

人工智能实践教程

从Python入门到机器学习



第一部分Python核心编程技术

- 所有代码及ppt均可以由以下链接下载
- <https://github.com/shao1chuan/pythonbook>
- <https://gitee.com/shao1chuan/pythonbook>

目录

CONTENT
S

1. 函数
2. 函数创建和调用
3. 变量作用域
4. 函数参数传递
5. 匿名函数
6. 递归函数

为什么需要函数?

```
print "          _oo0oo_ "
print "          o8888888o "
print "          88 . 88 "
print "          (| -_- |) "
print "          O\\ = /O "
print "         ___/`---'\___ "
print "        . \\\| |// \\\ "
print "       / \\\| | : | |// \\\ "
print "      / _||| -:- | | | - \\\ "
print "     | | \\||| - ||| | | "
print "     | |  |\\| | - \\| | | "
print "     | | \\| | |\\| -||| | | "
print "     \\ \\| | | |\\| -||| | | "
print "     _ \\ \\| | | |\\| -||| | | "
print "     . \\ \\| | | |\\| -||| | | "
print "     : \\ \\| | | |\\| -||| | | "
print "     | | : \\ \\| | | |\\| -||| | | "
print "     \\ \\| | | |\\| -||| | | "
print "     =====|-----|-----|-----|===== "
print "     |-----| "
print " "
print " ..... "
print "          佛祖镇楼          BUG辟易 "
print " 佛曰: "
print "    写字楼里写字间, 写字间里程序员: "
print "    程序员写程序, 又拿程序换酒钱。 "
print "    酒醒只在网上坐, 酒醉还来网下眠: "
print "    酒醒酒醒日复日, 网上网下年复年。 "
print "    但愿老死电脑间, 不愿鞠躬老板前: "
print "    奔驰宝马贵者趣, 公交自行程序员。 "
print "    别人笑我忒疯癫, 我笑自己命太贱: "
print "    不见满街漂亮妹, 哪个归得程序员?"
```

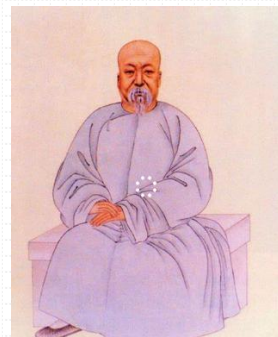
如果在开发程序时,需要某块代码多次,但是为了提高编写的效率以及代码的重用,所以把具有独立功能的代码块组织为一个小模块,这就是函数.

函数这个数学名词是**莱布尼兹**在**1694**年开始使用的，以描述曲线的一个相关量，如曲线的斜率或者曲线上的某一点。莱布尼兹所指的函数现在被称作可导函数，数学家之外的普通人一般接触到的函数即属此类。对于可导函数可以讨论它的极限和导数。此两者描述了函数输出值的变化同输入值变化的关系，是微积分学的基础。

中文的“函数”一词由清朝数学家李善兰译出。其《代数学》书中解释：“凡此變數中函（包含）彼變數者，則此為彼之函數”。



莱布尼兹, G. W.

