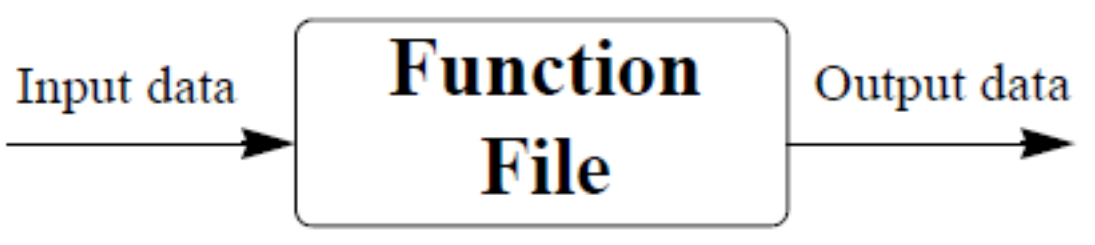
فصل ۷

توابع تعریف شده توسط کاربر و فایلهای توابع

MATLAB An Introduction With Applications, 6th Edition
Dr. Amos Gilat
The Ohio State University

Slide deck by
Dr. Greg Reese
Miami University

تابع یک برنامه به زبان MATLAB است که میتواند ورودی بگیرد و خروجی تولید کند. بعضی توابع ورودی نمی گیرند و/یا خروجی نمی دهند.



توابع توسط برنامه یا تابعی دیگر *فراخوانده یا اجرا* میشوند. آن برنامه میتواند به تابع فراخوانده شده ورودی ها را بدهد و خروجی های آن را بگیرد.

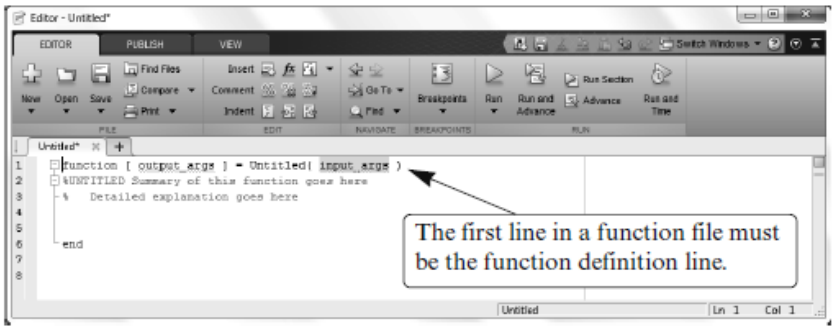
چرا از توابع استفاده میشود؟

- استفاده از یک کد در بیش از یک محل در برنامه بدون بازنویسی آن
- استفاده مجدد از کد با فراخواندن آن در برنامه های متفاوت
- آسان تر کردن عیب یابی با قرار دادن تمام کد مربوط به یک نوع عملکرد در یک قسمت

کد توابع به صورت *m-فایل* است

- میتوان این فایل را در هر برنامه ویرایشگری نوشت اما ساده ترین راه نوشتن آن در پنجره ویرایشگر MATLAB است

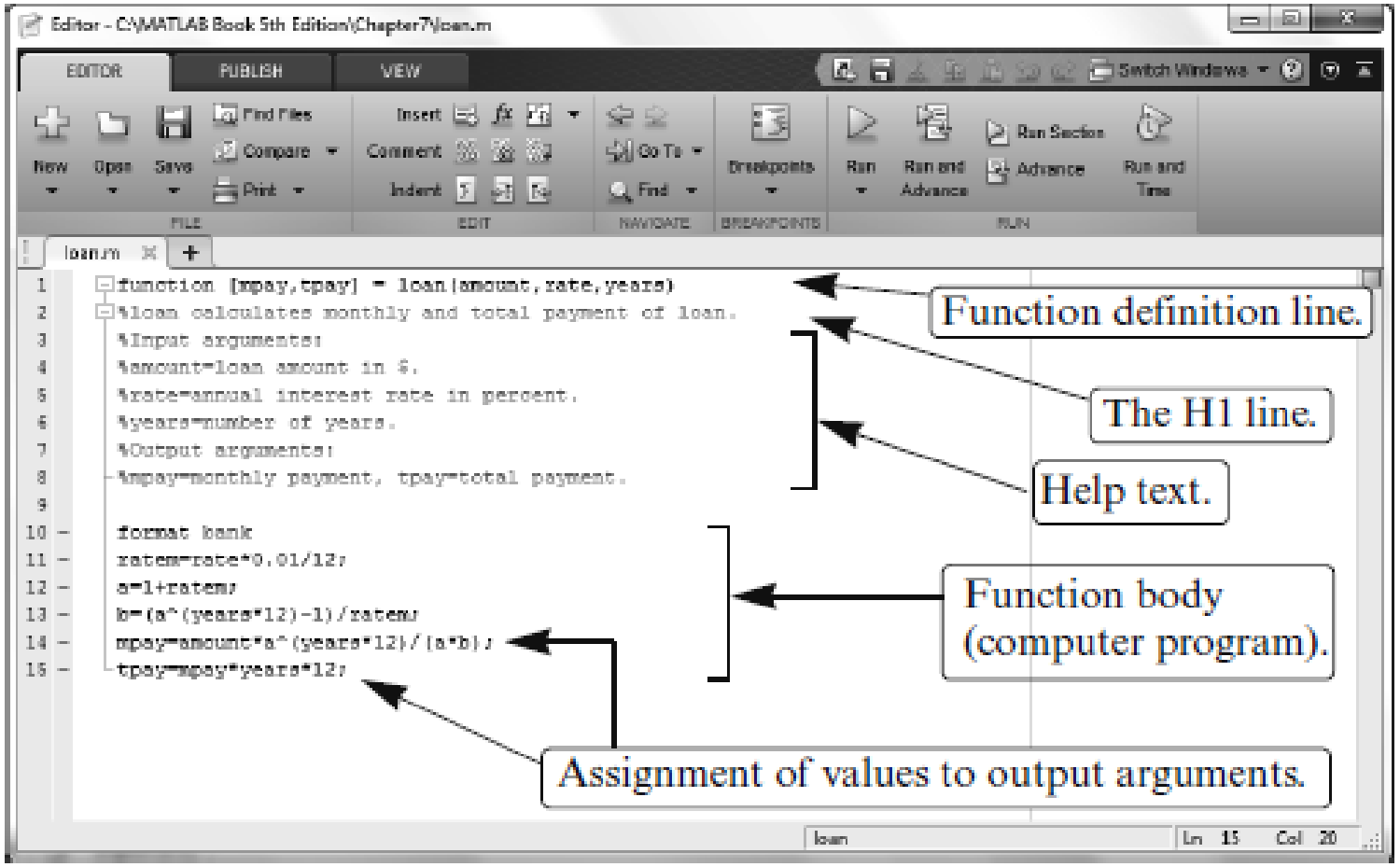
برای ایجاد یک m-فایل برای یک تابع جدید، در محیط اصلی MATLAB آیکن New را کلیک کرده و function را انتخاب کنید. پنجره ای به این شکل باز میشود



