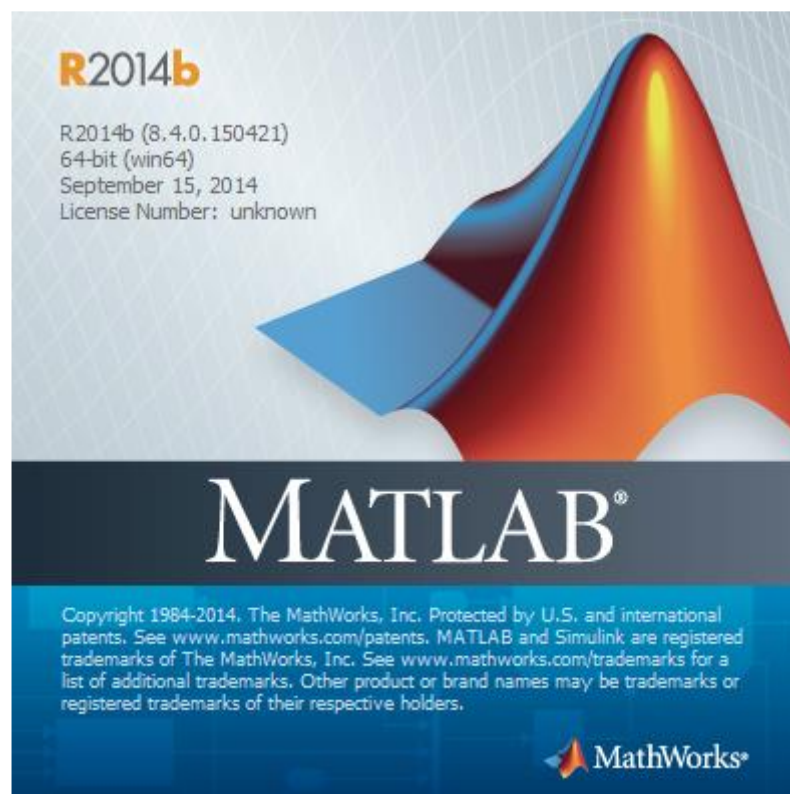


## 第4章

# MATLAB软件与应用



# 第四章 程序设计

- M文件
- 程序结构
- 函数及其调用
- 程序的调试

# M文件

# 程序设计 M文件

Mtatlalab既有传统高级语言的特征，又有独特优点，充分利用Matlab数据结构的特点，可以简化程序结构、提高编程效率。

## MATLAB的三种典型模式：

- M文件源代码（效率较高）
- 命令行窗口（简单问题适用）
- 其他形式（带图形界面的部分工具箱等）

# 程序设计 M文件

**2类M文件：** 命令/脚本文件， 函数文件

**脚本文件：**

又称**命令文件**，没有输入参数，不返回输出参数，相当于命令窗口多条命令的集合。

**函数文件：**

有或没有输入参数，返回或不返回输出参数；

类似于C/C++等编程语言的子程序；

函数文件相当于黑盒子，具有独立的工作空间；

通过输入/输出参数可以同外界交换信息；

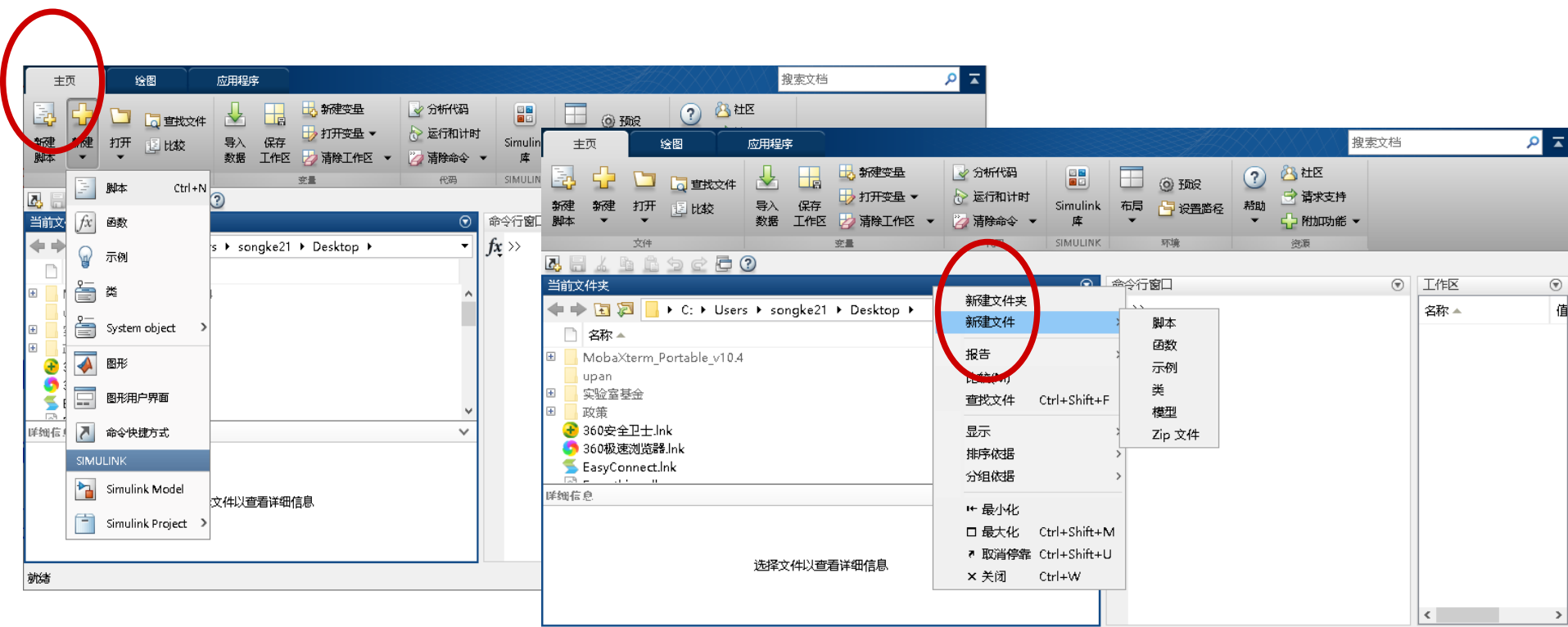
函数文件需要以**function**开始，需具有函数名。

# 程序设计 M文件创建方式

GUI方式：文件或主页-新建-脚本或函数  
或 主页菜单中的“新建脚本”

或 在当前文件夹窗口点右键，选“新建文件”

命令方式：命令窗口键入“edit 文件名”或“edit”



# 程序设计 M文件打开方式

GUI方式：文件或主页-打开，

或 在当前文件夹窗口中直接选中指定文件

命令方式：命令窗口键入“edit 文件名”

如果该文件位于当前文件夹，则可以正常打开，  
否则需要加上路径，例如

```