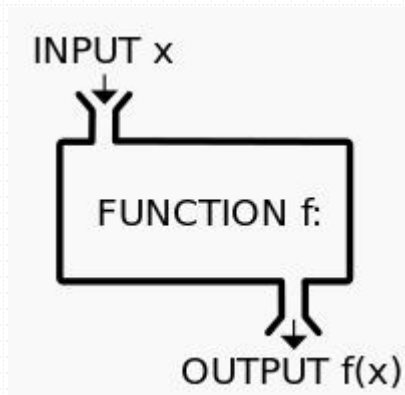


01

函数

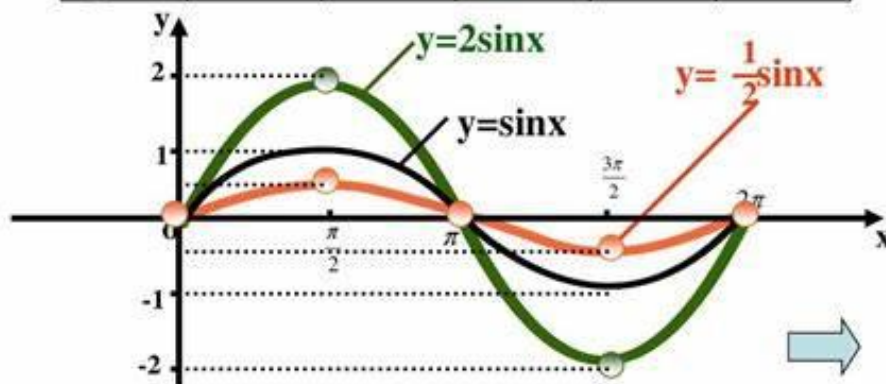
数学中的函数

初中数学水平都能理解一个大概。不管什么样子的函数，都可以用下图概括：



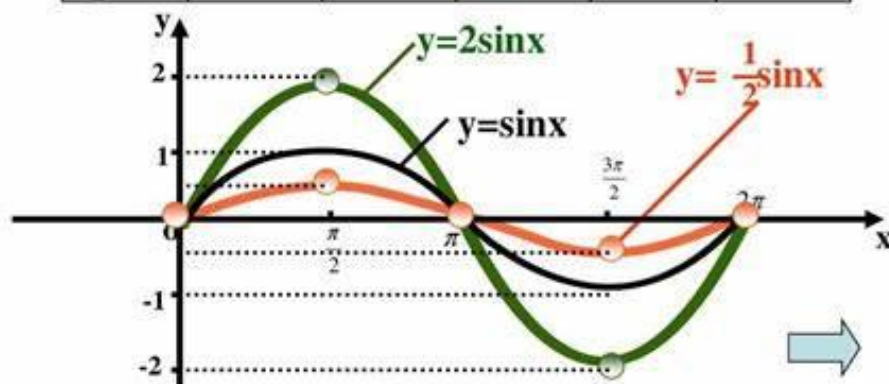
深入理解函数：变量 x 只能是任意数吗？

x	0	$\frac{\pi}{2}$	π	$\frac{3\pi}{2}$	2π
$\sin x$	0	1	0	-1	0
$2\sin x$	0	2	0	-2	0
$\frac{1}{2}\sin x$	0	$\frac{1}{2}$	0	$-\frac{1}{2}$	0



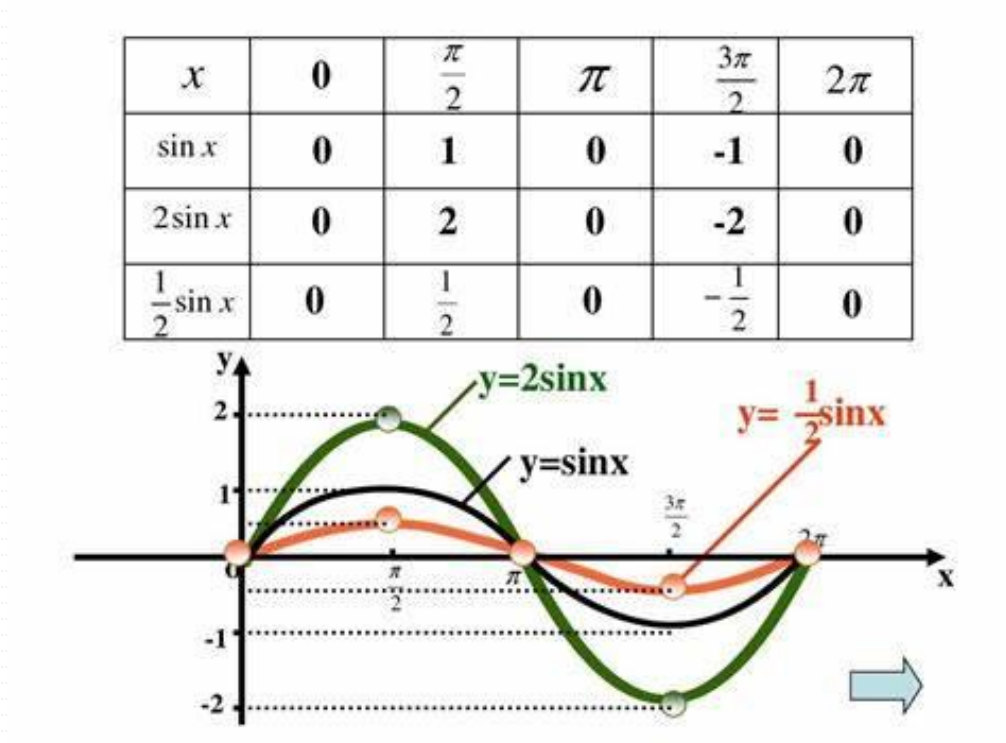
深入理解函数：变量 x 只能是任意数吗？

x	0	$\frac{\pi}{2}$	π	$\frac{3\pi}{2}$	2π
$\sin x$	0	1	0	-1	0
$2\sin x$	0	2	0	-2	0
$\frac{1}{2}\sin x$	0	$\frac{1}{2}$	0	$-\frac{1}{2}$	0



x ，不仅仅是数，可以是你认为的任何东西。

为什么需要函数？



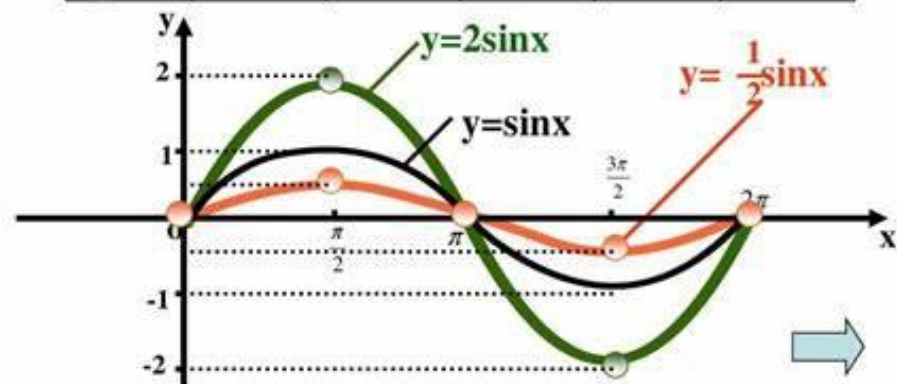
变量在本质上就是一个**占位符**。变量可以用 x ，也可以用别的符号，比如 y,z,k,i,j,\dots ，甚至用 α,β 这样的字母组合也可以。

