مقدمه ای بر برنامه نویسی با MATLAB

سعید سهیلی
استادیار گروه مکانیک
دانشگاه آزاد اسلامی مشهد


بسم الله الرحمن الرحیم

فصل پنجم

توابع تعریف شده توسط کاربر

مقدمه

- در فصول قبل با روش طراحی بالا به پایین آشنا شدیم.
- الگوریتم برنامه به تعدادی زیر مجموعه تقسیم می شود و هر زیرمجموعه به صورت مجزا برنامه نویسی می شود.
- در نهایت تمامی زیرمجموعه ها با هم ترکیب می شوند تا برنامه بزرگتر ساخته شود.
- در MATLAB این امکان وجود دارد که هر یک از زیرمجموعه ها به صورت یک تابع مجزا نوشته و هر تابع به طور مستقل از سایر زیرمجموعه ها آزمایش و رفع خطا شود.

مزایای استفاده از توابع

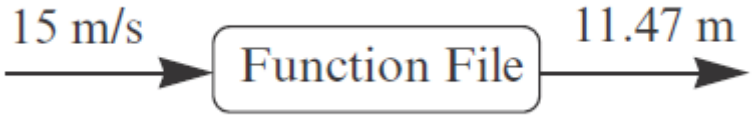
- آزمایش مستقل زیرمجموعه ها.
- کدهای قابل استفاده مجدد: ممکن است از برنامه نوشته شده در سایر برنامه ها نیز استفاده شود.
- عاری شدن برنامه از تاثیرات سهوی و جانبی.
- در توابع لیستی از متغیرها به نام لیست آرگومان های ورودی به داخل توابع فراخوانده می شوند و نتایج نیز از طریق لیست آرگومان های خارجی به برنامه بازگردانده می شوند. با این کار تابع از متغیرهای دیگر برنامه تاثیر نمی گیرد.

مقدمه ای بر توابع MATLAB

- تمام برنامه های مشاهده شده تا به حال فایل های نوشتاری (script file) بوده اند.
- وقتی یک فایل نوشتاری اجرا می شود، نتیجه مشابه وقتی است که در پنجره فرمان دستورات نوشته شوند.
- محیط کاری (Workspace) فایل نوشتاری با محیط کاری پنجره فرمان مشترک است و از متغیرهای هم استفاده می کنند.
- فایل نوشتاری هیچ گونه متغیر ورودی ندارد و هیچ نتیجه ای را بر نمی گرداند.
- تابع Matlab نوع خاصی از m-file است که در محیط کاری مستقل خود اجرا می شود. (متغیرها به صورت محلی درون تابع استفاده می شوند)
- داده های ورودی از طریق لیست آرگومان های ورودی دریافت و نتایج از طریق لیست آرگومان های خروجی بازگردانده می شوند.

maximum height that a ball reaches

$$h_{\max}(v_0) = \frac{v_0^2}{2g}$$



```
function [outarg1, outarg2, ...] = fname(inarg1, inarg2, ...)
% H1 comment line
% Other comment lines
...
(Executable code)
...
(return)
(end)
```

