

بسم الله الرحمن الرحيم

# *Perl Simple Tutorials*

# ماهي لغة البيرل؟؟؟

هي لغة برمجة عامة قد طورت هذه اللغة لكي يصبح في مقبورها معالجة

النصوص المكتوبة يدويا

يمكن اعتبار هذه الخطوة هي التطور التايخي لهذه اللغة

اهم الامور التي تستخدم فيها هي

تطوير صفحات الويب سايت

ادارة الانظمة

برمجة الشبكات

gui تطوير ال

الغرض من هذه اللغة هي ان تكون سهلة الاستخدام و متكاملة وتدعم كل من ال

procedural

o.o.p

كيف يتم تنفيذ برنامج في لغة البيرل

اولا وقبل ان تكتب البرنامج يجب ان يكون هذا البرنامج الذي انت بصدد ان

تنفذه يجب ان يكون قابل للتنفيذ يعني ماتجي تكتب اشياء ليس وتريدها تنفذ

اماعن الية التنفيذ يجب ان تكون كمايلي

command line

perl pro .pl

و كل شي انتهى

ام عن الطرق الاخرى فانا بلنسبة الي احب هذه الطريقة وعلى الرغم من هذا فانا

سوف اقوم بشرح الطرق الاخرى في الدروس القادمة

وهناك ايضا ملاحظة هامة جدا هي انه البرنامج المكتوب بلغة البيرل سواء

كام مكتوب من خطوة واحدة او اكثر فانه من غير المهم ان تتم كتابة هذه الدالة

main()

وهناك ايضا ملاحظة لابد من الوقوف عليها هي انه جملة البيرل تنتهي بعلامة

; الفارزة المنقوطة

اماعن التعليقات المكتوبة بلغة البيرل فأننا نكتب التعليق كما هو لكن فقط

#نضيف عليه ال

ex:-

CODE

```
# spawn love u all
```

-----  
جملة الطباعة وهي جملة سهلة و بسيطة ونكتب كمايلي

CODE

```
print "spawn love u";
```

هذه الحالة تكون عندماتكون الجملة التي نريدان نطبعها حرف يجب ان تحصر  
بين علامات الاقتباس  
اما عم الارقام فتكون كمايلي  
3x:-

CODE

```
print 1986;
```

من دون الحاجة الى علامات الاقتباس

-----  
perl variable type

انواع بيانات البيرل وهي تقسم الى ثلاثة انواع  
scalar  
arrays  
hashes

الان سوف نأخذ النوع الاول من المتغيرات الالوهو

scalar

اذا اخذنا معنى هذه الكلمة في اللغة العربية فنلاحظ ان هذه الكلمة هي تعني  
الموجهات او المدرجات  
الموجهات تقدم نوع واحد من البيانات وهي تتمثل كمايلي

CODE

```
my $name = "m_spawn";
```

CODE

```
my $friend = "all geeks";
```

ان قيم الموجهات هذه من الممكن ان تكون  
intgers من الممكن ان تكون  
قيم الموجهات من الممكن ان تكون فلو ت اي اعداد عشرية  
string قيم الموجهات من الممكن ان تون سلاسل حرفية او ميعرف ب  
المميزة وهي scalars هنالك عدد من ال  
line of noise  
punctution  
هذه الانواع من المتغيرات التي نكرناها من الممكن ان يتم استخدامها في كافة  
انواع الاهداف في لغة البيرل  
لكن الى حد الان نحن نتقدم في لغة البيرل و لانواجه صعوبات الان قدحان  
الوقت الذي يجب فيه ان نركز على هذه المتغير المهم الؤهو  
\$\_;

وهو المتغير الذي يطلق عليه برمجيا في لغة البيرل  
the default variable

والذي يستخدم عادة في ال  
the default argument

الان مبدئيا اصبح نعرف ما اهمية هذا المتغير و اين يتم استخدامه  
وهذا المتغير يجهز ضمنا  
by the certain looping constructors

-----  
arrays

النوعه الثاني من المتغيرات الذي يختلف عن سابقه في مايلي  
انه يقدم مجموعه من المتغيرات  
صيغته العامة هي