01

函数

深入理解函数：函数中为什么变量用x？

x	0	$\frac{\pi}{2}$	π	$\frac{3\pi}{2}$	2π
$\sin x$	0	1	0	-1	0
$2\sin x$	0	2	0	-2	0
$\frac{1}{2}\sin x$	0	$\frac{1}{2}$	0	$-\frac{1}{2}$	0



函数的定义与调用

```
>>> def test(a,b):  
...     "用来完成对2个数求和"  
...     print("%d"%(a+b))  
...  
>>>  
>>> test(11,22)  
33
```

查看test函数文档

```
>>> help(test)
```

定义一个什么事也不做的空函数,可以用 `pass` 语句;

`pass` 可以用来作为占位符,还没想好怎么写函数的代码,就可以先放一个 `pass` ,让代码能运行起来

```
def get_passwd(): 1). 定义函数, 不执行代码块
    pass          2). pass只是一个占位符, 不做任何操作

if __name__ == "__main__":
    get_passwd()
```

```
def f2c(fahrenheit):  
    """  
    将华氏温度转换为摄氏温度  
    """  
    #计算摄氏度  
    celsius = (fahrenheit - 32)/1.8  
    return fahrenheit, celsius  
  
if __name__ == "__main__":  
    fahrenheit, celsius = f2c(30)  
    print('%.1f华氏度转为摄氏度为%.1f' %(fahrenheit, celsius))
```

所谓“返回值”，就是程序中函数完成一件事情后，最后给调用者的结果。

没有返回值，默认返回None

```
def 函数名(参数1, 参数2, ..., 参数n):
```

```
    函数体（语句块）
```

- 1)函数名的命名规则要符合python中的命名要求。一般用小写字母和单下划线、数字等组合
匈牙利命名法(sUserName), 驼峰式大小写(userName), 帕斯卡命名法(UserName)
- 2)def是定义函数的关键词, 这个简写来自英文单词define
- 3)函数名后面是圆括号, 括号里面, 可以有参数列表, 也可以没有参数
- 4)千万不要忘记了括号后面的冒号
- 5)函数体（语句块）, 相对于def缩进, 按照python习惯, 缩进四个空格

02

函数创建和调用

函数应用:打印图形和数学计算

```
def printOneLine():  
    """打印一条横线"""  
    print("-"*30)  
  
def printNumLine(num):  
    """打印多条横线"""  
    i=0  
    # 因为printOneLine函数已经完成了打印横线的功能,  
    # 只需要多次调用此函数即可  
    while i<num:  
        printOneLine()  
        i+=1  
  
if __name__ == "__main__":  
    printNumLine(3)
```

程序显示结果是什么?

```
def sum3Number(a,b,c):  
    """求3个数的和"""  
    return a+b+c # return 的后面可以是数值,也可是一个表达式  
  
def average3Number(a,b,c):  
    """完成对3个数求平均值"""  
    # 因为sum3Number函数已经完成了3个数的就和,所以只需调用即可  
    # 即把接收到的3个数,当做实参传递即可  
    sumResult = sum3Number(a,b,c)  
  
    aveResult = sumResult/3.0  
    return aveResult  
  
if __name__ == "__main__":  
    # 调用函数,完成对3个数求平均值  
    result = average3Number(11,2,55)  
    print("average is %d"%result)
```

函数的嵌套调用的执行过程?

从理论上说，不用函数，也能够编程，我们在前面已经写了程序，就没有写函数，当然，用python的内建函数姑且不算了。现在之所以使用函数，主要是：

1. 降低编程的难度(分而治之的思想)
2. 代码重用。避免了重复劳动，提供了工作效率。


```
def save_money(money):  
    """存钱"""  
    allMoney = 100  
    print("存钱前:", allMoney)  
    allMoney += money  
    print("存钱后:", allMoney)  
  
def view_money():  
    """查询金额"""  
    allMoney = 500  
    print("查询金额:", allMoney)  
  
if __name__ == '__main__':  
    save_money(50)  
    view_money()
```

局部变量

```
def save_money(money):  
    """存钱"""  
    allMoney = 100  
    print("存钱前:", allMoney)  
    allMoney += money  
    print("存钱后:", allMoney)  
  
def view_money():  
    """查询金额"""  
    allMoney = 500  
    print("查询金额:", allMoney)  
  
if __name__ == '__main__':  
    save_money(50)  
    view_money()
```

局部变量

```
存钱前：100  
存钱后：150  
查询金额：500
```

- 局部变量,就是在函数内部定义的变量
- 不同的函数,可以定义相同的名字的局部变量,但是各用个的不会产生影响
- 局部变量的作用,为了临时保存数据需要在函数中定义变量来进行存储,这就是它的作用

