

تعلم لغة DART

ملخص مبسط جدا لفهم أساسيات لغة DART



Dart

جدول المحتويات

<i>Syntax</i>	3
<i>Datatype & variable</i>	4
<i>Arithmetic Operators</i>	10
<i>quality and Relational Operators</i>	12
<i>Type test Operators And Assignment Operators</i>	15
<i>Logical Operators</i>	19
<i>comment</i>	22
<i>if else statement</i>	23
<i>Switch case</i>	29
<i>For loop</i>	31
<i>While Loop</i>	36
<i>Do While Loop</i>	37
<i>break</i>	38
<i>continue</i>	40
<i>Number</i>	42
<i>String Method</i>	50
<i>List</i>	57
<i>MAP</i>	70
<i>Dyanmic Variables</i>	74
<i>final and const</i>	75
<i>Set</i>	76
<i>Convert between</i>	77
<i>List Map Set</i>	77
<i>Function</i>	79
<i>Function types</i>	83
<i>Function scope</i>	86
<i>import</i>	89
المسارات	90
<i>Runes</i>	92

Assert	94
Max And Min	95
DataType	96
Iterator And Iterable	106
Map Method	108
Try And Catch	110
shorthand if	111
OPP	112
Reset Variable	114
Setter And Getter	117
Static	119
Cascade operator	121
Constructor	122
inheritance	123
Override	124
Mult-level inheritate	125
Super Class	127
Abstract class	128
implements class	129
Collection	131
Enum	132
Regular expressions	134
Private	137
Mixin	139
Null Safety	141



Syntax

هو مجموعة القواعد لكتابة برنامج ما

الامر مشابه تماما للغات التي يتحدث بها البشر مثل الإنكليزية والعربية على سبيل المثال اذا اردنا قول القاء التحية على احد ما نقول بلغة العربية مرحبا وبلغة الإنكليزية Hello الكلمتين تختلفان بلفظ وطريقة الكتابة ولكن الكلمتين تؤديان نفس المعنى وهو الترحيب.

كذلك الامر بالنسبة للغات البرمجة يوجد العديد لغات البرمجة

The screenshot shows a code editor with the following Dart code:

```
1 void main () {  
2     print("ahmed");  
3 }  
4
```

A red arrow points from the word "print" in the code to a tooltip labeled "أمر الطباعة" (Print command). Below the code, the terminal output shows:

```
..  
ahmed  
Exited
```

A red arrow points from the word "ahmed" in the terminal output to a tooltip labeled "خرج البرنامج" (Program exited). A search bar at the bottom is labeled "Filter (e.g. text, !exclude)".

في هذا المثال نقوم بطباعة كلمة ahmed بلغة ال DART

وبهذه الصورة سوف نقوم بطباعة كلمة احمد بلغة Php

وخرج البرنامج سيكون نفسه مع اختلاف طريقة الكتابة واختلاف لغة البرمجة أيضا.

Datatype & variable

هي أسماء تخزن في ذاكرة الحاسوب وتأخذ قيم ويوجد عدة أنواع لل variable

أول نوع من ال variable هو ال String

من الضروري ان نقوم بوضع النص داخل المتغير String ضمن

(دبل كوتشن " ") او (سنجل كوتشن ' ')

```
String name = "wael";
print(name)
```

نقوم ب تخزين نص في داخله text تستطيع اختيار اسم المتغير الذي تريده ولكن
بشرط ان لا يحوي الاسم على رموز او ان لا يبدأ اسم المتغير برقم .

```
Run | Debug
void main () {
  String name = "wael" ;
  print("ahmed") ;
}
```

قيمة المتغير
اسم المتغير
نوع المتغير

لا نستطيع تخزين قيمة من نوع عدد في المتغير من نوع String

```
main.dart main
Run | Debug
void main () {
  String name =2 ;
  print(
```

A value of type 'int' can't be assigned to a variable
of type 'String'.
Try changing the type of the variable, or casting the
right-hand type to 'String'. dart(invalid_assignment)

Peek Problem (Alt+F8) Quick Fix... (Ctrl+.)

اذا قمنا كتابة المتغير داخل امر الطباعة سوف يقوم البرنامج بطباعة قيمة المتغير من دون أي مشاكل كما في الصورة بأسفل :

A screenshot of Visual Studio Code's terminal window. The code in the editor is:

```
Run | Debug
1 void main () {
2     String name = "wael";
3     print(name);
4 }
5
6
```

The terminal output shows:

```
wael
Exited
```

Annotations in red:

- An arrow points from the word "print" in the code to the word "wael" in the terminal output.
- The text "(امر الطباعة)" is written in red near the terminal output.
- Two arrows point from the terminal output to the word "wael" in the code.
- The text "خرج البرنامج" is written in red near the terminal output.

اذا لم نقوم بتعريف متغير لن يعمل البرنامج وسوف يظهر لنا خطأ

A screenshot of Visual Studio Code's terminal window. The code in the editor is:

```
Run | Debug
1 void main () {
2     print(name);
3 }
4
5
6
```

A tooltip appears over the word "name" in line 2, containing the text:

فقط بطباعة المتغير من دون تعريفه
واعطاء قيمة

The terminal output shows:

```
Build errors exist in your project.
Debug Anyway Show Errors Cancel
```

Annotations in red:

- An arrow points from the word "name" in the code to the tooltip.
- An arrow points from the tooltip to the word "name" in the code.
- The text "الخطأ الناتج" is written in red near the terminal output.

اذا اردنا وضع المتغير ضمن " " سوف يتم تعرف عليه على انه نص
واردنا لجهاز الحاسوب ان يتعرف عليه على انه متغير وهو ضمن الدوبل كوتتشين
فقط نقوم بوضع علامة او رمز دولار ساين \$ قبل اسم المتغير

A screenshot of Visual Studio Code's terminal window. The code in the editor is:

```
Run | Debug
1 void main () {
2     String name = "basel";
3     print("$name");
4 }
5
6
```

The terminal output shows:

```
basel
Exited
```

Annotations in red:

- An arrow points from the dollar sign (\$) in the code to the word "basel" in the terminal output.
- The text "تم التعرف عليه على انه متغير وليس نص" is written in red near the terminal output.
- Two arrows point from the terminal output to the word "basel" in the code.
- The text "خرج البرنامج" is written in red near the terminal output.

اذا قمنا بطباعة اسم المتغير من دون وضع علامة \$ قبل اسم المتغير وف يراه الحاسوب على انه نص عادي وليس متغير يحمل داخله قيمة وسوف يقوم بطباعة اسم المتغير فقط كما هو واضح في الصورة فب الأسفل :

```

Run | Debug
1 void main () {
2   String name = "basel";
3   print(["name"]);
4 }
5
6
...
name
Exited

```

لن يتم طباعة قيمة المتغير لأنه غير مسمى

ليس متغير لأنه ضمن دوبل كوتشن

خرج البرنامج

يوجد في لغة ال Dart مجموعة من الكلمات الأساسية التي لا يمكن ان نقوم باستخدامها في برنامجنا وهي عبارة عن كلمات معرفة مسبقا في اللغة نستطيع الوصول اليها ومعرفتها من خلال صندوق إرشادات المدرب .

```

Run | Debug
1 void main () {
2   String Async = "basel";
3   print("a String Async");
4
5
6
...

```

كلمة معرفة مسبقا من قبل DART

The value of the local variable 'Async' isn't used.
Try removing the variable, or using dart(unused_local_variable)

Peek Problem (Alt+F8) Quick Fix... (Ctrl+.)

basel
Exited

يوجد نوع ثانٍ من المتغيرات وهو INT

نستطيع من خلاله تخزين متغيرات الأرقام الصحيح فقط لا غير مثل (٩٠، ١، ٩٠) أي ان الأرقام التي لا تحوي على فاصلة عشرية .

```

1 void main () {
2     String name = "basel";
3     int age = 12;
4     print(name);
5 }
6

```

لا يجب وضع الرقم ضمن دوبل كوتشن وهو في متغير من نوع INT لأن سيعتبره
نص من نوع .String

```

1 void main () {
2     String name = "basel";
3     int age = "12";
4     print(name);
5 }
6

```

```

1 void main () {
2     String name = "basel";
3     int age = 34;
4     print(age);
5 }
6

```

لا نستطيع تعيين متغير أكثر من مرة في البرنامج يتم تعيينه مرة واحدة فقط
يمكننا إعادة تعيين ((قيمة المتغيرات))



Run | Debug

```
1 void main () {  
2     String name = "basel";  
3     int age = 34;  
4     age = 20;  
5     print(age);  
6 }
```

Filter (e.g. text, exclude)

20 Exited امر الطباعة

تعيين قيمة المتغير اول مرة
اعادة تعيين قيمة المتغير
واعطائه قيمة جديدة وهي 20

لأن محرر الأكواد في البرمجة يقوم بتنفيذ السطور البرمجية سطر سطر.
لا نستطيع طباعة قيمة أكثر من متغير في نفس الوقت من دون تعريف لحاسوب
والأسف سوف يعتبره متغير واحد كما في الصورة في الأسفل :

```
1 void main () {  
2     String name = "mohmmad";  
3     String lastname = "nour";  
4     print(name lastname);  
5 }
```

المتغير الاول المتغير الثاني

بوضع علامة دوبل كوتشن قبل كل متغير نستطيع تعريف الحاسوب اننا نقوم
طباعة متغيرين في نفس الوقت .

```
1 void main () {  
2     String name = "mohmmad";  
3     String lastname = "nour";  
4     print("$name $lastname");  
5 }
```

المتغير الاول المتغير الثاني

PROBLEMS OUTPUT DEBUG CONSOLE ... Filter (e.g. text, exclude)

mohmmad nour
Exited ← الخرج

نوع الثالث من المتغيرات وهو double
نستطيع من خلاله تخزين الأرقام العشرية .

نستطيع أيضا تخزين الأرقام الصحيحة ضمن متغير `double` ولكن لا نستطيع تخزين الأرقام العشرية ضمن المتغير `.int`.

A screenshot of a terminal window. The code in the editor is:

```
1 print("Hello")
2
3
4
5 double w = 2.3 ;
6 print("$w") ;
7
8
9
```

The terminal output shows:

```
2.3
Exited
```

Annotations in Arabic:

- A red arrow points from the text "تعريف المتغير" (Define variable) to the line `double w = 2.3 ;`.
- A red arrow points from the text "خروج البرنامج" (Program exit) to the word `Exited`.

نوع الرابع من المتغيرات وهو `bool` يرجع قيمة او `false` او `true`

Arithmetic Operators

نستطيع اجراء العمليات لحسابية على المتغيرات مثل ما هو واضح في الصورة في الأسفل

```
5 int a = 4 ;  
6 int b = 6 ;  
7 int c = a + b ;  
8 print(c) ;  
9  
10  
11 }
```

المتغير الاول
المتغير الثاني
العملية الحسابية
من الطباعة
الخرج

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Filter (e.g. text)

10 Exited

كما هو معروف في الرياضيات يوجد العديد من العمليات مثل الطرح والجمع والقسمة والضرب والتي نستطيع ان نستخدمها في برنامجنا الخاص لاجراء وظيفة معينة .

فقط في القسمة يجب ان نستخدم متغير من نوع double لأن القسمة ينتج عنها اعداد عشرية كما هو موضح في الصورة في الأسفل .

```
4  
5 int a = 12 ;  
6 int b = 6 ;  
7 double c = a / b ;  
8 print(c) ;  
9  
10  
11 }
```

من نوع double

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

2.0
Exited

نستطيع ان نقوم بتحويل خرج ناتج القسمة لعدد صحيح من خلال وضع علامة ~ قبل علامة القسمة بلغة الإنكليزية كما موضح في الصورة بلاسفل :

فمنا بتعريف المتغير من نوع عدد صحيح int

Shift + ⌘

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Filter (e.g. text, !exclude)

2 Exited خرج البرنامج

```
3
4 int a = 12 ;
5 int b = 6 ;
6 int c = a ~/ b ;
7 print(c) ;
8
9
10
11 }
```

ومن الإشارات المعروفة في الرياضيات وهي إشارة باقي القسمة %

كما هو موضح في المثال في الصورة بالأسفل :

(إشارة باقي القسمة)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Filter()

5 Exited خرج البرنامج

```
4
5 int a = 17 ;
6 int b = 6 ;
7 int c = a % b ;
8 print(c) ;
9
10
11 }
```

وأيضاً من العمليات كما هو موضح في الصورة في الأسفل :

تقوم بإضافة 1 الى خرج البرنامج ويوجد عكساً وهي -- ت تقوم بطرح 1 من خرج البرنامج .

تقوم بإضافة 1 الى المتغير

هي نفسها

c=c+1

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

11 Exited خرج البرنامج

```
2
3 int c = 10 ;
4 c++ ; |
5
6
7 print(c) ;
8
9
10
11 }
```

quality and Relational Operators

عمليات المساواة والمقارنة تقوم بأرجاع قيمة `false` او `true`

< اكبر

< اصغر

= يساوي

!= لا يساوي



A screenshot of a code editor showing a snippet of C code. The code defines two integer variables `a` and `b`, both initialized to 10. It then prints the result of the expression `a == b`. A red arrow points to the equality operator `==` in the code, with the label "إشارة المساواة" (equality sign) written in red. Another red arrow points to the output tab in the interface, which shows the word "true" in blue, with the label "خرج البرنامج" (program exited) written in red.

```
2
3 int a = 10 ;
4 int b = 10 ;
5
6 print(a == b) ; |
7
8 }
```

في المثال الموضح أعلاه اذا كانت قيمة المتغير `a` تساوي قيمة المتغير `b` سوف يقوم بأرجاع قيمة `true`.

والا اذا كانت قيمة المتغير `a` لا تساوي قيمة المتغير `b` سوف يقوم بأرجاع قيمة `false` كما هو واضح في الصورة في الاسفل.

```

1 void main(){
2
3 int a = 10 ;
4 int b = 20 ;
5
6 print(a == b) ;
7
8 }
9

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
 false Exited

اذا كانت قيمة المتغير a اكبر من قيمة المتغير b سوف يقوم بارجاع قيمة .true

الا اذا كانت قيمة المتغير a اصغر من قيمة المتغير b سوف يقوم بارجاع قيمة
false.

و اذا كانت قيمة المتغير a تساوي قيمة المتغير b سوف يقوم بارجاع قيمة
.false

كما هو موضح في الصورة في الأسفل :

Run | Debug
 1 void main(){
2
3 int a = 30 ;
4 int b = 20 ;
5
6 print(a > b) ;
7
8 }
9
 عملية مقارنة
 اشارة الاكبر

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Filter (e.g.
 true Exited

وتوجد حالات اخري للمقارنة على سبيل المثال اذا كانت قيمة المتغير a اكبر او
تساوي قيمة المتغير b سوف يقوم بارجاع قيمة .true

كما هو موضح في المثال في الصورة بلاسفل :

```
2
3 int a = 30 ;
4 int b = 30 ;
5
6 print(a >= b) ;
7
8 }
9
```

(إشارة المقارنة أكبر أو تساوي)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Filter (e.)

true
Exited خرج البرنامج

آخر اشارة لدينا في عمليات المقارنة وهي إشارة لا يساوي !=

إذا كانت قيمة المتغير a لا تساوي قيمة المتغير b سوف يقوم بإرجاع قيمة

.true

إذا كانت قيمة المتغير a تساوي قيمة المتغير b سوف يقوم بإرجاع قيمة .false

كما هو موضح في الصورة في الأسفل :

```
Run | Debug
1 void main(){
2
3 int a = 20 ;
4 int b = 30 ;
5
6 print([a != b]) ;
7
8 }
9
```

إشارة لا تساوي

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Filter (e.g. text, !exclude)

true
Exited خرج البرنامج

Type test Operators And Assignment Operators

عمليات الاختبار تستخدم هذه العمليات من اجل معرفة نوع المتغير وترجع قيمة .false او true اما

is هل يساوي

is! هل لا يساوي

كما هو موضح في الصورة في الأسفل يوجد عملية اختبار على المتغير `a` لأننا نقوم بتوجيه سؤال للحاسوب مضمونه هل المتغير `a` من نوع `int` كما هو موضح في المثال في الأسفل يكون الخرج `true` لأن المتغير من نوع `int` لأن المتغير قيمته عدد صحيح هو العدد ١٠ كما هو واضح في الصورة في الأسفل :

The screenshot shows a code editor with the following code:

```
1 int a = 10 ;
2
3 print(a is int ) ;
4
5 }
```

An arrow points from the text "عملية الاختبار" (Type test) to the line `print(a is int);`. Another arrow points from the text "خرج البرنامج" (Program output) to the word `true` in the terminal output.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

true
Exited

ولو ان المتغير قيمته ليست من نوع `int` سوف يقوم البرنامج بأرجاع قيمة `false` كما هو موضح في الصورة في الأسفل :

A screenshot of a Java IDE interface. At the top, there is a code editor with the following Java code:

```
String a = "w";
print(a is int);
```

The code editor has line numbers 1 through 8. Below the code editor is a navigation bar with tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL. The DEBUG CONSOLE tab is highlighted with a red arrow pointing to it. The terminal window shows the output:

false
Exited

كما يوجد مثال على نوع الثاني من عمليات الاختبار الا وهو `is`

A screenshot of a Java IDE interface. At the top, there is a code editor with the following Java code:

```
String a = "w";
print(a is int);
```

The code editor has line numbers 1 through 8. Below the code editor is a navigation bar with tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL. The DEBUG CONSOLE tab is highlighted with a red arrow pointing to it. The terminal window shows the output:

true
Exited

في المثال الموضح أعلاه نقوم بتوجيه سؤال للحاسوب مضمونه هل المتغير `a` ليس من نوع `int` كما هو موضح في المثال يكون الخرج `true` لأن المتغير ليس من نوع `int` لأن المتغير قيمته `.text`

في المثال الموضح أدناه لقد قمنا بإعادة تعيين قيمة المتغير `a` ولكن عند تشغيل البرنامج نتفاجئ ان قيمة المتغير `a` لم يتم اعدة تعينها وفي الدروس السابقة تكلمنا ان الحاسوب يقوم بتنفيذ الاكواد سطر سطر فما هو سبب عدم إعادة التعيين

```

2
3 int a = 10 ;
4 a ??= 20 ;
5 print(a) ;
6
7
8 }
9

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Filter (e.g. te)

10
Exited

(إعادة تعيين قيمة المتغير)

خرج البرنامج بعد التشغيل

وسبب في هذا الامر وهو إشارة مساواة `=??=` في حال كان المتغير لا يحتوي بداخله على أي قيمة وهذه الإشارة تعني ان اذا كان المتغير ليس له قيمة قم بأسناد قيمة له والقيمة التي سوف يقوم بأسنادها هي نفسها القيمة التي قمنا بادخالها للمتغير وهي ٢٠ كما هو موضح في الصورة في الأسفل :

```

2
3 int a ;
4 a ??= 20 ;
5 print(a) ;
6
7
8 }
9

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Filter (e.g.

20
Exited

المتغير لا يحتوي بداخله على اي قيمة)

خرج البرنامج

Null تعني ان المتغير معرف ولكن لا يحوي بداخله أي قيمة

```
2  
3     int a ;  
4  
5     print(a);  
6  
7 }  
8
```

متغير فارغ لا يحوي بداخله اي قيمة

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Filter (e.g. text, exclude)

null
Exited

خروج البرنامج بعد تشغيله

ومن الإشارات المهمة $a +$

اسم المتغير ثم وضع إشارة العملية الحسابية ثم قيمة مراد اضافتها الى المتغير
 تستطيع استخدام جميع العمليات الحسابية من ضرب وقسمة وطرح وجمع
 ولا ننسا ان نقوم بتغيير نوع المتغير في حال الفسمة الى double لان ناتج
 القسمة ينتج عنه اعداد عشرية .

```
3     int a = 1 ;  
4     a += 2;  
5  
6     print(a);  
7  
8 }
```

هي نفسها
 $a = a + 2$

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Filter

3
Exited

خروج البرنامج

Logical Operators

عمليات المقارنة وتقوم أيضا بارجاع قيمة من نوع bool أي ان الخرج سيكون اما .false او true

&& And

|| Or

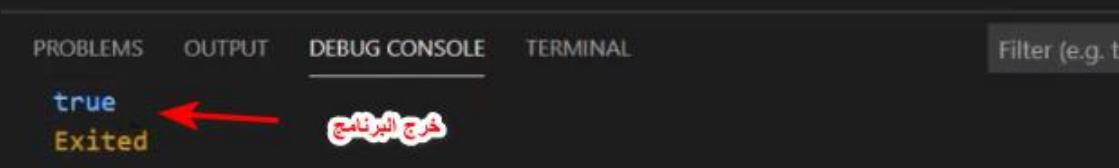
! Not

اول إشارة من عمليات المقارنة الا وهي ال AND (و)

نقوم بلضغط على زر SHIFT + 7

كما هو موضح في المثال في الأسفل فأن الشرط محقق بحيث ان المتغير a و متغير b اكبر من ١٠ فأن خرج البرنامج true .

