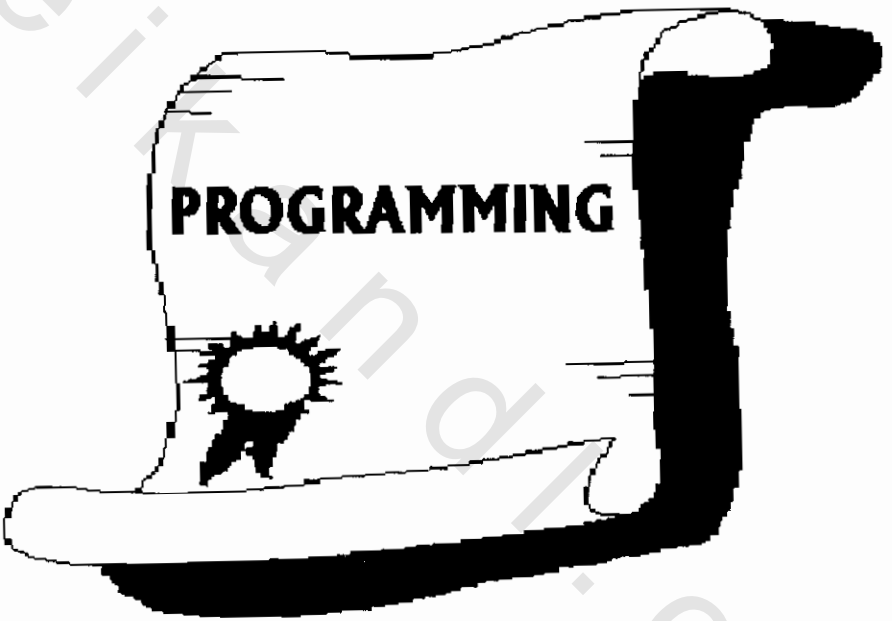
الباب السابع
البرمجة فى ماثيماتيكا



فى هذا الباب سوف نتعرف على أوامر برنامج ماثيماتيكا والخاصة بالموضوعات الآتية :

١. منظومة الإجراءات **Procedure**
٢. الحلقات التكرارية **Loops**
٣. أوامر الانتقال المشـروط **Conditionals**

الباب السابع

البرمجة فى ماثيماتيكا

عند بناء الحسابات فى ماثيماتيكا غالبا ما نحتاج أن نربط مجموعة الأوامر معا ويمكن أن يتم ذلك باستخدام منظومة إجراءات Procedure .

١ . منظومة الإجراءات Procedure

منظومة الإجراءات هى عبارة عن متتابعة من أوامر أو تعبيرات ماثيماتيكا بحيث يفصل بين كلا منها علامة الفصلة المنقوطة ; وقيمة التعبير الأخير فى المنظومة يمثل الناتج النهائى .

تعريف منظومة تتكون من مجموعة من الأوامر

Command1;Command2;...

```
In[1]:=r=(1+x)^2;r=Expand[r];r-1
Out[1]= 2 x + x^2
```

فى هذا المثـــال نلاحظ أن منظومة الإجراءات عبارة عن ثلاثة تعبيرات حســـابية يفصل كل منها عن الآخر بفصلة منقوطة كما نلاحظ أن الناتج النهائى هـــو قيمة التعبير الأخير (r - 1) فى منظومة الإجراءات .

ويمكن تعريف الدالة كمنظومة إجراءات `In[2]:= f[x_]:=(t=(1+x)^2;t=Expand[t])`

حيث نستخدم الأقواس ( ) لتوضيح

أن كل الإجراءات معا تمثل الدالة f

لحساب قيمة الدالة f عند x=a `In[3]:= f[a]`

`Out[3]= 1+2a+a^2`

عند الاستعلام عن قيمة t نجد أن المتغير `In[4]:= t`

t اصبح يمثل القيمة $1+2a+a^2$ `Out[4]= 1 + 2 a + a^2`

ويحتفظ بها حتى بعد الخروج من المنظومة .