如果一个变量,既能在一个函数中使用,也能在其他的函数中使用,这样的变量就是全局变量。

```
# 定义全局变量
allMoney = 100

def save_money(money):
    """存钱"""
    global allMoney
    print("存钱前:", allMoney)
    allMoney += money
    print("存钱后:", allMoney)

def view_money():
    """查询金额"""
    #allMoney = 500
    print("查询金额:", allMoney)

if __name__ == '__main__':
    save_money(50)
    view_money()
```

如果一个变量,既能在一个函数中使用,也能在其他的函数中使用,这样的变量就是全局变量。

```
# 定义全局变量
allMoney = 100

def save_money(money):
    """存钱"""
    global allMoney
    print("存钱前:", allMoney)
    allMoney += money
    print("存钱后:", allMoney)

def view_money():
    """查询金额"""
    #allMoney = 500
    print("查询金额:", allMoney)

if __name__ == '__main__':
    save_money(50)
    view_money()
```

```
存钱前： 100
存钱后： 150
查询金额： 150
```

- 在函数外边定义的变量叫做全局变量
- 全局变量能够在所有的函数中进行访问
- 如果在函数中修改全局变量,那么就需要使用`global`进行声明,否则出错
- 如果全局变量的名字和局部变量的名字相同,那么使用的是局部变量的
- 小技巧: 强龙不压地头蛇(就近原则)

不使用global声明全局变量时不能修改全局变量？

```
# 定义全局变量
allMoney = 100
operator = []

def save_money(money):
    """存钱"""
    global allMoney
    print("存钱前:", allMoney, operator)
    allMoney += money
    # 为什么operator不需要声明为全局变量?
    operator.append('存钱操作')
    print("存钱后:", allMoney, operator)

def view_money():
    """查询金额"""
    #allMoney = 500
    operator.append("查询金额操作")
    print("查询金额:", allMoney, operator)

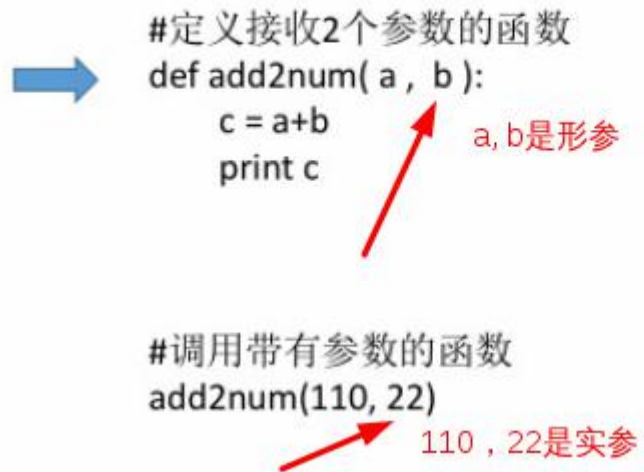
if __name__ == '__main__':
    save_money(50)
    view_money()
```

不使用`global`声明全局变量时不能修改全局变量？

```
存钱前：100 []  
存钱后：150 ['存钱操作']  
查询金额：150 ['存钱操作', '查询金额操作']
```

`global`的本质是声明可以修改全局变量的指向，即变量可以指向新的数据。

- 1). 不可变类型的全局变量: 指向的数据不能修改, 不使用`global`时无法修改全局变量。
- 2). 可变类型的全局变量: 指向的数据可以修改, 不使用`global`时可以修改全局变量。



```
#定义接收2个参数的函数
def add2num( a , b ):
    c = a+b
    print c

#调用带有参数的函数
add2num(110, 22)
```

a, b是形参

110, 22是实参

定义时小括号中的参数,用来接收参数用的,称为“形参”

调用时小括号中的参数,用来传递给函数用的,称为“实参”

调用函数时,如果参数个数不对,Python 解释器会自动检查出来,并抛出 `TypeError`;

- 如果参数类型不对,Python 解释器就无法帮我们检查。
- 数据类型检查可以用内置函数 `isinstance` 实现

需求: 定义一函数，计算x值的n次方。那如果计算x平方时只需要传入x值时怎么解决？

- 默认参数可以降低调用函数的难度。
 - 默认参数注意事项：
 - 有多个参数时,变化大放前面,变化小的放后面;
 - 必选参数在前,默认参数在后

默认函数容易出错点:

试一试:先定义一个函数,传入一个 `list`,添加一个 `END` 再返回.

可变参数就是传入的参数个数是可变的,可以是 1 个、2 个到任意个,还可以是 0 个。*args

- 以数学题为例子,给定一组数字 a,b,c,\dots ,
- 请计算 $a^2 + b^2 + c^2 + \dots$

如果已经有一个 list 或者 tuple,要调用一个可变参数怎么办?