```
1
2
3 int a = 20 ;
4 int b = 20 ;
5
6 print( a > 10 && b > 10 ) ; ← الشرط
7
8
9 } ← إشارة AND
10
```



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Filter (e.g. t)
true ←
Exited ← خرج البرنامج
```

عند استخدام AND في حال اختلال احد الشرطين سيرجع البرنامج قيمة False كما هو موضح في الصورة في الأسفل :

```

3   int a = 20 ;
4   int b = 5 ; ← الشرط غير متحقق
5
6   print( a > 10 && b > 10) ;
7
8
9 }
10

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Filter (e.g.)

false ← خرج البرنامج
Exited

الإشارة الثاني من علميات المقارنة الا وهي OR (او)

نقوم بـلضغط على زر \Shift + \

حتى يرجع البرنامج قيمة true يوجد عدة حالات اما ان يكون طرفين الشرط متحقق او احد الطرفين متحقق كما في المثال الموضح أدناه :

```

2
3   int a = 20 ; ← متغير الأول
4   int b = 5; ← متغير الثاني
5
6   print( a > 10 || b > 10) ; ← شرط
7
8
9 }
10

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Filter (e.g.)

true ← خرج البرنامج
Exited

حتى يرجع البرنامج قيمة `false` يجب اختلال طرفيين أي ان المتغير `a` و `b` قيمتهم ليست اكبر من ١٠ كما هو واضح في المثال في الأسفل :

```

3 int a = 5; ← المتغير الأول
4 int b = 5; ← المتغير الثاني
5
6 print( a > 10 || b > 10 );
7
8 } ← الشرط غير متحقق
9
10

```

The screenshot shows the VS Code interface with the DEBUG CONSOLE tab selected. The output window displays the text "false" and "Exited". A red arrow points from the word "false" to the text "خرج البرنامج بعد تشغيل" (Program exited) at the bottom of the output window.

الإشارة الثالثة وهي `! NOT!` وتقوم بعكس الحالة

نقوم بلضغط على زر `Shift + 1`

لنوضح المثال الذي في الأسفل لدينا متغيرين `a` و `b` وان قيمتهم اكبر من ١٠ والشرط متحقق ويقوم بأرجاع قيمة `true` في البرنامج ولكن لوجود إشارة `not` تقوم بعكس الحالة من `true` الى `false` وفي حال ارجع البرنامج قيمة `not false` تعود بعكس الحالة وطباعة `true` في خرج البرنامج .

```

3 int a = 20;
4 int b = 20;
5
6 print( !( a > 10 || b > 10 ) );
7
8 }
9
10

```

A red arrow points from the exclamation mark in the code to the text "إشارة NOT" (NOT indicator) below the code editor.

The screenshot shows the VS Code interface with the DEBUG CONSOLE tab selected. The output window displays the text "false" and "Exited". A red arrow points from the word "false" to the text "خرج البرنامج" (Program exited) at the bottom of the output window.

comment

التعليق لا يستطيع محرر الاكواود او الحاسوب قراءة او تنفيذ التعليق ضمن البرنامج.

يستخدم التعليق من اجل تسهيل قراءة الاكواود او مثل وضع عناوين على مجموع من الاسطرا البرمجية للتوضيح او شرح مهمة هذه الاكواود في البرنامج .

توجد التعليقات في كل لغات البرمجة .

تستطيع كتابة التعليق اما بضغط على زر / Ctrl + /

أو بكتابة // قبل الكود

او بكتابة /* comment */

كما موضح في المثال ادناه :

```
6 // ← (الشكل الاول للتعليق)
7 /* dsf sf ndsf sdf sf */
8 }
9 }
10 }
```

if else statement

اذا كان الشرط داخل if محقق true سوف يقوم البرنامج بتنفيذ الاكواد التي بداخل if.

```
8
9  /*
10
11  if ( condition = true ) {
12  | // code
13  }
14
15 */
16
17
18
```

The code editor shows the following snippet:

```
/*
if ( condition = true ) {
| // code
}
*/
```

Annotations in red:

- A red arrow points to the condition line with the text "اذا كان الشرط متحقق" (If the condition is true).
- A red arrow points to the code block with the text "سوف يقوم البرنامج بتنفيذ الكود" (The program will execute the code).



اذا كان الشرط داخل if غير متحقق false لن يتم تنفيذ ما داخل اقواس الدالة if.

```
/*
if ( condition = false | ) {
| // code
}
*/
```

The code editor shows the following snippet:

```
/*
if ( condition = false | ) {
| // code
}
*/
```

Annotations in red:

- A red arrow points to the condition line with the text "الشرط غير متحقق" (The condition is not true).
- A red arrow points to the code block with the text "لن يقوم بتنفيذ الكود" (The code will not be executed).

في المثال الذي لدينا في الأسفل ثالث متغيرات لهم قيم مختلفة من نوع int وقد قمنا بلاستعانة بدالة if الشرطية بتنفيذ الكود في حال تحقق الشرط داخل if سوف يقوم الحاسوب بتنفيذ الاكواد التي بداخل if كما نلاحظ ان price وهي قيمة المال التي معنا تساوي ٢٠٠٠ وان علينا الاختيار بين هذه الأجهزة الثلاث فنقوم بكتابة الشرط اذا كان price == s10 قم بطباعة (" سوف اشتري s10

("

s10 == 2000 price == 2000 وقيمة

كما هو موضع في المثال في الأسفل :

```

3 int iphone = 1000 ;
4 int s10     = 2000 ;
5 int s20     = 3000 ;
6
7 int price = 2000 ;
8
9 if (price == s10) {
10   print("سوق اشتري s10") ;
11 }
12
13

```

الشرط متحقق

سوق يقوم بتنفيذ الكود

Exit

خرج البرنامج بعد التشغيل

في حال كان الشرط غير متحقق false لن يقوم بتنفيذ الكود داخل if لأن price لا تساوي قيمة S20 كما هو واضح في الصورة في الأسفل :

```

3 int iphone = 1000 ;
4 int s10     = 2000 ;
5 int s20     = 3000 ;
6
7 int price = 2000 ;
8
9 if [price == s20] {
10   print("سوق اشتري s10") ;
11 }
12
13

```

الشرط غير متحقق

لم يقوم بتنفيذ الكود

Exit

لدينا مثال اخر ...

```
int iphone = 1000 ;
int s10 = 2000 ;
int s20 = 3000 ;
int price = 2000 ;
if (price == iphone) {
    print("سوف اشتري ايفون")
}

if (price == s10) {
    print("سوف اشتري اس 10")
}
if (price == s20) {
    print("سوف اشتري اس 20")
}
```

الدالة الشرطية الأولى
غير محققة

الدالة الشرطية الثانية
محققة

الدالة الشرطية الثالثة
غير محققة

Items OUTPUT DEBUG CONSOLE TERMINAL Filter (e.g.

في هذا المثال لدينا ثلاثة دالات شرطية سيقوم الحاسوب بفحص كل دالة في حال كان متحقق سيعمل بتنفيذ ما داخلها ونحران دالة الثانية هي متحققة كما هو ظاهر في خرج البرنامج في الأسفل :

```
11 if [price == s10] {
12     print("10")
13 }
14 if (price == s20) {
15     print("20")
16 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

سوف اشتري اس 10
Exited

خروج البرنامج بعد التشغل

أيضا مثال اخر لدينا

نلاحظ ان الشرط الأول وثاني محقق لان قيمة price اكبر من قيمة iphone و قيمة S10 ونلاحظ ان قيمة S20 في الشرط الثالث غير محققة لانها لاتساوي قيمة Price بهذا سوف يقوم الحاسوب بتنفيذ الشرط الأول والثاني فقط كما هو واضح في المثال في الأسفل :

```
1 int iphone = 1000;
2 int s10 = 2000;
3 int s20 = 3000;
4 int price = 2500;
5 if (price > iphone) { ← الشرط الأول متحقق
6     print("انا يمكنني شراء الايفون");
7 }
8
9 if (price > s10) { ← الشرط الثاني متحقق
10    print("انا يمكنني شراء الايم 10");
11 }
12 if [price > s20] { ← الشرط الثالث غير متحقق
13     print("انا يمكنني شراء s20");
14 }
15 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL ← خرج البرنامج بعد التشغيل
انا يمكنني شراء الايفون
انا يمكنني شراء الايم 10
Exited

لدينا أيضاً مثال عن `else` وتعني و (الا في حال عدم تحقق الشرط وارجاع قيمة `false` . يقوم الحاسوب بتنفيذ ما داخل دالة `false`

الآن لنقوم بتوضيح المثال الذي بلاسف نلاحظ ان قيمة `price` هي اقل من قيمة `iphone` ففي الدالة `if` سيقوم البرنامج بأرجاع قيمة `false` لأنها غير محققة ولأنها غير محققة ولو وجود ال `else` سوف يقوم البرنامج بتنفيذ ما بداخليها كما هو واضح في الأسفل .

٢١

```

2 int iphone = 1000;
3 int s10 = 2000;
4 int s20 = 3000;
5 int price = 500;
6 if (price > iphone) { // true or false
7     print("انا يمكنني شراء الايفون"); // true
8 }else {
9     print("انا لا يمكنني شراء الايفون"); // false
10 }
11
12 }
13 }
14 }
```

الشرط
فـي حال تحقق الشرط
وارجاع قيمة true
فـي حال عدم تتحقق الشرط
وارجاع قيمة false

في حال عدم تتحقق الشرط يتم تنفيذ الكود

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL

انا لا يمكنني شراء الايفون
Exited

خرج البرنامج

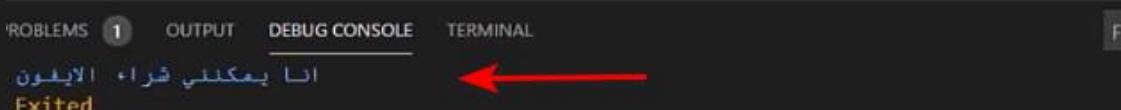
٢٢

٢٧

نلاحظ في المثال في الأسفل ان قيمة price هي اكبر من قيمة جميع المتغيرات الموجودة في البرنامج ومع هذا فإن الشرط الثاني محقق ومع ذلك لم يتم تنفيذه وطباعته لماذا ..؟

كما ذكرنا سابقا ان الدالة else لا تعمل الا في حال عدم تحقق الشرط الأول وارجاع قيمة false ولكن هنا الشرط الأول تحقق وقام بأرجاع قيمة true هنا سيقوم البرنامج بتنفيذ الشرط الأول فقط دون النظر الى الشرط الثاني بمعنى ان تقوم بإعطاء عدة احتمالات للحاسوب في حال تحقق الاحتمال الأول فإنه لن يقوم بلنظر الى باقي الاحتمالات لان الاحتمال الأول قد تحقق فعلا كما هو واضح في الصورة في الأسفل .

```
1 int iphone = 1000;
2 int s10 = 2000;
3 int s20 = 3000;
4 int price = 3500;
5
6 if (price > iphone) {
7     // true or false
8     print("انا يمكنني شراء الايفون"); // true ← الشرط الأول متحقق
9 } else if (price > s20) {
10    print("انا يمكنني شراء الايفون 20"); // ← الشرط الثاني متحقق
11 } else {
12    print("انا لا يمكنني شراء الايفون"); // false
13 }
14 []
15
```



ملاحظة : يمكنك وضع اكثر من شرط ضمن دالة If الشرطية

Switch case

تؤدي دالة Switch case نفس عمل الدالة if else ولكن فقط تختلف بطريقة الكتابة مثل ما هو واضح في المثال في الأسفل :

```
3 int price = 1000 ;  
4  
5 switch (price) {  
6     case 1000 : {  
7         تقوم بكتابه الكود بين فوسين  
8     }  
9     break ;  
10 }  
11  
12 }  
13  
14 }  
15  
16 }  
17  
18 }
```

الحالة الأولى

أخرج من الحالة الأولى

نجد في المثال هذا ان قيمة price تساوي ١٠٠٠ وان قيمة الحالة الأولى في البرنامج في أيضا ١٠٠٠ هذا يعني ان البرنامج سيقوم بتنفيذ الحالة الأولى .

```
3 int price = 1000 ;  
4  
5 switch (price) {  
6     case 1000 : {  
7         print("انا معى 1000 دولار")  
8     }  
9     break ;  
10  
11     case 2000 : {  
12         print("انا معى 2000 دولار")  
13     }  
14  
15 }  
16  
17  
18 }
```

قيمة المتغير

الحالة الأولى

الحالة الثانية

انا معى 1000 دولار
Exited

خروج البرنامج عند التشغيل

في نفس المثال لو قمنا بـأعطاء قيمة للمتغير `price = 3000` في هذه الحالة لن تتحقق الحالة الأولى وثانية وستقوم بأرجاع قيمة `false` ومن ثم سيقوم البرنامج بتنفيذ الأكواد بداخل `default` لأن `default` تعمل فقط في حالة عدم تحقق الحالات وراجعاً قيمة `.false`.

The screenshot shows a code editor with the following Python code:

```
21     default : { ←
22         print(" أنا لا اعلم المبلغ الموجود معك") ←
23     }
24
25
26     break ;
27
28
29
30 }
31
32
33
34 }
```

A red arrow points from the text "في حالة عدم تحقق حالة الأولى وثانية في البرنامج" to the `default` label at line 21. Another red arrow points from the text "انا لا اعلم المبلغ الموجود معك" to the `print` statement at line 23. A third red arrow points from the text "خرج البرنامج" to the word `Exited` in the terminal output.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

انا لا اعلم المبلغ الموجود معك
Exited ← خرج البرنامج

For loop

الحلقات التكرارية : لنفترض انك تريد طباعة الأعوام من من ١٩٠٠ الى ٢٠٢٠
بتأكيد انت لا تنوی ان تقوم بتكرار هذه العملية وتكرار امر الطباعة ١٢٠ مرة حتى
تصل الى هدفك من هنا تستطيع استخدام الحلقات التكرارية من خلالها تستطيع
توفير الوقت والجهد وأيضا سيكون أداء برنامجك افضل .

```
Run | Debug
1 void main(){
2
3     print(1900) ;
4     print(1901) ;
5     print(1902) ;
6     print(1903) ;
7
8 }
```

شكل طريقة كتابة الحلقة التكرارية : FOR :

```
Run | Debug
1 void main(){
2
3     // for (var = value ; condition ; operator ) { code }
4
5
6 }
```

الحلقة التكرارية ↑
تعريف متغير واعطائه قيمة ↑
 الشرط ↑
العمليات ↑
ال코드 المراد تنفيذه ↑

في هذا المثال نريد طباعة الأعوام من ١٩٠٠ إلى ٢٠٢٠

يجب كخطوة أولى تعريف المتغير ومن ثم كتابة حلقة FOR ومن ثم إعطاء قيمة ابتدائية للمتغير وتعتبر نقطة بداية لبدأ العد ثم لا ننسا كتابة الفاصلة المنقوطة ثم كتابة الشرط الا وهو ان تكون year اصغر او تساوي ٢٠٢٠ ومن ثم نقوم بادخال عداد الزيادة وهو بمقدار ١ فقط .

```
4
5 int year ; ← تعريف المتغير
6
7 for [ year = 1900 ; year <= 2020 ; year++ ] {
8
9     print(year) ;
10 }
11
12 } ← مقدار زيادة قيمة المتغير بمقدار 1 فقط
13
14
15 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

2014
2015
2016
2017
2018
2019
2020
Exited

خروج البرنامج بعد التشغيل

الية عمل الحلقة `for loop` كما هو واضح في الصورة في الأسفل خطوات ومراحل عمل الحلقة في البرنامج وسوف أقوم بشرح خطوة خطوة .

١ - يقوم بجلب القيمة الابتدائية للمتغير .

٢ - يقوم بإضافة مقدار الزيادة على قيمة المتغير ومقدار الزيادة هنا هو واحد كما شرحنا في الدروس السابقة ان كتابة اسم المتغير ثم وضع علامة `++` بعده معناها إضافة مقدارها واحد على قيمة المتغير .

٣- تحقق من الشرط : بعد إضافة مقدار الزيادة واحد الى المتغير يجب تأكيد من تحقق الشرط هل `1900 <= year <= 2020` هنا نجد ان الشرط متحقق وقام بإرجاع قيمة `true` ينتقل الى الخطوة الرابعة .

٤ - تنفيذ ما داخل الحلقة التكرارية

٥ - تنفيذ امر الطباعة

وإعادة العملية تكرار حتى يصل لمرحلة عدم تحقق الشرط وأرجاع قيمة `false` والخروج من الحلقة التكرارية بشكل نهائي.

```
5 int year ;
6
7 for [year = 1900 ; year <= 2020 ; year++ ] {
8
9     print(year)
10 }
11
12
13
14
15 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

2014

2015

2016

2017

2018

2019

2020

Exited

5

الآن لنقوم بشرح هذا المثال البسيط واستخدام دالة IF داخل الحلقة FOR

في هذا المثال تقوم الحلقة التكرارية بنفس الية عملها كما تكلمنا عنها سابقا حيث تقوم بعد الأعوام من ١٩٠٠ حتى تصل الى ١٩٥٠ هنا قد تتحقق شرط في دالة الشرطية IF هنا سوف يقوم البرنامج بتنفيذ امر الطباعة فيها ومن ثم يكمل العد حتى يصل الى ٢٠٢١ يقوم بتحقق منها وترجع قيمة FALSE لأن الشرط غير محقق وهنا يقوم البرنامج بخروج من الحلقة التكرارية .

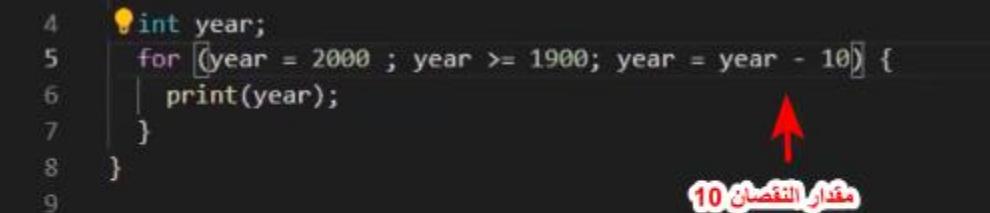
كما هو واضح في المثال في الأسفل :

```
10  
11     for ( year = 1900 ; year <= 2020 ; year++ ) {  
12         if (year == 1950) {  
13             print("year ===== 1950") ;  
14         }  
15         print(year) ;  
16     }  
17  
18
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
1947  
1948  
1949  
year ===== 1950 ← خرج الدالة if  
1950  
1951 ← خرج حلقة FOR  
1952  
1953
```

مثال اخر عن حلقة for كما هو واضح في الصورة في الأسفل لدينا حلقة for لكن مقدار الخطوة لدينا هو ١٠ وهو مقدار تناقص حيث ستقوم الحلقة for بنفس الية عملها حيث ستقوم بأنقاص ١ من النقطة الابتدائية ٢٠٠٠ حتى تصل الى ١٨٩٠ ترجع قيمة false وتخرج من الحلقة بأمكانك ادخال مقدار قيمة التناقص او الازدياد في خطوة مهما كانت بشرط ان لا ترجع قيمة .false



```
4 int year;
5 for [year = 2000 ; year >= 1900; year = year - 10] {
6     print(year);
7 }
8 }
```

مقدار التناقص 10



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
2000
1990
1980
1970
1960
1950
1940
1930
```

خروج البرنامج بعد التشغيل

While Loop

تشابه اليه عمل الحلقة While مع الحلقة for مع الاختلاف بطريقة الكتابة

كما هو واضح في الصورة في الأسفل :

```
2 int year = 1900 ; ← نقطة البداية
3
4 while [ year <= 2000 ] {
5
6     print(year) ; ← الشرط
7
8     year++ ; ← مقدار الخطوة
9 }
10
11 }
12 }
13 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
1994
1995
1996
1997 ← خرج البرنامج
1998
1999
2000
Exited
```

Do While Loop

تعمل حلقة Do while loop نفس عمل حلقة for ولكن تختلف عنهم انه تقوم بطباعة سواء الشرط تحقق او لن يتحقق كما سبق وذكرنا في الدورس السابقة انه في حال عدم تحقق الشرط في الحلقة ارجاع قيمة false لأن الكود الذي في داخل الحلقة لن يتم تنفيذه ولكن هذا الامر يختلف في حلقة while كما هو واضح في الصورة في الأسفل ان الشرط غير متحقق لأن ٢١٠٠ اكبر من ٢٠٠٠ ومع ذلك قام البرنامج بطباعة نقطة البداية والخروج من الحلقة.

