```
>> ce = {[1    2    3; 5    6    7], 'yes', 3 > 2};
>> ce {1}(2, 2)
ans =
      6
```

مثال:

```
>> x = rand (3, 3);
>> y = rand (3, 3);
>> z = rand (3, 3);
>> w {1} = x;
>> w {2} = y;
>> w {3} = z;
>> w
ans =
    [3×3 double]    [3×3 double]    [3×3 double]
```

مثال:

```
>> x {1} = rand (3, 3);
>> x {2} = rand (3, 3);
>> x {3} = rand (3, 3);
        .
        .
        .
        .
        .
>> x {9} = rand (3, 3);
>> x {1}
ans =
    0.8462    0.6721    0.6813
```

```
    0.5252    0.8381    0.3795
    0.2026    0.0196    0.8318
```

العنصر الموجود في السطر الثاني والعمود الثاني في مصفوفة (الخلية) الأولى (2, 2) {1} x >>

```
ans =
    0.8381
```

مثال: برنامج لجمع المصفوفات التسعة في المثال السابق في مصفوفة واحدة.

```
>> L = length (x);
>> sum1 = 0;
>> for i = 1: L
     b = x {i};
     sum1 = sum1 + b;
   end;
```

يجبر الإيعاز celldisp برنامج MATLAB على إظهار محتوى الخلايا بالنموذج العادي، واليك المثال التالي الذي يوضح ذلك:

```
>> celldisp (A)
A (1, 1) =
        1    2    3
        4    5    6
        7    8    9
  A (2, 1) =
        Ali Ahmed
  A (1, 2) =
        2.0000 + 3.0000i
  A (2, 2) =
        12    10    8    6    4    2    0
```

كما يِظُهر البرنامج محتوى الخلية المفردة عبر طلب محتوى الخلية باستخدام عنونة المحتوى، وهذا

يتم بشكل مختلف عن فهرسة الخلية التي تعّرف الخلية فقط دون الدخول إلى محتوى الخلية، فمثلا:

```
>> A {2, 2}
ans =
    12   10   8   6   4   2   0
>> A (2, 2)
ans =
    [1×7 double]
>> A (1, :)
ans =
    [3×3 double]    [2.0000 + 3.0000i]
```

لاحظ بأن البرنامج استخدم لجميع الخلايا السابقة الاسم ans، وذلك لان خلايا البيانات المخزونة ليس لها اسم محدد.

لقد استخدمنا سابقاً الأقواس المربعة لإنشاء المصفوفات العددية، وتعمل أقواس المجموعة نفس العمل بالنسبة للخلايا، وتفصل الأعمدة بفواصل بينما تفصل الأسطر بفواصل منقوطة. واليك المثال التالي:

```
>> B = {[1    2], 'John Smith';  2 + 3i, 5}
B =
    [1×2 double]          'John Smith'
    [2.0000 + 3.0000i]    [5]
```

من المألوف عند استخدام المصفوفات العددية أن تُملأ المصفوفة بعناصر صفرية ثم تُملأ من جديد بالبيانات اللازمة، ويمكن استخدام نفس المنهج في مصفوفات الخلايا، حيث ينشأ الايعاز cell مصفوفة خلية ويملؤها بمصفوفات عددية فارغة [ ] ولنأخذ المثال التالي:

```
>> C = cell (2, 3)
C =
    [ ]   [ ]   [ ]
    [ ]   [ ]   [ ]
```

ما إن يتم تعريف مصفوفة الخلية فأنة يمكن تعميم الخلايا عن طريق عنونة المحتوى وفهرسة الخلايا، كما يبينه المثال التالي:

```
>> C (1, 1) = 'The does n't work'
    Error
```