CODE

```
my @xxx = ("xxx", "xxx");
```

الان ناخذ مثال على هذا النوع من انواع البيانات

CODE

```
my @friend = ("geek 1", "geek2", "all geeks");
```

الان اخذنا مثال عليه الان راح نحلل هذا الكود  
z3r0 ان المصفوفة دائمتكون معنونة بالعنوان 1-  
اذا اردنا ان نطبع مقدار نعمل مايلي 2-

CODE

```
print my $friend[0];
```

وسوف تلاحظ النتيجة الاتية

geek1

الان ناتي و ناخذالمتغيرات المميزة في المصفوفات وهي المتغير الهام جداو  
السهل في نفس الوقت  
وهذا المتغير هو

`$#array`

وتكون صيغته العامة هي

**CODE**

```
print $array[$#array]
```

هذا المتغير تكون هذه صيغته العامة و هو يستخدم في الحالات التي تكون فيها  
اعدادالمتغيرات في المصفوفة كبيرة جدا لذا يتم استخدام هذا المتغير لكي يقوم  
باعطائنا المتغير الاخير!! في المصفوفة انن وظيفة هذا المتغير هو اعطاء مقدار  
المتغير الاخير في المصفوفة  
ملاحظة هامة ان المتغير

`$#array`

هو يعطيك مقدار المتغير الاخير وليس يطبعك موقعه  
لذا يجب الانتباه

هناك انواع اخرى من المتغيرات المميزة الا وهي

`1-@argv`

(the command line argument 2 ur script)

`2-@_`

(the argument passed to sub routine)

-----

**HASEHES**

النوع الثالث من انواع المتغيرات في لغة البيرل

ماهو الهاش؟؟!!

يمكن القول ان الهاش يقدم لك كل شئ وقرينه ايان المتغيرات في تكون على  
شكل ازواج

يعني كل شي وقرينه مايمكن ان نقول مثلا ان  
لون البطيخ احمر  
لون العنب اسود وهكذا  
يتم تمثيلها برمجيا كمايلي  
ناخدمثال على هذه الحالة

CODE

```
my %colour = ("apple", "red", "blackberry", "black");
```

وحالات ال  
=>

مسموحة يعني من الممكن تمثيل العبارة اعلاه كمايلي

CODE

```
my %fruit_color = (  
    apple => "red",  
    banana => "yellow",  
);
```

وهكذا  
لكن انا اردنا نطبع مقدار موجود في الهاش كيف تتم العملية ???  
ان العملية تتم كمايلي

CODE

```
print $colour{"apple"};
```

عند تنفيذ هذه العبارة كمايلي

red

الان نجى ناخذ الدوال المميزة في هذا النوع من المتغيرات  
الا وهو  
%env

وهي مختصر لكلمة  
environment

وهي تعني محيط او بيئة

### *perl data structures*

الان في الدرس الثاني سوف نتناول موضوع سهل ومهم لكل من يريد ان يبرمج  
برنامج في لغة البيرل  
ول شي راح ناخذ المتغيرات  
اول شي ان المتغيرات من المعروف انها من الثوابت المهمة في كل لغة وفي كل  
لغة تكون لها صيغة عامة اما الصيغة العامة لها في لغتنا هي

### CODE

```
my $var="value";
```

; هذه هي الصيغة العامة لها ولكن هالك نقطة مهمة هي انه ال  
مهمة جدا وضرورية لانه اذا تم تنفيذ الجملة من دون هذه العلامة فان الحاسبة  
لن تعلم بانتهاء الجملة و لنسوف تمر بمرحلة من الارتباك وسوف تظهر رسالة  
خطأ للمستخدم  
وهناك صيغة اخرى لتعريف المتغير هي

### CODE

```
$var="value"
```

اذن عرفنا انه كلمة

my

هي كلمة من الممكن ان يتم الاستغناء عنها يعني انا تم تنفيذ الجملة البرمجية المكتوبة بلغة البيرل من دون هذه الكلمة فان النتيجة هي نفس النتيجة الناتجة my منم الجملة المحتوية على كلمة ال

-----

الان بعد اخذنا نظرة عامة على المتغيرات العامة في لغة البيرل الان راح ناخذ conditional and looping constrctures

طبعا لغة البيرل مثل باقي اللغات تحتوي على جمل شرطية بكل حالاتها وانواعها

لكن هانل بعض النسخ منزوعة منها جملة ال case/switch

و لكنها متوافرة في النسخ الاحداث مثل الاصدار ال 5.8 والاصدارات الاعلى منه صيغة جملة الشرط في لغة البيرل هي كمايلي