```
2 int year = 2100;
3
4 do{
5     print(year);
6     year++;
7 }while (year <= 2000);
8
9
10 }
11 }
12 }
```

الخطوة الأولى: نقطة البداية

خطوة: خطوة

الشرط غير متحقق

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

2100
Exited

خروج البرنامج

break

مفهوم ال break مهمته هو الخروج من الحلقة التكرارية فقط لاغير وتعمل ال break مع جميع أنواع الحلقات التكرارية

لدينا المثال التالي :

نلاحظ ان لدينا حلقة تكرارية while ونقطة الابتدائية هي ١٩٠٠ ونريد عدد الأعوام من ١٩٠٠ الى ٢٠٠٠ ونلاحظ أيضاً تحقق الشرط الحلقة لأن قيمة year اصغر من ٢٠٠٠ وبالتالي ارجاع قيمة true وبسبب وجود break بعد تتحقق الشرط تقوم بخروج من الحلقة التكرارية بشكل نهائي وعدم تنفيذ امر الطباعة .

The screenshot shows a code editor with a Python script. The code is as follows:

```
1  #!/usr/bin/python
2  int year = 1900;           ← نقطة البداية
3
4  while (year <= 2000):    ← تحقق الشرط
5      break;
6      print(year);          ← الخروج من الحلقة
7      year++;               ← مقدار الخطوة
8
9
10
11
12
13
14 }
```

Annotations in Arabic are present in the code:

- نقطة البداية (Line 2)
- تحقق الشرط (Line 4)
- الخروج من الحلقة (Line 6)
- مقدار الخطوة (Line 7)
- خرج البرنامج (Line 10, pointing to the status bar)

The status bar at the bottom left shows "PROBLEMS 1" and "OUTPUT DEBUG CONSOLE TERMINAL".

ليكن لدينا دالة If داخل الحلقة التكرارية While نجد في الصورة في الأسفل ان الشرط حلقة محقق أي ان قيمة متغير year اصغر من ٢٠٠٠ بهذا فإن الشرط متحقق وسيقوم الحاسوب بدخول الى الحلقة وتحقق من شرط دالة if عند وصول العد الى ١٩٥٠ سيتحقق شرط if وسيقوم الحاسوب بعمل break والخروج من الحلقة بشكل نهائي كما هو واضح ولن يقوم بطباعة ١٩٥٠

```

2     int year = 1900;
3
4     while (year <= 2000) {
5
6         if (year == 1950) break ;
7
8         print(year);
9         year++;
10    }
11
12
13
14
15

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1943
1944
1945
1946
1947
1948
1949
Exited

continue

تقوم continue بنفس عمل break ولكن يوجد اختلاف الا وهو ان break تقوم بخروج من الحلقة التكرارية بشكل كامل ونهائي اما continue تقوم بخروج من الدورة او الخطوة لنفهم الامر بشكل افضل .

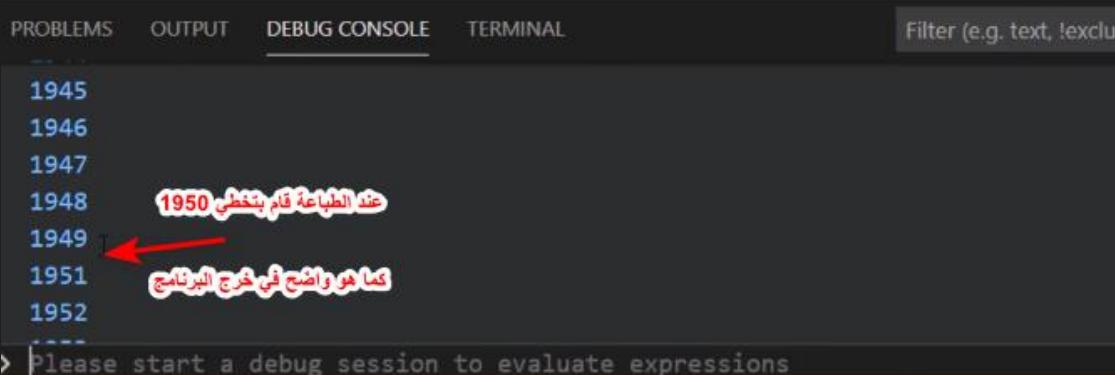
ليكن لدينا هذا المثال الموضح في الأسفل :

ليكن لدينا دالة If داخل الحلقة التكرارية for نجد في الصورة في الأسفل ان الشرط حلقة متحقق أي ان قيمة متغير year اصغر من ٢٠٠٠ بهذا فإن الشرط متحقق وسيقوم الحاسوب بدخول الى الحلقة وتحقق من شرط دالة if عند وصول العد الى ١٩٥٠ سيتحقق شرط if وسيقوم الحاسوب بعمل continue والخروج من دورة فقط كما هو واضح ولن يقوم بطباعة ١٩٥٠ لكن لن يقوم بخروج من الحلقة بشكل نهائي.

```
2 int year = 1900;
3 for (year = 1900 ; year <= 2000 ; year++){
4     if (year == 1950) continue ;
5     print(year) ;
6 }
7 }
```

الشرط متحقق

الخروج من الدورة



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Filter (e.g. text, !exclu

```
1945
1946
1947
1948      عند الطباعة قام بتخطي 1950
1949
1951      كما هو واضح في خرج البرنامج
1952
```

Please start a debug session to evaluate expressions

في هذا المثال المطابق تماماً أعلاه قمنا فقط بـأعطاء شرطين في دالة if هو ان عند وصول العد الى اكبر من ١٩٥٠ واصغر من ١٩٧٥ فم بعمل continue وتتجاهل الدورات من ١٩٥١ الى ١٩٧٤ وبذلك لن يقوم الحاسوب بطباعتها كما هو واضح في المثال في خرج البرنامج .

```
2     int year = 1900;
3
4     for (year = 1900 ; year <= 2000 ; year++){
5         if (year > 1950 && year < 1975) continue ;
6         print(year) ;
7     }
8 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
1945
1946
1947
1948
1949
1950 ←
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
```

سيقوم البرنامج بتتجاهل دورات من 1951 الى 1974

Number

العمليات على الأعداد في المثال الموضح أدناه لدينا لأننا نقوم بتوجيه سوال للحاسوب مضمونه هل المتغير قيمته محدودة وهذا ما تعنيه `isFinite` ماذا يعني رقم محدود ورقم غير محدود :

الرقم لمحدود مثل ١٠ او ٩٩ او ١ او -١

ونلاحظ أن قام البرنامج بأرجاع قيمة `true` لأن الرقم ١٠ رقم محدود.

```
3 | int a = 10 ;
4 |
5 | print(a.isFinite);
6 |
7 | }  
8 | هل متغير محدود
```

The screenshot shows a code editor interface with tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL. The DEBUG CONSOLE tab is selected. In the OUTPUT section, the word "true" is highlighted in blue and has a red arrow pointing to it from the text above. Below "true" is the word "Exited". In the DEBUG CONSOLE section, the text "خرج البرنامج" (Program exited) is displayed in red, with a red arrow pointing to it from the text above.

ولكن العدد الغير محدود مثل تقسيم عدد على صفر ترجع قيمة لا نهاية وهذا الأمور لا تتعلق بعلم الحاسوب انما هي قواعد رياضية كما هو واضح في الصورة في الأسفل :

```
3 double a = 10 / 0 ;  
4  
5 print(a.isFinite());  
6  
7 }  
8
```

عدد لا محدود
تقسيم عدد على صفر لا نهاية

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

false

Exited

خروج البرنامج عند التشغيل

ويوجد أيضا عكس `isInfinite` وهي `isFinite` هنا نقوم بتوجيه سؤال للحاسوب هل العدد `a` غير محدود سيقوم بأرجاع قيمة `true` لأن العدد فعلا غير محدود كما هو واضح في الصورة في الأسفل :

```
3 double a = 10 / 0 ;  
4  
5 print(a.isInfinite());  
6  
7 }  
8
```

هل العدد غير محدود

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

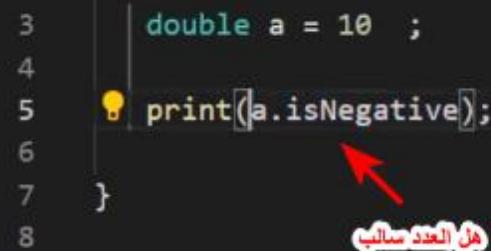
true

Exited

خروج البرنامج

هل العدد سالب IsNegative
يقوم برجاع قيمة false في خرج البرنامج ولو كان عكس ذلك سيقوم برجاع قيمة true كما هو واضح في الصورة في الأسفل :

```
3 | double a = 10 ;  
4 |  
5 | print(a.isNegative);  
6 |  
7 }  
8 |
```

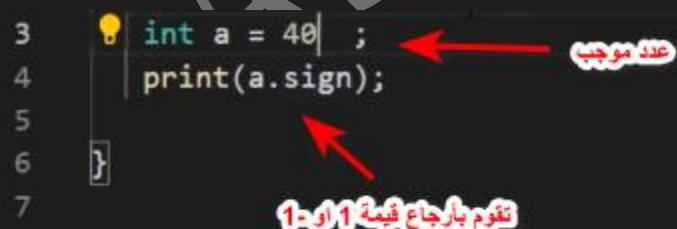


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

false Exited 

تقوم برجاع قيمة 1 للعد الموجب في خرج البرنامج او -1 للعدد السالب
وتقزم برجاع قيمة صفر اذا كان العدد صفر المثال كما هو واضح في الصورة في
الأسفل :

```
3 | int a = 40 ;  
4 | print(a.sign);  
5 |  
6 |  
7 |
```



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1 Exited 

IsEven هل العدد زوجي

في حال كان قيمة المتغير عدد زوجي سيقوم بأرجاع قيمة true واذا كان فردي سيقوم بأرجاع قيمة false كما هو واضح في اصورة في الأسفل :

```
3 | int a = 10 ;  
4 | print(a.isEven);  
5 |  
6 }  
7 |
```

هل العدد زوجي

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

true ← خرج البرنامج
Exited

IsOdd هل الرقم فردي

في حال كان قيمة المتغير عدد فردي سيقوم بأرجاع قيمة true واذا كان زوجي سيقوم بأرجاع قيمة false كما هو واضح في اصورة في الأسفل :

```
2 |  
3 | int a = 5 ;  
4 | print(a.isOdd);  
5 |  
6 }  
7 |
```

هل الرقم فردي

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

true ← خرج البرنامج
Exited

Abs() قيمة مطلقة (أي قيمة سالبة يبقوم بتحويلها الى قيمة موجبة)

كما هو واضح في المثال في الأسفل :

```
1
2
3     int a = -10 ;
4     print(a.abs());
5
6 }
7
```

قيمة مطلقة

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

10
Exited خرج البرنامج

Ceil() إعطاء قيمة تقريرية للعدد العشري(تحويله لعدد صحيح من دون فواصل).

ودائما ما يقرب للعدد الأكبر كما هو واضح في الصورة في الأسفل :

```
1
2
3     double a = 10.2 ;
4     print(a.ceil());
5
6 }
7
```

قيمة عشرية

تقريب العدد واعطاء قيمة صحيحة له

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

11
Exited خرج البرنامج

`compareTo(number)` يقوم بـلـمـقارـنـة بين عـدـد وـاـخـر وـيـرـجـع ثـلـاثـ قـيـمـة :

عـنـدـ التـسـاوـيـ العـدـدـيـنـ يـرـجـع قـيـمة صـفـرـ

عـنـدـ يـكـونـ العـدـدـ الـأـوـلـ أـكـبـرـ مـنـ العـدـدـ الثـانـيـ يـرـجـع قـيـمة 1ـ

عـنـدـ يـكـونـ العـدـدـ الـأـوـلـ أـصـغـرـ مـنـ العـدـدـ الثـانـيـ يـرـجـع قـيـمة -1ـ

كـمـاـ هـوـ وـاـضـحـ فـيـ المـثـالـ فـيـ الـأـسـفـلـ :

```
3 | int a = 10 ; ← العدد الأول
4 | print(a.compareTo(9)); ← العدد الثاني
5 |
6 | ← مقارنة بين عدد و عدد
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

خرج البرنامج

1 Exited

`Floor()` تقوم بـأـرـجـاعـ العـدـدـ العـشـرـيـ إـلـىـ عـدـدـ صـحـيـحـ وـتـجـاهـلـ الأـرـقـامـ الـتـيـ تـأـتـيـ

بعـدـ الفـاـصـلـةـ كـمـاـ هـوـ وـاـضـحـ فـيـ المـثـالـ فـيـ الـأـسـفـلـ :

```
3 | double a = 10.343434; ← عدد عشري
4 | print(a.floor()); ← ارجاع عدد صحيح
5 |
6 |
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

خرج البرنامج

10 Exited

تقوم بنفس عمل Ceil() ولكن تقوم بتقريب الى العدد الأقرب Round()

كما هو واضح في المثال في الأسفل :

```
4 | double a = 10.34 ;           عدد عشرى
3 | print(a.round());           ←
4 |
5 | }                         تقرير الاعداد
6 |
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
10 ← خرج البرنامج
Exited

: intToInt() تحويل قيمة المتغير من double الى

كما هو واضح في الصورة في الأسفل :

```
3 | double a = 10.44 ;
4 | print(a.toInt());           from double to int
5 |
6 |
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
10 ← خرج البرنامج
Exited

: double تحويل قيمة المتغير من int الى inttoDouble()

كما هو واضح في الصورة في الأسفل :

```
3 | int a = 232323 ; ← قيمة صحيحة
4 | print(a.toDouble()); ←
5 |
6 | from int to double
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

232323.0

Exited ← خرج البرنامج

int تحويل قيمة المتغير من نوع string الى Parse(variable)

كما هو واضح في الصورة في الأسفل :

```
3 | String a = "10" ; ← type string
4 | print(num.parse(a)); ←
5 |
6 | تحويل من نص الى عدد
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

10 ← خرج البرنامج

Exited

String Method

علامة + في متغيرات من نوع STRING هي دمع لعى عكس الأرقام تكون وظيفتها جمع الاعداد مثل ما هو واضح في المثال في الأسفل لدينا متغيران من نوع string اردنا وضع المتغيران في امر الطباعة ووضع مسافة بينهما باستخدام دوبل كوتشن.

```
1 void main() {  
2     String name = "wael"; ← المتغير الاول  
3     String lastname = "ahmed"; ← المتغير ثالث  
4     print(name + " | " + lastname);  
5 }  
6  
7 فراغ
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
wael ahmed ← خرج البرنامج
Exited

لا يمكن وضع الأرقام من نوع int ودمجها مع متغيرات من نوع string الا اذا كانت ضمن دوبل كوتشن الرقم هنا هو من نوع string لانه ضمن دوبل كوتشن.

```
2 |     String name = "wael";
3 |     String lastname = "ahmed";
4 |     print(name + " " + lastname + " 1");
5 |
6 }
7
```

type string

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
wael ahmed 1 ← خرج البرنامج
Exited

```
2 |     String name = "wael";
3 |     String lastname = "ahmed";
4 |     int a = 10; ← عملية جمع ارقام
5 |     print(name + " " + lastname + " ${a + 10} "); ← متغير من نوع int
6 |
7
```

فراغ ← دolar ساين خرج البرنامج بعد عملية الحسابية

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Filter (e:
wael ahmed 20 ← خرج البرنامج بشكل كامل
Exited

في المثال الموضح أعلاه نريد الربط بين متغيرات string ومتغيرات int وتنفيذ عملية جمع بين عددين دون ظهور خطأ في البرنامج.

نقوم بوضع المتغيران من نوع int ضمن {} ونضع قبله إشارة \$ ليقوم جهاز الكمبيوتر بتعريف عليهم على انهم اعداد صحيحة من نوع int .

Operations on variables of type String

نقوم بـ`IsEmpty` اذا اردنا ان نعرف اذا كان المتغير فارغ او لا ويقوم بـ`false` او `true`.

في حال كان لديه قيمة سبقت بـ`false` في قيمة `.isEmpty`

كما هو واضح في الصورة في الأسفل :

```
2 String name = "";
3
4 print(name.isEmpty) ;
5
6 }
7
```

هل متغير فارغ

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

true ← خرج البرنامج بعد التشغيل
Exited

ملاحظة مهمة جدا : المسافة او المسطرة لها قيمة مثلها مثل أي حرف او رقم او إشارة او علامة في الحاسوب.

```

2 String name = " ";
3
4 print(name.isEmpty) مسافة او مسطرة
5
6 }
7

```

هل المتغير فارغ

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

false ← خرج البرنامج بعد التشغيل
Exited لأن المتغير أو المسطرة لها قيمة

علينا التفريق بين متغير فارغ ومتغير ليس له قيمة كما في الصورة في الأسفل لأن المتغير الذي لا يحوي على قيمة يرجع قيمة null.

```

2 String name ;
3
4 print(name.isEmpty) متغير لا يحوي على قيمة
5
6 }
7

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Receiver: null ← null
Tried calling: isEmpty
#0 Object.noSuchMethod (dart:core-patch/object_patch.d
#1 main

string عدد الاحرف او طول string يقوم بعد الاحرف بلمتغير من نوع Length ويرجع قيمة من نوع int لدينا المثال في الأسفل نريد معرفة عدد الاحرف في متغير name وعند التنفيذ قام بطباعة عدد 8 عدد احرف هو ٧ بالإضافة للفراغ فقم بطباعة عدد ٨ في الخرج كما هو واضح في الصورة في الأسفل :

```

2 |     String name = 'wael abo';
3 |
4 |     print(name.length);           فراغ
5 |
6 |
7 |

```

عدد او طول نص

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

8
Exited

خرج البرنامج بعد التشغيل

تحويل من حروف الصغيرة الى حروف كبيرة

كما هو واضح في المثال في الأسفل :

```

2 |     String name = 'wael ahmed';
3 |     ^
4 |     print(name.toUpperCase());           احرف صغير
5 |

```

PROBLEMS OUTPUT DEBUG CONSOLE ...

Filter (e.g. te

WAEL AHMED

Exited

تحويل من احرف صغيرة الى احرف كبيرة

خرج البرنامج بعد التشغيل

تحويل من احرف كبيرة الى احرف صغيرة

كما هو واضح في الصورة في الأسفل :

```

2 | String name = 'WAEL';
3 | print(name.toLowerCase());
4 |
5 |

```

PROBLEMS OUTPUT DEBUG CONSOLE ... Filter (

wael
Exited

Trim() تقليل او حذف الفراغات سبق وذكرنا في الدروس السابقة ان الراغ له قيمة وعند استعمال ال trim في برنامجنا قام بتجاهل الفراغ وطباعة عدد احرف الكلمة التي تحوي على ٤ حروف فقط كما هو واضح في الصورة في الأسفل :

```

2 | String name = 'WAEL ';
3 | print(name.trim().length);
4 |
5 |
6 }
7

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

مقارنة بين متغيرين من نوع string يقوم بأرجاع قيمة compareTo("string") ١ و ١٠ و صفر.

في حال كان عدد احرف متغير الأول اكثرب من متغير ثاني سيقوم بأرجاع القيمة ١

في حال كان عدد احرف متغير الأول اقل من متغير ثاني سيقوم بأرجاع القيمة -١

في حال يوجد تطابق تام بينهم سيقوم بأرجاع قيمة صفر.

```

2 |     String name = 'wael'; ← المتغير الأول
3 |     ⚡
4 |     print(name.compareTo("w")); ← المتغير الثاني
5 |
6 } ← مقارنة
7

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1 ← خرج البرنامج
Exited

replaceAll(from , replase)

تستطيع من خلالها استبدال كلمة ما في المتغير سيقوم ببحث عن كلمة you واستبدلها بكلمة all .