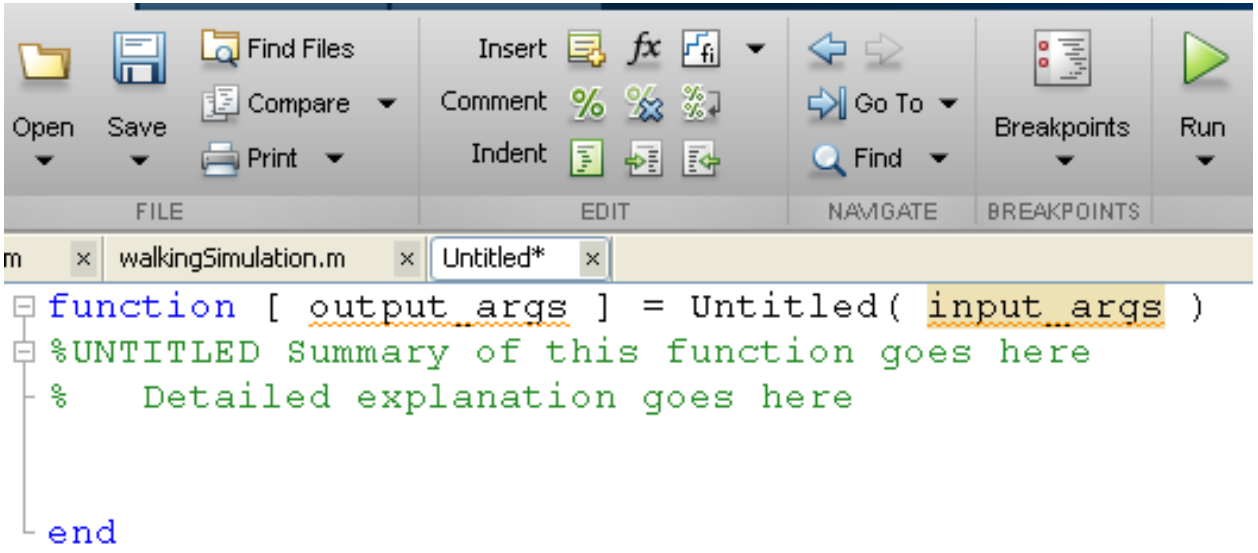
لیست متغیرهای خروجی

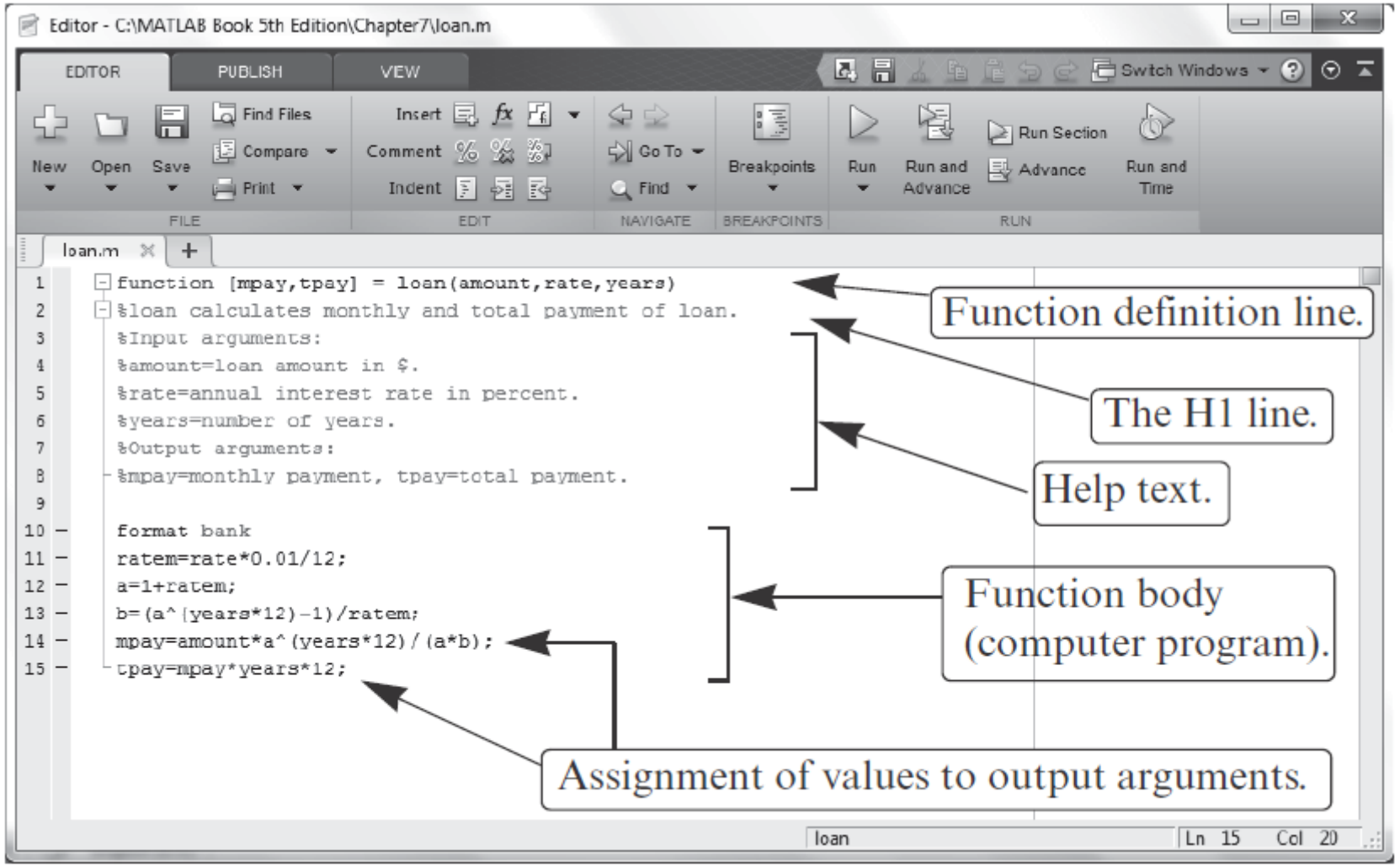
اسم تابع

لیست متغیرهای ورودی

خلاصه ای از هدف اصلی تابع.
قابل نمایش توسط دستور lookfor

توضیحات دیگر توسط دستور
help قابل نمایش است




Editor - C:\MATLAB Book 5th Edition\Chapter7\loan.m
EDITOR
PUBLISH
VIEW
Switch Windows
New
Open
Save
Find Files
Compare
Print
Insert
Comment
Indent
Go To
Find
Breakpoints
Run
Run and Advance
Run Section
Advance
Run and Time
FILE
EDIT
NAVIGATE
BREAKPOINTS
RUN
loan.m
function [mpay,tpay] = loan(amount,rate,years)
%loan calculates monthly and total payment of loan.
%Input arguments:
%amount=loan amount in $.
%rate=annual interest rate in percent.
%years=number of years.
%Output arguments:
%mpay=monthly payment, tpay=total payment.
format bank
ratem=rate*0.01/12;
a=1+ratem;
b=(a^(years*12)-1)/ratem;
mpay=amount*a^(years*12)/(a*b);
tpay=mpay*years*12;
Function definition line.
The H1 line.
Help text.
Function body (computer program).
Assignment of values to output arguments.
loan
Ln 15
Col 20

- یک تابع با نوشتن آن در پنجره فرمان یا قراردادن در یک فایل نوشتاری یا یک تابع دیگر، قابل فراخوانی است.
- هنگامی که تابع اجرا می شود، مقادیر ذخیره شده در لیست متغیرهای خروجی به کاربر برگردانده می شوند.
- به عنوان مثال تابع dist2 فاصله بین دو نقطه را محاسبه می کند.

```
function distance = dist2 (x1, y1, x2, y2)
%DIST2 Calculate the distance between two points
% Function DIST2 calculates the distance between
% two points (x1,y1) and (x2,y2) in a Cartesian
% coordinate system.
%
% Calling sequence:
%   distance = dist2(x1, y1, x2, y2)

% Define variables:
%   x1        -- x-position of point 1
```

- این تابع چهار متغیر ورودی و یک متغیر خروجی دارد.
- برای اجرای برنامه می توان از پنجره فرمان و دادن ورودی مناسب استفاده نمود.