CODE

```
if (conditions){
    some commands
}
else if (other commands){
    some commands
}
else(condition)
```

ولكن طبعا كما هو الحال مع التطور البرمجي المعهود في الانظمة المفتوحة المصدر نلاحظ هنالك حالة معاكسة لهذه الحالة وهي حالة النقيض لها وهي كمايلي

## CODE

```
unless (conditions){  
    some commands  
}
```

طبعاً هناك موضوع مهم يجب ان انبه اليه هو انه الشرط مهما كان صغيراً او قليل الحجم او حتى مهما كان غير مهم لكن يجب ان يتم وضعه بين الاقواس والا فان البرنامج سوف لن ينفذ  
الان راح ناخذ مثال بسيط على برنامج يحتوي على جملة ال  
if

## CODE

```
$a=55;  
if($a<85){  
    print "spawn love u"  
}
```

-----

while statement

تكون الصيغة العامة لهذه العبارة هي كما يلي

## CODE

```
while(expression){  
    some commands  
}
```

وكذلك هناك الحالة المعاكسة لهذه الحالة هي حالة ال

CODE

```
unless(expression)
{
    some commands
}
```

-----

for statement

الان سوف نأخذ عبارة برمجية تعتبر من أهم العبارات البرمجية الموجودة  
التركيب البرمجي و الصيغة العامة لهذه العبارة هي كما يلي

CODE

```
for($i=0;$i<$max;$i++){
    some commands
}
```

الان سوف نأخذ مثال برمجي على هذه الحالة

CODE

```
$a=33;
for($a=0;$a<32;$a++){
    print $a;
}
```

-----

build in function

عندما تنزل نظام الى حاسبتك نظام مفتوح المصدر تنزل معه عدة من التطبيقات  
و ينزل معه هذه اللغة و من الطبيعي فانه كل لغة برمجية يكون لها نوعان من  
الداول

النوع الاول هي النوع الذي يعرفه المبرمج اثناء كتابة برنامج  
الدوال المضمنة و المبنية داخل اللغة  
ونحن الان قد اخذنا بالفعل عددا من هذه الدوال منها جملة الطباعة وهي  
print

-----  
العمليات الرياضية في لغة البيرل

Mathematics

+ الجمع  
- الطرح  
\* الضرب  
% القسمة

-----  
المقارنات العددية numeric compartment

== equal to يساوي  
!= not equal to لايساوي  
< less than اقل من  
<= less than or equal to اقل من او يساوي  
> greater than اكبر من  
>= greater than or equal to اكبر من او يساوي

-----  
المقارنة بين السلاسل الحرفية sting copmeration

المساواة eq equality  
عدم المساواة NE INEQUALITY  
اقل من LT LESS THAN  
اكبر من GT GREATER THAN  
اصغر من او يساوي LE LES THAN OR EQUAL TO  
اكبر من او يساوي GE GREATER THAN OR EQUAL TO

-----  
وطبعاً نحن الآن نورد هذه العمليات ونمر عليها مرور الكرام لكن بعد ان نتوغل قليلاً في لغة البيرل سوف نورد كل عملية من هذه العمليات بشكل مفصل  
-----

## BOOLEAN LOGIC

و AND &&

او OR ||

نفي not !

-----  
miscellaneous منوعات

== assignment مساواة

ربط السلاسل الحرفية . string contact

x string multiplication

ان شاء الله فهتمم الدرس



## Perl reference tutorial

ان لغة البيرل واحدة من اهم اللغات الموجودة حاليا نظرا لمدى سهولتها ولمدى مرونتها في التعامل مع ال