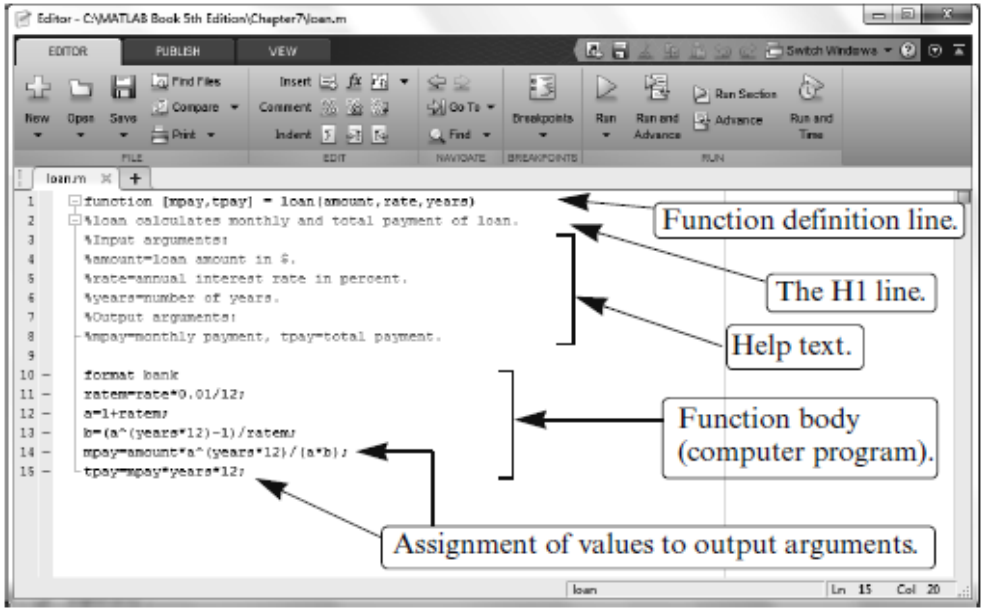
پنجره ویرایشگر خالی را نیز میتوان با کلیک کردن آیکن New Script در صفحه اصلی MATLAB باز کرد

همچنین پنجره ویرایشگر را میتوان از طریق پنجره فرمان با نوشتن `edit` که به صورت اختیاری نام فایل (بدون آپوستروف) نیز به دنبال آن بیاید باز کرد

- اگر فایل در پوشه جاری باشد، MATLAB آن را باز میکند
- اگر فایل در پوشه جاری نباشد، MATLAB آن را ایجاد میکند



