第二题：

这道题的主要的目的是计算两条序列的相似度。

相似度作者在这里是这样定义的（fraction=True）。

（1） 首先 A 序列和 B 序列长度相等，都为某值，如 n。

（2） 然后，计算 A 序列和 B 序列在对应位置（一一对应）相等的碱基的数量，如 m。

（3） 则，这两条序列的相似度，我们用 m/n 表示。

（4） 如果 fraction=False,则这两条序列的相似度，等于 m。

该代码的测试的运行结果：

```
(base) [xxzhang@mu02 python_dir]$ python code_2.py
I(ATGCATGCATGCATCGTAGCTACG,ATGCATGCATGCATCGTAGCTACG) = 1.00
I(ATGCATGCATGCATCGTAGCTACG,CCGATCGAGCTGTCTAGCTATCGC) = 0.12
I(ATGCATGCATGCATCGTAGCTACG,AAATGCGGACACGTAGCTGTAGCC) = 0.33
I(ATGCATGCATGCATCGTAGCTACG,ATGAATACATACAACGAAGCTACG) = 0.79
I(CCGATCGAGCTGTCTAGCTATCGC,ATGCATGCATGCATCGTAGCTACG) = 0.12
I(CCGATCGAGCTGTCTAGCTATCGC,CCGATCGAGCTGTCTAGCTATCGC) = 1.00
I(CCGATCGAGCTGTCTAGCTATCGC,AAATGCGGACACGTAGCTGTAGCC) = 0.17
I(CCGATCGAGCTGTCTAGCTATCGC,ATGAATACATACAACGAAGCTACG) = 0.12
I(AAATGCGGACACGTAGCTGTAGCC,ATGCATGCATGCATCGTAGCTACG) = 0.33
I(AAATGCGGACACGTAGCTGTAGCC,CCGATCGAGCTGTCTAGCTATCGC) = 0.17
I(AAATGCGGACACGTAGCTGTAGCC,AAATGCGGACACGTAGCTGTAGCC) = 1.00
I(AAATGCGGACACGTAGCTGTAGCC,ATGAATACATACAACGAAGCTACG) = 0.29
I(ATGAATACATACAACGAAGCTACG,ATGCATGCATGCATCGTAGCTACG) = 0.79
I(ATGAATACATACAACGAAGCTACG,CCGATCGAGCTGTCTAGCTATCGC) = 0.12
I(ATGAATACATACAACGAAGCTACG,AAATGCGGACACGTAGCTGTAGCC) = 0.29
I(ATGAATACATACAACGAAGCTACG,ATGAATACATACAACGAAGCTACG) = 1.00
```

下面，正式介绍代码的主体：

```
def identity(s1,s2,fraction=True):
    if len(s1)!=len(s2):
        return None
    identity = 0
    for n1,n2 in zip(s1,s2):
        if n1==n2:
            identity+=1
    if fraction:
        return identity/len(s1)
    else:
        return identity
```

同样地，作者使用了 def 关键词定义了一个名为“identity”的函数，其输入的参数有三

个。s1,s2 和 fraction，其中 fraction 为布尔变量（仅有 true 或 false 两种取值），该参数默认为 True。

```
if len(s1)!=len(s2):
    return None
```

下面我们一行行来看。到了这里的 if 语句的这一部分，同样的，这个 if 语句的条件是 len(s1)!=len(s2)，这里的 len()和咱们第一题讲的一样，表示的是求这个字符串的长度。而这里的“！=”指的是“不等于”，所以这里的条件指的就是，如果 s1 字符串的长度不等于 s2 字符串的长度。那么，此函数返回 None。同样的，如果这个时候，函数已经返回了 None 值，则函数接下来的语句便不再运行了。此时返回的 None 值就是空值。

那么，如果 if 语句的条件不成立，则程序接下来继续运行。

作者定义了数值型的变量 identity，并将其初始化为 0。

```
identity = 0
```

然后，就到了这里的这个 for 循环，这个 for 循环中有一个函数 zip（）。

```
for n1,n2 in zip(s1,s2):
    if n1==n2:
        identity+=1
```

这里掩饰一下，zip（）函数的作用。从下面可以看到，这个 zip 的作用就是从等长的两个字符型变量中，成对的按照位置取出字符，并将其分别赋值给 s1 和 s2。在循环的过程中，按照位置依次类推，取出每对值。

```
>>> for s1,s2 in zip("AACG","ATAC"):
...     print (s1,s2)
...
A A
A T
C A
G C
```

所以我们也好理解在 for 循环内部的 if 语句了，即，如果我们从这两条序列中成对取出

的这两个字符相等，则 identity+1。所以，最后 for 循环，结束，我们得到的是，这两个长度相等的字符串，有多少个位置是相同的字符。

上述 for 循环的片段结束之后，又到了下一段的 if 语句（注意在 Python 中，空格表示的是逻辑层次，在相同对齐的位置，在逻辑上是平等的，不是从属关系）。这一段 if 语句决定输出。如果 fraction=True，则返回的值为 identity 除以输入序列的长度。否则，则输出 identity 的值。

```
if fraction:
    return identity/len(s1)
else:
    return identity
```