complex data structures

وبما ان لغة البيرل لها القدرة على التعامل

complex data structures

مثل

المصفوفات ذات البعد الثنائي

الهاشات المتعددة او المتشعبة

وان لغة البيرل كان لابد لها من ان تقدم بعض الخائص لدعم كل هذه الامكانيات

او الذي يعرف في اللغة العربية reference لنقدمت لغة البيرل شئ يدعى ب

بالمصدر ان استخدام هذا المصدر هو اشبه باستخدام المفتاح للتعامل مع

complex data structures

الان قديطرح احدسؤال ويقول لمانا قديحتاج احد لا وهو لمانا قديحتاج

complex data structures شخص الى هذه

والجواب على هكنا سؤال هو انه المبرمجين السابقين في لغة البيرل وبالاخص

لغة البيرل ذات الاصدار الرابع

ان من اصعب الاشياء لديهم كانت هي الهاشات التي كانت بشكل قائمة وكيفية

تمثيلها وتقديمها ولكن الان ايضا قديسأل احد لمانا قديحتاج شخص الى هذه الى

استعمال هذه الهاشات وكما ذكرنا سابقا ان فكرة عمل الهاش هي انه هو كل

ونظيرة يعني

تفاح احمر

موز اصفر

وهذه اشارة الى انه لون التفاح احمر ولون الموز اخضر

والان ناخذ مثال عملي لنفرض كان لديك هذا الهاش كيف منتسوف تحله

Baghdad-> Iraq

Mosul -> Iraq

Cairo -> Egypt

Gaza -> Palastine

Jerusalem->Palastine

الان الغاية من هذاالمقطع البرمجي لو كان لديك هكنا فهل كنت سوف تحله  
بهذه الطريقة الطويلة الجواب لا طبعاً والحل الطبيعي على هكنا مقطع هو  
انه نعمل على ترتيب المقاطع كمايلي

Iraq: Mosul,Baghdad

Egypt: Cairo

Palastine : Gaza ,jerusalem

هنسوف يكون الحل على هكنا سوال الان يجب ان نعرف ان كانت قيم الهاش  
من غير الممكن ان تكون قوائم يمكن القول في هذه الحال ان الكود سوف لن ينفذ  
في لغة البيرل ذات الاصدار الرابع الهاشات كانت من غير الممكن ان تكون سوى  
على شكل ستريينك وفي هذه الحالة لو كنت من مشنخدمين لغة البيرل ذات  
الاصدار الرابع كان عليه ان يدمج كل هذه المدن في ستريينك واحدة فقط بطريقة  
ما وعندما يحين الوقت لكي الناتج او تطبعه عليك ان تكسر هذه ال ستريينك الى  
قائمة ثم تقوم بترتيب هذه العناصر ثم اعادتها الى ستريينك واحدة وهذا يمكن  
اعتباره الفوضى العارمة بذاتها

## THE SOLUTION

مع تطور الزمن تقدمت لغة البيرل وقدمت لنا الاصدار الخامس والذي فيه

hashes value must be a scalar value

Reference الحل لهذه الحالة هو ال

قديسأل احدهما هو هذا ال Reference الان بعد كل هذا الحديث عن هذا ال

Reference

Reference انن يمكن تعريف ال

التي من الممكن ان تشير الى المصفوفة بكاملها او scalar على انه قيمة من نوع  
من الممكن ان تشير الى الهاش بكامله

في لغة البيرل هي اشبة بالاسم للمصفوفات والهاشات Reference ال

يشير الى شي واحد فقط Reference ال

الى مصفوفة يمكن ان تشير الى المصفوفة بكاملها من Reference وانا كان لديك الى هاش من الممكن ان تشير الى الهاش كله Reference خلاله وانا كان لديك Reference من خلال ال

array وانت لايمكنك ان تملك هاش تكون القيم التي بداخله من نوع لكن كام نكرنا لكم ان ال scalar قيم الهاش يجب فقط ان تكون من نوع من الممكن ان يشير الى مصفوفة كاملة وكما نكرنا ان قيم ال Reference لذا من الممكن الحصول على لكن من الممكن الحصول scalar هي Reference على  
a hash of  
references to arrays,

## Syntax

references الان سوف نتكلم عن طرق الحصول على الطريقة الاولى هي اذا علامة

\  
في بداية المتغير او امام المتغير فانك في مثل هذه الحالة سوف تحصل على لذلك المتغير references

EX:-

\$aref=\@array 1:- \$aref now holds a reference to @array

\$href=%hash 2:- \$href now holds a reference to %hash

\$sref=\$scalar 3:- \$sref now holds a reference to \$scalar

الى مصفوفة reference الان تمسك 1:-

الى هاش reference الان تمسك 2:- \$href

الى scalar reference الان تمسك 3:- \$sref

من الطريقة الاولى اصبحت سهلة ان reference و الان اتوقع طريقة عمل ال شاء الله

\$href , \$aref في المتغير مثل reference ولكن ماذا يحدث عندما يخزن ال  
في هذه الحالة من الممكن ان تقوم بعملية نسخه تماما كما هو الحال مع اي قيمة  
scalar من نوع