وفى كثير من الأحيان نحتاج الى أن نجعل المتغيرات المستخدمة داخل أى منظومة إجراءات كمتغيرات موضعية (Local) بمعنى أن هذه المتغيرات تحتفظ بالقيم داخل منظومة الإجـــراءات فقط ولكن تفقد هذه القيم بعد إنهاء الحسابات فى المنظومة والخروج منها ويتم عمل ذلك فـــى ماثيماتيكا باستخدام الأمر Block أو الأمر Module كالآتى :

للإعلان عن أن المتغيرات x,y,... تمثل متغيرات موضعيـــة داخـــل منظومـــة الإجـــراءات procedure

```
Block[{x, y, ...}, procedure]
Or
Module[{x, y, ...}, procedure]
```

تحديد القيم الابتدائية x = xo , y = yo, ... للمتغيرات الموضعية داخل منظومة الإجـــراءات procedure

```
Block[{x=xo, y=yo, ...}, procedure]
Or
Module[{x = x0, y=yo, ...}, procedure]
```

وبذلك فإنه بواسطة الأمر Block أو الأمر Module يمكن تعريف أى متغيرات داخل منظومة الإجراءات الواحدة دون التأثير على القيم بخارج المنظومة وبالتالى يمكن تعريف نفس المتغير الموضعى داخل اكثر من منظومة إجراءات .

عند استخدام الأمر Block فى تعريف
الدالة g فان u يعامل كمتغير موضعى

```
In[5]:=
g[x_]:=Block[{u},u=(1+x)^2;u=Expand[u]]
```

لحساب قيمة الدالة g عند x = a

```
In[6]:= g[a]
Out[6]= 1 + 2a + a^2
```

عند الاستعلام عن قيمة u نلاحظ عدم
وجود قيمة لأن u متغير موضعى

```
In[7]:=u
Out[7]=u
```

تعريف x=5 خارج المنظومة ثم استخدام
الأمرModuleواعتبار أن x متغيرموضعى
وطباعة قيمة عشوائية للمتغير x

```
In[8]:=x=5;
Module[{x},x=Random[];Print[x];]
Out[8]=0.863718
```

عند الاستعلام عن قيمة x نلاحظ أن
x = 5 وهى القيمة الموجودة خارج
الأمر Module

```
In[9]:=?x
Out[9]=Global`x
x = 5
```

٢ . الحلقات التكرارية Loops

منظومة الإجراءات تسمح بتنفيذ مجموعة من تعبيرات ماثيماتيكا وفقا لترتيب هذه التعبيرات داخل منظومة الإجراءات وفى كثير من الأحيان نحتاج الى تنفيذ بعض العمليات بصورة متكررة ويتم ذلك داخل ماثيماتيكا باستخدام أوامر خاصة بالحلقات المتكررة Loops حيث تعمل الحلقات المتكررة على تكرار مجموعة متتالية من الأوامر بصورة متكررة لعدد محدود من المرات مع إمكانية التغيير الأوتوماتيكى لقيم المتغيرات داخل الحلقات التكرارية ، ومن أوامر ماثيماتيكا الخاصة بالحلقات التكرارية هو الأمر Do والذى يستخدم بصورة مشابهة كما فى لغات البرمجة مثل فورتران FORTRAN والأمر Do له استخدامات متعددة وصيغته العامة موضحه بالجدول الآتى :

الوظيفة التى يقوم بها الأمر	الصيغة العامة للأمر
حساب قيمة expr عدد n من المرات	Do[expr,{n}]
حساب قيمة expr بصورة متكررة وفقا للعداد i من i=1 الى i=imax بخطوة تساوى 1	Do[expr,{i,imax}]
حساب قيمة expr بصورة متكررة وفقا للعداد i من i=imin الى i=imax بخطوة تساوى 1	Do[expr,{i,imin,imax}]
حساب قيمة expr بصورة متكررة وفقا للعداد i من i=imin الى i=imax بخطوة تساوى istep	Do[expr,{i,imin,imax,istep}]
حساب قيمة expr لقيم i , j المعطاة	Do[expr,{i,imin,imax}, {j,jmin, jmax}]

```
In[1]:=t=x;Do[t=2(1+t),{3}];t
Out[1]=2 (1 + 2 (1 + 2 (1 + x)))
```

في هذه الحلقة يتم حســـــاب التعبير t=2(1+t) ثلاث مرات حيث t=x

```
In[2]:=Do[Print[m^2],{m,3}]
Out[2]= 1
        4
        9
```

في هذه الحلقة يتم طباعة m^2 لقيم m من 1 الى 3

```
In[3]:=Do[a=I^2+3j;Print[a],{I,2},{j,I}]
Out[3]= 4
        7
        10
```