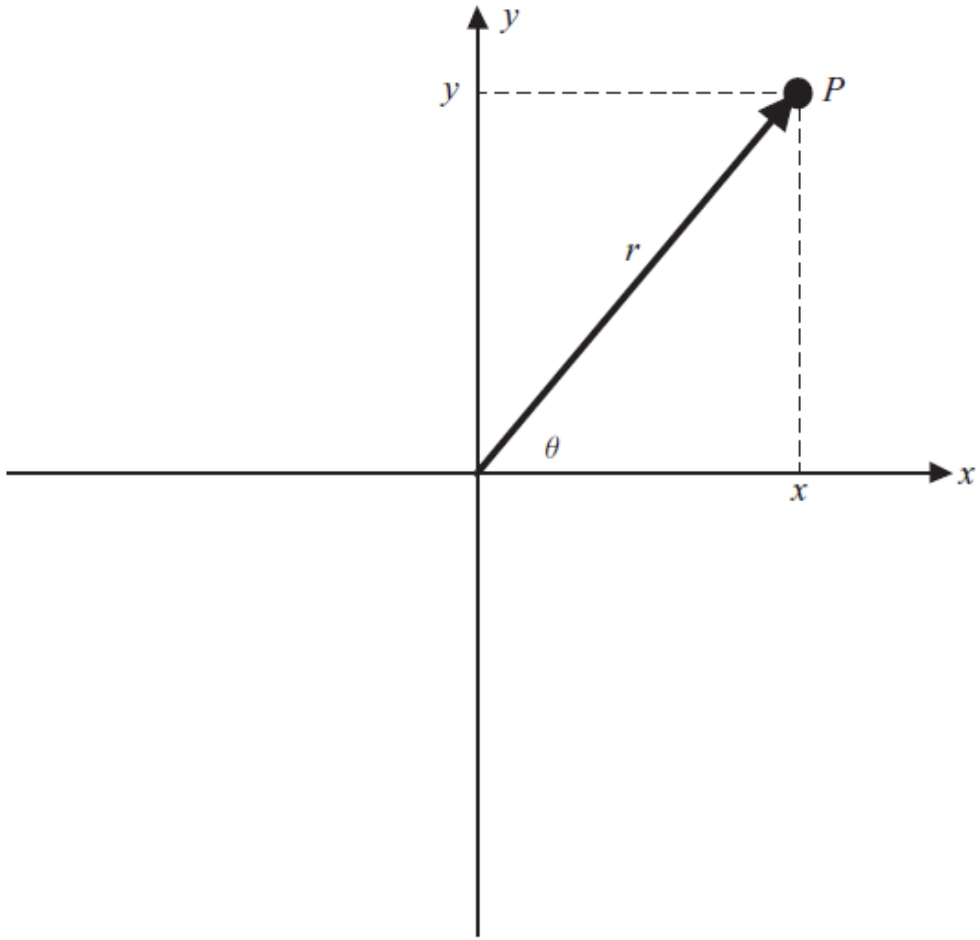
```
res = dist2(1, 1, 4, 5)
```

- این تابع را می توان از داخل یک فایل نوشتاری فراخوانی نمود.

- توابع به صورت مستقیم نباید اجرا شوند.

- محیط کار تابع با محیط کار عمومی متفاوت است.
- در توابع نیازی به نوشتن فرمان clear نیست

مثال) تبدیل مختصات قائم به قطبی



$$x = r\cos\theta$$
$$y = r\sin\theta$$
$$r = \sqrt{x^2 + y^2}$$
$$\theta = \tan^{-1}\frac{y}{x}$$

1. بیان مساله: تابعی بنویسید که مکان با مختصات قائم را به مختصات قطبی تبدیل نماید.
2. ورودی ها و خروجی ها: x, y مختصات قائم و خروجی ها r, θ در مختصات قطبی می باشد.
3. الگوریتم مساله:

```
r ← sqrt(x.^2 + y.^2)
theta ← 180/pi * atan2(y,x)
```

4. تبدیل الگوریتم به عبارت های MATLAB:

```
function [r, theta] = rect2polar(x,y)
%RECT2POLAR Convert rectangular to polar coordinates
% Function RECT2POLAR accepts the rectangular coordinates
% (x,y) and converts them into the polar coordinates
% (r,theta), where theta is expressed in degrees.
%
% Calling sequence:
%   [r, theta] = rect2polar(x,y)
% Define variables:
%   r         -- Length of polar vector
%   theta     -- Angle of vector in degrees
%   x         -- x-position of point
%   y         -- y-position of point
% Record of revisions:
%     Date          Engineer         Description of change
%     ====          ========         =====================
%   02/01/10     S. J. Chapman      Original code
r = sqrt(x.^2 + y.^2);
theta = 180/pi * atan2(y,x);
```