- 1). Python 允许你在 list 或 tuple 前面加一个 * 号;
- 2). 把 list 或 tuple 的元素变成可变参数传进去;

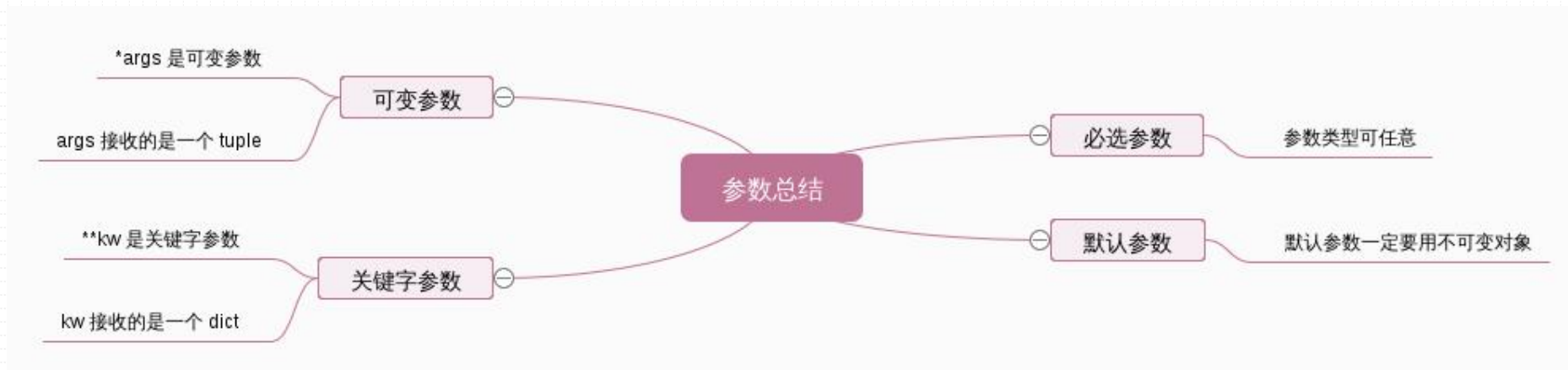
```
largs = [1,2,3]
```

```
func(largs[0],largs[1],largs[2])
```

```
func(*largs)
```

- 关键字参数允许传入 0 个或任意个含参数名的参数;
- 这些关键字参数在函数内部自动组装为一个 dict;
- 关键字参数用**kwargs;

- 参数组合是指可以必选参数、默认参数、可变参数和关键字参数一起使用。
- 参数定义的顺序必须是:必选参数、默认参数、可变参数和关键字参数。



对于任意函数,都可以通过类似 `func(*args, **kw)` 的形式调用它。

匿名函数指一类无须定义标识符的函数或子程序。Python用lambda语法定义匿名函数，只需用表达式而无需申明。(省略了用def声明函数的标准步骤)

```
# 定义匿名函数
sum = lambda arg1, arg2: arg1 + arg2

#调用sum函数
print("Value of total : ", sum( 10, 20 ))
print("Value of total : ", sum( 20, 20 ))
```

```
Value of total : 30
Value of total : 40
```


lambda函数的语法只包含一个语句,如下:

```
lambda [arg1 [,arg2,.....argn]]:expression
```

lambda函数能接收任何数量的参数但只能返回一个表达式的值

匿名函数

应用场合1: 函数作为参数传递

```
def fun(a, b=2, myfun=pow):  
    """a接收, b接收, fun接收函数名"""  
    print('a =', a, end='; ')  
    print('b =', b, end='; ')  
    print('result = ', myfun(a, b))
```

```
fun(5)  
fun(5, 3)  
fun(5, 3, lambda x, y: x ** y)  
fun(5, 3, lambda x, y: x + y)
```

```
a = 5; b = 2; result = 25  
a = 5; b = 3; result = 125  
a = 5; b = 3; result = 125  
a = 5; b = 3; result = 8
```

```

# 用来生成美观的ASCII格式的表格的第三方模块
import prettytable as pt

def show(goods):
    """友好的显示商品信息"""
    # 实例化表格对象
    table = pt.PrettyTable()
    # 表头信息设置
    table.field_names = ["Name", "Count", "Price"]
    # 依次添加每一行信息到表格中;
    for good in goods:
        table.add_row(good)
    print(table)

goods = [
    # 商品名称 商品数量 商品价格
    ('Python核心编程', 200, 70.30),
    ('Java核心编程', 40, 54.40),
    ('Php核心编程', 40, 85.30),
    ('Ruby核心编程', 1000, 50.60),
]

# 1). 按照商品数量进行排序
print("按照商品数量进行排序")
goods.sort(key=lambda x: x[1])
show(goods)

# 2). 按照商品价格进行排序
print("按照商品价格进行排序")
goods.sort(key=lambda x: x[2])
show(goods)

```

按照商品数量进行排序

Name	Count	Price
Java核心编程	40	54.4
Php核心编程	40	85.3
Python核心编程	200	70.3
Ruby核心编程	1000	50.6

按照商品价格进行排序

Name	Count	Price
Ruby核心编程	1000	50.6
Java核心编程	40	54.4
Python核心编程	200	70.3
Php核心编程	40	85.3

问题描述1: 有一个整数列表(10个元素), 要求调整元素顺序, 把所有的奇数放在前面, 偶数放在后面,

问题描述2: (2018-携程-春招题)

给定一个整形数组, 将数组中所有的0移动到末尾, 非0项保持不变;

在原始数组上进行移动操作, 勿创建新的数组;

#输入: 数组的记录;0 7 0 2

#输出: 调整后数组的内容; 7 2 0 0

已知: 函数可以调用函数。结论: 一个函数在内部**调用自己**本身, 这个函数就是递归函数。

需求:

计算阶乘 factorial: $n! = 1 * 2 * 3 * \dots * n$

```
def factorial(num):  
    """非递归方法求num的阶乘"""  
    # 初始化结果为1  
    result = 1  
    # 依次遍历1, 2, 3, ..... num  
    for item in range(1, num+1):  
        # result = 1 * 2 * 3 * ..... * num  
        result *= item  
    # 返回阶乘的结果  
    return result  
  
if __name__ == '__main__':  
    print("2的阶乘:", factorial(2))  
    print("3的阶乘:", factorial(3))  
    print("5的阶乘:", factorial(5))
```

```
2的阶乘: 2  
3的阶乘: 6  
5的阶乘: 120
```

1. 阶乘的规律

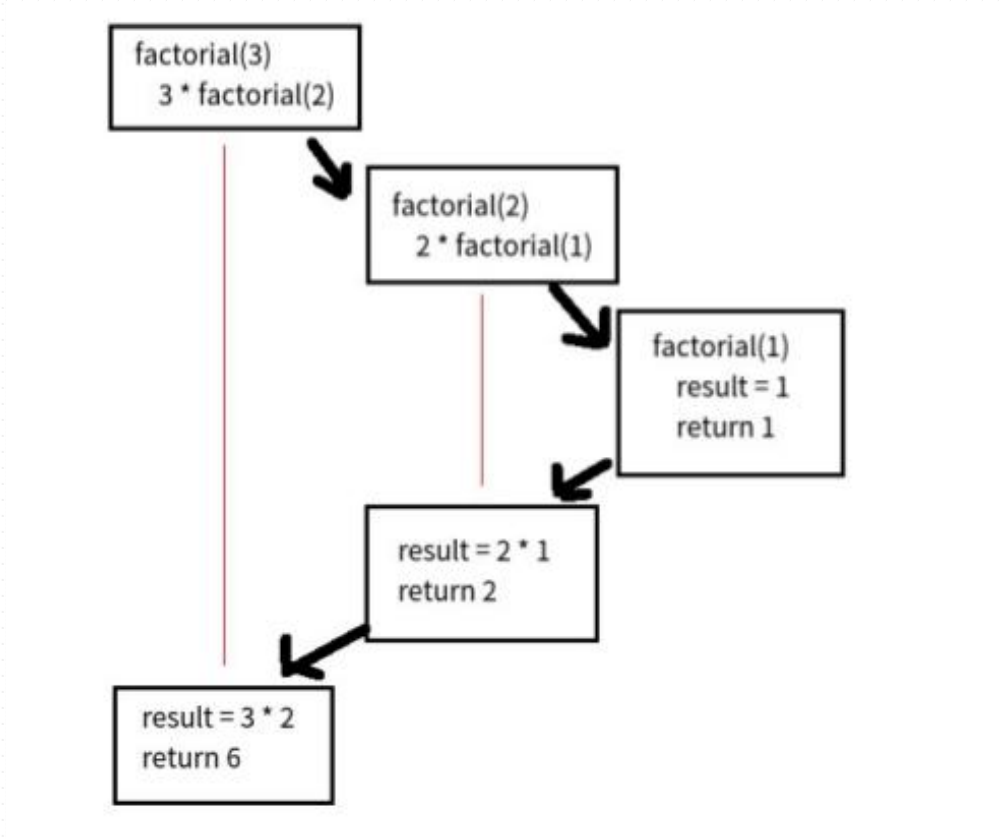
```
1! = 1
2! = 2 × 1 = 2 × 1!
3! = 3 × 2 × 1 = 3 × 2!
4! = 4 × 3 × 2 × 1 = 4 × 3!
...
n! = n × (n-1)!
```

2. 代码实现

```
def factorial(num):
    """递归方法求num的阶乘"""
    # 递归一定要有跳出条件；
    if num >= 1:
        result = num * factorial(num - 1)
    else:
        result = 1
    return result

if __name__ == '__main__':
    print("2的阶乘:", factorial(2))
    print("3的阶乘:", factorial(3))
    print("5的阶乘:", factorial(5))
```

```
2的阶乘: 2
3的阶乘: 6
5的阶乘: 120
```



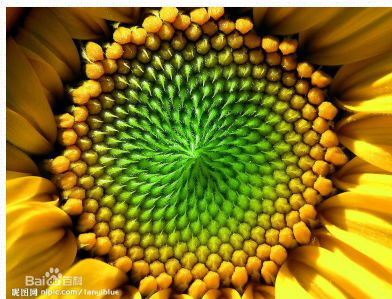
常用的递归函数: 斐波那契数列([视频链接](#))(1分18秒到2分06秒)

斐波那契数列 (Fibonacci sequence)，又称黄金分割数列，指的是这样一个数列：1、1、2、3、5、8、13、21、34、.....在数学上，斐波那契数列以如下被以递推的方法定义：

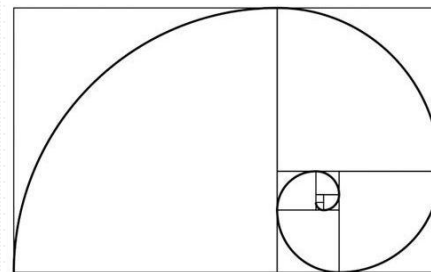
$$F(1)=1, F(2)=1, F(n)=F(n-1)+F(n-2) \quad (n \geq 3, n \in \mathbb{N}^*)$$