edit C:\myfile.m
```

其他方式：在win系统中找到\*.m文件直接双击打开。

# 程序设计 M文件运行方式

- 当前文件夹窗口中右键点击指定文件，选“运行”
  - 在打开M文件后，选菜单“编辑器”-“运行”
  - 在命令窗口或其他程序中通过文件名（不带.m后缀），可直接运行该文件；
- 
- 以上三类方式用于函数文件要注意，是否有正确的输入输出
  - 函数文件一般调用格式为[输出列表]=函数名(输入列表)，这种方式可以在命令窗口、脚本文件、其他函数文件中使用，仍然要注意输入和输出是否正确。

参考示例文件 `mysind.m` `myplotsin.m` `mydeg2rad.m`



# 程序设计    程序结构

- 顺序结构
- 选择/条件结构（if , switch , try）
- 循环结构（while, for）

以上3种基本结构可组成很复杂的程序。

# 程序设计      顺序结构

按语句的排列顺序依次执行。

一般涉及输入和输出、简单计算或处理等。

## 1、input函数可实现从键盘输入数据

```
A=input(提示信息, 选项)
```

提示信息为字符串，提示用户输入什么样的数据。

```
A=input(‘输入A矩阵:’)
```

执行该命令，首先会在命令窗口看到提示信息，然后可在命令窗口键入数据对A赋值

# 程序设计      顺序结构

## 2、用于命令窗口输出的函数主要有disp

disp(输出项)

【例】      `A=' Good afternoon everyone. ' ;`  
             `disp(A)`

输出结果:    Good afternoon everyone.