في هذه الحلقة يتم حســـــاب التعبير $a = i^2 + 3j$ ثم طباعته لقيم i,j المعطاة

وفى برنامج ماثيماتيكا يمكن تكرار تطبيق نفس الدالة عدد محدود من المرات على تعبـــير معـــين باستخدام الأمر **Nest** كالآتى :

Nest[f,expr,n]	تطبيق الدالة **f** على التعبير **expr** لعدد **n** من المرات

```
In[4]:=Nest[f,x,3]
Out[4]= f[f[f[x]]]
```

لحساب f (f (f (x)))

```
In[5]:=f[x_]=(x+1)^2;Nest[f,x,2]
Out[5]=(1 + (1 + x)^2 )^2
```

تعريف الدالة $f(x) = (x+1)^2$
ثم حساب f (f (x))

```
In[6]:= Nest[f,1,2]
Out[6]= 25
```

لحساب f (f (1))

وفى برنامج ماثيماتيكا يمكن بناء الحلقات بحيث يتم إيقاف تنفيذ التكرار إذا لم يتحقق شرط معين وذلك باستخدام الأوامر **For , While** كالآتي :

الوظيفة التى يقوم بها الأمر	الصيغة العامة للأمر
حساب قيمة **start** والتحقق من الشرط **test** لتنفيذ **body** مع تكرار إضافة **step**	**For[start,test,step,body]**
يتم تكرار تنفيذ **body** إذا كان الشرط **test** متحقق	**While[test,body]**

حلقة باستخدام **For** حيث يتم البدء بقيمة **i=0** والتحقق من الشرط **i<3** لتنفيذ طباعة **i** مع إضافة **1** الى **i**

```
In[7]:=For[i=0,i<3,i=i+1,Print[i]]
Out[7]=
        0
        1
        2
```

تنفيذ الحلقة السابقة باستخدام **While**

```
In[8]:=i=0;While[i<3,Print[i];i=i+1]
Out[8]=
        0
        1
        2
```

وعند استخدام أوامر الحلقات فى ماثيماتيكا خاصة مع الأوامر For , While غالبا ما نحتاج الى تكرار تعديل قيم فى بعض المتغيرات المستخدمة داخل الحلقة ، وتوجد بعض الطرق المختصرة لأجراء مثل هذه التعديلات فى قيم المتغيرات والجدول الآتى يوضح ذلك .

العملية المختصرة	معنى العملية المختصرة
i++	زيادة قيمة i بمقدار 1 فيما يستجد مع الاحتفاظ بقيمة i السابقة داخل i++
i--	نقصان قيمة i بمقدار 1 فيما يستجد مع الاحتفاظ بقيمة i السابقة داخل i--
++i	زيادة قيمة i بمقدار 1 وجعل i هى القيمة الجديدة آي أن ++i تمثل i+1
--i	نقصان قيمة i بمقدار 1 وجعل i هى القيمة الجديدة آي أن --i تمثل i-1
i+ = d	زيادة قيمة i بمقدار d آي أن i=i+d
i- = d	نقصان قيمة i بمقدار d آي أن i=i-d
x*=c	ضرب x فى العدد c آي أن x = x * c
x/=c	قسمة x على العدد c آي أن x = x / c
{x,y}={y,x}	استبدال قيم x , y آي تغيير قيمة x لتصبح y وتغيير قيمة y لتصبح x

```
In[9]:=i=5;Print[i++];Print[i]
Out[9]=
        5
        6
```

في هذا المثال يتم طباعة قيمة i++ وقيمة i
ونلاحظ أن قيمة i++ هي قيمة i قبل الزيادة

```
In[10]:=i=5;Print[++i];Print[i]
Out[10]=
        6
        6
```

في هذا المثال يتم طباعة قيمة ++i وقيمة i
ونلاحظ أن قيمة ++i هي قيمة i بعد الزيادة

```
In[11]:=r=x;r+=3y;r
Out[11]=x + 3 y
```

في هذه المنظومة تم وضع r=x ثم زيادة قيمة r
بمقدار 3y فتصبح قيمة r الجديدة x+3y

```
In[12]:=a=3;b=7;Print[{a,b}];
        {a,b}={b,a};Print[{a,b}]
Out[12]=
        {3, 7}
        {7, 3}
```

استبدال قيم a , b

```
In[13]:=x=1;y=2;z=3;Print[{x,y,z}];
        {x,y,z}={z,x,y};Print[{x,y,z}]
Out[13]=
        {1, 2, 3}
        {3, 1, 2}
```

