🡪

整形数字通过对10取余可以获得最后一位，除以10可以获得前半部分。

🡪

整形数字如果要一位位以字符放在数组中时，可以转变为str类型，str字符串便可以进行数组下的操作。

🡪

判断回文串。两种方式，可以使用先后index, 当前面index和后面index碰面时，还没有不相同便是回文串。或者直接使用[::-1]生成倒置后的结果。判断是否与原字符串相同便可。

🡪

Python字符串可以直接当list使用，通过Index找寻下标。如果要获得list类型的，list(string)便行。

🡪

通过循环删除Python 数组中元素

例如：

test = [1, 2, 3, 3, 3, 6, 7]

要求删除数组中所有等于3的元素。

方法1：

i=0

while i < len(test):

if test[i] == 3:  
 test.pop(i)  
 else:  
 i += 1

此方法可行。由于将指针放在开头index=0, 当等于需要删除的元素时候删除，不移动指针。Python数组会将剩余元素自动前移。当不等于需要删的元素时候，移动指针。并且由于在循环中删除元素，所以数组的长度，index最大值会改变。所以需要在while条件中采用len(test)，相当于每次循环动态的获取现在数组的长度。但是如果向获取原数组长度，进行循环则会发生index报错问题。

lenTest = len(test)

i = 0  
while i < lenTest: # index报错，相当于固定index最大值。每次删除数组元素以后，数组长度将变小  
 if test[i] == 3:  
 test.pop(i)  
 else:  
 i += 1

并且由于方法1直接更改数组，不需要开辟新的内存空间。原数组的地址和改变以后数组的内存地址相同。

方法2：

test = [1, 2, 3, 3, 3, 6, 7]

for element in test:

print(element)  
 if element == 3:  
 test.remove(element)

# 打印出1 2 3 3 7，test运行后结果为[1, 2, 3, 6, 7] 并没有完全去除元素3， 因为在循环中删除元素，所以for element in test会在每次循环读取，根据index读取下一个元素，而在某次循环中删除元素后，剩余元素自动上移，index所对应的元素发生改变。发生跳过元素的情况。

方法3：

test = [element for element in test if element != 3]

采用list comprehension. 可行，但和方法一的区别是，由list comprehenstion所得出的结果相当于在内存空间中开辟一个新的空间保存生成数组。所以原数组test内存地址id,和新数组test内存地址将不同。

🡪