【例】      `A=[1 2 3;4 5 6;7 8 9];`  
             `disp(A)`

输出结果:

1	2	3
4	5	6
7	8	9

# 程序设计      顺序结构

【例】：求解一元二次方程  $ax^2 + bx + c = 0$

```
a=input('a=');  
b=input('b=');  
c=input('c=');  
d=b^2-a*4*c;  
x=[(-b+sqrt(d))/(2*a),(-b-sqrt(d))/(2*a)]; %用户输入解表达式  
disp(['x1=',num2str(x(1)), ' x2=',num2str(x(2))])
```

a=4

b=78

c=54

x1=-0.7188   x2=-18.7812

# 程序设计      选择结构

也称条件结构，根据给定条件成立或不成立，分别执行不同的语句。

①if语句

②switch语句

③try语句

# 程序设计 选择结构 if

## ① 单分支if语句

```
if 条件  
    语句组  
end
```

## ② 双分支if语句

```
if 条件  
    语句组1  
else  
    语句组2  
end
```

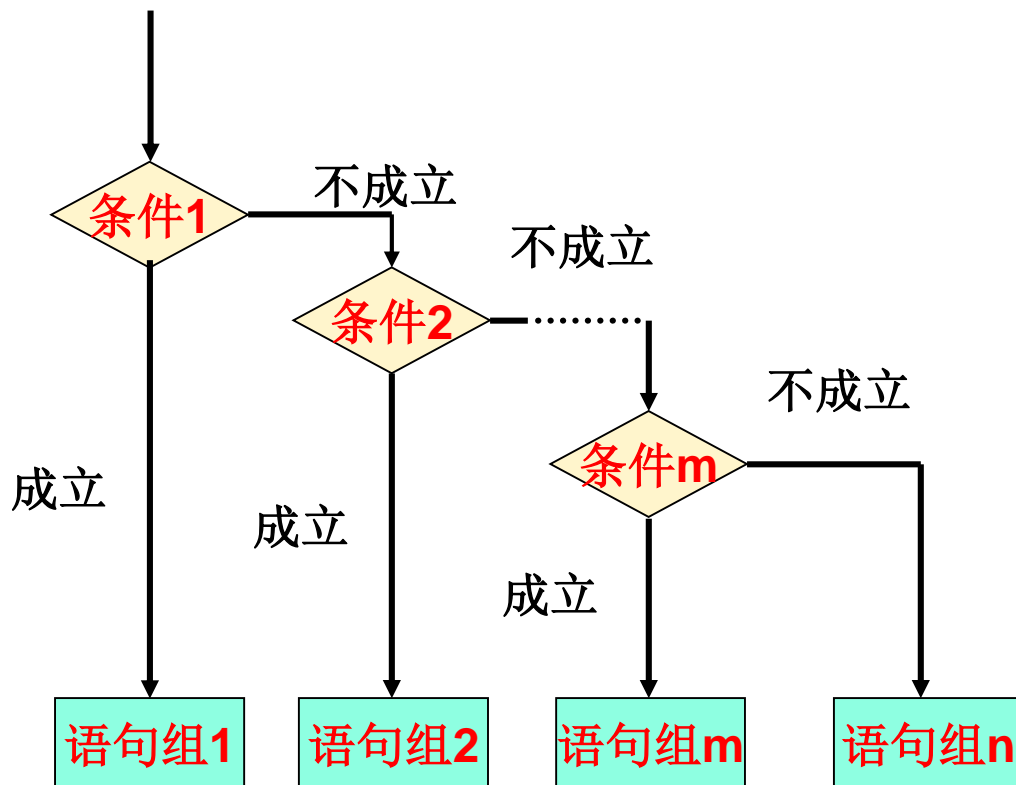
## ③ 多分支if语句

```
if 条件1  
    语句组1  
elseif 条件2  
    语句组2  
.....  
elseif 条件m  
    语句组m  
else  
    语句组n  
end
```