لقد استخدمنا هنا في الجانب الأيسر دليل الخلية وبالتالي، يجب أن يكون الطرف الأيمن خلية وهذا ما سبب ظهور الخطأ، وليس كوننا لم نُخط محتوياتها بأقواس مجموعة.

```
>> C (1, 1) = {'The does n't work'}
C =
    'The does n't work'    [ ]    [ ]
    [ ]                    [ ]    [ ]
>> C (2, 3) = {'This works too'}
C =
    'This does work'       [ ]              [ ]
    [ ]                    [ ]              'This works too'
```

وبسبب وجود أقواس المجموعة في الجانب الأيسر من العبارة الأخيرة، فان برنامج MATLAB سيضع الخيط الرمزي في الخلية المعنونة. ويوجد هنا مرة أخرى عنونة محتوى، بينما تعتبر العبارة الأصلية مثالاً عن فهرسة المصفوفة.

التعامل مع مصفوفة الخلية

يمكن أن نستخدم الأقواس المربعة أيضاً لضم مصفوفات الخلايا ضمن مصفوفات اكبر، كما هو الحال للمصفوفة العددية، واليك المثال التالي:

```
>> A
A=
    [3×3 double]           [2.0000 + 3.0000i]
    'Ali Ahmed'            [1×7 double]

>> B
B =
    [1×2 double]           'John Smith'
    [2.0000 + 3.0000i]     [5]
```

```
>> C = [A; B]          (متساوية الأبعاد)
C =
    [3×3 double]        [2.0000 + 3.0000i]
    'Ali Ahmed'         [1×7 double]
    [1×2 double]        'John Smith'
    [2.0000 + 3.0000i]  [5]
```

يمكن تحديد خلايا جزئية لإنشاء خلايا جديدة عبر استخدام تقنيات مناسبة لعنونة مصفوفة الخلية كما في المثال التالي:

```
>> D = C ([1 3], :)
D =
    [3×3 double]        [2.0000 + 3.0000i]
    [1×2 double]        'John Smith'
```

ويمكن حذف سطر مصفوفة الخلية عبر استخدام الخلية الفارغة.

```
>> C (3, :) = [ ]
C =
    [3×3 double]        [2.0000 + 3.0000i]
    'Ali Ahmed'         [1×7 double]
    [2.0000 + 3.0000i]  [5]
```

ويستخدم الايعاز reshape لتغير مواضع الخلايا، ولكنه لا يستطيع إضافة أو حذف الخلايا وليبان ذلك، نأخذ المثال التالي:

```
>> x = cells (3, 4);
>> size (x)
ans =
     3    4
>> y = reshape (x, 6, 2);
>> size (y)
ans =
```

6 2

أي إن الايعاز reshape يعيد تشكيل أية مصفوفة بدون تغير نوعها، وكذلك يعيد الايعاز size حجم أي نوع من المصفوفات.

كذلك يعيد الايعاز repmat بالتعامل مع مصفوفات الخلايا حيث يعمل على تكرارها وفقاً للتكرار المطلوب.

مثال:

```
>> y
y =
    [ ]  [ ]
    [ ]  [ ]
    [ ]  [ ]
    [ ]  [ ]
    [ ]  [ ]
    [ ]  [ ]
>> z = repmat (y, 1, 3)
z =
   [ ]  [ ]  [ ]  [ ]  [ ]  [ ]
   [ ]  [ ]  [ ]  [ ]  [ ]  [ ]
   [ ]  [ ]  [ ]  [ ]  [ ]  [ ]
   [ ]  [ ]  [ ]  [ ]  [ ]  [ ]
   [ ]  [ ]  [ ]  [ ]  [ ]  [ ]
   [ ]  [ ]  [ ]  [ ]  [ ]  [ ]
```

السلاسل الرمزية

تكمن قوة برنامج MATLAB الحقيقية في القدرة على التعامل مع الأرقام، ولكنه يحتاج أحياناً إلى التعامل مع النصوص كما في حالة وضع العناوين وأسماء المحاور على الرسومات.