到这里，def identity（）这个函数体基本上讲完了。

接下来作者就在这个 for 循环中，使用了这个 identity 函数，求给定的两个序列的相似程度。

这里 sequence 是一个列表，列表中存储了四个等长的不同的序列，下面这个 for 循环，就是依次的从这些序列中取出一个序列，然后进行两两比较，计算相似性。

```
sequences = [
    "ATGCATGCATGCATCGTAGCTACG",
    "CCGATCGAGCTGTCTAGCTATCGC",
    "AAATGCGGACACGTAGCTGTAGCC",
    "ATGAATACATACAACGAAGCTACG",
]
```

```
for seq1 in sequences:
    for seq2 in sequences:
        print("I(%s, %s) = %.2f" %
              (seq1,seq2,identity(seq1,seq2)))
```

下面演示遍历过程（一共有 16 种可能）：

Seq1	Seq2
"ATGCATGCATGCATCGTAGCTACG",	"ATGCATGCATGCATCGTAGCTACG",
	"CCGATCGAGCTGTCTAGCTATCGC",
	"AAATGCGGACACGTAGCTGTAGCC",
	"ATGAATACATACAACGAAGCTACG",
"CCGATCGAGCTGTCTAGCTATCGC",	"ATGCATGCATGCATCGTAGCTACG",
	"CCGATCGAGCTGTCTAGCTATCGC",
	"AAATGCGGACACGTAGCTGTAGCC",
	"ATGAATACATACAACGAAGCTACG",
"AAATGCGGACACGTAGCTGTAGCC",	"ATGCATGCATGCATCGTAGCTACG",
	"CCGATCGAGCTGTCTAGCTATCGC",
	"AAATGCGGACACGTAGCTGTAGCC",
	"ATGAATACATACAACGAAGCTACG",
"ATGAATACATACAACGAAGCTACG",	"ATGCATGCATGCATCGTAGCTACG",
	"CCGATCGAGCTGTCTAGCTATCGC",
	"AAATGCGGACACGTAGCTGTAGCC",
	"ATGAATACATACAACGAAGCTACG",

最后解释一下，print（）函数中间这一堆是什么意思。这一段实际上是对输出格式的要求。%前面的“”引号内的内容，表示的是输出的内容。其中的%s,%s 和%.2f 代指的是后面括号内的字符串 seq1，字符串 seq2 以及 identity（）的输出值。

而%s 和%.2f 在代指变量的时候有什么区别呢?

%s：表示的是代指的是字符串型的变量；

%f：表示的是代指的是数值型的变量，且为**浮点数**(可以理解为小数)。中间的“.2”表示的是浮点数保留两位小数。

```
print("I(%s, %s) = %.2f" %
          (seq1,seq2,identity(seq1,seq2)))
```

上面讲的输出格式，通过看下面的输出就可以比较好的理解。

I(seq1,seq2)=X.XX

```
(base) [xxzhang@mu02 python_dir]$ python code_2.py
I(ATGCATGCATGCATCGTAGCTACG,ATGCATGCATGCATCGTAGCTACG) = 1.00
I(ATGCATGCATGCATCGTAGCTACG,CCGATCGAGCTGTCTAGCTATCGC) = 0.12
I(ATGCATGCATGCATCGTAGCTACG,AAATGCGGACACGTAGCTGTAGCC) = 0.33
I(ATGCATGCATGCATCGTAGCTACG,ATGAATACATACAACGAAGCTACG) = 0.79
I(CCGATCGAGCTGTCTAGCTATCGC,ATGCATGCATGCATCGTAGCTACG) = 0.12
I(CCGATCGAGCTGTCTAGCTATCGC,CCGATCGAGCTGTCTAGCTATCGC) = 1.00
I(CCGATCGAGCTGTCTAGCTATCGC,AAATGCGGACACGTAGCTGTAGCC) = 0.17
I(CCGATCGAGCTGTCTAGCTATCGC,ATGAATACATACAACGAAGCTACG) = 0.12
I(AAATGCGGACACGTAGCTGTAGCC,ATGCATGCATGCATCGTAGCTACG) = 0.33
I(AAATGCGGACACGTAGCTGTAGCC,CCGATCGAGCTGTCTAGCTATCGC) = 0.17
I(AAATGCGGACACGTAGCTGTAGCC,AAATGCGGACACGTAGCTGTAGCC) = 1.00
I(AAATGCGGACACGTAGCTGTAGCC,ATGAATACATACAACGAAGCTACG) = 0.29
I(ATGAATACATACAACGAAGCTACG,ATGCATGCATGCATCGTAGCTACG) = 0.79
I(ATGAATACATACAACGAAGCTACG,CCGATCGAGCTGTCTAGCTATCGC) = 0.12
I(ATGAATACATACAACGAAGCTACG,AAATGCGGACACGTAGCTGTAGCC) = 0.29
I(ATGAATACATACAACGAAGCTACG,ATGAATACATACAACGAAGCTACG) = 1.00
```