语句组也允许  
含有其他if结构

# 程序设计 选择结构 if

多分支if语句



# 程序设计 选择结构 if

例：计算分段函数  $y = \begin{cases} \cos(x+1) + \sqrt{x^2+1} & x=10 \\ x\sqrt{x+\sqrt{x}} & x \neq 10 \end{cases}$

```
x=input('请输入X的值:');  
y=cos(x+1)+sqrt(x*x+1);  
if x~=10 %单分支if  
    y=x*sqrt(x+sqrt(x));  
end
```

```
x=input('输入X的值:');  
if x==10 %双分支if  
    y=cos(x+1)+sqrt(x*x+1);  
else  
    y=x*sqrt(x+sqrt(x));  
end
```

```
x=input('输入X的值:');  
if x~=10 %双分支if  
    y=x*sqrt(x+sqrt(x));  
else  
    y=cos(x+1)+sqrt(x*x+1);  
end
```



# 程序设计 选择结构 switch

根据表达式的取值不同，分别执行不同的语句

```
switch 表达式
```

```
case 值1
```

```
    语句组1
```

```
case 值2
```

```
    语句组2
```

```
...
```

```
otherwise %可以空缺
```

```
    语句组n %可以空缺
```

```
end
```



一个或多个值

# 程序设计 选择结构 switch

例：按照以下消费价值计算折扣率

$price < 200$	没有折扣
$200 \leq price < 500$	3% 折扣
$500 \leq price < 1000$	5% 折扣
$1000 \leq price < 2500$	8% 折扣
$2500 \leq price < 5000$	10% 折扣
$5000 \leq price$	14% 折扣

# 程序设计 选择结构 switch

```
price=input('请输入商品价格:')
```

```
switch fix(price/100) %与fix相似的函数还有floor或ceil
```

```
    case {0,1}          %商品价格小于200
```

```
        rate=0;
```

```
    case{2,3,4}          %商品价格大于200小于500
```

```
        rate=3/100;
```

```
    case num2cell(5:9) %商品价格大于500小于1000
```

```
        rate=5/100;
```

```
    case num2cell(10:24) %商品价格大于1000小于2500
```

```
        rate=8/100;
```

```
    case num2cell(25:49) %商品价格大于2500小于5000
```

```
        rate=10/100;
```

```
    otherwise          %商品价格大于等于5000
```

```
        rate=14/100;
```

```
end
```

```
fprintf('折扣率=%f', rate*100),disp('%') %输出折扣率
```

## 一种试探执行语句

```
try  
    语句组1  
catch  
    语句组2  
end
```

先试探执行 语句组1，如果在执行过程中出现错误，则将信息保留到lasterr变量，并转去执行 语句组2。

# 程序设计

## 选择结构

### try

例：矩阵乘法要求两矩阵形状满足一定要求。

```
A=[1 2 3;4 5 6];  
B=[7 8 9;10 11 12];  
try  
    C=A*B;  
catch  
    C=A.*B;  
end  
C,lasterr
```

输出结果

```
C =  
    7    16    27  
   40    55    72  
ans =  
Error using ==> mtimes  
Inner matrix dimensions must agree.
```

# 程序设计 循环结构 for 和 while

按照给定的条件，重复执行指定语句。

```
for 变量=初值:增量:结束值  增量默认为1  
    语句组  
end
```

```
while 逻辑表达式  满足条件才执行  
    语句组  
end
```