كما هو واضح في المثال في الأسفل :

```

2 |     String name = 'how are you';
3 |     ⚡
4 |     print(name.replaceAll("you", "all")); ← الكلمة الجديدة
5 |
6 } ← اسم المتغير ← الكلمة المراد استبدلها ← دالة الاستبدال
7

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

how are all ← خرج البرنامج
Exited

List

القوائم او مصفوفة : وهي نوع من أنواع المتغيرات في البرمجة.

تقبل عدة قيم بداخلها بمختلف أنواعها اعداد او نصوص نفصل بين القيمة والأخرى بأشارة الكومة او فاصلة (،) .

عند كتابة مجموعة من القيم بداخلها فأن اول قيمة فيها تحجز قيمة صفر في ذاكرة الحاسوب وثاني قيمة تأخذ قيمة ١ وثالث قيمة تأخذ قيمة ٢ وهكذا الى لا نهاية.

كما هو واضح في الصورة في الأسفل :

```
-  
3 List names = [ "wael" , "basel" , "ahmed"] ;  
4 print(names[0]) ; |  
5  
6 }  
7 رقم العنصر المحجوز في ذاكرة الحاسوب
```



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
wael ← خرج البرنامج بعد التشغيل
Exited

نستطيع كتابة او انشاء list داخل list انشاء عدد لا نهائي من list بداخل بعضهم البعض.

كما هو واضح في الصورة في الأسفل :

```

2
3 List names = [ "wael" , [ "1" , "2" , "3"] , "basel" , "ahmed" , "mohammad" ];
4 print(names[1]) ; 0 1 2 3 4
5 }
6 }
7
  رقم العنصر المحفوظ في ذاكرة الحاسوب
  خرج البرنامج بعد الطباعة
  list داخل list
  list الأولى
  list الثانية
  list

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Filter (e.g. text, exclude)

[1, 2, 3] Exited

ليكن لدينا المثال الذي في الأسفل نلاحظ انه يوجد list أخرى ونريد الوصول وطباعة عنصر معين على سبيل المثال الرقم ٢ من ال list الثانية من ال list الثانية نلاحظ ان حجز list الثانية في list الأولى وتأخذ قيمة [1] في ذاكرة الحاسوب وان الرقم ٢ يحجز أيضا قيمة [1] من list ثانية

اذا اردنا طباعة العنصر نقوم بكتابة اسم list ثم رقم حجز list ثانية قيمته [1] ثم رقم حجز العنصر ٢ داخل list ثانية وقيمه أيضا [1]

كما هو واضح في الصورة في الأسفل :

```

2
3 List names = [ "wael" , [ "1" , "2" , "3"] , "basel" , "ahmed" , "mohammad" ];
4 print(names[1][1]) ; 0 1 2 3 4
5 }
6 }
7
  list الأولى
  list الثانية
  list
  list
  list
  list

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Filter (e.g. text, exclude)

2 Exited

يمكننا ان نقوم بـأعادة تعيين قيمة العناصر من داخل list كما هو واضح في الصورة لدينا في الأسفل :

```
1
2
3 List names = [ "wael" , "basel" , "ahmed" , "mohammad"] ;
4 names[0] = "ggg" ;
5 print(names[0]) ; ← (إعادة تعيين قيمة المتغير)
6
7
8 }
9
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL خرج البرنامج بعد التشغيل Filter (

ggg
Exited

إضافة عنصر جديد من خلال add كما هو واضح في الصورة في الأسفل :

```
1
2
3 List names = [ "wael" , "basel" , "ahmed" , "mohammad"] ;
4 names[0] = "ggg" ;
5 names[1] = "ww" ;
6 names.add("kamal") ;
7 print(names) ; ← (إضافة عنصر جديد لل list)
8
9
10 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL ← (العنصر الجديد) Filter (

[ggg, ww, ahmed, mohammad, kamal]
Exited

لو اردنا على سبيل المثال ان نقوم بطباعة جميع عناصر list بلتأكيد يمكننا ان نقوم باستخدام امر الطباعة وطباعة كل عنصر على حدا كما هو واضح في الصورة في الأسفل :

```
3 List names = [ "wael", "basel" , "ahmed" , "mohammad"] ;
4
5 print(names[0]) ;
6 print(names[1]) ;
7 print(names[2]) ;
8 print(names[3]) ;
9
10
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Filter (e.g. text)

```
wael
basel
ahmed
mohammad
Exited
```

ولكن في حال كان لدينا عناصر كثير جدا في list هل نقوم بتكرار امر طباع على عدد عناصر list بلتأكيد هذا الامر سيأخذ منا وقت طويل جدا .

لهذا فإن لدينا حل لمشكلة التكرار كما ذكرنا في الدروس السابقة يمكننا استعمال الحلقات التكرارية لطباعة عناصر list .

في هذا المثال الموضح في الأسفل :

لدينا مصفوفة تحوي على 4 عناصر نريد ان نطبع عناصر المصفوفة باستخدام الحلقة التكرارية for نقوم بإنشاء متغير نسميه x لنقوم ب تخزين عناصر المصفوفة ونعطيه قيمة ابتدائية صفر ثم نقوم بكتابة الشرط داخل for ان متغير x يجب ان يكون اصغر من عدد عناصر المصفوفة وكما نعمل من الدروس السابقة ان length تستخدم للعد عناصر او محارف المتغيرات ثم ندخل مقدار الخطوة قيمته واحد لان (1 = x++) ثم نقوم بتنفيذ امر الطباعة .

The screenshot shows a Java code editor and terminal window. The code defines a list of names and uses a for loop to print each name. Red annotations explain the code's execution:

- A red arrow points to the index 0 above the first element "wael".
- A red arrow points to the index 1 above the second element "basel".
- A red arrow points to the index 2 above the third element "ahmed".
- A red arrow points to the index 3 above the fourth element "mohammad".
- A red box highlights the line "print(names[x]);" with the text "المتغير الذي قمنا بتخزين عناصر المصفوفة به".
- A red box highlights the line "x++" with the text "يقوم بعد عناصر المصفوفة".
- A red arrow points from the word "Exit" in the terminal to the output line "Exited".
- A red arrow points from the word "Output" in the terminal to the printed names.
- A red arrow points from the word "Terminal" in the terminal to the "Exited" line.

```
3 List names = [ "wael", "basel" , "ahmed" , "mohammad"] ;
4           0      1      2      3
5 for (int x = 0 ; x < names.length; x++) {
6     print(names[x]);
7 }
8
9 المتغير الذي قمنا بتخزين عناصر المصفوفة به
10
11
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Filter (e.g. text)
wael
basel
ahmed
mohammad
Exited
```

لدينا طريقة اخر لطباعة عناصر المصفوفة وهي دالة مخصصة لـ list وهي (forEach(variable)) نقوم بتعريف متغير لنقوم ب تخزين عناصر المصفوفة بداخله ثم نقوم بطباعة هذا المتغير كما هو موضح في الصورة في الأسفل :

```

1 void main() {
2     List names = ["wael", "basel", "ahmed", "mohammad"];
3
4     names.forEach((name){
5         print(name);
6     }) ;
7 }
8

```

المتغير الذي يقوم ب تخزين عناصر المصفوفة بداخله
 دالة مختصرة لطباعة عناصر المصفوفة

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Filter (e.g. t)

wael
basel
ahmed
mohammad
Exited

خريج البرنامج بعد التشغيل

سيقوم بجلب اول عنصر من list First

كما هو واضح في الصورة في الأسفل :

```

2 List names = ["wael", "basel", "ahmed", "mohammad"];
3 print(names.first) ;
4
5
6 }
7

```

أول عنصر في مصفوفة

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Filter (e.g. t)

wael
Exited

خريج البرنامج

سيقوم بجلب اخر عنصر من list

كما هو واضح في الصورة في الأسفل :

```
2 List names = ["wael", "basel", "ahmed", "mohammad"];
3 print(names.last) ;
4
5 }
6
7
```

آخر عنصر في المصفوفة

خروج البرنامج

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Filter

mohammad
Exited

هل المصفوفة فارغة سيقوم بأرجاع قيمة false او true

كما هو واضح قام بأرجاع قيمة false لأن المصفوفة تحوي على عناصر

```
2 List names = ["wael", "basel", "ahmed", "mohammad"];
3 print(names.isEmpty) ;
4
5 }
6
7
```

هل المصفوفة فارغة

خروج البرنامج

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Filter

false
Exited

هل المصفوفة غير فارغة أيضا سيقوم بأرجاع قيمة false او true

كما هو واضح في المثال في الأسفل :

```
2 List names = ["wael"];
3 print(names.isEmpty) ;
4
5
6 }
7
```

هل مصفوفة غير فارغة

خروج البرنامج
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
true
Exited

يقوم بعكس عناصر المصفوفة ولكن لا يرجع الخرج على شكل list Reversed
كما هو واضح في المثال في الأسفل :

```
2 List names = ["wael" , "basel"];
3 print(names.reversed) ;
4
5
6 }
7
```

يقوم بعكس عناصر المصفوفة

خروج البرنامج
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
(basel, wael)
Exited

اذا اردنا طباعة الخرج على شكل list نقوم بكتابة ()reversed.tolist()

هل العنصر وحيد في حال كان عنصر وحيد في المصفوفة يقوم بأرجاع قيمته وطباعته في حال كان يوجد عدة عناصر باظهار رسالة خطأ كما هو واضح في الصورة في الأسفل :

```

3   List names = ["wael" , "basel" , "ahmed"];
4
5   print(names.single) ;
6
7
8 }
9

```

هل العنصر وحيد

خروج البرنامج المصفوفة تحتوي على عدة عناصر

The screenshot shows a Dart development environment. In the terminal tab, there is an unhandled exception message:

```

Unhandled exception:
Bad state: Too many elements
#0    List.single (dart:core-patch/growable_array.dart:261:5)
#1    main
#2    _startIsolate.<anonymous closure> (dart:isolate-patch/isolate_patch.dart:301:19)

#3    _RawReceivePortImpl._handleMessage (dart:isolate-patch/isolate_patch.dart:168:12)
Exited (255)

```

إضافة عناصر list اخرى كما هو واضح في الصورة في الأسفل :

```

2   List names = ["wael" , "basel"] ;
3   names.addAll(['mohmmad' , 'ahmed']) ;
4   print(names) ;
5
6

```

أضافة عناصر مصفوفة لمصفوفة اخرى

خروج البرنامج بعد اضافة عناصر المصفوفة

PROBLEMS OUTPUT DEBUG CONSOLE

DEBUG CONSOLE

[wael, basel, mohmmad, ahmed]

Exited

يقوم Insert بـأضافة عنصر منها مثل add ولكن مع إمكانية تحديد مكان العنصر في list كما هو واضح في الصورة في الأسفل :

```

2 | List names = ["wael" , "basel"] ;
3 | names.insert( 1 , "mohmmad") ;
4 | print(names) ;
5 |
6 |

```

Annotations in the code:

- (أضافة عنصر مع إمكانية تحديد مكانه)
- تحديد مكان العنصر بعد إضافته
- العنصر مراد إضافته

Output window:

خروج البرنامج

PROBLEMS OUTPUT DEBUG CONSOLE

DEBUG CONSOLE

[wael, mohmmad, basel]

Exited

يقوم insertAll(index , []) بـأضافة list كاملة الى list أخرى مع إمكانية تحديد موقعها كما هو واضح في الصورة في الأسفل :

```

2 | List names = ["wael" , "basel" , "wa"] ;
3 | names.insertAll(1 , ['mohmmad' , 'ahmed']) ;
4 | print(names) ;
5 |
6 |

```

Annotations in the code:

- (أضافة مصفوفة كاملة مع مكانية تحديد مكانها)
- تحديد موقع مراد إضافتها فيه
- عناصر المصفوفة المراد إضافتها

Output window:

عناصر المصفوفة المضافة

خروج البرنامج بعد التشغيل

PROBLEMS OUTPUT DEBUG CONSOLE

DEBUG CONSOLE

[wael, mohmmad, ahmed, basel, wa]

Exited

replacerange(start , end , reblase)

تقبل ثلاثة متغيرات :

Start هو العنصر الأول.

End هو العنصر الأخير.

Reblase القيمة أو العناصر المراد اضافتها.

يبدأ من الصفر ولكن ينتهي عند ٢ ولا يقوم بأسبدالها.

يقوم بأسبدال العناصر من 0 الى 2 حتى ولو كان هناك عنصر واحد في Reblase

كما هو واضح في الصورة في الأسفل :

```
2  
3 List names = ["wael" , "basel" , "wa"] ;  
4 names.replaceRange(0 , 2 , ["1" , "2"]);  
5 print(names);  
6 }  
7 }  
8 }
```

خروج البرنامج بعد التشغيل

DEBUG CONSOLE

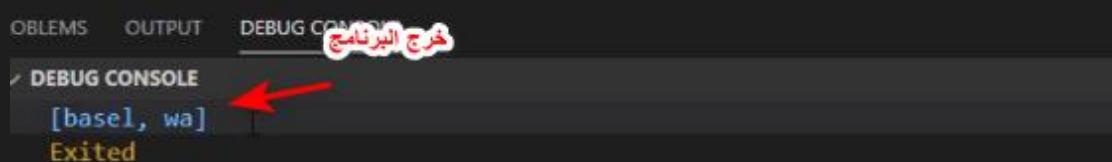
[1, 2, wa]
Exited

Remove يقوم بحذف عنصر من المصفوفة يقوم ببحث عن الكلمة المطابقة تماما للكلمة المراد حذفها أي انه حساس جدا لحالة المحارف.

كما هو واضح في الصورة في الأسفل :

```
3     List names = ["wael" , "basel" , "wa"] ;
4     names.remove("wael") ;
5     print(names);
6
7 }
8
```

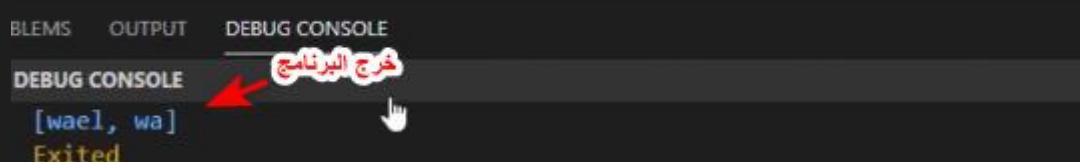
حذف عنصر من مصفوفة



removeAt حذف عنصر ما من المصفوفة ولكن بتحديد رقم موضعه في المصفوفة كما هو واضح في الصورة في الأسفل :

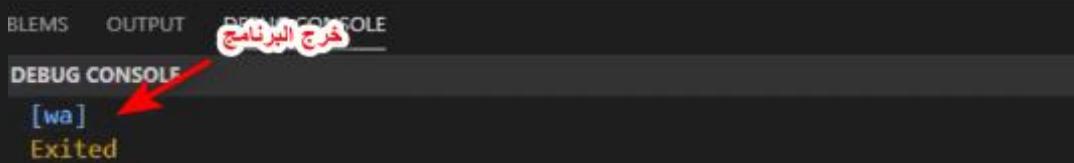
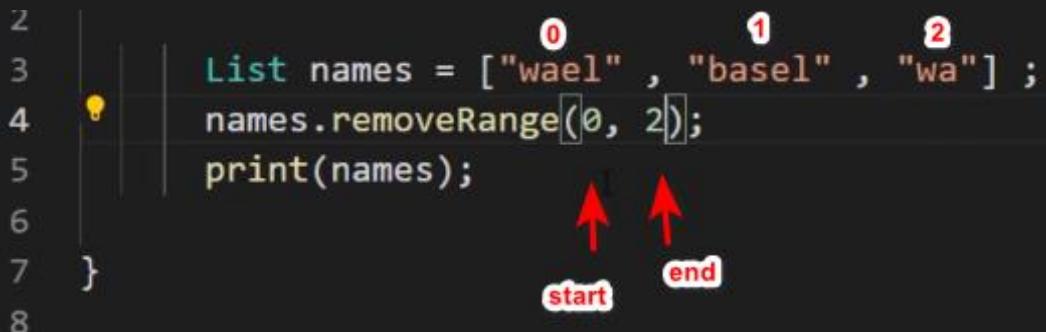
```
2
3     List names = ["wael" , "basel" , "wa"] ;
4     names.removeAt(1) ;
5     print(names);
6
7 }
8
```

حذف عنصر ما من المصفوفة
ولكن بتحديد رقم موضعه في المصفوفة



يقوم بحذف عدة عناصر من المصفوفة وذلك بتحديد مجال الحذف من صفر وينتهي عن ٢ ولكن بدون حذف عنصر ٢ كما هو واضح في المثال في الأسفل :

```
2
3     0   1   2
4 List names = ["wael" , "basel" , "wa"] ;
5     names.removeRange(0, 2);
6     print(names);
7 }
8
```



```
PROBLEMS OUTPUT DEBUG CONSOLE خرج البرنامج
DEBUG CONSOLE
[wa]
Exited
```

MAP

وهي نوع من أنواع المتغيرات تقبل بداخلها عدة عناصر وقيم

على شكل key : value

كما هو واضح في الصورة في الأسفل

```
3 |         صورة
4 | Map info = { "name" : "wael" , "age" : 12 , "price" : 2000 } ;
5 | print(info) ;
6 |
7 |
```

MAP اسم
key فاصل بين كل عنصر وآخر
value الفاصل بين كل عنصر وآخر

خروج البرنامج بعد طباعة

```
PROBLEMS OUTPUT DEBUG CONSOLE
DEBUG CONSOLE
{name: wael, age: 12, price: 2000}
Exited
```

في حال اردنا طباعة key من map كما هو واضح في الصورة في الأسفل :

كما هو الحال لو اردنا طباعة value نفس الامر تماما

```
4 | Map info = { "name" : "wael" , "age" : 12 , "price" : 2000 } ;
5 | print(info.keys) ;
6 |
7 |
```

طباعة فقط key

خروج البرنامج

```
PROBLEMS OUTPUT DEBUG CONSOLE
DEBUG CONSOLE
(name, age, price)
Exited
```

نستطيع اجراء العمليات على MAP :

.Map هل isNotEmpty ليس فارغة.

.isEmpty هل Map فارغة.

.Length عدد العناصر في Map

.Clear تفريغ Map بشكل كامل من عناصر.

واستخدام ForEach لطباعة key او value كما هو واضح في الصورة في الأسفل:

```
3  
4 Map info = { "name" : "wael" , "age" : 12 , "price" : 2000 } ;  
5 info.forEach((key, value) {  
6     print(value) ;  
7 }) ;  
8 }  
9
```

طباعة عناصر MAP

The screenshot shows a Java IDE interface with a code editor and a debug console. The code in the editor is a simple loop that prints the values of a Map named 'info'. The debug console shows the output: 'wael', '12', and '2000', each followed by a red arrow pointing to it. A red box highlights the word 'Exit' at the end of the console output. The title bar of the window says 'خرج البرنامج' (Program Exit).

Key Remove حذف عنصر من Map وتعامل فقط مع Key : Value

لأنه يعتبر عنصر واحد

```

2 | Map info = {"name": "wael", "age": 12, "price": 2000};
3 |
4 | info.remove("name");
5 | print(info);
6 }
7

```

يقوم بحذف عنصر من ال map

```

PROBLEMS OUTPUT DEBUG CONSOLE خرج البرنامج
DEBUG CONSOLE {age: 12, price: 2000}
Exited

```

Map إضافة عنصر الى Key : Value addAll

كما هو واضح في الصورة في الأسفل :

```

2 | Map info = {"name": "wael", "age": 12, "price": 2000};
3 | info.addAll([ "lastname" : "ahmed" ]);
4 | print(info);
5 }
6

```

إضافة عنصر الى Map

```

PROBLEMS OUTPUT DEBUG CONSOLE خرج البرنامج
DEBUG CONSOLE {name: wael, age: 12, price: 2000, lastname: ahmed}
Exited

```

نستطيع كتابة عدد لا نهائي من Map list داخل بعضهم البعض كما هو واضح في الصورة في الأسفل :

```
2 | Map info = {1 : ["wael" , "basel" , {"name" : "wael" , 1 : {"wael" : 23}} ] } ;  
3 | print(info) ;  
4 | }  
5 |
```

Problems Output DEBUG CONSOLE  Filter (e.g. te)

```
DEBUG CONSOLE  
{1: ["wael", "basel", {"name": "wael", 1: {"wael": 23}}]}  
Exited
```

Dyanmic Variables

هو نوع من أنواع المتغيرات مثله مثل int او String ولكن متغير ديناميكي var يقبل أي قيمة مهما كان نوعها سواء رقم او نص او مصفوفة ويتم تعينه لمرة واحدة فقط كما هو واضح في الصورة في الأسفل :

```
1 void main() {  
2     var name = 2 ; ↑↑ تعيين أول قيمة للمتغير  
3     name = "wael" ; ↑↑ إعادة تعين قيمة للمتغير  
4     print(n A value of type 'String' can't be assigned to a  
5         variable of type 'int'.  
Try changing the type of the variable, or casting  
the right-hand type to  
'int'. dart(invalid_assignment)  
  
PROBLEMS 1 OUTPUT  
DEBUG CONSOLE 2 Exited  
Peek Problem (Alt+F8) Quick Fix... (Ctrl+.)
```

يمكنا إعادة تعين قيمة للمتغير في حال كان نفس نوع المتغير الأول :

المتغير الأول من نوع string اذا اردنا إعادة تعين قيمة يجب ان يكون حسرا من نوع string كما هو واضح في الصورة في الأسفل :

```
1 void main() {  
2     var name = "wael" ;  
3     name = "basel" ; ↑↑ إعادة تعين قيمة  
4     print(name) ;  
5 }
```

PROBLEMS OUTPUT DEBUG CONSOLE
DEBUG CONSOLE basel خروج البرنامج
Exited

final and const

لا يستطيع احد تغيير قيمة المتغير او إعادة تعينه consts

كما هو واضح في الصورة في الأسفل : لدينا متغير من نوع string اسمه name بعد ما قمنا باضافة const قبله لا يمكننا إعادة تعين قيمته.