تصویر ۲-۷: نمونه ای از یک فایل تابع کامل



هدف

- محاسبه اقساط ماهیانه و کل مبلغ بازپرداخت وام

ورودی ها

- مقدار وام، نرخ سود، زمان بازپرداخت

خروجی ها

- اقساط ماهیانه و کل مبلغ بازپرداخت

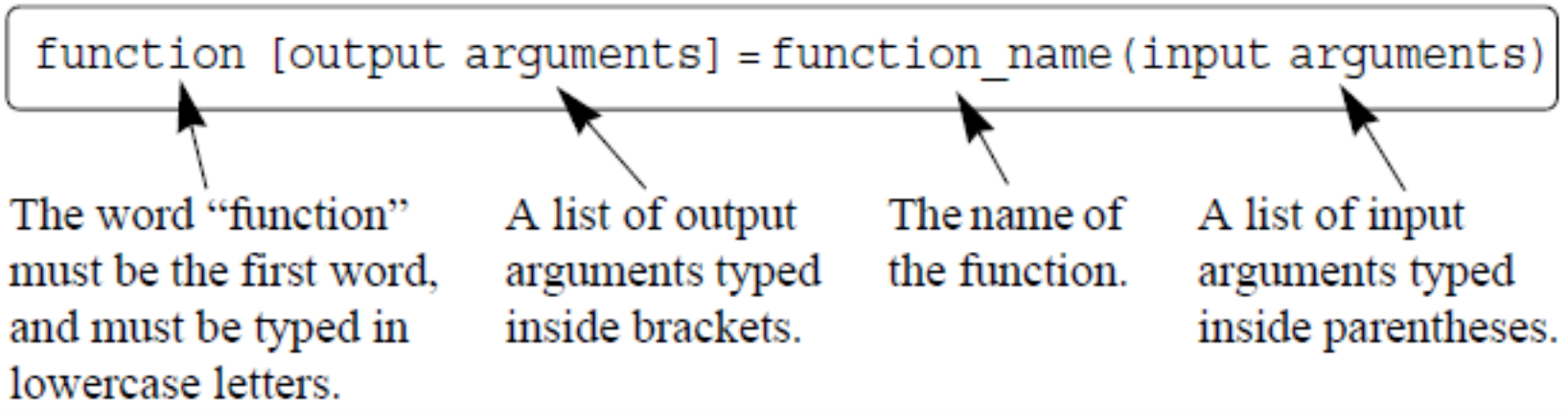
خط اجرایی خطی از کد است که MATLAB میتواند آن را اجرا کند

- خطی است که خط توضیحی، خط خالی، یا خطی که فقط شامل space و tab است نباشد

خط تعریف تابع

- باید اولین خط اجرایی در فایل تابع باشد
  - در غیر اینصورت، MATLAB فایل را یک فایل برنامه معمولی در نظر میگیرد
- فایل را به عنوان فایل تابع تعریف میکند
- نام تابع را تعریف میکند
- تعداد و ترتیب متغیرهای ورودی و خروجی را تعریف میکند

خط تعریف تابع به این صورت است



نام تابع

- از حروف، اعداد و underscore تشکیل شده است
- نمیتواند دارای فاصله باشد
- از قوانین مشابه نام متغیرها پیروی میکند

از نام گذاری به نام توابع داخلی MATLAB خودداری کنید

متغیرهای ورودی

- برای انتقال داده ها به تابع از برنامه فراخواننده استفاده میشوند
- میشود هیچ یا چند متغیر ورودی وجود داشته باشد
  - متغیرها را با کاما از هم جدا کنید
- میتوانند عدد، بردار یا آرایه باشند
- دارای مقادیر مشخص شده توسط برنامه فراخواننده هستند

متغیرهای خروجی

- برای انتقال داده ها از تابع به برنامه فراخواننده به کار میروند
- میشود هیچ یا چند متغیر خروجی وجود داشته باشد
  - در خط تعریف تابع
    - اگر هیچ متغیری وجود نداشته باشد، میتوان عملگر تخصیص (=) را حذف کرد
    - اگر فقط یک متغیر خروجی وجود داشته باشد، میتوان کروشه را حذف کرد
    - اگر چند متغیر وجود دارد، آنها را با *کاما* جدا کنید
- میتوانند عدد، بردار یا آرایه باشند
- تابع پیش از پایان اجرا، باید به همه متغیرهای خروجی مقداری نسبت دهد

نمونه هایی از خطوط تعریف توابع

Function definition line	Comments
function[mpay,tpay]= loan(amount,rate,years)	Three input arguments, two output arguments.
function [A] = RectArea(a,b)	Two input arguments, one output argument.
function A = RectArea(a,b)	Same as above; one output argument can be typed without the brackets.
function [V, S] = SphereVolArea(r)	One input variable, two output variables.
function trajectory(v,h,g)	Three input arguments, no output arguments.

خط H1 و خطوط متن راهنما خطوط توضیحی هستند (با علامت درصد (%) شروع میشوند)

- برای دادن اطلاعاتی در خصوص تابع به کاربر استفاده میشوند
- هر دو آنها اختیاری هستند

اگر دنبال دستوری از MATLAB برای انجام محاسبه خاصی میگردید، میتوانید از یک کلمه یا عبارت توصیف کننده آن برای جستجوی نام دستور استفاده کنید. این کار را با دستور lookfor انجام دهید:

```
lookfor 'word or phrase'
```

- کلمه یا عبارت را بین آپوستروف ها بنویسید

مثال

فرض کنید میخواهید دستور محاسبه ریشه دوم یک عدد را پیدا کنید. یک کلمه محتمل در توصیف محاسبه "square" است

```
>> lookfor 'square'
magic - Magic square.
realsqrt - Real square root.
sqrt - Square root.
lscov - Least squares with known covariance.
sqrtm - Matrix square root.
cgs - Conjugate Gradients Squared Method.
```