Ex:-

```
$xy = $aref;      1:- $xy now holds a reference to @array  
$p[3] = $href;    2:- $p[3] now holds a reference to %hash  
$z = $p[3];       3:- $z now holds a reference to %hash
```

الى مصفوفة reference الان تحمل \$xy

الى هاش reference الان تحمل \$p[3]

الى هاش reference الان تحمل \$z

reference هذه الامثلة تعلم الشخص كيف يعمل  
الى متغير يحمل اسم

## MaKe RuLE TwO

الان بعد ان تناولنا الطريقة الاولى واصبحت مفهومة الى الجميع بان الله  
الان شوف تناول الطريقة الثانية  
وهي ال

جديدة و عيالناتج الى تلك anonymous array وهي تصنع [ITEMS]  
الى تلك المصفوفة references

الى ذلك references وتعيد الى anonymous hash تصنع [ITEMS] و ال  
الهاش

EX:-

```
$href = { spawn => 5 , perl => 4 };
```

\$href now holds a references to a hash

```
$aref = [ "spawn", "soqor", "securitygurus" ];
```

\$aref now holds a references to an array

### 1:-NOT3\*

بس احب الى ان انبه انه في مثال الهاش انه عدلا احرف هو الذي قمت  
مكونة من 5 احرف وكذا الحال مع كلمة بيرل spawn باستخدامه يعني انه كلمة  
فهي مكونة فقط من الربع احرف للحيث ان انبه

### 2:-NOT3\*

عندي ملاحظة ثانية لكنها يمكن القول انه مجرد تنبيه مو اكثر هو انه  
فارغ anonymous hash عندما تكتب بلا عناصر اي فارغة فانها تصنع {} :-1  
anonymous array عندما تكتب فارغة بدون اي عناصر فانها تصنع [] :-2  
فارغة  
وهذه ملاحظة هامة نوعا ما

## UsIng ReFeReNcEs

الان قد يسأل شخص ماهي الفائدة المرجوة من او ما 1 يمكن ان نعمل  
حالما نحصل عليه ولقد علمنا مسبقا انه عبارة REFERENCES باستخدام ال  
وانه من الممكن ان يتم استعادته في اي وقت مثل اي scalar عن قيمة من نوع  
والان سوف نستعمل طريقتان من اجل استخدامه scalar قيمة من نوع

### UsE RuLe 1

في الاقواس المتعرجة {} هذه الاقواس array references دائما يمكنك باستخدام  
اعني كمثال يعني

This

“@ {aref}”

instead of

\$aref

Some Examples

1:-@a

@ { \$aref }

an array

2:-reverse @a      reverse @{\$ref}      reverse an array  
 3:-\$a[3]      \${\$aref}[3]      an element of the array  
 4:-\$a[3]=[144]      \${\$aref}[3]=[144]      Assigning an element

1:- مصفوفة عادية

2:- تعني عكس او قلب في اللغة العربية reverse عكس المصفوفة حيث كلمة -:  
 عنصر في المصفوفة -:3

4:- تحديد عنصر في المصفوفة

**\*NOT3:-3**

الان على الجانبين توجد تعابير و ان التعابير التي على الطرف الايمن هي  
 ذات نفسها التعابير التي على الطرف الايسر هي ذات نفسها الموجودة على الطرف  
 الايمن الان كيف لو كان نريد ان نطبق بعض العبارات البرمجية على و بشكل  
 في الوقت الذي انا كل ماملك هو عبارة عن ال (Loop) خاص عبارات ال  
 اذا كان لديك هكنا سوال فان الجواب عليه يكون كمايلي reference

array في حالة ال

```
for my $element (@array){
    xxxxxx;
}
```

نقوم بعمل تعديل وهو كمايلي reference هذه في الحالات العادية لكن مع ال

```
for my $element (@{$aref}){
    xxxxxxxx;
}
```

ونفس السؤال مع حالة الهاش عندما نقول كيف نطبع هاش بينما نحن كل ما  
 reference نملك هو