# 程序设计 循环结构 for

例：1~100中奇数求和

```
%for循环求和  
s=0;  
for n=1:2:100;  
    s=s+n;  
end  
s
```

```
%while循环求和  
s=0;  
n=1;  
while n<=100  
    s=s+n;  
    n=n+2;  
end  
s
```

```
%专用函数求和  
x=1:2:100;  
s=sum(x)
```

# 程序设计 循环结构 while

例：从键盘输入若干个数，当输入0时结束运算，并求这些数的平均值以及它们的和。

```
s=0;
n=0;
x=input('输入一个数(以0结束):');
while(x~=0)
    s=s+x;
    n=n+1;
    x=input('输入一个数(以0结束):');
end
if (n>0)
    s
    mean=s/n
end
```



# 程序设计 流程控制

① break

② continue

③ pause

④ keyboard

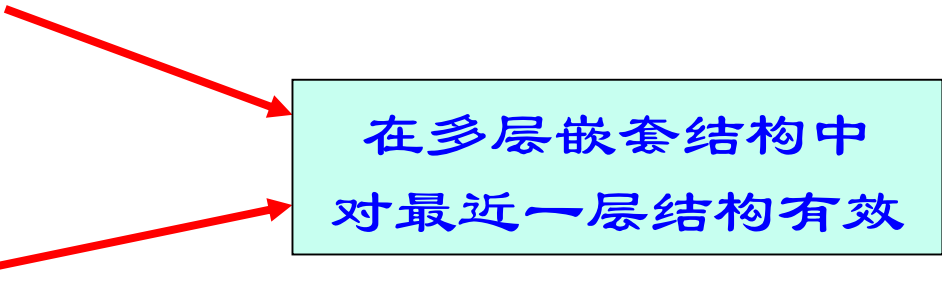
其他 input return error warning echo

## break

完全退出**当前**for或while循环结构。

## continue

结束**本轮**for或while循环，继续进行下轮循环。



在多层嵌套结构中  
对最近一层结构有效

# 程序设计    流程控制

例：输出103到120之间的第一个能被7整除的整数

```
for n=103:120
    if rem(n, 7) ~= 0
        continue %不能整除
        disp('b') %b从不显示出来
    end
    disp('a')
    break %得到第一个数就终止
end
n
```

结果是

a

n=105

```
for n=103:120
    if rem(n, 7) == 0
        break %能整除
    end
    disp('a')
end
n
```

结果是

a

a

n=105

# 程序设计 流程控制

`pause(n秒)` 执行至此延迟n秒后继续执行

若缺省秒数，则一直延迟，直到用户按任意键再继续。  
在命令窗口按`Ctrl+C`强行终止程序的执行。

`keyboard` 遇到该指令，当前程序流程暂停，转到命令窗口等待用户输入和执行其他指令，直到命令窗口中输入和执行`return`，就再次返回原程序继续执行。

`echo` 控制在M文件执行中是否显示每条命令，请参考 `help echo`

# 程序设计 流程控制

## error 和 warning

（可在程序中设置检查环节）

【例】输入半径，计算圆周长。

```
r=input('请输入半径') %尝试输入 0, -1和10查看运行结果
if r==0
    warning('半径为零, 无限小的圆') %提示并继续运行
elseif r<0
    error('半径为负, 无法计算') %提示并终止运行
end
c=2*pi*r;
disp('周长为')
c
```

# 程序设计 函数文件

M文件中有一类是函数文件，类似于C/C++等编程语言的子程序；函数文件相当于黑盒子，具有独立的工作空间；通过输入/输出参数可以同外界交换信息；matlab大部分内置函数例如sin也是以函数文件形式存在。

## 需要关注以下事项

- 函数的输入与输出
- 函数文件名和函数名
- 函数文件的搜索路径
- 函数的注释说明
- 函数中的return语句
- 函数调用和参数传递
- 函数的递归调用
- 主函数与子函数
- 局部变量和全局变量

# 程序设计 函数文件的输入/输出

而不是必须

可以输入参数，可以返回输出参数。

类似于C等编程语言的子程序，胜任复杂问题。

无返回值函数

**function** 函数名(输入形参表)

...