با خواندن توضیحات میتوانید بفهمید که دستور مورد نظرتان "sqrt" است

مثال

یک کلمه محتمل دقیق تر "square root" است

```
>> lookfor 'square root'
realsqrt - Real square root.
sqrt - Square root.
sqrtm - Matrix square root.
```

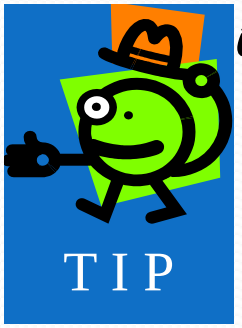
تعداد پیشنهادات از حالت قبل کمتر است

نکته – با استفاده از خطوط H1، میتوان کاری کرد که `lookfor` برای توابع نوشته شده توسط کاربر هم کار کند، نه فقط توابع داخلی MATLAB

خط *H1* اولین خط توضیح پس از خط تعریف تابع است

- اختیاری است
- فقط یک سطر طول دارد
- معمولاً شامل نام تابع و تعریف کوتاه تابع است

lookfor 'word' تمامی فایلهایی که آنها را H1 خط میشناسد به دنبال "word" می گردد (شامل آنهایی که شما نوشته اید). اگر آن را پیدا کند اسم فایل و خط H1 آن را چاپ میکند



وقتی در خط H1 مینویسید، از کلماتی استفاده کنید که استفاده از آنها توسط کاربران برای lookfor محتمل تر است

- خوب – استفاده از "square"و"root"

```
% mySqrt(x): computes square root of x using Reese algorithm
```

- بد – استفاده از "half"و"power"

```
% mySqrt(x): computes x to half power using Reese algorithm
```

دستور help خط H1 و تمام خطوط توضیح ادامه آن را تا رسیدن به خط خالی یا حاوی کد در فایل تابع نمایش میدهد

- برای فایل هایی که شما نوشته اید نیز کار میکند

برای اینکه help فقط یک خط خالی را نمایش دهد، از یک خط فقط با یک علامت درصد استفاده کنید

مثال – فایل تابع lowpass.m

```
function y = lowpass( x, fHigh )
% LOWPASS - lowpass filter with specified cutoff frequency
% USAGE: y = lowpass( x, fHigh )
%
% AUTHOR: Joe Blow
 X = fft( x );
...
```

```
>> help lowpass
LOWPASS - lowpass filter with specified cutoff frequency
USAGE: y = lowpass( x, fHigh )

AUTHOR: Joe Blow
```

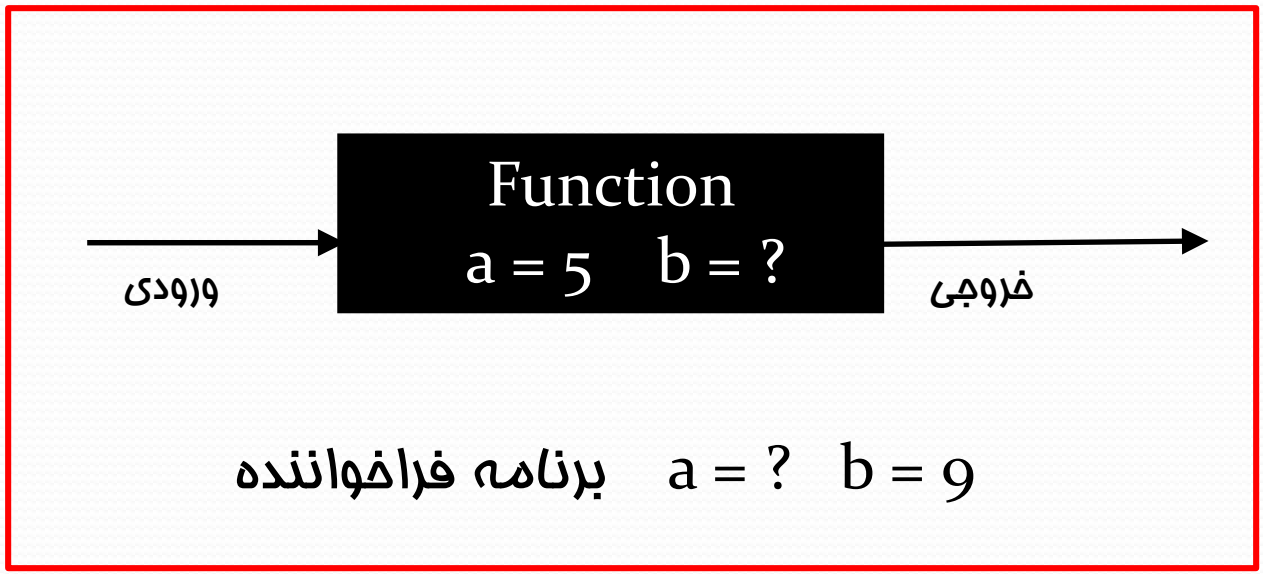
خط خالی ایجاد شده به وسیله خط * بالا

بدنه تابع

- حاوی کدی است که محاسبات را انجام میدهد
- کد میتواند از این موارد استفاده کند:
  - همه قابلیتهای برنامه نویسی MATLAB که تا اینجا بیان شده است
  - دستورات خاص MATLAB که فقط برای توابع هستند