استبدال قيم x , y , z

In[14]:=For[i=1;t=x,i^2<10,i++,t=t^2+i;Print[t]]

Out[14]=

$$1+x^2$$
$$2+\left(1+x^2\right)^2$$
$$3+\left(2+\left(1+x^2\right)^2\right)^2$$

هذه الحلقة على الصورة **For[start,test,step,body]** حيث

start	**i=1 ; t=x**
test	**i^2<10**
step	**i++**
body	**t=t^2+i ; Print[t]**

٣ . أوامر الانتقال المشـروط Conditionals

عند بناء منظومة الإجراءات Procedure فى ماثيماتيكا غالبا ما نحتاج الى تنفيذ بعض العمليات إذا تحقق شروط معينة ويتم ذلك فى ماثيماتيكا باستخدام أوامـــر الانتقـــال المشـــروط الآتية :

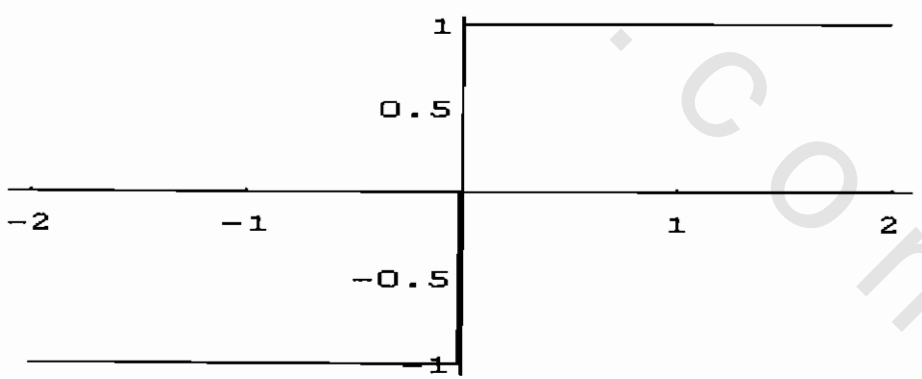
الوظيفة التى يقوم بها الأمر	الصيغة العامة للأمر
يتم تنفيذ then إذا كـــان test تحقـــق وخلاف ذلك يتم تنفيذ else	If[test,then,else]
إذا كان test تحقق يتم تنفيذ then وإذا كان test غير متحقـــق يتـــم تنفيـــذ else وخلاف ذلك يتم تنفيذ unknown	If[test,then,else,unknown]
يتم تنفيذ value المناظرة الى أول اختبـــار testi يتحقق	1,value1,test2,value 2, ...]

```
In[1]:= x=3;y=5;
        If[ x > y , Print[x] , Print[y] ]
Out[1]=
          5
```

إدخال قيم x , y ثم طباعة العدد الأكبر وقد تم استخدام أمر الانتقال المشروط If على الصورة If[test,then,else] حيث

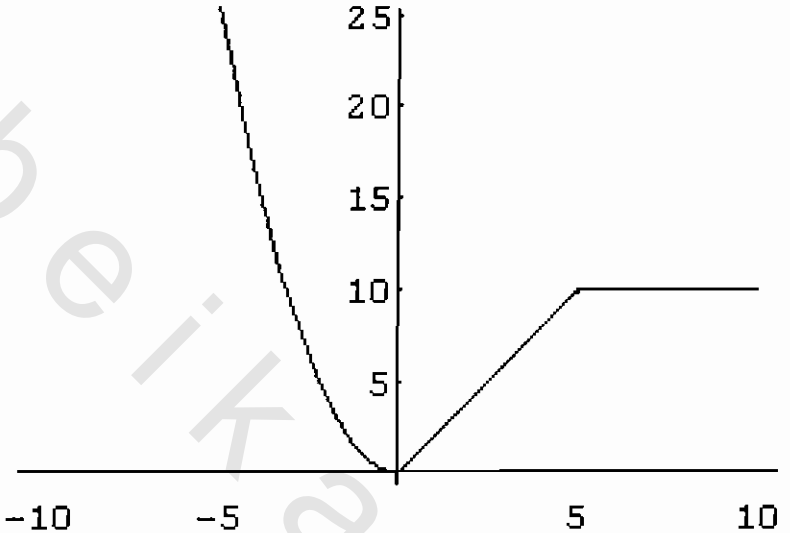
test	x > y
then	Print[x]
else	Print[y]

```
In[2]:=f[x_]:=If[x>0,1,-1];
                Plot[f[x],{x,-2,2}]
```



