

控制结构和函数编程练习

《用 Python 玩转数据》 by Dazhuang@NJU

经典问题的 **Python** 实现，用于练手，后续学习了序列等数据结构有些问题的解决或许会更方便一些。如果小伙伴以往没有较好的程序设计基础，自己多码代码才能对程序有一个大致的了解，多多练习找找感觉吧！**Fighting^_^**

1. 身体质量指数（BMI，Body Mass Index）是国际上常用的衡量人体肥胖程度和是否健康的重要标准，计算公式为： $BMI = \text{体重} / \text{身高的平方}$ （国际单位 kg/m^2 ）。中国的成年人 BMI 数值定义为：

过轻：低于 18.5

正常：18.5-23.9

过重：24-27.9

肥胖：高于 28

请输入体重和身高，输出相应的 BMI 值和体重肥胖程度判断结果（too thin、normal、overweight 或 fat）。

[输入样例]

60,1.6

[输出样例]

Your BMI is 23.4

normal

[提示]

程序中体重和身高的输入可用“`weight, height = eval(input())`”语句表示。

2. 按公式： $C = 5/9 \times (F - 32)$ ，将华氏温度转换成摄氏温度，并产生一张华氏 0~300 度与对应的摄氏温度之间的对照表（每隔 20 度输出一次）

3. 角谷静夫是日本的一位著名学者，他提出了一个猜想（称为角谷猜想）：对于一个正整数 n ，若为偶数则除以 2，若为奇数则乘以 3 加 1，得到一个新的数后按照之前的两条规则继续演算，若干次后得到的结果必然为 1。输入任一正整数，输出演算过程。

[输入样例]

10

[输出样例]

10/2=5

5*3+1=16

16/2=8

8/2=4

4/2=2

2/2=1

4. 输入 n ，用递推法（例如前项之间的关系推导后项，本题为一重循环）编程求 $1+2!+3!+\dots+n!$ 的和并输出。

[输入样例]

5

[输出样例]

153

5. 编程求解 1-4 这 4 个数字可以组成多少个无重复的三位数，按从小到大的顺序输出这些数字。

6. 验证命题：如果一个三位整数是 37 的倍数，则这个整数循环左移后得到的另两个 3 位数也是 37 的倍数。（注意验证命题的结果输出方式，只要输出命题为真还是假即可，而非每一个三位数都有一个真假的输出）

7. 一个数如果等于它的因子之和则称这个数为完数，例如 6， $6=1+2+3$ ，编程计算 1000 以内的所有完数并输出。

8. 验证哥德巴赫猜想之一：2000 以内的正偶数（大于等于 4）都能够分解为两个质数之和。每个偶数表达成形如： $4=2+2$ 的形式。

【参考代码】

1.

```
#-*-coding:utf-8-*-
```

```
weight, height = eval(input())
```

```
bmi = weight / height ** 2
```

```
print("Your BMI is {0:.1f}".format(bmi))
```

```
if bmi < 18.5:
```

```
    print("too thin")
```

```
elif bmi < 24:
```

```
    print("normal")
```

```
elif bmi < 27.9:
```

```
    print("overweight")
```

```
else:
```

```
    print("fat")
```

2.

```
#-*-coding:utf-8-*-
```

```
for f in range(0, 301, 20):
```

```
    c = 5 / 9 * (f - 32)
```

```
    print("{} f = {:.0f} c".format(f, c))
```

3.

```
#-*-coding:utf-8-*-
```

```
n = int(input())
```

```
while n != 1:
```

```
    if n % 2 == 0:
```

```
        print("{} / 2 = {}".format(n, n // 2))
```

```
        n //= 2
```

```
    else:
```

```
print("{}*3+1={}".format(n, 3*n+1))
n = 3*n+1
```

4.

```
#-*-coding:utf-8-*-
```

```
n = int(input())
```

```
s = term = 1
```

```
for i in range(2, n+1):
```

```
    term *= i
```

```
    s += term
```

```
print(s)
```

5.

```
#-*-coding:utf-8-*-
```

```
for i in range(1, 5):
```

```
    for j in range(1, 5):
```

```
        for k in range(1, 5):
```

```
            if i != k and i != j and j != k:
```

```
                print(100*i+10*j+k)
```

6.

```
#-*-coding:utf-8-*-
```

```
for num in range(100, 1000):
```

```
    if num % 37 == 0:
```

```
        num_new_1 = num % 100 * 10 + num // 100
```

```
        num_new_2 = num % 10 * 100 + num // 10
```

```
        if num_new_1 % 37 != 0 or num_new_2 % 37 != 0:
```

```
            print("It's a false proposition.")
```

```
            break
```

```
else:
```

```
    print("It's a true proposition.")
```

7.

```
#-*-coding:utf-8-*-
```

```
for i in range(1, 1001):
```

s = 0 # 注意 s 初值的位置，设处置的位置根据需要来定而非一定放在所有循环的外面

```
    for j in range(1, i):
```

```
        if i%j == 0:
```

```
            s += j
```

```
    if s == i:
```

```
        print("\n", i, " ", end = "")
```

```
        print("its factors are ", end = "")
```

```
        for j in range(1, i):
```

```
            if i%j == 0:
```

```
print(j, end = " ")
```

8.

```
#-*-coding:utf-8-*-
```

```
import math
```

```
def prime(x):
```

```
    if x == 1:
```

```
        return False
```

```
    n = int(math.sqrt(x))
```

```
    for i in range(2, n+1):
```

```
        if x % i == 0:
```

```
            return False
```

```
    return True
```

```
def GC(n):
```

```
    k = 3
```

```
    while k < n:
```

```
        t = n - k
```

```
        if t < k:
```

```
            break
```

```
        if prime(k) and prime(t):
```

```
            return k, t
```

```
        k += 2
```

```
n = int(input())
```

```
if n > 4:
```

```
    a,b = GC(n)
```

```
    print("{}={}+{}".format(n, a, b))
```

```
elif n == 4:
```

```
    print("{}={}+{}".format(4, 2, 2))
```