متغیرهای اعلام شده داخل یک تابع، برای آن تابع *محلی* هستند، یعنی فقط داخل آن تابع قابل استفاده هستند

- برنامه فراخواننده نمیتواند به هیچ کدام از متغیرهای داخل آن تابع دسترسی داشته باشد
- تابع نمیتواند به هیچ کدام از متغیرهای داخل برنامه فراخواننده دسترسی داشته باشد



متغیرهای داخل و خارج تابع میتوانند نام یکسان داشته باشند، ولی همچنان متغیرهای متفاوتی هستند. تغییر یکی دیگری را عوض نمیکند

مثال – test.m

```
function test( n )
fprintf( 'On entering function n = %d\n', n );
n = 10;
fprintf( 'Before leaving function n = %d\n', n );
```

```
>> n = 5
n = 5
>> test(n)
On entering function n = 5
Before leaving function n = 10
>> n
n = 5
```

← n پس از اجرای تابع هنوز ۵ است

به طور پیشفرض، متغیرها محلی هستند. برای اینکه یک متغیر در بین توابع مختلف و/یا فضای کاری شناخته شده باشد، باید آن را *سراسری (global)* معرفی کنید. این کار را با این دستور انجام دهید

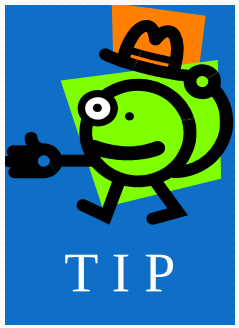
```
global variable_name
```

میتوانید چند متغیر سراسری را در یک خط با جدا کردن آنها با فاصله اعلام کنید، مثلاً

```
global v1 v2 v3
```

- متغیر را باید در هر فایلی که میخواهید در آن از آن متغیر استفاده کنید سراسری اعلام کنید
  - پس از آن متغیر فقط بین این فایلها مشترک است
- دستور `global` باید قبل از استفاده از متغیر نوشته شود
  - عادت خوبی است که دستور را در بالای فایل بنویسید
- دستور `global` باید در پنجره فرمان و یا یک برنامه معمولی نوشته شود تا متغیر در فضای کاری شناخته شده باشد

- مقدار متغیر را میتوان در هر جایی که شناخته شده باشد (سراسری اعلام شده باشد) تعیین کرد یا تغییر داد
- رایج است که از اسامی توصیفی طولانی (یا اسامی با همه حروف بزرگ) برای متغیرهای سراسری استفاده میکنند تا از متغیرهای معمولی متمایز شوند



نکته – استفاده از متغیرهای سراسری در کدنویسی جدید عادت خوبی محسوب نمیشود. تا میشود از آنها پرهیز کنید

یک فایل را پیش از استفاده از آن باید ذخیره کنید

- روی آیکن Save کلیک کنید
- به شدت توصیه میشود که نام فایل همان نام تابع و به دنبال آن .m باشد
- به شدت توصیه میشود که از پسوند فایل .m استفاده کنید
- اگر پسوند فایل را ننویسید، MATLAB به طور خودکار .m را اضافه میکند

مثال

Function definition line	File name
function [mpay,tpay] = loan(amount,rate,years)	loan.m
function [A] = RectArea(a,b)	RectArea.m
function [V, S] = SphereVolArea(r)	SphereVolArea.m
function trajectory(v,h,g)	trajectory.m

توابع تعریف شده توسط کاربر مانند توابع داخلی فراخوانده میشوند

- از پنجره فرمان
- از یک کد معمولی
- از یک تابع دیگر

برای استفاده از تابع فایل آن باید در پوشه جاری باشد

متغیرهای ورودی میتوانند عدد ثابت، عبارت، یا متغیر (که دارای مقدار است) باشند

ورودی و خروجی با فراخواندن تابع به ترتیبی که در خط تعریف تابع نوشته شده است مبادله میشوند

مثلاً،

```
function [month total]=loan(amount,rate,years)
```

مثال

از اکنون تا زمانی که فرزندم ۱۸ ساله شود، برای یک وام ۶ درصدی به مبلغ ۱۰۰۰۰ دلار چقدر پول ماهانه و به طور کلی باید بپردازم؟

در شکل ۲-۷ خط تعریف تابع به این صورت است

```
function [mpay tpay] = loan(amount,rate,years)
```

مثال کاربرد:

```
>> kidAge = 11;
>> rate = 6;
>> [mpay tpay] = loan( 10000, rate, 18-kidAge )
```