الحل في هذه الحالة

```
for my $keys (keys %hash){
    xxxxxxxx;
}
```

هذه في الحالة العادية و الان سوف نرى ما نلوف يحدث في حالة ال

reference

```
for my $keys (%{$herf}){  
    print "$key => ${$href}{$key}\n";  
}
```

reference الان بعد ان اخذنا الطريقة الاولى من طرق صنع ال  
الان نأتي الى الطريقة الثانية



## Use Rule 2

ان استخدام الطريقة الاولى هو حقيقة كل ما تريد لانها كل ماتحتاج تقريبا لانها  
reference تخبرك كيف تماما ماناتفعل او تريد ان تفعل مع ال  
وان استخدام الطريقة الاولى والتدوين فيها يمكن القول انه مزعج لذا فانه في  
هذه الطريقة يوجد اختصار  
في الطريقة الاولى كنا نقوم بعمل هذه  
“\${\$ref}[3]”

الان وفي هذه الطريقة الجديدة سوف نقوم بعمل هذا التغير  
\$aref->[3]

هي العنصر [3]->\$aref للمصفوفة فأن reference تحمل ل \$aref اذا كانت  
الرابع من المصفوفة وقلنا العنصر الرابع لانه نكرنا في دروس ماضية انه تعداد  
العنصر في المصفوفة يبدأ من الرقم 0  
هذا التغير يدل على الطريقة المعقدة الموجودة في الاعلى وان كلاً المقارين هما  
نفس الشيء يعني الان بعد انا قمنا بعملية التغير فأن كل من المقارين الذين  
في الاعلى والذي في الاسفل كلاهما نفس المقدار هذه الحالة كانت مع  
المصفوفة الان سوف ننقل الى الهاش  
\${\$href}->{red}

هذه كانت الطريقة المتبعة في الطريقة الاولى الان وبعد التغير سوف تتحول  
الى  
\$href->{red}

وايضا الان كل من المقدرين هما انفس الشيء و ان اختلفت طرق الكاتبة  
الان سوف نأخذ مثال سريع لترى كل هناك هو مفيد

```
@spawn = ([1, 2, 3],  
           [4, 5, 6],  
           [7, 8, 9]  
           );
```

هي عبارة عن مصفوفة مكونة من ثلاث عناصر وكل واحد هو @spawn  
الى مصفوفة reference

التي تتكون من 3 عناصر وهي references هو واحد من هذه ال \$spawn[1]  
تشير الى مصفوفة وهذه المصفوفة مكونة من 3 عناصر وطبقا للطريقة  
الثانية فانه من الممكن ان نحصل على عنصر من المصفوفة عند طباعة مايلي  
\$spawn[1]->1;

عند طباعة هذه الجملة سوف نحصل على المقدار 5 لانه ثاني عنصر من  
المصفوفة 1 لذا نحن الان لدينا شيء يشبه المصفوفات البعد الثنائي  
ان عملية التدوين لاتزال صعبة ومزعجة الى حد الان لذا يمكن القول انه هناك  
اختصار واحد بعدوه هو  
اذا كانت لديك هذه المصفوفة

```
@spawn = ([1, 2, 3],  
           [4, 5, 6],  
           [7, 8, 9]  
           );
```

بدلا من ان تقوم بطباعة هذه الجملة من اجل الحصول على رقم 5 مثلا

```
print $spawn[1]->[1];
```

من الممكن ان نقوم بعمل مايلي من اجل الحصول على الناتج ولكن بخطة اقصر  
وهي

```
print $spawn[1][1];
```

وعند تنفيذ هذه الجملة سوف نحصل على نفس الناتج في كلتا الحالتين ويمكنك ان تتأكد بنفسك

الآن وبعد ان انتهى هذا الدرس البسيط في ال  
اتمنى ان اكون قد وفقت في تقديم كل ما هو مفيد وجيد REFERENCES

GR33Tz To :  
HACKERS PAL (Th3 Und3r W0rld M45t3r )  
GaCkEr (H0P1355 C453)

And SpEcial Gr33tz to

[www.soqor.net](http://www.soqor.net)

[www.securitygurus.net](http://www.securitygurus.net)

ThIs SiMpLe TuTorIal is wRot3 By

*M\_sPaWn*

*Mail Me:- mahmoud najafy@hotmail.com*

THX A LOT 4 READING MY SIMPLE TUTORIAL

تم بحمد الله