Run | Debug

```
1 void main() {  
2     const String name = "wael" ;  
3     name = "basel" ;  
4     print(name) ;  
5 }
```

اظهور رسالة الخطأ بعد تشغيل البرنامج

اعطاء قيمة نهائية للمتغير

اعادة تعين قيمة

Visual Studio Code

Build errors exist in your project.

Debug Anyway Show Errors Cancel

PROBLEMS 1 OUTPUT DEBUG CONSOLE

DEBUG CONSOLE

نستطيع تمرير قيمة للمتغير مرة واحدة فقط Final

كما هو واضح في الصورة في الأسفل :

تمرير قيمة للمتغير مرة واحدة فقط

```
1 void main() {  
2     final String name = "wael" ;  
3     name = "basel" ;  
4     print(name) ;  
5 }
```

رسالة الخطأ بعد تشغيل البرنامج

Build errors exist in your project.

Debug Anyway Show Errors Cancel

PROBLEMS 1 OUTPUT DEBUG CONSOLE

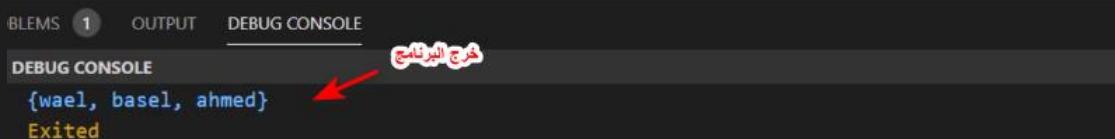
Set

هي نوع من أنواع المتغيرات أيضا مثل String و int و Map ولكن تختلف عن Map انها لا تقبل قيم من نوع key و value كما هو الحال في Map ولكن تقبل قيم عاديّة مثلها مثل List تتميز Set بعدم التكرار أي لو فرضنا اننا قمنا بتكرار قيمة معينة مثل ما هو واضح في المثال في الأسفل :

لا يقوم البرنامج بأظهار خطأ انه يوجد داخل Set قيمة متكرر ولكن سيقوم بعدم طباعتها في خرج البرنامج.

```
2     Set names = { "wael" , "basel" , "ahmed" , "ahmed" } ;  
3     print(names) ;  
4 }
```

القيمة المتكررة



نستطيع اجراء العمليات على Set مثلها مثل أي نوع من أنواع المتغيرات مثل singel و last و add و first و addAll وغيرها من أنواع العمليات على المتغيرات.

يمكننا اضافة عدة عناصر عن طريق addAll لـ list مثل addAll لـ Set و .Set لـ Set

Convert between

List Map Set

تحويل list الى Set

كما هو واضح في المثال في الأسفل :

```
2     Set names = {"wael", "basel"};
3
4     print(names.toList());
5 }
6
```

茅غور من نوع Set

茅حويل الى list

PROBLEMS OUTPUT DEBUG CONSOLE

✓ DEBUG CONSOLE خرج البرنامج
[wael, basel] ←
Exited

تحويل list الى Set نلاحظ عدد تكرار قيمة basel كما سبق وذكرنا ان Set تتميز بعدم التكرار على عكس List و Map كما هو واضح في المثال في الأسفل :

```
2     List names = ["wael", "basel", "basel"];
3
4     print(names.toSet());
5 }
6
```

茅غور من نوع list

茅حويل الى Set

PROBLEMS OUTPUT DEBUG CONSOLE

✓ DEBUG CONSOLE خرج البرنامج
{wael, basel} ←
Exited

تحويل Map او Set الى list

لنقوم بشرح المثال الذي في الأسفل :

لقد قمنا بتعريف List فارغة وقمنا بأعطائها اسم name

قمنا بتعريف Map واضافة عناصر فيها

الآن نريد اجراء عملية لتفريغ عناصر Map وجعلها في List نقوم باستخدام forEach نقوم بادخال اسم المصفوفة فارغة ثم Add ونقوم بتحديد نوع العناصر المراد جلبها من Map سواء كان Key او value .

The screenshot shows a Java code editor with the following code:

```
2 List name = [] ; ← تعرف مصفوفة فارغة
3 Map info = {"name" : "wael" , "age" : 22} ; ← تعرف Map
4           key   value   key   value
5 info.forEach((key, value) { name.add(value) ; }) ;
6           ← استخراج عناصر Map
7 print(name) ; ← اضافة عناصر Map الى المصفوفة الفارغة
8           ← طباعة value من Map
9 }
```

Below the code, the DEBUG CONSOLE tab is selected, showing the output:

PROBLEMS OUTPUT DEBUG CONSOLE خرج البرنامج
DEBUG CONSOLE [wael, 22] ←
Exited

في حال اردنا طباعة عناصر key من Map

The screenshot shows a Java code editor with the following code:

```
3 Map info = {"name" : "wael" , "age" : 22} ; ← key   value   key   value
4           key   value
5 info.forEach((key, value) { name.add(key) ; }) ;
6           ← key
7 print(name) ;
8
9 }
```

Below the code, the DEBUG CONSOLE tab is selected, showing the output:

PROBLEMS OUTPUT DEBUG CONSOLE خرج البرنامج
DEBUG CONSOLE [name, age] ←
Exited

Function

بفرض اننا قمنا بكتابة مجموعة من الاكواد والاسطر البرمجية ونريد استخدامها في اكثر من مكان في برنامجنا هل من المعقول ان نقوم بتكرار نفس الأوامر في اكثر من مكان في برنامجنا.

من هنا تأتي أهمية **Function** في لغات البرمجة اذ اننا نستطيع كتابة الكود مرة واحدة وضعه في **Function** واستدعائه عند الحاجة.

التصريح او ابسط طريقة لكتابة **Function** في لغة البرمجة dart.

The screenshot shows a code editor with a Dart file named `main.dart`. The code defines a function `printName` and calls it twice. Red annotations with arrows explain the code:

- A red box labeled "تصريح" (Declaration) points to the first line of the function definition: `// 1 - namefunction () { //code }`.
- A red box labeled "كتابة الأوامر" (Writing the command) points to the second line of the function definition: `// 2 - namefunction () ;`.
- A red box labeled "الاستدعاء" (Invocation) points to the call to the function: `namefunction();`.

```
// 1 - namefunction () { //code }
// 2 - namefunction () ;
namefunction();
```

كما هو واضح في المثال البسيط في الأسفل نريد طباعة اسم wael قمنا
بتصریح عن **Function** وكتابة امر طباعة في داخلها وعند استدعاءها يتم
تنفيذ الكود بعدد المرات التي يقوم فيها المبرمج بستدعائها كما هو واضح في
المثال في الأسفل:

```

5
6     printName() { print("wael") ; }
7     printName() ;
8     printName() ;
9     printName() ;
10    printName() ;
11 }

```

الاستدعاء

كود المراد تنفيذه

PROBLEMS OUTPUT DEBUG CONSOLE

DEBUG CONSOLE

wael
wael
wael
Exited

خروج البرنامج

وأيضا لدينا المثال نريد ان نقوم بطباعة ناتج جمع عددين صحيحين :

```

6 sumNumber() {
7     int a = 10 ;
8     int b = 20 ;
9     int c = a + b ;
10    print(c) ;
11 }

```

تصریح عن Function

متغير الأول

متغير الثاني

عملية الجمع

امر الطباعة

استدعاء ال Function

قم بطباعة ناتج الجمع مرتين
لأننا قمنا بإستخدامه مرتين

1 sumNumber() ;
2 sumNumber() ;

BLEMS OUTPUT DEBUG CONSOLE

DEBUG CONSOLE

30
30
Exited

خروج البرنامج

الآن لنقوم بأخذ مثال قوي نستطيع من خلاله فهم فائدة **Function** في لغات البرمجة من خلال برنامج طباعة ناتج جمع عددين ولكن هذه المرة من دون إعطاء قيم للمتغيرات قمنا بكتابة المتغيرات داخل **Function** ولكن على شكل برمترات لتقوم بأخذ قيم مختلفة عند الاستدعاء ثم قمنا بكتابة عملية جمع عددين داخل **Function** وفي الاستدعاء الأول قمنا بأخذ ١٠ و ٢٠ وناتج جمع هذين العددين هو ٣٠ كما هو واضح نفس الامر بالنسبة للاستدعاء الثاني بلقييم المختلفة الذي يحويها وناتج الجمع هو ٥٠ ونستطيع كتابة عدد لا نهائي من الاستدعاء.



```

5 البرامترات
6 sumNumber(int a , int b) {
7     int c | = a + b ;
8     print(c) ;
9 }
10 int a   int b
11 sumNumber(10 ,20 ) ; 1 استدعاء
12 sumNumber(20 ,30 ) ; 2 Function
13

```

البرامترات

تصريح عن Function

تنفيذ عملية الجمع

استدعاء Function (الأول والثاني ولكن بقيم مختلفة)

خروج البرنامج

Exit

بهذا تكون قد قمنا بتوفير وقت وجهد كبير بدلاً من تكرار الكود كما هو واضح :

```
14  
15     int a = 10 ;    ①  
16     int b = 20 ;  
17     int c = a+ b ;  
18     print(c) ; |  
19     a = 30 ;  
20     b = 50 ;    ②  
21     c = a + b ;  
22     print(c) ;  
23
```

PROBLEMS OUTPUT DEBUG CONSOLE

▼ DEBUG CONSOLE

```
30 ①  
80 ②  
Exited
```

Adel Aboba'

٨٢

Function types

تنقسم أنواع **Function** إلى :

لا يرجع قيمة **Void**

يرجع قيمة **Return**

ما هو الفرق الجوهرى بين **void** و **..return**؟؟

نوع **void** (لا يرجع قيمة) أي انه يقوم فقط بتنفيذ الكود داخل **Function** فقط.

اما **return** (يرجع قيمة) يقوم بأرجاع قيمة قمت بتحديدها حتى أقوم بتسجيلها في متغير.

جميع الأمثلة السابقة كانت **Function** من نوع **.void**

اما المثال الذي في الأسفل هو من نوع **return** فلما قمنا فقط بكتابة كلمة **return**

داخل **Function** وتحديد نوع القيمة الذي سيقوم بأرجاعها ثم تخزين اسم

مع البرامترات في متغير اسمه **r** وقمنا بطباعة هذا المتغير كما هو

واضح في الصورة في الأسفل :

The screenshot shows a code editor with Java code. Red annotations explain the code elements:

- Line 7: **تحديد نوع** (Type definition) points to the **int** keyword before **sumNumber**.
- Line 10: **نوع return** (return type) points to the **return c;** statement.
- Line 13: **ارجاع قيمة** (Return value) points to the **int r = sumNumber(10, 20);** call.
- Line 13: **تخزين Function في متغير** (Store Function in variable) points to the **r** variable in the assignment statement.
- Line 14: **امثلة الطباعة** (Print examples) points to the **print(r);** statement.
- Line 18: **خروج البرنامج** (Program exit) points to the output in the DEBUG CONSOLE: **Exited**.

```
int sumNumber(int a , int b ) {  
    int c = a + b ;  
  
    return c ;  
}  
  
int r = sumNumber(10 , 20 ) ;  
print(r) ;  
  
PROBLEMS OUTPUT DEBUG CONSOLE  
DEBUG CONSOLE  
30 Exited
```

يمكننا ارجاع أي قيمة في Function من نوع return قمنا بأرجاع قيمة متغير a وتخزينه في متغير آخر وطباعة المتغير كما هو واضح في المثال في الأسفل :

```

7 int sumNumber(int a , int b ) {
8     int c = a + b ;
9     1 print(c) ;
10    return a ; | ← ارجاع قيمة a
11 }
12 int r = sumNumber(10 ,20 ) ; → a=10
13
14 2 print(r) ;
15
16
17

```

PROBLEMS OUTPUT DEBUG CONSOLE

DEBUG CONSOLE خرج البرنامج

30 1
10 2
Exited

في حال كانت Function من نوع void وقمنا ب تخزين قيمة معينة داخل متغير من Function سيقوم البرنامج بأظهار رسالة خطأ لأنه لا يمكن ان يعيد قيمة ويخرنها كما هو واضح في المثال في الأسفل :

```

6 void sumNumber(int a, int b)
7 void | su This expression has a type of 'void' so its value can't be used.
8     int c Try checking to see if you're using the correct API, there might be a
9     print( function or call that returns void you didn't expect. Also check type
10    } parameters and variables which might also be
11
12 int r = sumNumber(10 ,20 ) ; void. dart(use_of_void_result)
13
14 print(r) ; Peek Problem (Alt+F8) No quick fixes available
15
16
17

```

لنقوم الان بتوضيح بعض الأمور عن **Function** وعلى أي أساس يتم تحديد نوعها..!!

يتم تحديد نوعها على حسب الكود المكتوب بداخل **Function**

في حال لم يتم كتابة Return داخل **Function** مع الكود سيتم اعتبارها من نوع Void او اذا تم كتابة Return في الكود داخل **Function** يتم اعتبارها من النوع الثاني وهو Return .

في حال لم يتم تحديد نوع القيمة التي سوف ي RETURN ي يقوم بأرجاعها فأن الحاسوب سيقوم بفحص نوع القيمة بشكل اتوماتيكي

قام بتحديد نوع Function انه من نوع int بناء على قيمة التي سوف يرجعها وكما هو الواضح فإن المتغير a قيمته عدد صحيح من نوع int كما هو واضح في المثال الذي في الأسفل :

```
5     int sumNumber(int a, int b)
6     sumNumber(int a , int b ) {
7         int c = a + b ;
8         print(c) ;
9         return a ;
10    }
```

قام بتحديد نوع
Function
بناء على قيمة التي سوف يرجعها

Function scope

.variable يعبر عن مدى إمكانية الوصول لل Scope

في المثال الموضح في الأسفل لدينا function يقوم بتعريف متغير داخلها واعطائه قيمة ٢٠ ثم ننتقل الى خارج function لطباعة هذا المتغير وعند تشغيل البرنامج سيقوم الحاسوب بأظهار رسالة خطأ هذا يعني اذا قمنا بتعريف متغير داخل function نستطيع استخدامه فقط داخل function التي قمنا بتعريفه بداخلاها.

A screenshot of Visual Studio Code showing a code editor and a message dialog. The code editor contains the following Java-like pseudocode:

```
6 sumNumber(int a, int b) {  
7     int c = a + b;  
8     int y = 20 ;  
9     print(c);  
10 }  
11 print(y) ;  
12 sumNumber(10, 20); function
```

The code is highlighted with boxes and annotations:

- A green box surrounds the entire function body (lines 6-11).
- A red arrow points from the text "تعريف متغير داخل function" to the line "int c = a + b;".
- A red arrow points from the text "طباعة المتغير خارج ال function" to the line "print(y) ;".
- A red arrow points from the text "رسالة الخطأ بعد التشغيل" to the message dialog.

The message dialog says: "Build errors exist in your project." with buttons "Debug Anyway", "Show Errors", and "Cancel".

لدينا حالة أخرى للمتغير : لو قمنا بتعريف المتغير خارج function وقمنا بتنفيذ امر الطباعة من داخل ال function سيعمل البرنامج من دون أي برماج كما هو واضح في الصورة في الأسفل :

A screenshot of Visual Studio Code showing the code editor and a terminal window (DEBUG CONSOLE). The code editor contains the same pseudocode as before, with annotations:

- A green box surrounds the entire function body (lines 6-11).
- A red arrow points from the text "تعريف المتغير خارج function" to the line "int c ;".
- A red arrow points from the text "أمر الطباعة" to the line "print(c);".
- A red arrow points from the text "function" to the closing brace of the function definition.

The terminal window shows the output:

```
PROBLEMS OUTPUT DEBUG CONSOLE  
DEB CONSOLE  
30  
Exited
```

A red arrow points from the text "خروج البرنامج بعد التشغيل" to the word "Exited".

في حال قمنا بتعريف function خارج Void Main() ستعمل من دون أي مشاكل كما هو واضح في المثال في الأسفل :

```

8     sumNumber(int a, int b) {
9         int c = a + b;
10        print(c);
11    }
12

```



PROBLEMS OUTPUT DEBUG CONSOLE

DEBUG CONSOLE خرج البرنامج
30
Exited



نستطيع أيضاً تعريف المتغيرات خرج Void Main() واستخدام هذا المتغير في أي function في البرنامج كما هو واضح في المثال في الأسفل :

```

1 int c; ← تعرف متغير خارج
Run | Debug
2 void main() { ← Void Main()
3 // 1 - namefunction ( argument ) { //code }
4 // 2 - namefunction () ;
5 // 3 - type function a - void b - return
6 // 4 - scope
7 c = 500 ; ← تنفيذ أمر الطباعة من داخل
8 print(c) ; ← Void Main()
9 sumNumber(10, 20);
10
11
12 sumNumber(int a, int b) {

```

Void Main()

PROBLEMS OUTPUT DEBUG CONSOLE

DEBUG CONSOLE خرج البرنامج
500
30
Exited

ملاحظة : Void Main() هي function معرفة مسبقا في لغة Dart.