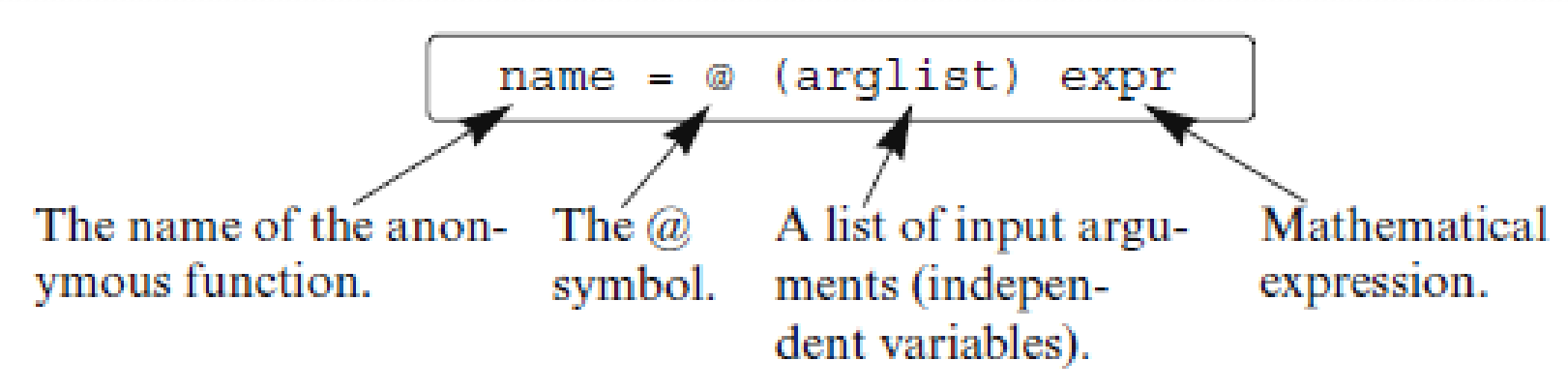
ثابت متغیر عبارت

Characteristic	Function	Script
File extension	.m	.m
First executable line	Function definition line	Any
Variables recognized in Command Window?	No	Yes
Can use variables defined in workspace?	No	Yes
Can accept input arguments?	Yes	No
Can return output arguments?	Yes	No

تابع ناشناس تابعی است که بدون استفاده از یک فایل تابع جداگانه تعریف می شود

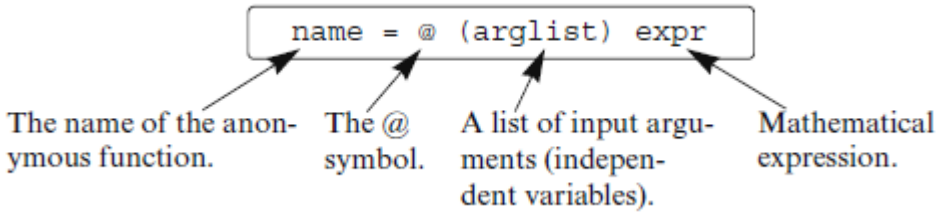
- فقط برای توابع کوتاه (یک خطی) است
- معمولاً توسط کاربر نوشته میشود
- در داخل یک تابع دیگر یا بدنه یک کد، و نه در یک فایل جداگانه تعریف میشود
- اغلب توسط توابعی از MATLAB که روی توابع دیگر عمل میکنند استفاده میشود، مثلاً
  - یافتن مساحت زیر نمودار یک تابع
  - یافتن مشتقات یک تابع
  - یافتن مقدار بهینه یک تابع

توابع ناشناس را میتوان در پنجره فرمان، یک کد معمولی، یا داخل یک تابع تعریف شده توسط کاربر ایجاد کرد. صورت کل آنها اینگونه است:



به عنوان مثال

```
cube = @(x) x^3
```



- Name – نام تابع، بر اساس قواعد نام توابع تعریف شده توسط کاربر
- @ – دستگیره تابع، شیئی که اطلاعاتی در مورد تابع دارد
- arglist – هیچ یا چند متغیر تابع، که مشابه توابع تعریف شده توسط کاربر تعریف میشوند
- expr – یک عبارت به زبان MATLAB

تابع ناشناس

- میتواند شامل هر تابع داخلی یا تعریف شده توسط کاربر باشد
- مشابه توابع تعریف شده توسط کاربر فراخوانده میشود

```
>> triple = @(n) 3 * n
triple =
    @(n)3*n
>> triple( 4 )
ans = 12
>> x = 1.5;
>> y = triple( x )
y = 4.5000
```

توابع ناشناس میتوانند در بدنه خود هر متغیری که پیش از تعریف آنها تعریف شده باشد را استفاده کنند

مقادیری که تابع استفاده میکند آنهایی هستند که متغیرها در زمان تعریف تابع دارند، نه زمان فراخوانی آن

مسأله – متغیر تعریف شده پیش از تعریف تابع ناشناس

```
>> p = 5;
>> combo = @(x,y) p*x + q*y;
>> combo(2,8)
??? Undefined function or variable 'q'.
Error in ==> @(x,y)p*x+q*y
```

راه حل

```
>> p = 5;
>> q = 4;
>> combo = @(x,y) p*x + q*y;
>> combo(2,8)
ans = 42
```

تابع ناشناس از تغییر مقدار متغیر پس از تعریف تابع اثر نمی پذیرد

```
>> p = 5;
>> q = 4;
>> combo = @(x,y) p*x + q*y;
>> combo( 2, 8 )
ans = 42
>> p = 0
p = 0
>> q = 0
q = 0
>> combo( 2, 8 )
ans = 42
```