有返回值函数

**function** [输出形参表]=函数名(输入形参表)

...

无输入值函数

**function** [输出形参表]=函数名()

...

# 程序设计    函数文件名与函数名

- Matlab以函数文件名来识别函数
- 函数文件名建议命名为： 函数名 + 后缀.m
- 函数文件名区分大小写，但一般建议为小写
- 函数文件名应避免同matlab内置函数重名
- 函数文件应存储于当前路径或搜索路径

# 程序设计 函数文件的注释说明

函数文件的基本注释说明不是必须的，一般有三部分

① 紧随引导行之后以%开头的第一注释行。

一般包括大写的函数文件名和函数功能简要描述，供lookfor关键词查询和help在线帮助时使用。

② 第一注释行及之后连续的注释行。

通常包括函数输入/输出参数的含义及调用格式说明等信息，构成全部在线帮助文本。

③ 与在线帮助文本相隔一空行的注释行。

包括函数文件编写和修改的信息，如作者和版本等。



# 程序设计

## 函数文件中的return

在函数文件中，遇到return语句就结束函数的执行，转到调用该函数的位置，执行之后的语句。

return在函数文件中不是必须的。

例：编写函数文件，求半径为r的圆的面积和周长。

```
function [s,p] = mycircle(r)
% MYCIRCLE calculate the area and perimeter of a circle
% r      圆半径
% s      圆面积
% p      圆周长
    s = pi*r*r;
    p = 2*pi*r;
    return;
    disp( 'return之后的语句不会执行' )
```

# 程序设计      函数文件中的return

以上保存为文件mycircle.m，在命令窗口调用。

```
[s,p] = mycircle(10)
```

输出结果是：

```
s =
```

```
314.1593
```

```
p =
```

```
62.8319
```

采用help, doc或lookfor命令可显示出注释说明部分的内容。

```
help mycircle
```

显示信息为

```
MYCIRCLE calculate the area and perimeter of a circle
```

```
r      圆半径
```

```
s      圆面积
```

```
p      圆周长
```

# 程序设计      函数调用和参数传递

函数文件的调用格式：

[输出1, 输出2, ...]=函数名 (输入1, 输入2, ...)

- 函数文件具有独立工作空间
- 通过输入、输出参数实现函数内外的数据传递
- 输入参数在函数中的任何赋值，不会传递出去

本质上是变量值的传递，而不是变量地址传递（如fortran语言）

# 程序设计

# 函数调用和参数传递

【例】用matlab编写M文件和M函数来绘制函数在区间 $[-6, 6]$ 中的图形。

$$y(x) = \begin{cases} \sin x & x \leq 0 \\ x & 0 < x \leq 3 \\ -x + 6 & x > 3 \end{cases}$$

```
%脚本文件命名为myscripl.m
```

```
tic
```

```
x=[-6:0.1:6];leng=length(x);
```

```
for m=1:leng;
```

```
    if x(m)<=0
```

```
        y(m)=sin(x(m));
```

```
    elseif x(m)<=3
```

```
        y(m)=x(m);
```

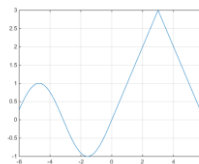
```
    else y(m)=-x(m)+6;
```

```
    end
```

```
end
```

```
plot(x,y); grid;
```

```
toc
```