لا نستطيع تنفيذ أي امر خارج Void Main() بل مختصر نستطيع فقط تعريف المتغيرات في الخارج كما هو واضح في الصورة في الأسفل :

```
1 int a = 10 ;
2 print(a); ← تنتفيذ امر الطباعة خارج Void Main()
3 void main() {
4     // 1 - namefunction ( argument ) { //code }
5     // 2 - namefunction () ;
6     // 3 - type function a - ← رسالة الخطأ بعد التشغيل
7     // 4 - scope
8
9     sumNumber(10, 20);
10 }
```

رسالة الخطأ بعد التشغيل

في حال كان لدينا function اسمها sumNumber() وقمنا بتعريف متغير ما داخل Void Main() وقمنا بتنفيذ امر الطباعة داخل هذه function سيقوم بأظهار رسالة خطأ كما هو واضح في المثال في الأسفل :

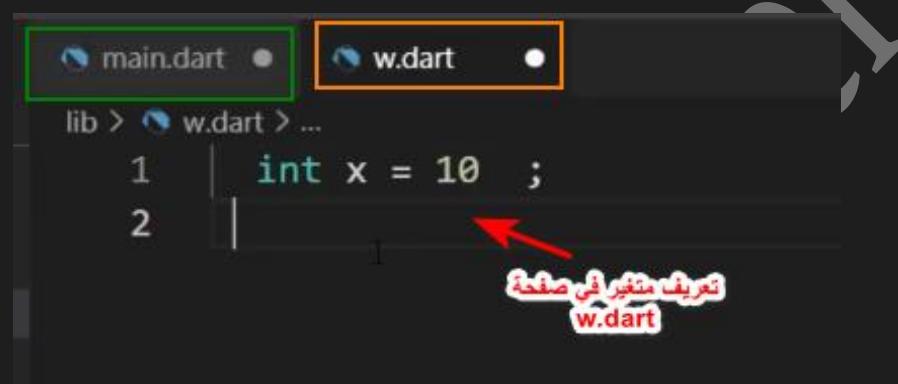
```
8 int c; ← تعريف متغير في Void Main()
9
10 sumNumber(10, 20);
11 }
```

```
4 sumNumber(int a, int b) { ← رسالة خطأ بعد التشغيل
5     c = a + b;
6     print(c); ← function sunNumber
7 }
```

رسالة خطأ بعد التشغيل

import

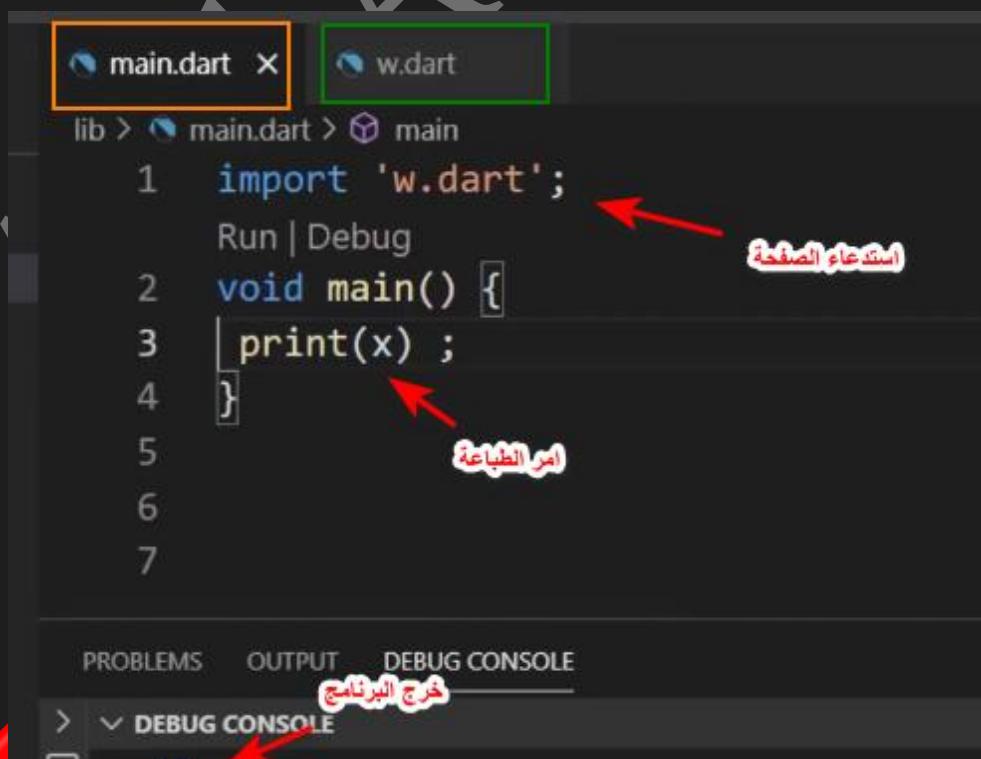
والمقصود به وهو الاستدعاء سنقوم بمثال بسيط عن الاستدعاء سنقوم بعمل ملف باسم w.dart وسنقوم بتعريف متغير فيه ونسميه x ونعطيه قيمة 10 كما هو واضح في المثال في الأسفل:



```
main.dart • w.dart •
lib > w.dart > ...
1 | int x = 10 ;
2 |
```

تعريف متغير في صفحة
w.dart

ثم نتوجه الى صفة main() ونقوم بكتابه الاستدعاء خارج ()



```
main.dart ✘ w.dart •
lib > main.dart > main
1 import 'w.dart';
Run | Debug
2 void main() {
3   print(x) ;
4 }
```

استدعاء الصفة

أمر الطباعة

DEBUG CONSOLE

خروج البرنامج

```
10
Exited
```

المسارات

في حال اردنا الدخول الى مجلد معين نقوم بكتابة `import` ثم نقوم بعمل دوبل كوتشن " " ونكتب بداخلها /. بهذا الشكل :

```
import 'w.dart';
```

او في حال اردنا الخروج او رجوع للخلف من مجلد معين نفس العملية تماما نكتب داخل دوبل كوتشن " " ونكتب بداخلها /.. بهذا الشكل :

```
import '../w.dart';
```

لعرض محتويات المجلد نقوم بكتابة نقطة داخل دوبل كوتشن " " بهذا الشكل :

```
import '...';
```

يوجد طريقة اسهل للاستدعاء نستطيع من خلالها أيضا استدعاء الصفحة المعرف فيها المتغير من خلال الوقوف فوق المتغير ونقوم بضغط على كما هو ظاهر في الأسفل :

main.dart > ... Undefined name 'x'.
Try correcting the name to one that is defined,
or defining the
void main name. dart(undefined_identifier)
Peek Problem (Alt+F8) Quick Fix... (Ctrl+,.)

```
print(x);
```

A red arrow points from the text "Quick Fix... (Ctrl+,.)" to the "Quick Fix..." button in the UI.

ثم نجد اسم الملف الذي يوجد به المتغير الذي قمنا بتعريفه نضغط عليه كما هو واضح في الصورة في الأسفل :

Run | Debug
void main() {

 print(x);
}

A red arrow points from the text "Import library 'w.dart'" in the dropdown menu to the same option in the UI.

- Import library './w.dart'
- Import library 'r/w.dart'
- Import library 'w.dart'**
- Import library 'w/a/myfunction.dart'
- Import library 'w/myfunction.dart'
- Create local variable 'x'

```
import 'w.dart';
```

Runes

تحويل الى ترميز في هذا المثال المعروض في الأسفل نلاحظ خرج البرنامج عبارة عن list كلها ارقام صحيحة اذ أن كل حرف او علامة او حتى فراغ له ترميز معين محفوظ في جهاز الكمبيوتر مثل الحرف W ترميزه ١١٩ والحرف a ترميزه ٩٧ كما هو واضح في المثال في الأسفل :

```
5  
4 String name = "wael";  
5 print(name.codeUnits) ;  
6  
7 }  
8  
9
```

تحويل الاحرف الى ترميز

ITEMS OUTPUT DEBUG CONSOLE **خرج البرنامج**
DEBUG CONSOLE [119, 97, 101, 108] Exited

```
4 String name = "wael" ;  
5  
6 print(name.runes) ;  
7  
8 }  
9
```

تحويل الى ترميز

ITEMS OUTPUT DEBUG CONSOLE **خرج البرنامج**
DEBUG CONSOLE (119, 97, 101, 108) Exited

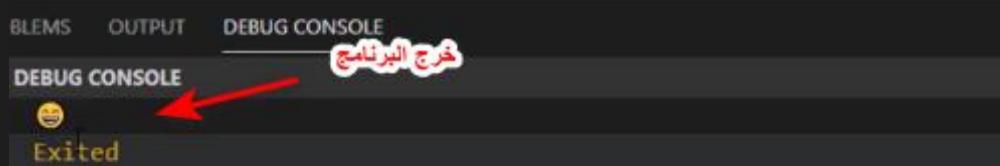
نستطيع الاستفادة من codeUnits في اخذ ترميز ايموجي معين واستخدامه

Native [1]	Apple [2]	Android [3]	Android [3]	Symbola [4]	Twitter [5]	Unicode	Bytes (UTF-8)	Description
😁	😊	_ANDROID_	😊	😊	😁	U+1F601	\xF0\x9F\x98\x81	grinning face with smiling eyes
😂	😂	_ANDROID_	😂	😊	😂	U+1F602	\xF0\x9F\x98\x82	face with tears of joy
😃	😃	_ANDROID_	😃	😊	😃	U+1F603	\xF0\x9F\x98\x83	smiling face with open mouth
😄	😄	_ANDROID_	😄	😊	😄	U+1F604	\xF0\x9F\x98\x84	smiling face with open mouth and smiling eyes
😅	😅	_ANDROID_	😅	😊	😅	U+1F605	\xF0\x9F\x98\x85	smiling face with open mouth and cold sweat
😆	😆	_ANDROID_	😆	😊	😆	U+1F606	\xF0\x9F\x98\x86	smiling face with open mouth and tightly-closed eyes
😉	😉	_ANDROID_	😉	😊	😉	U+1F609	\xF0\x9F\x98\x89	winking face
😊	😊	_ANDROID_	😊	😊	😊	U+1F60A	\xF0\x9F\x98\x8A	smiling face with smiling eyes

كما هو واضح في المثال في الأسفل :

```
3 String emoji = "\u{1F601}";  
4 print(emoji); |  
5  
6  
7  
8 }  
9
```

ترميز الایموجي



Assert

ليكن لدينا متغير ما في برماجنا واردنا تأكد من ان قيمته لا تساوي null

هنا نلجأ الى مفهوم assert والتي تستخدم من قبل المبرمجين وتتألف syntax الخاص بها من جزئين الأول هو كتابة الشرط والثاني الرسالة المراد عرضها في debug console كمل هو واضح في الصورة في الأسفل :

```
int a;
assert(a != null , "a != null");
// assert ( condition , "message")
print(a);
}
```

الشرط ← syntax ← الرسالة المراد اظهارها

مثال حي عن Assert كما هو واضح ان قيمة المتغير a تساوي null وعند طباعة قام بأظهار رسالة الخطأ

The screenshot shows a Dart code editor with the following code:

```
int a;
assert(a != null , "a != null");
print(a);
```

Annotations in Arabic point to specific parts of the code:

- A red arrow points to the word "null" in the assert condition, with the label "شلوى null".
- A red arrow points to the message string "a != null", with the label "تتأكد من ان قيمة المتغير لا تساوى Null".
- A green arrow points to the "syntax" part of the explanatory text above, with the label "الرسالة المراد اظهارها".
- A red arrow points to the "message" part of the explanatory text, with the label "الرسالة المراد اظهارها".
- A red arrow points to the "assert" keyword in the code, with the label " الشرط".
- A red arrow points to the "syntax" label in the explanatory text, with the label " الشرط".

The terminal below shows the output of the run command:

```
PS C:\Users\wael\Desktop\course> dart main.dart
Unhandled exception:
'file:///C:/Users/wael/Desktop/course/lib/main.dart': Failed assertion: line 3 pos 10: 'a != null': is not true.
#0    _AssertionError._doThrowNew (dart:core-patch/errors_patch.dart:46:39)
#1    _AssertionError._throwNew (dart:core-patch/errors_patch.dart:36:5)
#2    main
#3    _startIsolate.<anonymous closure> (dart:isolate-patch/isolate_patch.dart:301:19)
#4    _RawReceivePortImpl._handleMessage (dart:isolate-patch/isolate_patch.dart:168:12)

Exited (255)
```

A red arrow points from the explanatory text "مخرج البرنامج رسالة الخطأ" to the error message in the terminal.

Max And Min

هي `function` جاهزة من مكتبة `math` في لغة `dart` تستخدم لطباعة العدد الأكبر أو الأصغر.

مثال عن طباعة العدد الأكبر كما هو واضح في المثال في الأسفل :

```
Run | Debug
3 void main() {
4     int a = 20;           المتغير الأول
5     int b = 30;           المتغير الثاني
6     print(max(a, b));   دالة طباعة العدد الأكبر
7 }
8 
```

PROBLEMS OUTPUT DEBUG CONSOLE

DEBUG CONSOLE

```
30 خرج البرنامج
Exited
```

مثال عن طباعة العدد الأصغر كما هو واضح في المثال في الأسفل :

```
Run | Debug
3 void main() {
4     int a = 70;           المتغير الأول
5     int b = 30;           المتغير الثاني
6
7     print(min(a, b));   دالة طباعة العدد الأصغر
8 }
```

PROBLEMS OUTPUT DEBUG CONSOLE

DEBUG CONSOLE

```
30 خرج البرنامج
Exited
```

Data Type

(List - Set - Map)

فيما سبق قمنا بشرح أنواع المتغيرات ولكن في هذا الدرس سننتمق بدراسة أنواع المتغيرات والحالات التي نستطيع اجرائها عليها.

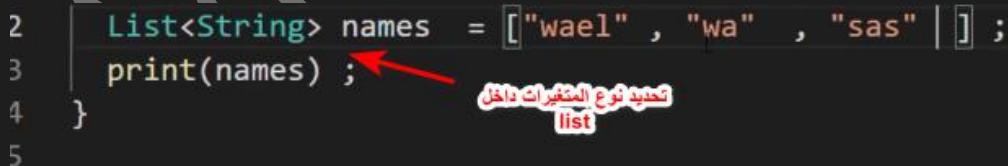
كما نعلم ان list تقبل مجموعة من أنواع مختلفة من متغيرات وتكون dynamic كما هو واضح في المثال في الأسفل :



```
Run | Debug
void ma List<dynamic> names
{
    List names = ["wael" , "wa" , "sas" , 2 , {"name" : "wael"} ] ;
    print(names) ;
}
```

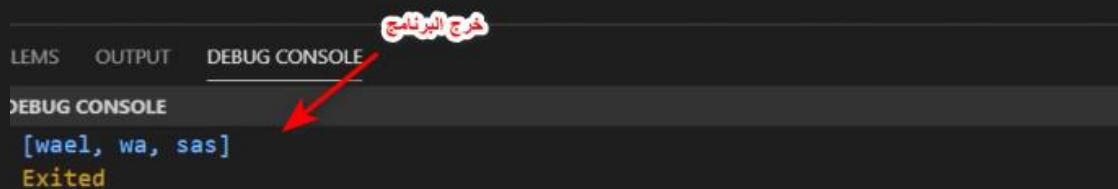
أنواع مختلفة من متغيرات

نستطيع تحديد نوع المتغير داخل list ولنفرض انه يكون من نوع string بهذه الحالة لا نستطيع الاكتابة قيمة من نوع string داخل list كما هو واضح في المثال في الأسفل :



```
2 List<String> names = ["wael" , "wa" , "sas" ] ;
3 print(names) ;
4 }
5 }
```

تحديد نوع المتغيرات داخل list



خروج البرنامج

DEBUG CONSOLE

```
[wael, wa, sas]
Exited
```

نلاحظ ان لدينا في هذا المثال list قمنا بتحديد القيمة التي يجب فقط ان تحويها .Map بداخلها وهي

بطريقة متقدمة اكثرا نستطيع تحديد نوع القيم داخل Map لكل من key و value

كما هو واضح في المثال في الأسفل ان كل من قيمة key و value هي من نوع : String

The screenshot shows a code editor with a Dart file named main.dart. The code defines a list of maps where both keys and values are strings:

```
void main() {
  List<Map<String, String>> names = [
    {"name": "wael", "age": "23"},
    {"name": "basel", "age": "55"}
  ];
  print(names);
}
```

Annotations in the code editor highlight the types: 'key' and 'value' are labeled 'تحديد نوع المصفوفة' (Type annotation), and the entire list is labeled 'جميع قيم المصفوفة من نوع String' (All values in the list are of type String).

مثل هذه الطريقة تصلح لكل من list و Map و Set كما هو واضح :

```

2 | Set<String> names = {"wael" , "basel"} ;
3 | print(names) ;
```

تحديد نوع القيمة من نوع
String

لا تقبل قيمة إلا من نوع
String

LEMS OUTPUT DEBUG CONSOLE خرج البرنامج

DEBUG CONSOLE {wael, basel}

Exited

(FirstWhere - AsMap - WhereType)

يقوم بجلب جزء نقوم بتحديده من list **Sublist**

تبدأ بي (Start)

وتنتهي بي (End) نقوم بتحديده ولكن لا يقوم البرنامج بطباعته

كما هو واضح في المثال في الأسفل :

```

void main() { 0   1   2   3
? List names = ["wael", "basel", "mohammad" , "ahmed"];
  List sublist = names.sublist(1 , 3) ;
  print(sublist) ;
```

Start End

أمر الطباعة

تقوم بجلب قيمة محددة من مصفوفة

LEMS OUTPUT DEBUG CONSOLE خرج البرنامج

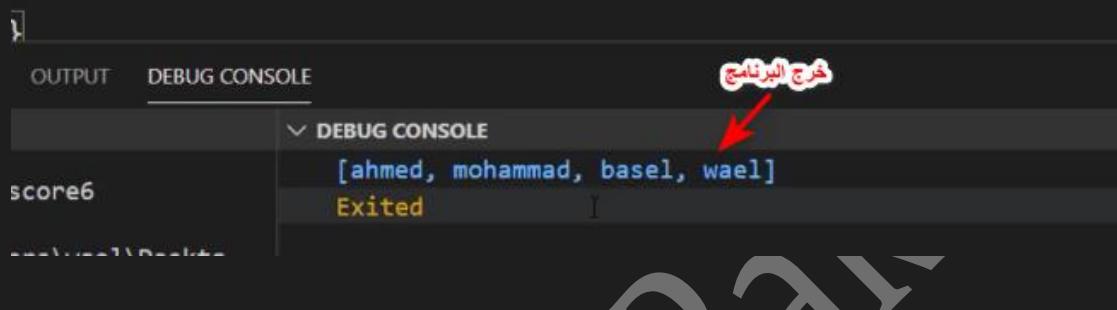
DEBUG CONSOLE [basel, mohammad]

Exited

يقوم ب إعادة ترتيب عناصر list في كل مرة نقوم ب إعادة تشغيل **Shuffle**

البرنامج كما هو واضح في المثال في الأسفل :

```
List names = ["wael", "basel", "mohammad" , "ahmed"];  
names.shuffle() ; |  
print(names) ;  
نقوم ب إعادة ترتيب عناصر المصفوفة
```



تحويل list الى Map كما هو واضح في المثال في الأسفل : **asMap**

```
List names = ["wael", "basel", "mohammad" , "ahmed"];  
Map namesmap = names.asMap() ;  
print(namesmap) ;  
امر الطباعة | المصفوفة | تحويل مصفوفة الى Map
```

```
OUTPUT DEBUG CONSOLE  
DEBUG CONSOLE  
pscore6 {0: wael, 1: basel, 2: mohammad, 3: ahmed}  
Exited
```

ارجاع مجموعة من قيم المحددة من list مع تحديد نوع هذه **whereType**

القيم سواء كانت int او string من مختلف أنواع المتغيرات كما هو ظاهر المثال في الأسفل :

```

void main() {
    List names = ["wael", 100, "basel", "mohammad", 22, "ahmed", 1002,
    var namesmap = names.whereType<String>();
    print(namesmap);
}

```

اسم list
ارجاع مجموعة قيم محددة من list
تحديد نوع القيمة

OUTPUT DEBUG CONSOLE

DEBUG CONSOLE

(wael, basel, mohammad, ahmed)

Exited

خروج البرنامج

يقوم بفحص اول قيمة محققة للشرط وتقوم بتخزينها في firstWhere المتغير الذي نقوم بتعريفه كما هو واضح في الصورة :

```

List names = [100, 2, 4, 6, 10];
var namesmap = names.firstWhere((n) => n > 10);
print(namesmap);

```

اسم List
يقوم بفحص اول قيمة محققة للشرط
return مختصرة
 الشرط

OUTPUT DEBUG CONSOLE

DEBUG CONSOLE

100

Exited

خروج البرنامج

نستطيع استخدامها أيضا مع متغيرات من نوع string كما هو واضح في المثال في الأسفل :

```

List names = ["wael", "basel", "g"];
var namesmap = names.firstWhere((n) => n.length > 4);
print(namesmap);

```

عدد الاحرف
 الشرط

OUTPUT DEBUG CONSOLE

DEBUG CONSOLE

basel

Exited

خروج البرنامج

Any

تعني أي قيمة محققة للشرط وترجع قيمة اما true او false في المثال

الموضح لدينا في الأسفل :

نريد تأكيد من ان أي قيمة في list عدد احرفها اكبر من ٣ وبتأكد فأن الشرط محقق ويكتفي واحد من عناصر list ان يكون متحقق للشرط حتى تقوم بارجاع

.true

```

List names = ["wawe" , "basel"];
var namesmap = names.any((e) => e.length > 3) ;
print(namesmap) ;

```

True
False

تقوم بارجاع قيمة
return
مختصرة

عدد الاحرف
الشرط

خرج البرنامج
true
Exited

Every

يجب ان يكون جميع عناصر list متحققة للشرط حتى تقوم بارجاع قيمة true والا ستقوم بارجاع قيمة false

لدينا المثال في الأسفل نريد التأكيد من ان جميع عناصر list عدد احرف كل عنصر فيها اكبر من ٥ عند التنفيذ سيقوم البرنامج بارجاع قيمة false لا الشرط غير متحقق لوجود كلمة تتألف من ٣ حروف فقط كما هو واضح في المثال :

```

List names = ["waw" , "basell"];
var namesmap = names.every((e) => e.length > 5 ) ;
print(namesmap) ;
}

```

تقوم بإرجاع قيمة true or false

الشرط

عدد الأحرف

OUTPUT DEBUG CONSOLE

خرج البرنامج

false
Exited

Take يقوم بإرجاع مجموعة محددة من عناصر مثلا نقوم بكتابة رقم 5

بداخلها تقوم بإرجاع اول 5 عناصر من list كما هو واضح في المثال في الأسفل :

```

List names = ["wael" , "basel" , 100 , "ahmed" , 23 ] ;
var namesmap = names.take(5).toList() ;
print(namesmap) ;

```

تقوم بإرجاع مجموعة من عناصر

عدد العناصر

المراد إرجاعها

تحويل إلى مصفوفة

OUTPUT DEBUG CONSOLE

خرج البرنامج

[wael, basel, 100, ahmed, 23]
Exited

Where تقوم بنفس عمل firstWhere ولكن الفرق بينهما ان

تقوم بإرجاع اول قيمة (قيمة واحدة فقط) من list محقق للشرط وتخزينها.