مثال

برای محاسبه $sin(2\pi t)$ یک تابع ناشناس بنویسید و از دستور quad برای یافتن مساحت زیر نمودار تابع از ۰ تا ۱ استفاده کنید.

```
>> mysine = @(t) sin( 2*pi*t );
>> quad( mysine, 0, 1 )
ans =
     0
```

یک تابع MATLAB که تابع دیگری را به عنوان ورودی میگیرد *تابع تابع* نامیده میشود
به عنوان مثال

- `quad` یافتن مساحت زیر نمودار تابع داده شده
- `fzero` یافتن نقاط صفر تابع داده شده
- `fminbnd` یافتن مقدار کمینه تابع داده شده

دستگیره تابع:

از دستگیره های تابع برای دادن توابع به توابع توابع استفاده میشود

- توابع داده شده میتوانند توابع داخلی، تعریف شده توسط کاربر، یا ناشناس باشند

برای ساختن یک دستگیره تابع برای توابع داخلی یا تعریف شده توسط کاربر، علامت @ را در برابر نام تابع قرار دهید

- مثلاً، دستگیره تابع برای تابع `cos` به صورت `@cos` است

اگر بخواهید از دستگیره تابع بیش از یک بار استفاده کنید، مناسب خواهد بود که آن را به یک متغیر نسبت دهید، مثلاً

```
>> cosHandle = @cos;
```

نوشتن یک تابع تابع که دستگیره تابع را به عنوان متغیر ورودی بگیرد:

برای نوشتن یک تابع تابع

- از یک اسم مجاز برای دستگیره تابع که قرار است به آن داده شود استفاده کنید
  - از @ در نام استفاده نکنید
  - نامی که انتخاب میکنید اسم *ساختگی* نامیده میشود
  - نام تابع ورودی میتواند هر چیزی باشد، مشابه متغیرهای ورودی
  - نمونه اسم تابع: f, fun, F, Func
- در تابع تابع، تابع ورودی را مشابه تابع عادی فراخوانی کنید، مثلاً Func(x)

مهم – تمام موارد مورد نیاز تابع را مستند سازی کنید، تا کاربر بداند که چگونه تابع را برای ورود بنویسد

- ورودی ها – چند تا، به چه ترتیبی، چه نوع داده ای (اسکالر، بردار سطری یا ستونی، ماتریس و ابعاد آن)
- خروجی ها – به همین صورت

A name for the function that is passed in.

```
function xyout=funplot(Fun,a,b)
% funplot makes a plot of the function Fun which is passed in
% when funplot is called in the domain [a, b].
% Input arguments are:
% Fun:  Function handle of the function to be plotted.
% a:  The first point of the domain.
% b:  The last point of the domain.
% Output argument is:
% xyout: The values of x and y at x=a, x=(a+b)/2, and x=b
% listed in a 3 by 2 matrix.

x=linspace(a,b,100);
y=Fun(x);
xyout(1,1)=a; xyout(2,1)=(a+b)/2; xyout(3,1)=b;
xyout(1,2)=y(1);
xyout(2,2)=Fun((a+b)/2);
xyout(3,2)=y(100);
plot(x,y)
xlabel('x'), ylabel('y')
```

Using the imported function to calculate $f(x)$ at 100 points.

Using the imported function to calculate $f(x)$ at the midpoint.

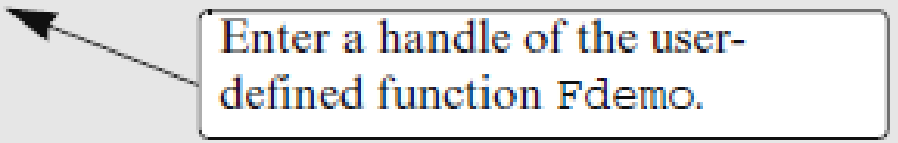
دادن یک تابع تعریف شده توسط کاربر به یک تابع تابع:

با استفاده از علامت @ قبل از نام تابع، تابع تعریف شده توسط کاربر را به عنوان ورودی بدهید

مثال

```
function y=Fdemo(x)
y=exp(-0.17*x).*x.^3-2*x.^2+0.8*x-3;
```

```
>> ydemo=funplot(@Fdemo,0.5,4)
ydemo =
    0.5000   -2.9852
    2.2500   -3.5548
    4.0000    0.6235
```



دادن یک تابع ناشناس به عنوان ورودی یک تابع تابع:

نام تابع ناشناس را بدون استفاده از علامت @ بدهید

مثال

```
>> FdemoAnony=@(x) exp(-0.17*x).*x.^3-2*x.^2+0.8*x-3
FdemoAnony =
    @(x) exp(-0.17*x).*x.^3-2*x.^2+0.8*x-3

>> ydemo=funplot(FdemoAnony,0.5,4)
ydemo =
    0.5000   -2.9852
    2.2500   -3.5548
    4.0000    0.6235
```

Create an anonymous function for $f(x)$.

Enter the name of the anonymous function (FdemoAnony).