تركيب السلسة الرمزية

تتألف السلاسل الرمزية في لغة MATLAB من مصفوفات عددية خاصة من قيم ASCII والتي تعيد أظهار السلسة الرمزية، فمثلاً:

```
>> t = 'How about this character string?'
t =
    How about this character string?
>> size (t)
ans =
      1    32
>> whos                    إيعاز عرض أسماء المتغيرات وحجومها وعدد بياناتها وصنفها
```

Name	Size	Bytes	Class
ans	1×2	16	double array
t	1×32	64	character array

```
   Grand total is 34 elements using 80 bytes
```

إن السلاسل الرمزية ببساطة هي نص محاطة بفاصلة علوية مفردة. ويعتبر كل حرف في السلسة عنصراً من مصفوفة، والتي نحتاج إلى بايتين لتخزين كل حرف، ونختلف بذلك عن 8 بايت المخصصة لكل عنصر من عناصر المصفوفة العددية أو مضاعفة الدقة. ولرؤية التمثيل ASCII لسلسلة رمزية

نحتاج فقط لإجراء بعض العمليات الرياضية على السلسلة أو استخدام الايعاز double، وكما في المثال التالي:

```
>> u = double (t)
u =
    Columns 1 through 12
      72   111   119   32   97  98   111   117   116   32   116   104
    Columns 13 through 24
      105  115   32    99   104 97   114   97    99    116  101   114
    Columns 25 through 32
      32   115   116   114  105 110  103  63
>> char (u)
ans =
      How about this character string?
>> char (u (1))
ans =
      H
```

وبما إن السلاسل هي مصفوفات، لذلك يمكن التعامل معها بكل أدوات التعامل مع المصفوفات المتوفرة في لغة MATLAB، فمثلاً:

```
>>  u = t (16: 24)
u =
   character
```

وتعنون السلاسل تماماً كما تعنون المصفوفات، وتحوي العناصر من 16 إلى 24 في المثال أعلاه الكلمة character

```
>> u = t (24: -1: 16)
u =
   retcarahc
```

وهنا تمت تهجئة الكلمة character بشكل عكسي.

```
>> u = t (16: 24)'
u =
    c
    h

      a
      r
      a
      c
      t
      e
      r
```

وتم هنا تحويل كلمة character إلى مصفوفة عمود عبر عملية مدور(منقول).

يمكن دمج المصفوفات الرمزية وكالاتي:

```
>> u = 'Hameed ';
>> v = 'Aiad';
>> w  = [u  v]
w =
    Hameed Aiad
```

ويسمح لنا الايعاز disp إظهار السلسلة بدون طباعة اسم المتغير كما في المثال التالي:

```
>> disp (u)
   Hameed
```

ويمكن أن تتملك السلاسل الرمزية (كما في باقي المصفوفات) عدة اسطر، ولكن يجب أن يحوي كل سطر عدداً متساوياً من الأعمدة، لذلك يجب استخدام الفراغات لجعل طول كل الأسطر متساوية كما في المثال التالي:

```
>> v = ['character strings having more than   '
        'one row must have the same number'
```

```
        'of columns just like arrays!          ']
v =
    character string having more than
    one row must have the same number
    of columns just like array!
```

وينشئ الايعاز char مصفوفة نصية متعددة الأسطر انطلاقاً من سلاسل مستقلة مختلفة الطول، كما في المثال التالي:

```
>> legends = char ('Wilt', 'Russel', 'Kareem', 'Bird', 'Magic', 'Jordan')
legends =
          Wilt
          Russel
          Kareem
          Bird
          Magic
          Jordan
>> size (legends)
ans =
      6     6
```

تحويل الأعداد إلى سلاسل رمزية وبالعكس

قد نرغب في العديد من الحالات بتحويل النتائج العددية إلى سلاسل رمزية واستخراج البيانات العددية من السلاسل الرمزية. يزودك برنامج MATLAB بالإيعاز num2str و int2str و fprintf وغيرها لتحويل النتائج العددية إلى سلاسل رمزية، واليك الأمثلة التالية على التحويل:

```
>> int2str (35)
ans =
      35
>> class (ans)
 ans =
```