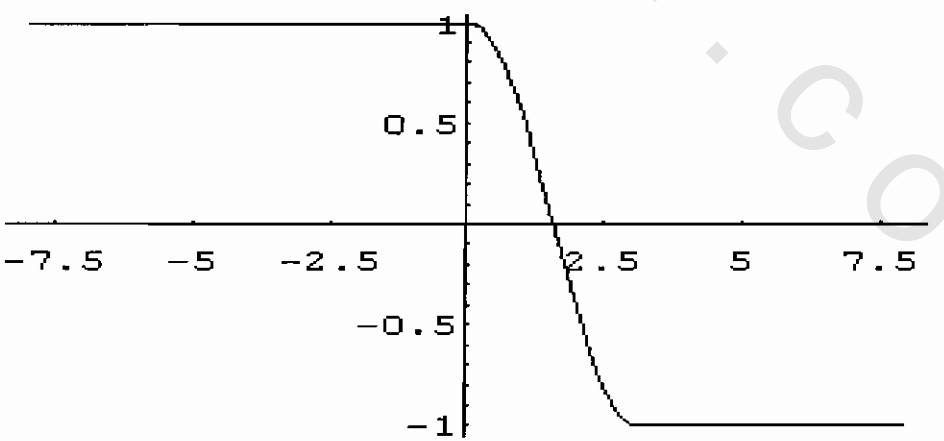
استخدام الأمر If في تعريف الدالة f[x] ثم رسم الدالة

```
In[3]:=r1[x_]:=Which[x<0,x^2,x>0 && x<5 ,2x,True,10];
       Plot[r1[x],{x,-10,10}]
```



استخدام الأمر Which في تعريف الدالة r1[x] ثم رسم الدالة

```
In[4]:=r2[x_]:=Which[x<0,1,x<N[Pi],Cos[x],True ,-1];
       Plot[r2[x],{x,-8,8}]
```



استخدام الأمر Which في تعريف الدالة r2[x] ثم رسم الدالة

منظومة تم فيها تعريف قائمة a من الأعـــــداد الحقيقية ثم حســـــاب وطباعـــــة العدد الأصغر من القائمة

```
In[5]:=a={5,2,7,55,-4,9,3,10,-24,44,65,-21};mi=a[[1]];
      Do[If[mi>a[[i]],mi=a[[i]],++i],{i,1,Length[a]-1}];
      Print["Minimum of the list  a = ",mi]

Out[5]= Minimum of the list  a =  -24
```

منظومة تم فيها حســـــاب وطباعـــــة العـــــدد الأكبر من القائمة a

```
In[6]:= ma=a[[1]];
      Do[If[ma<a[[i]],ma=a[[i]],++i],{i,1,Length[a]-1}];
      Print["Maximum of the list  a = ",ma]

Out[6]= Maximum of the list  a = 65
```

منظومة تم فيها حســـــاب وطباعة العـــــدد الأصغر والعدد الأكبر من القائمة a

```
In[7]:= mi=a[[1]];ma=a[[1]];
      Do[Which[mi>a[[i]],mi=a[[i]],ma<a[[i]],ma=a[[i]]];++i,
         {i,2,Length[a]-1}];
       Print["Minimum of the list a = ",mi];
       Print["Maximum of the list a = ",ma]


Out[7]= Minimum of the list a = -24
        Maximum of the list a = 65
```

وباستخدام الدوال الموجودة داخل بناء ماثيماتيكا يمكن حساب

العدد الأصغر من القائمة مباشرة باستخدام الدالة Min

والعدد الأكبر من القائمة مباشرة باستخدام الدالة Max

```
In[8]:=Min[a]
Out[8]=-24

In[9]:=Max[a]
Out[9]=65
```

المراجـــع

[1] - Wolfram , Stephen
Mathematica : A System for Doing Mathematics by Computer ,
Second Edition , Addison Wesley , 1991 .

[2] - Wolfram , Stephen
Mathematica : The Student Book ,
Addison Wesley , 1994 .

[3] - Abell , Martha L . and Braselton, Tames P . ,
The Mathematica Handbook
Academic Press , 1992 .

[4] - Maeder , Roman ,
Programming in Mathematica ,
Addison Wesley , 1992 .