استفاده از نام تابع برای دادن یک تابع به یک تابع تابع

- از این روش دیگر استفاده نمی شود. کتاب را برای اطلاعات بیشتر ببینید

میتوان چند تابع را در یک فایل تابع داشت

- توابع پشت سر هم نوشته میشوند
  - اولین تابع *تابع اولیه* نامیده میشود
  - بقیه توابع *زیرتابع* نامیده میشوند
- هر تابع با یک خط تعریف تابع آغاز میشود
- نام فایل تابع باید همنام تابع اولیه باشد (همراه با .m)

- هر تابعی در فایل میتواند توابع دیگر فایل را فرا بخواند
- توابع به متغیرهای داخل توابع دیگر دسترسی ندارند (مگر اینکه آن متغیرها سراسری اعلام شده باشند)
- توابع خارج فایل، کدها، یا پنجره فرمان تنها میتوانند تابع اولیه را فرا بخوانند

زیرتابع ها

- معمولاً برای واضح تر کردن تابع اولیه به وسیله انجام دادن بخشی از محاسبات آن به کار میروند
- معمولاً به تنهایی یا برای فراخوانده شدن به وسیله کدی غیر از تابع اولیه مفید نیستند

مثال

```
function [me SD] = stat(v)                    The primary function.
n=length(v);
me=AVG(v,n);
SD=StandDiv(v,me,n);

function av=AVG(x,num)                        Subfunction.
av=sum(x)/num;

function Sdiv=StandDiv(x,xAve,num)            Subfunction.
xdif=x-xAve;
xdif2=xdif.^2;
Sdiv= sqrt(sum(xdif2)/(num-1));
```

```
>> Grades=[80 75 91 60 79 89 65 80 95 50 81];
>> [AveGrade StanDeviation] = stat(Grades)
AveGrade =
   76.8182
StanDeviation =
   13.6661
```

تابع تو در تو تابعی است که در داخل یک تابع دیگر نوشته میشود

- تابع تو در تو با یک خط تعریف تابع شروع میشود و با یک دستور end خاتمه می یابد
- تابعی که تابع تو در تو در آن قرار دارد نیز باید با end تمام شود
- یک تابع تو در تو میتواند حاوی یک تابع تو در توی دیگر باشد، که آن هم یک تابع تو در تو داشته باشد و ...

یک تابع تو در تو

صورت کلی یک تابع تعریف شده توسط کاربر A (که *تابع اولیه* نامیده میشود) که شامل یک تابع تو در توی B است به این صورت است:

```
function y=A(a1,a2)
.......
    function z=B(b1,b2)
    .......
    end
.......
end
```

به دستور end در انتهای A و B توجه کنید

```
function y=A(a1,a2)
.......
  function z=B(b1,b2)
  .......
  end
.......
end
```

تابع تو در توی B میتواند به فضای کاری تابع اولیه A دسترسی داشته باشد و برعکس

```
function y=A(a1,a2)
.......
  function z=B(b1,b2)
  .......
  end
.......
end
```

- تابع A میتواند تابع B را فرا بخواند

```
function y=A(a1,a2)
fprintf( 'In function A\n' );

  function z=B(b1,b2)
  fprintf( 'In function B\n' );
  z = b1 + b2;
  end

fprintf( 'Calling B from A\n' );
y = B( a1, a2 );
end
```

```
>> y =  A( 1, 2 )
In function A
Calling B from A
In function B
y = 3
```

```
function y=A(a1,a2)
.......
  function z=B(b1,b2)
  .......
  end
.......
end
```

تابع B نمیتواند تابع اولیه A را فرابخواند (در کتاب اشتباه نوشته شده)

```
function y=A(a1,a2)
fprintf( 'In function A\n' );
y = B( a1, a2 );

  function z=B(b1,b2)
  fprintf( 'In function B\n' );
  fprintf( 'Calling A from B\n' );
  z = A( b1, b2 );
  end

end
>> y =  A( 1, 2 )
In function A
In function B
Calling A from B
In function A
In function B
Calling A from B
...
Maximum recursion limit of
500 reached.
```

دو (یا بیشتر) تابع تو در تو در سطح یکسان

صورت کلی یک تابع تعریف شده توسط کاربر A که شامل دو تابع تو در توی B و C در سطح یکسان است به این صورت است:

```
function y=A(a1,a2)
.......
  function z=B(b1,b2)
  .......
  end
.......
  function w=C(c1,c2)
  .......
  end
.......
end
```

```
function y=A(a1,a2)
.......
  function z=B(b1,b2)
  .......
  end
.......
  function w=C(c1,c2)
  .......
  end
.......
end
```

- تابع A میتواند توابع B و C را فرا بخواند
- تابع B میتواند تابع C را فرا بخواند، ولی نه تابع A را
- تابع C میتواند تابع B را فرا بخواند، ولی نه تابع A را

شماره تمرین های منتخب

- ۴
- ۷
- ۸
- ۱۰
- ۱۲
- ۱۴
- ۱۶
- ۱۹
- ۲۰
- ۲۲
- ۲۳
- ۲۴
- ۲۵
- ۲۸
- ۲۹
- ۳۰
- ۳۱
- ۳۳
- ۳۵
- ۳۸
- ۳۹