يقوم بـأرجاع مجموعة من القيم من list المحققة للشرط وتخزينها على هيئة Iterable .

كما هو واضح في المثال في الأسفل :

```

List names = ["wael" , "basel" , "ahmed" , "ahmed" ];
var namesmap = names.where((e) => e.length > 4);
print(namesmap);
}

OUTPUT DEBUG CONSOLE
DEBUG CONSOLE
(ed.
(basel, ahmed, ahmed)
Exited
new cross-platf
rShell https://

```

ارجاع اول قيمة محققة للشرط حيث يقوم بـأرجاع رقم القيمة المحققة للشرط من list .

بمعنى لدينا مثال في الأسفل الشرط ان يكون عدد احرف العنصر اكبر من 4 وعند فحص عناصر list واحد تلو الاخر نلاحظ ان اول قيمة محققة للشرط هي basel ورقمها يساوي 1 فيقوم البرنامج بطباعة الرقم 1 عند تنفيذ امر الطباعة .

```

void main() {
  0   1   2   3
List names = ["wael" , "basel" , "ahmed" , "ahmed" ];
var namesmap = names.indexWhere((e) => e.length > 4);
print(namesmap);
}

OUTPUT DEBUG CONSOLE
DEBUG CONSOLE
(ed.
1
Exited
new cross-platf
rShell https://

```

يقوم بفحص اول حرف من الكلمة في حال كان محقق يرجع **startsWith**

قيمة true واذا كلن غير محقق يرجع قيمة false.

هل الكلمة wael تبدأ بحرف W ..؟ اذن سيقوم بأرجاع قيمة true

```
String name = "wael" ;  
print(name.startsWith("w")) ;  
}
```

يقوم بارجاع قيمة
True or false

AS OUTPUT DEBUG CONSOLE

AL DEBUG CONSOLE خرج البرنامج
true
Exited

يقوم بفحص اخر حرف من الكلمة في حال كان متحقق يرجع **EndsWith**

قيمة true واذا كلن غير متحقق يرجع قيمة false.

هل الكلمة basel تنتهي بحرف L ..؟ اذن سيقوم بأرجاع قيمة false

```
String name = "basel" ;  
print(name.endsWith("l")) ;  
}
```

ارجاع قيمة
true or false

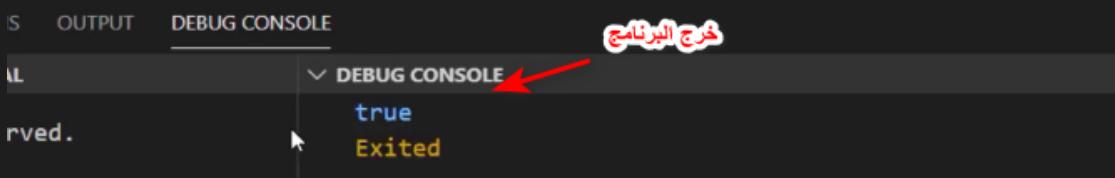
AS OUTPUT DEBUG CONSOLE

AL DEBUG CONSOLE خرج البرنامج
false
Exited

يقوم بفحص كل حرف من الكلمة في حال كان محقق يرجع قيمة **Contains**
 .false واذا كلن غير متحقق يرجع قيمة true

هل الكلمة wael يوجد فيها حرف a ..؟ اذن سيقوم بأرجاع قيمة true

```
String name = "basel" ;
print(name.contains("a")) ;
```



يقوم بفحص كل حرف من الكلمة وارجاع قيمة موقع الحرف بكلمة ونستطيع تحديد قيمة للبدأ منها.

كلمة baselle يوجد فيها حرفين (e) ولكن قمنا بتحديد قيمة للبدأ منها وهي 2
 سيقوم الحاسوب بتجاهل اول ثلاثة احرف من الكلمة كما هو واضح في المثال في الأسفل :

```
String name = "beaslle" ;
print(name.indexOf("e", 2)) ;
```

الحروف بالكلمة

ارجاع قيمة موقع

ابدا التتحقق من

الحرف الثاني

هذه الاخرت

ستقوم الحاسوب بتجاهله



Iterator And Iterable

Iterable هو مفهوم عام يشمل جميع المتغيرات التي نستطيع عمل تكرار عليها ومن ضمنها Set و List و Iterable هو الذي يمكن ان نعمل عليه iterator هنا تعني التكرار (تشمل جميع أنواع الحلقات التكرارية For او while).

```
List names = ["wael" , "ahmed" , "Mohammad"] ;  
for (String name in names ) {  
    print(name) ;  
}  
S OUTPUT DEBUG CONSOLE  
L PowerShell خرج البرنامج  
ght (C) Microsoft Corporation. All rights reserved.  
new cross-platform PowerShell https://aka.ms/pscore6  
wael  
ahmed  
Mohammad  
Exited
```

ميثود التكرار ليكن لدينا متغير من نوع iterator نسميه iter ثم نقوم بكتابة حلقة تكرارية من نوع while ونقوم بكتابة الشرط بداخلها والذي يتضمن المتغير true الذي سنقوم بعمل moveNext() عليه الذي بدورة يقوم بإرجاع اما true او false وسيبقى الشرط محقق حتى يصل لآخر عنصر يقوم بطباعته ليقوم بأعادة الفحص فيرجع قيمة false لانه قام بطباعة جميع عناصر list لا يوجد عنصر تالي ومن ثم يخرج البرنامج من الحلقة while كما هو واضح في المثال في الأسفل :

```
List names = ["wael" , "ahmed" , "Mohmmad"] ;  
        ↑ نوع المتغير  
        ↑ صناعة التكرار  
Iterator iter = names.iterator ;  
        ↑ يقوم بإرجاع قيمة  
        ↑ true or false  
while(iter.MoveNext()){  
    print(iter.current) ;  
}  
        ↑ أمر الطباعة
```

S OUTPUT DEBUG CONSOLE

IL

```
s PowerShell  
ght (C) Microsoft Corporation. All rights reserved.  
e new cross-platform PowerShell https://aka.ms/pscore6
```

DEBUG CONSOLE

```
wael  
ahmed  
Mohmmad  
Exited
```

خرج البرنامج

Map Method

تقوم بعمل iterator على list او Map او Set و تقوم بتخزين عناصرها في متغير.

.Iterable تقوم بأرجاع القيم على هيئة Map Method
كما هو واضح في المثال في الأسفل طباعة طول الكلمات في list :

The screenshot shows a debugger interface with a code editor and a debug console. The code in the editor is:

```
List names = ['wael' , 'basel' , "mohammad"] ;
var lengthElement = names.map((e) => e.length) ;
print(lengthElement) ;
```

Annotations in the code editor explain the process:

- اسم المتغير نوع المتغير : lengthElement
- (أمر الطباعة) : print(lengthElement) ;
- يقوم بعمل iterator على list : names.map((e) => e.length)
- طول الكلمات في المصفوفة : e.length

The debug console shows the output:

```
VS OUTPUT DEBUG CONSOLE
NAL
WS PowerShell
ight (C) Microsoft Corporation. All rights reserved.

DEBUG CONSOLE
(4, 5, 8)
Exited
```

An annotation points to the output with the text: خرج البرنامج على هيئة Iterable

وأيضا طباعة الكلمات وتحويلها الى list كما هو واضح في المثال في الأسفل :

The screenshot shows a debugger interface with a code editor and a debug console. The code in the editor is:

```
List names = ['wael' , 'basel' , "mohammad"] ;
var lengthElement = names.map((e) => e).toList() ;
print(lengthElement) ;
```

Annotations in the code editor explain the process:

- يقوم بعمل iterator على list : names.map((e) => e)
- return المختصرة : () => e
- عناصر المصفوفة : e
- تحويل الى مصفوفة : .toList()

The debug console shows the output:

```
VS OUTPUT DEBUG CONSOLE
NAL
WS PowerShell
ight (C) Microsoft Corporation. All rights reserved.

DEBUG CONSOLE
[wael, basel, mohammad]
Exited
```

An annotation points to the output with the text: خرج البرنامج

نستطيع عمل الكثير من العمليات على Map Method كمثال استخدام الدالة if الشرطية داخلها كما هو واضح في المثال في الأسفل :

```

3 List names = ['wael' , 'basel' , "mohammad"] ;
4
5 var lengthElement = names.map((e) [
6   if (e == "wael") { ← الشرط
7     return "Yes Wael Exist" ;
8   }
9   return "No" ;
10 ])
11 print(lengthElement) ;
12

```

ارجاع القيمة على هيئة Iterable

خرج المخرج على هيئة Iterable

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
the new cross-platform PowerShell https://aka.ms/pscore6

PROBLEMS OUTPUT DEBUG CONSOLE

MINIMAL DEBUG CONSOLE

(Yes Wael Exist, No, No)
Exited

العملية نفسها تطبق على Map نريد استخراج قيم key من Map سيقوم البرنامج بأرجاعها على هيئة Iterable نقوم بكتابة امر تحويل الى List لان المتغير الذي نقوم بتخزين القيم داخله هو من نوع list في حال لم نقم بكتابة امر تحويل سيقوم البرنامج بأظهار رسالة خطأ.

```

3 Map info = {"name" : "wael" , "age" : 12 , "phone" : 243531} ;
4   Key Value
5
6 List infokey = info.entries.map((e) => e.key).toList() ; ← ارجاع قيمة Key
7
8 print(infokey) ;
9
10
11 }
12

```

ارجاع قيمة Key

خرج المخرج

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PROBLEMS OUTPUT DEBUG CONSOLE

TERMINAL DEBUG CONSOLE

[name, age, phone]
Exited

Try And Catch

معناها ان يكون لدينا كود من المتوقع حدوث خطأ فيه نضعه داخل try

مثال على هذا لدينا المثال في الأسفل نريد تقسيم عدد صحيح على صفر بلتأكيد سيقوم بإرجاع خطأ ويتوقف البرنامج عن العمل لأن التقسيم على صفر لا نهاية.

نقوم بوضع عملية القسمة وطباعتها ضمن try ومن ثم بكتابة Catch ونضع فيه بارامتر لنجعل بتخزين قيمة الخطأ فيه ثم نقوم بطباعة هذا البارامتر كما هو واضح لدينا في المثال في الأسفل :

```
int a = 10 ;
int b = 0 ;
try{
    int res = a ~/ b ;
    print(res) ;
} catch(e) {
    print("Error : $e") ;
}
```

الخطأ هنا لا يمكن التقسيم على صفر

Exit

shorthand if

الدالة الشرطية (المختصرة) : تقوم بنفس عمل الدالة الشرطسة تماما ولكن تختلف عنها بطريقة الكتابة انها مختصرة جدا كما هو واضح في المثال في الأسفل

```
int a = 2;
a > 5 ? print("a > 5") : print("a < 5");
    ↑       ↑       ↑       ↑       ↑
    الشرط   if     في حال تتحقق الشرط
              تنفيذ الأمر الثاني
    else      في حال عدم تتحقق الشرط
              الشرط الأول
```

```
PowerShell
: (C) Microsoft Corporation. All rights reserved.
```

خروج البرنامج

a < 5
Exited

وبليتأكيد يكننا كتابة اكثـر من شـرط كما هو واضح في المثال في الأسـفل :

```
String name = "basel";
(name == "wael") || ((name == "basel")) ? print("name equal wael Or Basel") : print("name");
}
    ↑       ↑       ↑       ↑       ↑
    الشرط الأول   او   الشرط الثاني   if   else
```

```
PowerShell
: (C) Microsoft Corporation. All rights reserved.
```

خروج البرنامج

name equal wael Or Basel
Exited

OPP

برمجة كائنية التوجه : هي مخطط يساعدنا على تنظيم العمل.

عندما نقوم بتعريف كلاس من ضروري ان نقوم بت كتابة اول حرف كبير من اسم الكلاس.

ماذا يمكننا كتابة داخل كلاس :

Variable

method

Constructors

Set & Get

يمكننا عمل نموذج من الكلاس وكما سبق وذكرنا ان الكلاس هو عبارة عن مخطط عمل يجب الالتزام بكل ما يوجد فيه كما هو واضح :

```
3  Mobile mobile = new Mobile() ;
4  mobile.screen = "6.4";
5  print(mobile.screen) ;
6
7
8 }
9
10 class Mobile {
11   // Var
12   String screen ;
13   String name ;
```

نموذج عن الكلاس

تعريف الكلاس

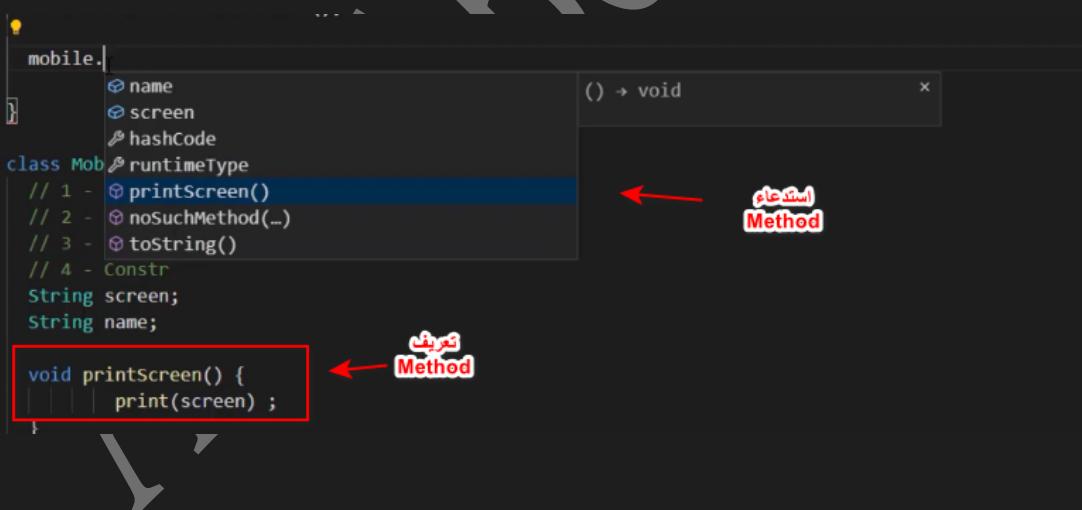
يمكننا أيضاً تعين قيمة او إعادة تعين قيمة للمتغير من خلال النموذج كما هو واضح في الصورة في الأسفل :

```
Mobile mobile = new Mobile();  
mobile.screen = "6.4";  
print(mobile.screen);  
}  
  
class Mobile {  
    // Var  
    String screen;  
    String name;
```

يمكننا تعين قيمة من خلال النموذج

عندما نريد الوصول الى مكونات الكلاس يجب بناء نموذج عنه حتى نستطيع الوصول اليه.

نفس الامر لل method يمكننا كتابتها في كلاس واستدعائها في نموذج كما هو واضح في الصورة في الأسفل :



The screenshot shows an IDE interface with code completion suggestions for the variable 'mobile'. The suggestion 'printScreen()' is highlighted in blue, indicating it is the selected method. A tooltip labeled 'استدعاء Method' (Invocation Method) is shown above the suggestion. Another tooltip labeled 'تعريف Method' (Method Definition) is shown below the code completion bar, pointing to the definition of the 'printScreen()' method in the 'Mobile' class. The code completion bar also shows other methods like 'name', 'screen', and 'hashCode'.

```
mobile.  
    name  
    screen  
    hashCode  
class Mobile {  
    runtimeType  
    // 1 - printScreen()  
    // 2 - noSuchMethod(...)  
    // 3 - toString()  
    // 4 - Constr  
    String screen;  
    String name;  
    void printScreen() {  
        print(screen);  
    }  
}
```

Reset Variable

طرق إعادة تعيين قيمة للمتغير ضمن الكلاس.

: يوجد عدة طرق لإعادة تعيين :

User.name = " value"

cascade opreator

Set & Get

Constractor

الطريقة الأولى هي عن طريق كتابة اسم النموذج ثم كتابة اسم المتغير المعرف مسبقاً في الكلاس ثم إسناد قيمة له كما هو واضح في الصورة في الأسفل :

```
class Mobile {  
    // 1 - var  
    // 2 - method  
    // 3 - set And Get  
    // 4 - Constr  
  
    String name ;
```

تعريف متغير

نموذج عن كلاس

```
Mobile mobile = new Mobile();  
mobile.name = "wael" ;  
print(mobile.name) ;
```

إعادة تعيين قيمة

الطريقة الثانية عن طريق cascade operator

بكتابة نقطتين قبل نموذج الكلاس ثم اسم المتغير ثم اسناد القيمة الجديدة له كما هو واضح في الصورة في الأسفل :

```
User user = new User()..name = "wael";  
//1 - user.name = "wael";  
print(user.name);
```

cascade operator

اعادة تعيين قيمة

الطريقة الثالثة وهي Constractor

نستطيع إعادة تعيين قيمة متغير من تمرير قيمته من خلال Constractor كما هو واضح في الصورة في الأسفل :

```
User(val){  
    name = val ;  
}  
  
User user = new User("wael");  
//1 - user.name = "wael" ;  
// 2 - cascade Opreator  
// 3 - Constr  
print(user.name);
```

Constractor

تمرير قيمة المتغير من خلال
Constractor

يوجد طريقة مختصرة لاعادة التعيين عن طريق Constractor عن طريق this يمكن استخدامها مع method أيضا وتكون عائدية على الكلاس الموجودة ضمنها (او انها تشير الى الكلاس الموجود فيه حاليا) كما هو واضح في الصورة في الأسفل :

```
User(this.name) ;
```

Constractor

AdelAbobaker

Setter And Getter

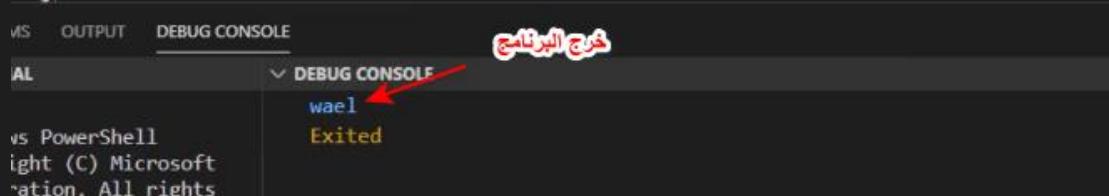
Set & Get هي طريقة الـslam لــtting قيمة متغير او جلب قيمة متغير من كلاس معين بالإضافة للميزات الإضافية التي تتمتع بها.

ـtting قيمة متغير من داخل الكلاس.

