实验 函数（一）

```
# 打印你的学号和姓名
print('姓名：梁皓然')

print('学号：2018302100035')
```

```
姓名：梁皓然
学号：2018302100035
```

1. 尝试运行下面的语句，观察结果。如果有错，则思考原因，并以注释形式将错误原因写在程序中。

```
def change(i, s):
    i += s # 函数没有使用return返回值，函数无效

i=5
change(i, 10)
print("i =", i)
```

修改和测试结果如下

```
def change(i, s):
    z = i + s
    return z

i=5
change(i, 10)
print("i =", change(i, 10))
```

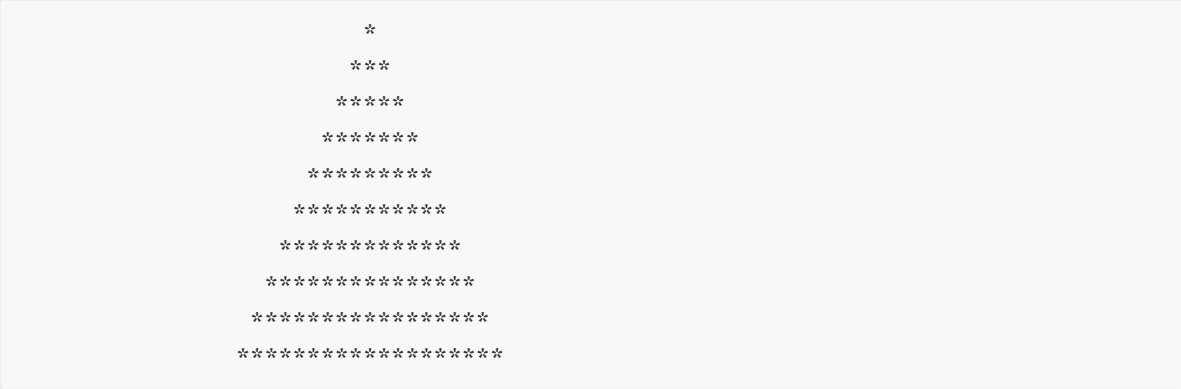
```
i = 15
```

```
def change(i, s):
    i[0] += s

i=[5]
change(i, 10)
print("i =", i)
```

```
i = [15]
```

```
def print_star(n):
    print(('*'*n).center(50))
for i in range(10):
    print_star(2*i+1)
```



```
def isEven(n):
    if n%2==0:
        return True
    else:
        return False

for i in [12, 3, 45, 6]:
    if isEven(i): print(i, end=' ')
```

```
12 6
```

```
def f1():
    "simple function"
    pass
print(f1.__doc__)
```

```
simple function
```

```
def judge(param1, *param2):
    print(type(param2))
    print(param2)
judge(1,2,3,4,5)
```

```
<class 'tuple'>
(2, 3, 4, 5)
```

```
def judge(param1, **param2):
    print(type(param2))
    print(param2)
judge(1,a=2,b=3,c=4,d=5)
```

```
<class 'dict'>
{'a': 2, 'b': 3, 'c': 4, 'd': 5}
```

2. 编写程序，定义一个求整数阶乘的函数fact(n)，然后编写测试代码，由用户输入需要计算阶乘的整数n（n>=0），输出计算结果。

例如，输入整数n(n>=0)：5
输出 5! = 120

```
n = int(input('请输入一个整数：'))

def fact(n):
    sum1 = 0
    if n == 0:
        sum1 = 1
    else:
        sum1 = n * fact(n - 1)
    return sum1

print("{0} != {1}".format(n, fact(n)))
```

```
请输入一个整数：5
5 != 120
```

3. 编写程序，定义一个求斐波那契数列的函数fib(n)，并编写测试代码，输出前20项（每项宽度5个字符位置，右对齐），每行输出10个。

```
def fib(n):
    if n == 1 or n == 2:
        return 1
    else:
        return fib(n - 1) + fib(n - 2)


for i in range(20):
    print(format(fib(i + 1), '>10.0f'), end='')
    if (i + 1) % 10 == 0:
        print()
```

```
         1         1         2         3         5         8        13        21
        34        55
        89       144       233       377       610       987      1597      2584
      4181      6765
```