5. امتحان کردن برنامه:

```
» [r, theta] = rect2polar(4,3)
r =
     5
theta =
   36.8699
» [r, theta] = rect2polar(-4,3)
r =
     5
theta =
  143.1301
» [r, theta] = rect2polar(-4,-3)
r =
     5
theta =
 -143.1301
» [r, theta] = rect2polar(4,-3)
r =
     5
theta =
  -36.8699
```

به اشتراک گذاشتن داده با استفاده از حافظه سراسری (global memory)

- تمام متغیرهای درون تابع محلی (local) هستند.
- به عبارت دیگر فقط درون تابع خوانده و استفاده می شوند.
- برنامه ها، داده ها را با توابع از طریق لیست آرگومان های آن ها مبادله می کنند.
- می توان متغیرهای دلخواه را در بین توابع به صورت سراسری (global) به اشتراک گذاشت
- از هر نوع محیط کاری می توان به حافظه جهانی دسترسی یافت.

- متغیرهای سراسری به صورت زیر تعریف می شوند.

```
global GRAVITY_CONST FrictionCoefficient
```

- بهتر است این متغیرها با حروف بزرگ نوشته شوند.
- متغیرهای جهانی را بلافاصله بعد از توضیحات اولیه و قبل از اولین عبارت قابل اجرا تعریف کنید.
- در هر تابع (یا جای دیگر) که بخواهیم از متغیرها استفاده کنیم، باید دستور global نوشته شود.

مثال) مولد اعداد تصادفی

- عملکرد اکثر طرح های مهندسی قبل از مرحله ساخت به وسیله شبیه سازی نرم افزاری مورد بررسی قرار می گیرد.
- برای این کار مدل شبیه سازی شده به وسیله داده تست می شود.
- در جهان واقعی اندازه گیری ها همراه با خطا است.
- برای واقعی تر شدن نتایج شبیه سازی، داه ها با خطا وارد سیستم می شوند.
- خطا به وسیله مولد اعداد تصادفی ایجاد می شود.
- یکی از الگوریتم های تولید اعداد تصادفی، استفاده از تابع باقیمانده در هنگام اعمال روی اعداد بزرگ است.

$$n_{i+1} = \text{mod}(8121\ n_i + 28{,}411,\ 134{,}456)$$

$$ran_i = \frac{n_i}{134456}$$

- برای استفاده از این فرمول ها با یک مقدار اولیه n_0 نیاز است.

1. بیان مساله به طور واضح

- تابعی بنویسید که آرایه Ran شامل اعداد تصادفی در محدوده ۰ تا ۱ را ایجاد کند.
- تابع دارای یک یا دو آرگومان ورودی (n, m) است. اگر فقط یک آرگومان ذکر شود، باید آرایه مربعی با اندازه n@#را ایجاد کند و اگر دو آرگومان ذکر شود آرایه با اندازه m@#را ایجاد می کند.

2. تعریف ورودی ها و خروجی ها

- در این مساله دو تابع وجود دارد. تابع seed عددی صحیح را به عنوان ورودی می گیرد و از آن برای آغاز دنباله اعداد تصادفی استفاده می کند.
- ورودی تابع random0 یک یا دو عدد است که اندازه آرایه تصادفی را مشخص می کند و خروجی آرایه اعداد تصادفی در محدوده ۰ تا ۱ است.

3. توصیف الگوریتم

- شبه کد تابع random0 به صورت زیر است.

```
function ran = random0 ( m, n )
Check for valid arguments
Set n ← m if not supplied
Create output array with "zeros" function
for ii = 1:number of rows
   for jj = 1:number of columns
      ISEED ← mod (8121 * ISEED + 28411, 134456 )
      ran(ii,jj) ← ISEED / 134456
   end
end
```

- مقدار ISEED در حافظه جهانی قرار می گیرد.

```
function seed (new_seed)
new_seed ← round(new_seed)
ISEED ← abs(new_seed)
```

4. تبدیل الگوریتم به عبارات matlab

- هر دو تابع به عبارات matlab تبدیل می شوند.

