三角数通过连续的正整数相加来生成（如 1=1，3=1+2，6=1+2+3，10=1+2+3+4，…）。

例子：

```
g=generate_triangles()
for _ in range(5):
    print(next(g))      #输出 1，3，6，10，15
```

使用生成器 generate_triangles()，编写函数 generate_triangles_under(n)，返回小于 n 的所有三角数。

13. Decorator:

使用装饰器的方法重做实验 3.5。代码框架如下，请补充函数 timed。

```
from time import time

def timed(fn):    #fn stands for function
        # 你的代码


@timed
def test_plus():
    result = []
    for i in range(100000):
        result = result + [i]

@timed
def test_append():
    result = []
    for i in range(100000):
        result.append(i)

test_plus()
test_append()
```