4. 编写程序，利用可变参数定义一个求任意2个或更多数中最小值的函数min_n(a, b, *c)，并编写测试代码。

例如，print(min_n(8, 2))，print(min_n(16, 1, 7, 4, 15))

```
def min_n(a, b, *c):
    list_a = [a, b]
    list_a.extend(c)
    return min(list_a)


print("最小值为 ", min_n(8, 2))
print("最小值为 ", min_n(16, 1, 7, 4, 15))
```

```
最小值为  2
最小值为  1
```

5. 编写程序，利用元组作为函数的返回值，求序列数据中的最大值、最小值和元素个数，并编写测试代码。假设测试数据分别为s1=[9,7,8,3,2,1,55,6]，s2=["apple", "pear", "meton", "kiwi"]和s3="TheQuickBrownFox"。运行效果如下：

list=[9,7,8,3,2,1,55,6]
最大值=55, 最小值=1, 元素个数=8
list=["apple", "pear", "meton", "kiwi"]
最大值=pear, 最小值=apple, 元素个数=4
list=TheQuickBrownFox
最大值=x, 最小值=B, 元素个数=16

```
def print_info(x):
    i = max(x)
    j = min(x)
    l = len(x)
    print("最大值是%s" % i)
    print("最小值是%s" % j)
    print("元素个数是{0}".format(l))
    return i, j, l


s1 = [9, 8, 7, 3, 2, 1, 55, 6]
s2 = ['apple', 'pear', 'melon', 'kiwi']
s3 = 'TheQuickBrownFox'
q = print_info(s1)
print(q)
w = print_info(s2)
print(w)
e = print_info(s3)
print(e)
```

```
最大值是55
最小值是1
元素个数是8
(55, 1, 8)
最大值是pear
最小值是apple
元素个数是4
('pear', 'apple', 4)
最大值是x
最小值是B
元素个数是16
('x', 'B', 16)
```

5. 你可以编写更多程序。（选做）

（1） 编写程序，定义一个函数IsPerfectNumber(n)判断一个整数是否是完全数。如果一个整数的所有因子的和等于该整数，它就是完全数。用该函数找出1~1000内的所有完全数。

下面程序定义一个函数IsPerfectNumber(n)判断一个整数是否是完全数。

```python
import math
def isPerfectNumber():
    n = int(input('input a number(>0): '))
    while(n <= 0):
        n = int(input('input a number(>0): '))
    sum = 0
    k = int(math.sqrt(n))
    for i in range(1,k+1):
        if 0 == n%i:
            sum += n/i
            sum += i
    sum -= n  #1肯定是因子，多加了一个它自己。
    if n == sum:
        return 1
    else:
        return 0
def main():
    tag = isPerfectNumber()
    if tag:
        print("This is perfect number")
    else:
        print("This isn't perfect number")

if __name__=='__main__':
    main()
```

```
input a number(>0): 496
This is perfect number
```

下面程序用函数IsPerfectNumber(n)找出1~1000内的所有完全数。

```
import math
def isPerfectNumber(k):
    n = k
    while(n <= 0):
        n = int(input('input a number(>0): '))
    sum = 0
    k = int(math.sqrt(n))
    for i in range(1,k+1):
        if 0 == n%i:
            sum += n/i
            sum += i
    sum -= n  #1肯定是因子，多加了一个它自己。
    if n == sum:
        return i
    else:
        return 0
def main():
    for k in range(2,1001):
        i = isPerfectNumber(k)
        if i:
            print(k)

if __name__=='__main__':
    main()
```

```
6
28
496
```

（2）编写并测试一个函数toNumbers(strList)，满足以下规格说明。

strList 是一个字符串列表，每个字符串表示一个数字。
修改列表，将每一项转换为数字。
假设aList = ["123", "456", "789"]，则通过调用toNumbers(aList)使得aList = [123, 456, 789]

```
def toNumbers(lis):
    return [int(num) for num in lis]
```

```
aList = ["123", "456", "789"]
```

```
b=toNumbers(aList)
```

```
b
```

```
[123, 456, 789]
```