# 程序设计

# 函数调用和参数传递

调用函数使用该名字

%这是一个函数文件，文件名为myf1.m

**function y=f1(x)**

**y=(x<=0).\*sin(x)+(x>=0&x<=3).\*x+(x>3).\*(-x+6);**

%灵活运用逻辑运算可使程序简化

%以下可在命令窗口运行，或通过命令/脚本文件运行

**x=-6:0.1:6;**

**y=myf1(x);** %调用函数myf1.m

**plot(x,y); grid**

编写函数程序，保存为myf1.m并置于当前文件路径中，在命令窗口输入**y=myf1(x)**来调用该函数。（输入参数x要预先就绪）。

# 程序设计      函数调用和参数传递

函数文件的输入/输出参数数目可以改变

调用函数时，两个预定义变量nargin和nargout分别记录实际输入和输出的参数数量。

例 nargin用法示例，函数文件examp.m:

```
function [x,y] = myf2(a,b,c)
fprintf('nargin=%d\n',nargin);
fprintf('nargout=%d\n',nargout);
if nargin==0 x=0;end
if nargin==1 x=a;y=99;end
if nargin==2 x=a+b;end
if nargin==3 x=a+b+c;end
```

%调用myf2

[x]=myf2() %x=0，或不写()

[x, y]=myf2() %调用错误

[x]=myf2(2, 3, 4) %x=9

[x, y]=myf2(2) %x=2, y99

myf2(2) %结果ans=2

# 程序设计      函数调用和参数传递

函数所传递参数也可以是矩阵

例：编写函数文件，计算圆的周长和面积

函数文件如下：

```
function [s,p] = mycircle(r)
% MYCIRCLE calculate the area and perimeter of a circle
% r          圆半径
% s          圆面积
% p          圆周长
    s = pi*r.*r;  %注意元素乘法
    p = 2*pi*r;
```

%调用该函数

```
[s,p] = mycircle([1,3,5])
```

```
[s,p] = mycircle(1:2:5)
```

# 程序设计      函数的递归调用

函数可以嵌套调用，即一个函数可以调用别的函数。  
函数甚至可以调用自身，称为**递归调用**。

例：利用函数的递归调用，求n！。

n！本身就是以递归的形式定义的：

$$n! = \begin{cases} 1, & n \leq 1 \\ n(n-1)!, & n > 1 \end{cases}$$

求n！，就要求(n-1)！，可采用递归调用。函数myfactor.m如下：

```
function f = myfactor(n)
if n<=1
    f = 1;
else
    f = myfactor(n-1)*n;  %递归调用求(n-1)!
end
```



# 程序设计      函数的递归调用

在脚本文件中调用该函数，求  $s = 1!+2!+3!+4!+5!$ 。

```
s = 0;  
for i = 1:5  
    s = s + factor(i);  
end  
s
```

在命令窗口运行脚本文件，结果如下：

```
s =  
  
153
```

# 程序设计      主函数和子函数

- 一个函数文件可包含多个函数，最前面的一个是主函数，后续的为子函数
- 子函数只能被本文件中的主函数或子函数调用
- 同一文件中主函数和子函数的工作空间相互独立
- help和lookfor命令不能提供子函数的帮助信息

# 程序设计 主函数和子函数

例：函数文件myfactor.m

```
function f = myfactor(n)
%MYFACTOR get n!
%input   : n
%output  : f
if n<=1
    f = 1;
else
    f = myfactor(n-1)*n; %递归求(n-1)!
end
myfactor_sub1(n,f) %调用子函数
```

主函数

```
function myfactor_sub1(x,y)
fprintf('%d!=%d\n',x,y)
```

子函数

# 程序设计      全局变量和局部变量

Matlab的主工作空间、各函数的工作空间是独立的，函数内用到的变量通常是局部变量。

特殊情况：全局变量（函数间传递信息的另一手段）

`global`    变量名

全局变量破坏了函数对变量的封装，降低了程序的可读性，尽量不用或采用醒目的命名方式（如global\_a）并把全局变量的定义语句放在程序的前部以示突出。

# 程序设计      程序调式

- Matlab可检查大部分语法错误并给出提示信息
- 调式菜单结合断点、pause等可胜任大部分调试工作
- 可结合命令窗口和工作空间的信息进行调试
- 灵活运用调试快捷键
- 调试中光标放置在变量名上会给出一些有用信息

