Untitled 3

Lesson 3: 程序的控制: 判断与循环

考虑这样一个简单的问题，让用户输入一个数，若为偶数则要求重新输入，若为奇数则直接输出

问题本身不复杂，但要想处理这个问题，只依赖原先的顺序逻辑是行不通的，我们的程序不能傻傻的一条语句一条语句向下执行，必须有根据当前情况控制自身如何运行的灵活性。

判断语句

判断是最基础的控制方式，可以让程序根据当前情况决定要执行哪个分支。为了让程序理解现在是什么情况，我们要向判断语句传递一个值(可以是变量的值，也可以是表达式的值)，然后让程序根据这个值来选择分支继续运行

下面是一个简单的例子的三种实现

```
# even_or_odd.py

num = int(input('please input a integer:'))
if num < 0:
    print('error')         # 判断是否为正数
elif num % 2 == 0:         # 判断是否为偶数
    print('even')
else:                      # 判断是否为奇数
    print('odd')
```

```
# even_or_odd.py

num = int(input('please input a integer:'))
if num >= 0:                    # if语句可以任意嵌套
    if num % 2 == 0:
        print('even')
    else:
        print('odd')
else:
    print('error')
```

```
# even_or_odd.py

num = int(input('please input a integer:'))
if num >= 0 and num % 2 == 0:              # 用好and or not来表达更为复杂的条
件
    print('even')
elif num >= 0 and num % 2 == 1:
    print('odd')
else:
    print('error')
```

这三种实现，我最推荐第一种，这主要基于两个原则

1. 宁可分支数多，也要尽可能避免较高的嵌套数，过多的嵌套会使代码的可读性下降。因此我们推荐先处理没有后续更小分支的分支
2. 尽可能将复杂条件拆分为多层if，因为复杂条件难以阅读且难以保证正确覆盖所有情况

判断语句中传递的可以是表达式，也可以是变量。这实际上进行了一次类型转换，把0值/空列表/空元组/空值转为False，其余则为True

```
# print_or_not.py

s = 'hello'
num = 0
empty = []

if len(s):
    print(s)

if str:
    print(s)

if num:
    print(num)

if empty:
    print(len(empty))
```

循环

判断语句带来的分支结构极大的拓展了程序的灵活性，但是判断语句是一次性的，有时我们希望持续判断输入，直到满足要求再进行下一步；有时我们希望对同一个语句执行给定的次数；

有时我们希望持续进行某个步骤直到目标完成...这就需要用到循环语句

常用: for循环

用于遍历一个可迭代的对象，对迭代得到的每个元素做同样的操作

```
# sum.py

sum = 0
sum_odd = 0
for i in range(1, 101):    # 根据左闭右开原则，i的取值为[1,100]
    sum += i

for i in range(1,101,2):  # 2表示遍历时的步长
    sum_odd += i

print(sum)
print(sum_odd)
```

```
s = 'hello'
for letter in s:         # 遍历str的所有子元素
    print(letter)
```

```
s = 'hello'
for idx, letter in enumerate(s):    # 同时遍历str的所有下标与子元素
    print(idx,':‘, letter)
```

while循环

相比for循环，while不强调循环的范围，而强调继续循环的条件

```
# sum.py
''' 计算1到100的和 '''
N = 100
num = 1
sum = 0
while num <= N:  # 满足括号内条件时，循环继续，否则循环结束
    sum += num
    num += 1
print(sum)
```

循环的控制: continue & break

有时候我们不希望程序傻傻的完成整个循环，而希望程序能在满足一定条件后直接进入下一次循环/跳出循环

```
s = 'hello, world!'
for letter in s:
    if letter == ',' or letter ==  ' ':
        continue          # 不做处理，直接进入下一个循环
    if letter == l:
        break             # 直接退出循环
    print(letter)
```

习题

1. 输入一个字符串和一个字符，在字符串中寻找这个字符，输出这个字符在这个字符串中首次出现的位置(若不存在则输出不存在)
2. 输入两个整数，输出它们的最大公因数与最小公倍数
3. 输入两个字符串，判断其中一个字符串是否是另一个字符串的一部分