斐波那契

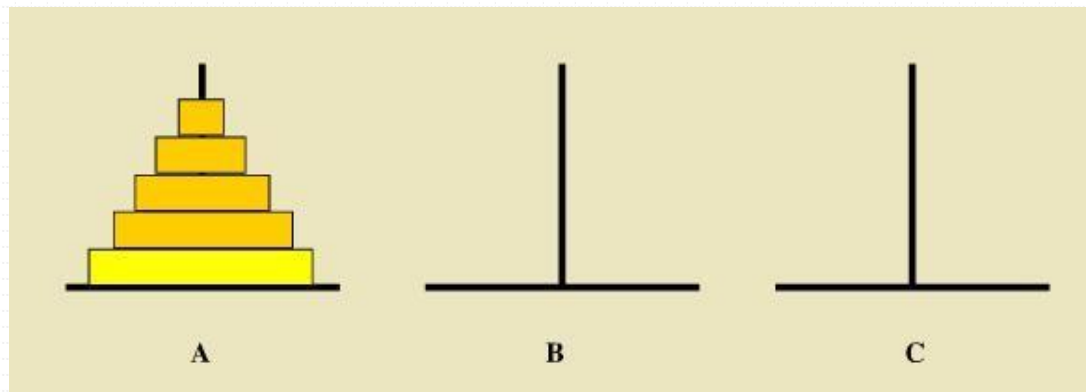


自然界中“巧合”



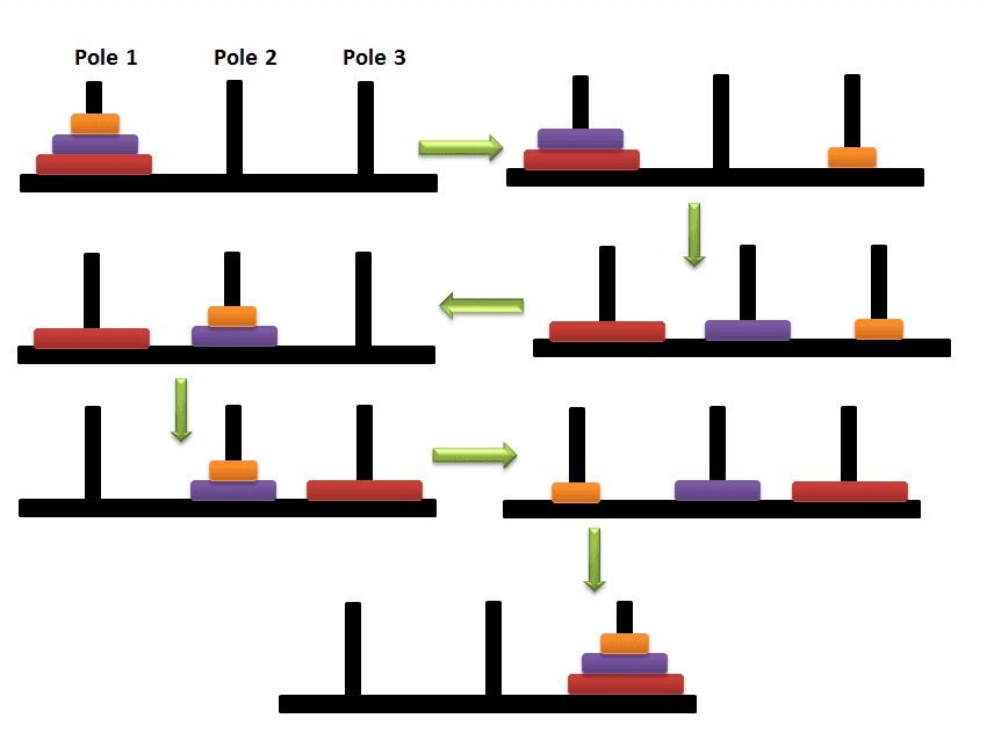
斐波那契弧线

印度的古老传说： 在世界中心贝拿勒斯（在印度北部）的圣庙里，一块黄铜板上插着三根宝石针。印度教的主神梵天在创造世界的时候，在其中一根针上从下到上地穿好了由大到小的64片金片，这就是所谓的汉诺塔。不论白天黑夜，总有一个僧侣在按照下面的法则移动这些金片：**一次只移动一片**，不管在哪根针上，**小片必须在大片**上面。僧侣们预言，当所有的金片都从梵天**穿好的那根针上移到另外一根针**上时，世界就将在一声霹雳中消灭，而梵塔、庙宇和众生也都将同归于尽。