# 程序设计 其他

## inline (建立内联函数)

【例】计算 $f(x) = \sin(x) + y$

`fx=inline('sin(x).*y','x','y');` %'x'和'y'指定了函数的自变量

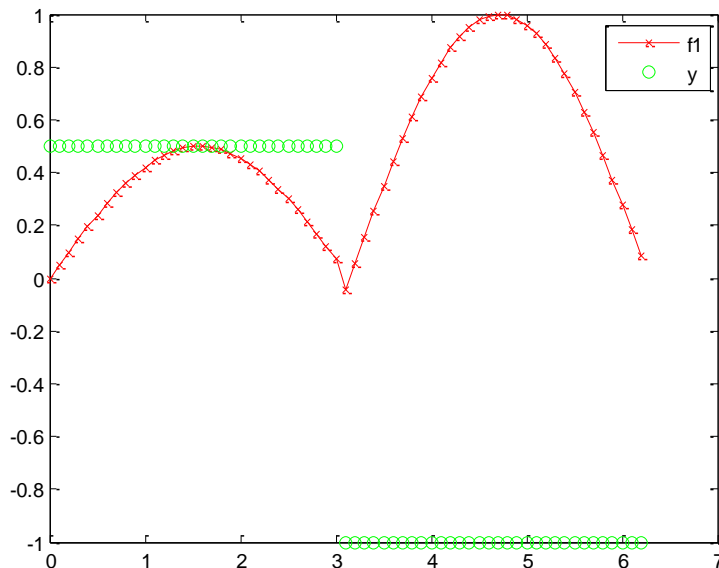
`x=[0:0.1:2*pi];`

`y(1:31)=0.5; y(32:63)=-1;`

`f1=fx(x,y);`

`plot(x,f1,'rx-',x,y,'go ')`

`legend('f1','y')`



内联函数不用写M文件，很便捷，但功能有限制，只能由一个表达式组成，只能返回一个变量，不能调用另一个内联函数。

# 程序设计 其他

## @ 匿名函数 (内联函数升级版)

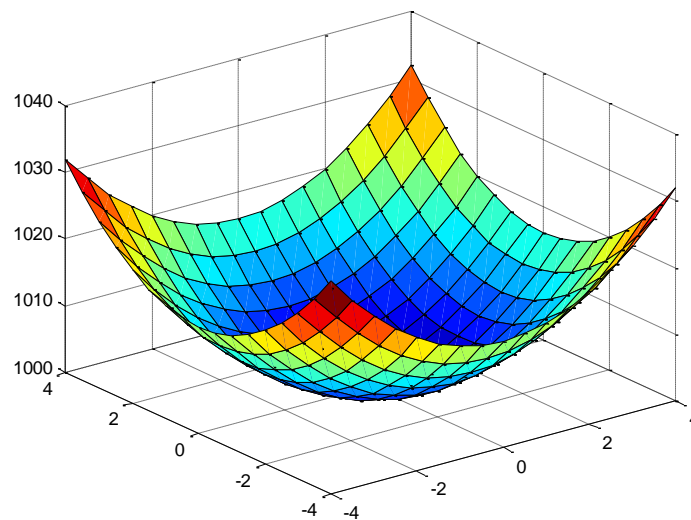
【例】计算  $f(x) = x^2 + y^2 + 1000$

```
m=1000; f=@(a,b)a.^2+b.^2+m; %匿名函数除了自变量a与b, 还可  
%以有额外参数传递, 例如 m
```

```
x=-4:0.5:4; y=x;  
[x,y]=meshgrid(x,y);  
z=f(x,y); %自变量  
surf(x,y,z);
```

多重匿名函数, 例如

```
f=@(x,y)@(a)x.^2+y.^a;  
f1=f(2,3); f2=f1(4) %结果是f2=85
```



# 第4章，程序设计

1. **for while** 熟练运用
2. **If swich** 熟练运用
3. **break continue** 熟练运用
4. 输入输出 **input disp**
5. 函数文件的书写和调用（可不写注释，但要满足基本要求）
6. 递归调用、全局变量、匿名函数、内联函数的基本运用

**重视综合运用**