5. امتحان برنامه نهایی

- **seed(1024)**
- **random0(4)**
- **random0(4)**
- **seed(1024)**
- **random0(4)**
- **random0(2,3)**

حفظ داده ها در حین استفاده از یک تابع

- وقتی تابع به پایان اجرای خود می رسد، محیط کار ایجاد شده برای آن از بین می رود و تمامی متغیرهای محلی آن پاک می شوند.
- گاهی اوقات حفظ بعضی اطلاعات محلی درون یک تابع بین فراخوانی های متوالی مفید است.
- حافظه persistent نوع خاصی از حافظه است که تنها از درون تابع می توان به آن دسترسی پیدا کرد و محتویات آن بین فراخوانی های یک تابع بدون تغییر حفظ می شود.

```
persistent var1 var2 var3 ...
```

مثال) میانگین گیری در حین اجرا

- فرض کنید برای آرایه ای از اعداد میانگین و انحراف معیار محاسبه شده باشد.
- می خواهیم با وارد کردن یک مقدار جدید، میانگین و انحراف معیار بدون نیاز به وارد کردن تمامی اعداد در حین اجرا محاسبه شود.

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad s = \sqrt{\frac{N\sum_{i=1}^{N} x_i^2 - \left(\sum_{i=1}^{N} x_i\right)^2}{N(N-1)}}$$

- هر چه انحراف معیار بزرگتر باشد، پراکندگی نقاط روی مجموعه نقاط بیشتر است.
- اگر N، مجموع مقادیر Σx ، مجموعه مربعات مقادیر Σx^2 را داشته باشیم، می توان میانگین و انحراف معیار را در هر زمان محاسبه نمود.

- این تابع باید مقادیر را یکی یکی وارد و مقادیر $N, \Sigma x, \text{ and } \Sigma x^2$ را ذخیره و حفظ نماید. برای این کار از حافظه persistent استفاده می شود.

1. بیان مساله

- تابعی ایجاد نمایید که میانگین و انحراف معیار مجموعه ای از اعداد را در حالیکه مقادیر جدید وارد می شوند، حساب نماید. تابع باید قابلیت صفر کردن مجدد مجموع ها را داشته باشد.

2. ورودی ها و خروجی های تابع

- رشته ورودی 'reset' برای صفر کردن مجموع ها
- مقادیر عددی جهت محاسبات آماری
- خروجی مقدار متوسط و انحراف معیار داده ها است.

3. طراحی الگوریتم

```
Check for a legal number of arguments
Check for a 'reset', and reset sums if present
Otherwise, add current value to running sums
Calculate and return running average and std dev
   if enough data is available. Return zeros if
   not enough data is available.
```

```
Check for a legal number of arguments
if x == 'reset'
   n ← 0
   sum_x ← 0
   sum_x2 ← 0
else
   n ← n + 1
   sum_x ← sum_x + x
   sum_x2 ← sum_x2 + x^2
end

% Calculate ave and sd
if n == 0
   ave ← 0
   std ← 0
elseif n == 1
   ave ← sum_x
   std ← 0
else
   ave ← sum_x / n
   std ← sqrt((n*sum_x2 - sum_x^2) / (n*(n-1)))
end
```

4. تبدیل به کد matlab

5. آزمایش برنامه

- برای آزمایش این تابع یک فایل نوشتاری ایجاد می شود که runstats را صفر کند، داده های ورودی را بخواند و تابع را فراخوانی کند و نتایج محاسبات آماری را نمایش دهد

3., 2., 3., 4., 2.8

آرگومان های اختیاری

- قبلا استفاده از تابع plot با حداقل دو آرگومان ورودی و حداکثر هفت آرگومان ورودی استفاده نموده ایم.
- تابع max یک یا دو آرگومان خروجی را پوشش می دهد.
- با استفاده از توابع خاصی در matlab می توان اطلاعات آرگومان های اختیاری را استفاده نمود.