递归函数

常用的递归函数: 汉诺塔问题([在线游戏地址](#))



n代表汉诺塔盘子的个数

$$n=1, \quad \text{sum} = 1(2^1-1) \text{次}$$

$$n=2, \quad \text{sum} = 3(2^2-1) \text{次}$$

$$n=3, \quad \text{sum} = 7(2^3-1) \text{次}$$

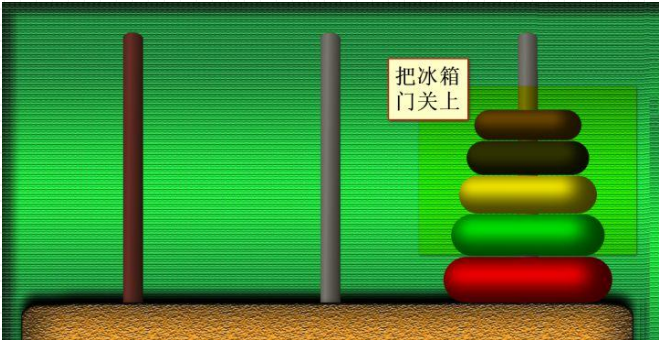
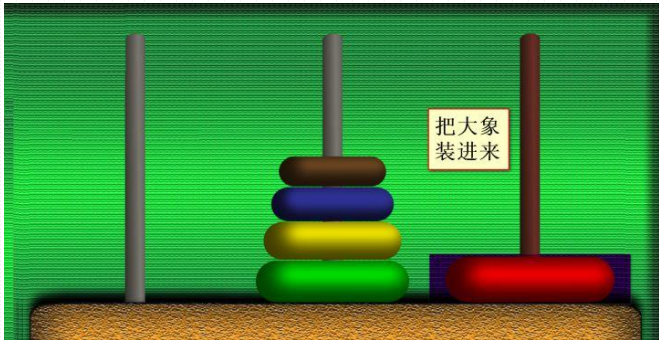
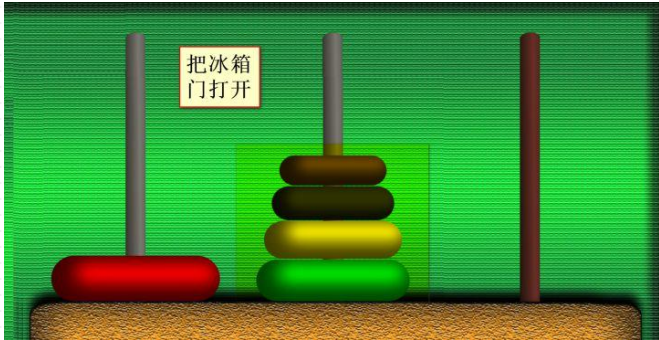
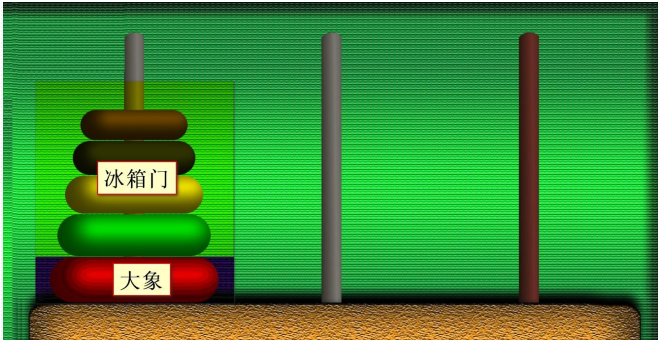
....

$$n= n, \quad \text{sum} = 2^n - 1$$

递归函数

常用的递归函数: 汉诺塔问题

算法思路:



对于问题N, 如果N-1已经解决了, 那么N是否很容易解决?

算法思路:

1. 编程实现 9*9乘法表(循环嵌套的复习)。

2. 用函数实现求100-200里面所有的素数。

提示:素数的特征是除了1和其本身能被整除,其它数都不能被整除的数

3. 请用函数实现一个判断用户输入的年份是否是闰年的程序。

提示:能被4整除,但是不能被100整除的年份或者能被400整除的年份.

算法思路:

1. 用函数实现输入某年某月某日,判断这一天是这一年的第几天?闰年情况也考虑进去。

2. 编写“学生管理系统”,要求如下:

- 必须使用自定义函数,完成对程序的模块化
- 学生信息至少包含:姓名、年龄、学号,除此以外可以适当添加
- 必须完成的功能:添加、删除、修改、查询、退出

题目需求:对于一个十进制的正整数, 定义 $f(n)$ 为其各位数字的平方和, 如:

$$f(13) = 1^2 + 3^2 = 10$$

$$f(207) = 2^2 + 0^2 + 7^2 = 53$$

下面给出三个正整数 k, a, b , 你需要计算有多少个正整数 n 满足 $a \leq n \leq b$, 且 $k \cdot f(n) = n$

输入: 第一行包含3个正整数 k, a, b , $k \geq 1, a, b \leq 10^{18}, a \leq b$;

输出: 输出对应的答案;

范例:

输入: 51 5000 10000

输出: 3

题目描述:给定一个正整数，编写程序计算有多少对质数的和等于输入的这个正整数，并输出结果。输入值小于1000。

如，输入为10, 程序应该输出结果为2。（共有两对质数的和为10,分别为(5,5),(3,7)）

输入描述: 输入包括一个整数n,($3 \leq n < 1000$)

输出描述: 输出对数

示例1:

输入: 10

输出: 2

感谢聆听！

THANK YOU!