第四题：

这道题理解起来特别的简单。它主要做了什么呢?

假设我们有一个字母表 ABCD，这个代码主要做的东西可以理解为一个数学的问题，从 ABCD 四个字母中又放回的取两个字母，如果两次取得的字母相同，则将其值对应为“match”的值；如果两次取得的字母不同，则将其值对应为“mismatch”的值。这样就建立起来了 16 种（根据字母表的长度不同，结果不同）一一对应的关系，将其存储到字典这样的一种数据结构中。

下面介绍代码的具体的过程：

同样的，其使用 def 关键词定义了一个名为“create_substitution_matrix”的函数，其输入参数为 alphabet，match 和 mismatch。其中 alphabet 为一个值为字母的列表，而 match 和 mismatch 都为数值型变量。

```
def create_substitution_matrix(alphabet, match, mismatch):
```

首先，使用 len（）函数，求得**列表**的长度为 n。

```
n = len(alphabet)
```

然后，定义一个空的**字典** submat。

```
submat = {}
```

接下来就到了我们比较熟悉的地方（前面我们已经多次介绍了 for 循环和 if……else 语句了，相信现在应该有基本的概念了）。

这是两个嵌套的 for 循环，变量的取值范围都是 range（n）。

我们来看看这个 range(n)的意思是什么？我们取 n=5，发现 i 依次取值为 0，1，2，3，4（也就是说取从 0 开始的 5 个数）。

注：当你自己看代码的时候，你不理解它的意思，最直接的方法就是 print 输出，看看得

到的是什么。然后再根据结果，推测该行代码的功能。

```
>>> n = 5
>>> for i in range(n):
...     print (i)
...
0
1
2
3
4
```

```
for i in range(n):
    for j in range(n):
    if alphabet[i] == alphabet[j]:
        submat[alphabet[i] + alphabet[j]] = match
    else:
        submat[alphabet[i] + alphabet[j]] = mismatch
return submat
```

然后到最里面的 if……else 的判断结构。If 的成立的条件是 alphabet[i]==alphabet[j]，注意这里用的是“==”。当我们进行字符型的数据的判断的时候，用的是双等号，这里的意思也就是说，当 alphabet 的第 i 个值和 alphabet 的第 j 个值相等的时候，怎么样？（嵌套 for 循环，我们之前也讲过，一共要循环 n*n 次）如果 if 语句后面的条件成立，则执行 submat[alphabet[i]+alphabet[j]] =match 这行代码。其中我们要明白 submat 其为字典型的变量，这种格式在其[]里面的是字典的“键”，后面等号是对应“键”的赋值。在此处，[]里面的 alphabet[i]+alphabet[j]是字符之间的拼接，类似于“A”+“C”到“AC”,这个我们在第一题的时候也讲过。所以，也就是说，将字典中的键赋为某值，构建其一个键-值的 pair。如果两者字符相同，则赋值为 match，不同，则赋值为 mismatch。

下面这个表同样是演示 for 循环的遍历过程。

I 的取值	J 的取值
0	0
	1
	2
1	0
	1
	2
……	0

最后，就是调用函数，使用不同的输出参数去看程序的运行结果。

```
#Test Algorithm
alphabet = ["A","T","G","C"]
submat_dna = create_substitution_matrix(alphabet,5,-4)
print("Submat für DNA Alphabet:")
print(submat_dna)
alphabet = ["A","R","N","D","C","Q","E"]
submat_aa = create_substitution_matrix(alphabet,5,-4)
print("Submat für AA Alphabet:")
print(submat_aa)
```

下为程序测试的运行结果：