- nargin: تعداد آرگومان های ورودی واقعی که در فراخوانی تابع استفاده می شود بر می گرداند.
- nargout: تعداد آرگومان های خروجی واقعی که در فراخوانی تابع استفاده می شود بر می گرداند.
  - از توابع nargin و nargout تنها در درون تابع می توان استفاده نمود.
  - در هر بار فراخوانی تعداد آرگومان های ورودی و خروجی واقعی را به ما می دهد.
- nargchk: در صورتیکه تابعی با تعداد بسیار زیاد یا بسیار کمی از آرگومان ها فراخوانی شود، یک پیغام خطا می دهد
  - شکل کلی این تابع به صورت زیر است.
  - message = nargchk(min_args,max_args,num_args);
  - min_args حداقل تعداد آرگومان ها و max_args حداکثر تعداد آرگومان ها و num_args تعداد واقعی آرگومان ها است
  - اگر تعداد آرگومان ها خارج از محدوده باشد پیغام خطا می دهد.

- error: اگر در تابعی خطای اساسی رخ دهد، پیغام خطا صادر می کند و به کار تابع پایان می دهد.
  - صورت کلی این تابع به صورت زیر است
  - error('msg')
    - که msg کاراکتر رشته ای حاوی پیغام خطا است.
    - این تابع با nargchk به خوبی کار می کند. به عنوان مثال
      - function CheckInputs(x, y, z)
      - error(nargchk(2, 3, nargin))
- warning: پیغام هشدار نشان می دهد ولی برنامه را ادامه می دهد.
  - صورت کلی به فرم زیر است.
  - Warning('msg')
- inputname: نام واقعی متغیری که با شماره آرگومان خاصی مطابقت دارد به ما می دهد.
  - صورت کلی به فرم زیر است
  - name = inputname(argno);
    - argno شماره آرگومان است.
    - اگر آرگومان یک متغیر باشد، نام آن برگردانده می شود.

مثال) استفاده از آرگومان های اختیاری

- تابع مقدار x, y را می گیرد و به نمایش قطبی معادل شامل اندازه و زاویه تبدیل می کند
- اگر یک آرگومان وارد شود، تابع فرض می کند که y صفر است و فقط اندازه را به ما می دهد.
- ابتدا تابع با تعداد کم یا زیادتر از حد مجاز آرگومان بررسی می شود.

```
» [mag angle] = polar_value
??? Error using ==> polar_value
Not enough input arguments.
» [mag angle] = polar_value(1,-1,1)
??? Error using ==> polar_value
Too many input arguments.
```

- در ادامه تابع در حالت های مختلف چک می شود.

```
» [mag angle] = polar_value(1)
mag =
1
angle =
0
» [mag angle] = polar_value(1,-1)
mag =
1.4142
angle =
-45
```

```
» mag = polar_value(1,-1)
mag =
1.4142
» [mag angle] = polar_value(1,-1)
mag =
1.4142
angle =
-45
```

- فراخوانی تابع با مقدار ۰ و ۰ منجر به پیغام warning می شود

```
» [mag angle] = polar_value(0,0)
Warning: Both x any y are zero: angle is meaningless!
> In d:\book\matlab\chap6\polar_value.m at line
32
mag =
0
angle =
0
```

خلاصه

- توابع نوع خاصی از m-file هستند که داده را از طریق آرگومان های ورودی دریافت و نتیجه را از طریق آرگومان های خروجی برمی گردانند.
- هر تابع دارای محیط کار مستقل خود است.
- تابع nargin تعداد واقعی آرگومان های ورودی را معین می کند.
- داده ها را می توان در حافظه جهانی قرار داد.
- می توان داده های داخلی درون یک تابع را با قرار دادن در حافظه دائم، حفظ نمود.