Problem 1

Complete the function below, which takes arguments `a` and `b` and you can assume they will be integer numbers. The function should test whether `a` is evenly divisable by `b`. If it is evenly divisable, it should return a string that reads

```
"a is evenly divisible by b!"
```

Please note that `a` and `b` should be replaced with their actual values given in the function call. In the event that `a` is not evenly divisable by `b`, the function should store a return

```
"a is not evenly divisible by b. The remainder is c."
```

where `c` is the remainder. Again, `a`, `b`, and `c` should be replaced by thier actual values in the function call. For example, calling the function with

`is_evenly_divisible(4, 2)`

should return

`"4 is evenly divisible by 2!"`

and calling the function with

`is_evenly_divisible(5, 3)`

should return

`"5 is not evenly divisible by 3. The remainder is 2."`

In [ ]:

```
def is_evenly_divisible(a,b):
    my_rema= a % b
    if my_rema == 0:
     return  str(a) + " is evenly divisible by " + str(b) + "!"
    else:
     return  str(a) + " is not evenly divisible by " + str(b) + ", the remainder is 

     #字符串和数字不能直接相加，需要先将数字类型转化成字符串类型 采用Str函数。
```

In [ ]:

```
x=is_evenly_divisible(4,2)
```

In [ ]:

```
print(x)
```

```
4 is evenly divisible by 2!
```

In [ ]:

```
y=is_evenly_divisible(15,7)
```

In [ ]:

```
print(y)
```

```
15 is not evenly divisible by 7, the remainder is 1.
```

Problem 2

Complete the function below. Assume the arguments `a` and `b` are integer or floating point numbers. The argument `operation` is a string that has one of the following values exactly: `'add', 'subtract', 'multiply', 'divide'`.

Given one of those values for `operation`, design a series of tests that will return the cooresponding mathematical operation for the numbers defined in `a` and `b`. So if `operation = 'add` then you should return the sum of `a` and `b`. Likewise for the other operations. In the event that the user inputs in invalid value for `operation`, have the function return the string:

```
'Operation must be one of: ["add", "subtract", "multiply", "divide"]'
```

This string has been placed in a comment below for your assistance. The following examples should clarify the desired implementation of the function. If the function is called with

`math_operation(4, 2, 'add')`

will return `6`.

`math_operation(4, 2, 'multiply')`

will return `8`.

`math_operation(4, 2, 'divide')`

will return `2`.

`math_operation(4, 2, 'plus')`

will return `'Operation must be one of: ["add", "subtract", "multiply", "divide"]'`.

In [ ]:

```
def math_operation(a, b, operation):

    #return #function should return the correct calculation or the statement:
           #'Operation must be one of: ["add", "subtract", "multiply", "divide"]'

    operationlist = ["add", "subtract", "multiply", "divide"]
    if operation == operationlist[0]:
        return a + b
    elif operation == operationlist[1]:
        return a - b
    elif operation == operationlist[2]:
        return a * b
    elif operation == operationlist[3]:
        return a / b
    else:
        return "Operation must be one of: ['add', 'substract', 'multiply', 'divide']"
```

In [ ]:

```
x = math_operation(4, 2, "add")
```

In [ ]:

```
print(x)
```

```
6
```

In [ ]:

```python
y = math_operation(4, 2, "plus")
```

In [ ]:

```python
print(y)
```

```
Operation must be one of: ['add', 'substract', 'multiply', 'divide']
```