كما هو واضح في المثال في الأسفل قمنا بــtting قيمة المتغير من خلال Set

```
User user = new User();
user.changeUsername = "wael" ;
print(user.username) ;
```

ـtting قيمة



قمنا قــim method من نوع void لا ترجع قيمة وسميناها newname وقمنا بــt入 بــarmenter داخــle اسمــinaها changeUsername لتــخــzin قيمة المتــغــير داخــle ثم قــim بــtــdidiــfــ المتــغــير المــوجــod ضــمــنــ كــلاــs عن طــرــيقــ كما هو واضح في المثال في الأسفل :

```
void set changeUsername(newname){
    this.username = newname ;
}
```

ــarmenter

ــtــr لــجــلــb قيمة متــغــir ما من داخل الكــلاــs.

```

String get newname{
    return username ;
}

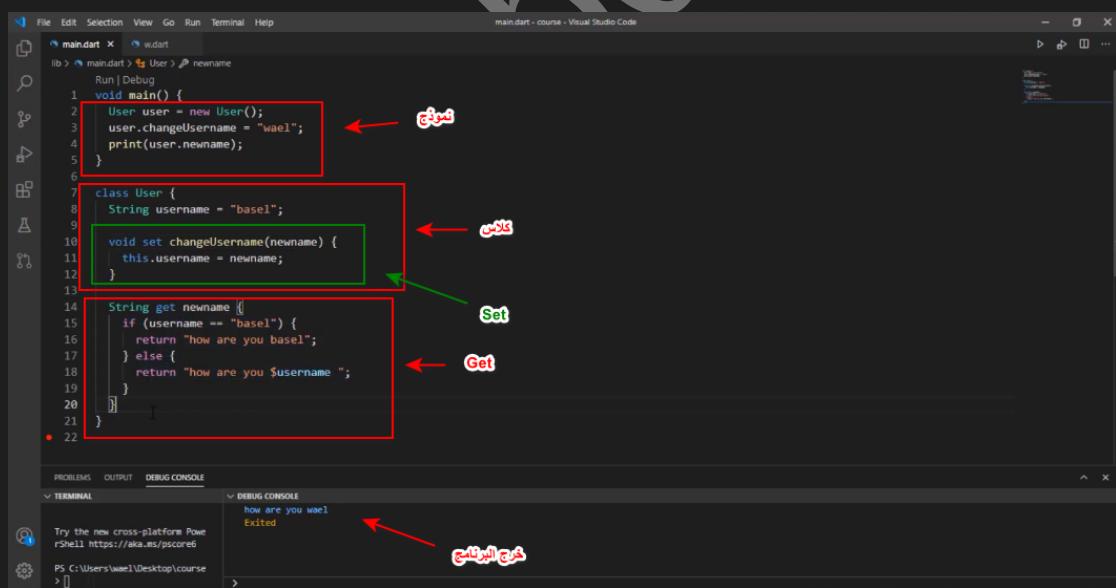
```

نستطيع استخدام Get method مع كمت هو واضح في المثال في الأسفل :

```

String get newname{
    if (username == "basel"){
        return "how are you basel" ;
    }else {
        return "how are you $username " ;
    }
}

```



Static

هي قيمة ثابتة لا يمكن الوصول اليها الا من خلال الكلاس نفسه لا نستطيع الوصول اليها عن طريق `instance` النموذج يتم ذلك عن طريق طباعة اسم الكلاس ثم تنفيذ الامر المطلوب سواء كان إعادة تعيين قيمة او امر طباعة كام هو واضح في الافضل يف الأسف :

The screenshot shows an IDE interface with Java code. The code defines a class `User` with a static variable `username` set to "wael". Two instances of `User` are created: `userone` and `usertwo`. The static variable `username` is then assigned the value "basel" for both instances. The output console shows two "basel" entries, indicating that both objects share the same static variable.

```
1 User userone = new User();
2 User usertwo = new User();
3 // userone.username = "basel";           إعادة تعيين قيمة
4 User.username = "basel" ;               ←
5 print(User.username);
6 // usertwo.username = "basel";
7 print(User.username);
8 }
9
10 class User {
11     static String username = "wael";
12 }
```

الكلمات المميزة في الشاشة:

- قيمة ثابتة لا يمكن الوصول اليها الا عن طريق الكلاس نفسه
- خريطة البرنام

وكل نعلم انه من الممكن انشاء عدة `instance` وكل واحد منهم نستطيع إعطائه قيم مختلفة عن الآخر ولكن في حال وجود `Static` سيتشارك جميع `instance` نفس قيمة المتغير الموجود ضمن الكلاس.

كما هو واضح في المثال في الأسفل :

```
2 |     User userone = new User(); ← instance one
3 |     User usertwo = new User(); ← instance two
4 |     userone.username = "basel";
5 |     print(userone.username);
6 |     print(usertwo.username);
7 |
8 |
9 | class User {
10 |     String username = "wael";
11 | }
12 |
```

الصورة توضح الكود و выводه في конسول الديباغ ، حيث نلاحظ أن هناك اثنين من الاكتشافات (instances) ، الأولى تحمل اسم `instance one` والثانية تحمل اسم `instance two` .

نستطيع الوصول لقيمة Static عن طريق Get مهما قمنا بتعريف instance سيقوم بأخذ القيمة مشتركة من متغير `username` كما هو

واضح في المثال في الأسفل :

```
1 |     User userone = new User();
2 |     User usertwo = new User();
3 |     User.username = "basel" ; ← instance two
4 |     print(userone.myname);
5 |     print(usertwo.myname);
6 |
7 |
8 | class User {
9 |     static String username = "wael";
10|     String get myname { ← Get
11|         return username;
12|     }
13| }
```

الصورة توضح الكود و نلاحظ أن المتغير `myname` يأخذ قيمة `username` من المثلثة `User` .

Cascade operator

عادة عندما نقوم بـ**استدعاء method** معينة من داخل كلاس معين فأننا نقوم بتعريف اسم **instance** ثم كتابة اسم **instance** ثم كتابة اسم الميثود المراد استدعائهما من الكلاس كما هو واضح في المثال في الأسفل :

The screenshot shows a Java code editor and a debug console. The code editor has a green box highlighting the following lines:

```
3 Mobile mob = new Mobile() ;
4     mob.testone() ;
5     mob.testtwo() ;
6 }
7
8 class Mobile {
```

A green arrow points from the word "instance" in the explanatory text above to the variable "mob" in the highlighted code. The debug console below shows the output:

PROBLEMS OUTPUT DEBUG CONSOLE

MINAL DEBUG CONSOLE خرج البرنامج

```
the new cross-platf
PowerShell https://
.ms/pSCORE6
```

method one
method two
Exited

A red arrow points from the text "خرج البرنامج" (Program exited) in the debug console back to the explanatory text above.

Below the code editor, there is a large stylized logo of the letters "AL".

void testone() {
 print("method one") ;
}
void testtwo() {
 print("method two") ;
}

ولكن يوجد طريقة مختصرة من خلالها نستطيع استدعاء method من كلاس معين عن طريق cascade operator كما هو واضح في المثال في الأسفل :

```

new Mobile()..testone()..testtwo();
}
instance
class Mobile {
    void testone() {
        ...
    }
}

void testtwo() {
    ...
}

IS OUTPUT DEBUG CONSOLE
AL DEBUG CONSOLE
e new cross-platf
werShell https://
/pscore6
method one
method two
Exited
Cascade operator

```

Constructor

او ما يسمى التابع الباقي : يتم استدعائه مباشرة (بشكل تلقائي) عند تعريف instance من كلاس معينة يتم تعريف التابع الباقي بكتابة اسم الكلاس ثم فتح قوسين وإدخال البرمترات كما هو واضح لدينا في المثال في الأسفل :

```

Mobile mobile = new Mobile.wael("wael");
}

class Mobile {
    name Constructor
    Mobile.wael(screen){
        print(screen) ;
    }
}

Constructor ( التابع الباقي )
IS 1 OUTPUT DEBUG CONSOLE
AL DEBUG CONSOLE
e new cross-platf
werShell https://aka.ms/pscore6
wael
Exited
خرج البرنامج

```

inheritance

الوراثة : هو ان يتم توريث جميع محتويات كلاس معين لكلas اخر .

ليكن لدينا كلاس mobile يحوي على عناصر أساسية مثل cpu و camera و memory و screen و قلنا بعمل كلاس اخر سميته Samsung واردنا توريث جميع محتويات كلاس mobile الى كلاس Samsung كما هو واضح لدينا في المثال في الأسفل :

```
7 class Mobile {  
8     String screen ;  
9     String camera ;  
10    String cpu ;  
11    String memory ;  
12 }  
  
13 class Samsung extends Mobile {  
14 }  
  
المورث  
محفوظات  
الكلasse  
توريث
```

نلاحظ انه تم توريث جميع محتويات كلاس mobile لクラス Samsung كما هو واضح لدينا في الصورة :

```
Samsung s20 = new Samsung();           ← instance class Samsung
print(s20.)
}
    ↗ camera
    ↗ cpu
    ↗ memory
    ↗ screen
وراثة جیب محتويات کلاس
Mobile
```

Override

المقصود منها هي إعادة تعيين لمتغير او لميثود معينة داخل كلاس.

ويف حال اردنا إعادة تعيين لقيمة متغير ما يجب الالتزام بنوع المتغير في حال كان نفس الاسم كما هو واضح لدينا في الأسفل :

```
class Mobile {  
    String screen = "6.3";  
}
```

كلاس الاب

```
class Samsung extends Mobile {  
    String screen = "6.5" ;  
}
```

اعادة تعيين

في حال قمنا بتغيير نوع المتغير لن يعمل البرنامج وسيقوم بأظهار رسالة خطأ كما هو واضح لدينا في المثال في الأسفل :

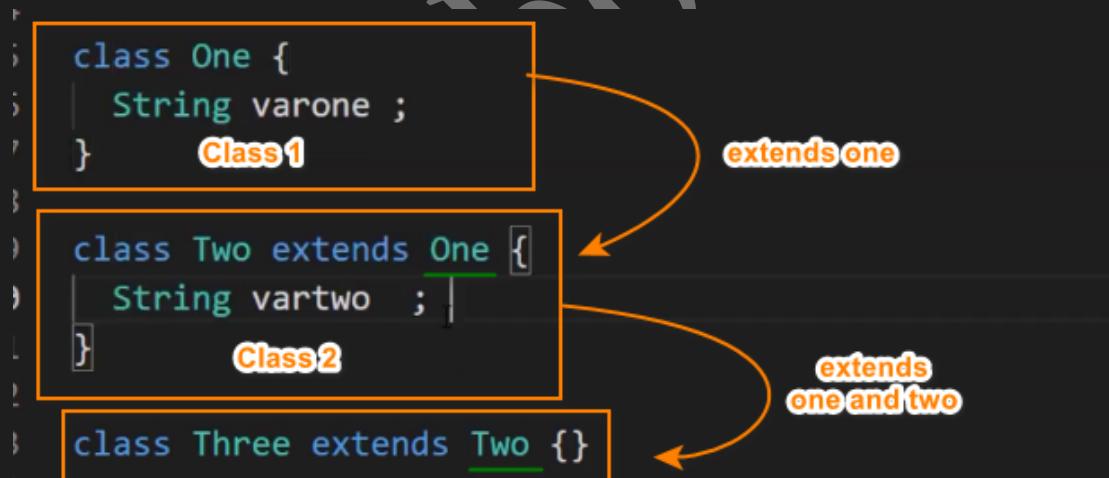
```
7 class Mobile {  
8     String screen = "6.3";  
9 }  
10  
11 class Samsung extends Mobile {  
12     double screen = "6.5" ;  
13 }  
14
```

Mult-level inheritance

مستويات الوراثة : تعرفنا في الدروس السابقة ان وراثة هي وراثة كلاس من كلاس اخر ولكن الوراثة لها عدة مستويات.

في هذا المثال قمنا بتعريف ثلاث كلاسات وقمنا بتعريف متغيرين في الكلاس class one و class two و class three الى class one و class two و class three

كما هو واضح في الصورة في الأسفل :



: class three فأن class two ورث جميع محتويات class one

```
2 three th = new Three() ; ← instance three
3 th.| ←
4 } ←
5     vartwo ←
6     varone ←
7     cla ←
8     hashCode ←
9     S ←
10    runtimeType ←
11 } ←
12     noSuchMethod(...) ←
13     toString() ←
14 class Two extends One { ←
```

الوراثة من
Class one & Class two

Super Class

تعني الإشارة الى الكلاس الرئيسي (الاب) كما هو واضح لدينا في المثال في الأسفل :

```
6  class One {  
7 |     String varone= "one" ;  
8 }  
9  class Two extends One {  
10 |     String get getvar {  
11 |         return super. ;  
12 |     }  
13 }  
14  
PROBLEMS 2 OUTPUT DEBUG CONSOLE RMINAL ms/pscore6
```

الإشارة الى الكلاس الاب
الإشارة الى الكلاس الاب
وراثة
الكلام الاب
الكلام الاب
متغيرات
Class one
one
Exited

اذا كان الكلاس الرئيسي (الاب) يحوي على Constructor فيجب عندها ان يكون الكلاس الابن يحوي على Constructor وفي حال كنا نستخدم نفس الاسم يجب كتابة : ثم ("name Constructor") كما هو واضح لدينا في الامثل في الأسفل :

```
void main() {  
    Two two = new Two("wael") ;  
    print(two.username) ; instance two  
}  
1  class One {  
2 |     final username ;  
3 |     One(this.username) ;  
4 }  
5  class Two extends One {  
6 |     Two(username) : super(username) ;  
7 }  
LEMS OUTPUT Constructor one Constructor two DEBUG CONSOLE basel Exited
```

الإشارة الى الكلاس الرئيسي
Constructor one
Constructor two
خروج البرنامج

Abstract class

بساطة مهمته فقط التوريث ولا يمكن عمل instance منه

عند محاولتنا تعريف instance من class Mobile كما هو واضح لدينا في المثال في الأسفل :

```
2 |     Mobile mob = new Mobile() ;           ← instance class
3 |     print(mob.brand) ;
4 |
5
6 abstract class Mobile {
7     String brand = "samsung" ;
8 }
9
```

```
5
6 abstract class Mobile {
7     String      brand = "unknwon" ;
8 }
9 class Samsung extends Mobile {
10    String      brand = "samsung" ;
11 }
12
```

← abstract class

← abstract class

كلasse ورث

implements class

هو الذي يجبر الكلاس الاب ان يرث جميع خواص الكلاس الاب بمعنى أي ان جميع الخواص الموجودة في الكلاس الرئيسي (الاب) يجب استعمالها في الكلاس الاب كما هو واضح في المثال في الأسفل ظهر خطأ لعدم وراثة الكلاس الاب (جميع) خواص الكلاس الرئيسي :

```
6 class Mobile {  
7     String brand = "unknwon";  
8     String screen ; |  
9 }  
  
10  
11 class Samsung implements Mobile {  
12     String brand ; ← ظهور خطأ  
13 }  
14
```

الكلasse الرئيسي

implementes class

يتم تعيين القيم في الكلاس الاب أي ان implements تمنع اخذ القيم من الكلاس الرئيسي (الاب).

أي انه عند كتابة الكلاس الرئيسي لا داعي من تعيين القيم.

```
main.dart > Samsung  
5 class Mobile {  
6     String brand ;  
7     String screen ; ← كلام الرئيسي  
8 }  
9  
10  
11 class Samsung implements Mobile {  
12     String brand = "samsung" ; ← تعيين قيم للمتغيرات في كلام الاب  
13     String screen = "6.3" ;  
14 }
```

خرج البرنامج

DEBUG CONSOLE

FINAL :/nscore6 DEBUG CONSOLE samsung

لا مشكلة من إضافة Properties الى الكلاس الابن في حال وجود implements ولكن في حال إضافة جديدة الى الكلاس الرئيسي يجب اضافتها الى الكلاس الابن لتفادي حدوث خطأ في البرنامج.

```
class Mobile {  
    String brand = "unknwon";  
    String screen ;  
}  
  
class Samsung implements Mobile {  
    String brand ;  
    String screen ;  
    String t ;  new Properties  
}  
  

```

يجب الانتباه لاسماء المتغيرات لأن implements حساس للمحارات.

```
class Mobile {  
    String brand = "unknwon";  
    String screen ;  
}  
  
class Samsung implements Mobile {  
    String brand ;  
    String screenw ;   implements حساس للمحارات  
}
```

Collection

بلدروس السابقة اذا اردنا تعريف list نقوم بكتابه نوع + اسم list الان بعد التعرف على OPP نستطيع تعريف list او set او map بصيغة instance كما هو واضح لدينا في المثال في الأسفل :

```
1 void main() {  
2     List names = new List(); | ← تعريف مصفوفة فارغة  
3     names.add("wael"); | ← اضافة قيم للمصفوفة  
4     names.add("basel"); |  
5     print(names); |  
6 } |  
7 }
```



نستطيع كتابة instance بدون استدعاء اسم الكلاس وسيتم تعرف اخذ النوع تلقائيا على حسب قيمة ولا يشترط كتابة new عند تعريف instance من كلاس معين كما هو واضح لدينا في المثال في الأسفل :

خروج البرنامج

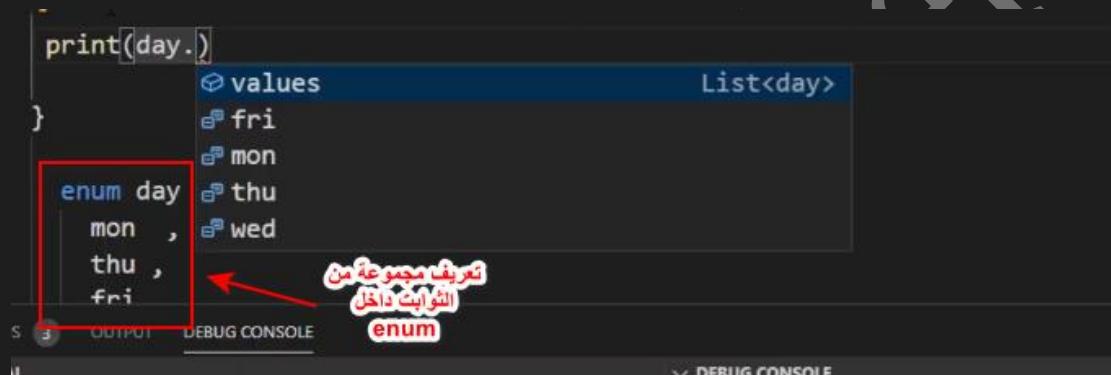
INAL
Windows PowerShell
right (C) Microsoft Corporation. All rights reserved.
[wael, basel]
Exited

```
var mob = Mobile();  
|  
|← class Mobile  
|← new  
|  
|← instance
```

Enum

المقصود فيها نستطيع تعريف مجموعة من ثوابت في برنامجنا نستطيع استخدامها في برنامجنا بشكل دائم وثابت.

قمنا بتعريف أيام الأسبوع داخل enum كما هو واضح لدينا في المثال في الأسفل :



The screenshot shows a debugger interface with a code editor and a debug console. In the code editor, there is an enum definition:

```
print(day.)  
}  
  
enum day {  
    mon ,  
    thu ,  
    fri  
}
```

A tooltip appears over the enum keyword with the text "تعريف مجموعة من الثوابت داخل enum". The debug console shows the output of the print statement:

```
values  
List<day>  
fri  
mon  
thu  
wed
```

Below the code editor, the debug console shows the output of the print(day.values); statement:

```
print(day.values) ;  
}  
  
enum day {  
    mon ,  
    thu ,  
    fri  
}
```

The debug console output is:

```
PowerShell  
ht (C) Microsoft Corporation. All rights reserved.  
[day.mon, day.thu, day.fri, day.wed]  
Exited
```

A red arrow points from the tooltip to the enum keyword in the code editor. Another red arrow points from the word "Exit" in the debug console output to the text "خرج البرنامج" (Program exited).

مثال اخر عن enum قمنا بتعريف الثوابت التالية :

```
enum Status {  
    online ,  
    offline  
}
```

مجموعة من الثوابت

واردنا كتابة برنامج لمعرفة الشخص اذا كان متصل الان او غير متصل الان

(المثال اقرب لتطبيق الدردشة واتس اب)

٢١

```
1 void main(){  
2     enum name الوظيفة الحالية متصل الان  
3     Status statuscurrent = Status.online;  
4     if (statuscurrent == Status.offline) {  
5         print(" الشخص غير متصل"  
6     }else {  
7         print(" الشخص متصل"  
8     }  
9 }  
10
```

BLEMS OUTPUT DEBUG CONSOLE

VINAL

dows PowerShell
yright (C) Microsoft Corporation. All rights reserved.

DEBUG CONSOLE

الشخص متصل
Exited

خروج البرنامج

Regular expressions

مهمته التحقق من المدخلات وترجع قيمة اما `true` او `false`.

regular expressions هو بحد ذاته علم ليس مطلوب منك انت كمبرمج ان تعمق فيه انما تقوم بجلب ال pattern جاهزة من على الانترنت في الأمثلة القادمة ستعلم ما معنى هذا المصطلح..!!

على سبيل المثال عند تسجيل دخول الى حسابك على الفيسبوك يطلب منك الموقع ادخال اسم المستخدم وكلمة المرور مثلا..!!

وунدها نتوجه للحقل البريد الالكتروني وقمنا بادخال معطيات خطأ مثلا قمنا كتابة كلمة (ahmad) قمنا بادخال مجموعة من الأرقام او الإشارات عند النقر على الزر تسجيل الدخول سيقوم الموقع بأظهار رسالة خطأ مفادها ان (البريد الالكتروني غير صالح).

من هنا نطرح سؤال كيف عرف الموقع ان البريد غير صالح ..؟؟

لكتابة البريد الالكتروني لابد من مجموعة من القواعد والخصائص التي يحتويها ليكون بريد الكتروني نقوم ببرمجتها ليقوم الحاسوب بتعرف عليه على انه بريد الكتروني كما هو واضح لدينا في المثال يف الأسفل :



ومن هنا تأتي أهمية regular expressions وهي التحقق من المدخلات وسنقوم بطرح مثال عنها بشكل مفصل.

لدينا المثال الذي بأسفل نريد التحقق من ان المدخل فعلا هو بريد الكتروني

---الخطوة الأولى قمنا بجلب patternEmail من الانترنت جاهزة.

--الخطوة الثانية قمنا بـاستدعاء instance من كلاس RegExp في لغة dart.

--الخطوة الثالثة قمنا بتعريف متغير لتخزين القيمة داخله لتحقق منها.

-- الخطوة الأخيرة قمنا بكتابة امر الطباعة داخل instance RegExp مع

وعند تشغيل البرنامج قام البرنامج بأرجاع قيمة true لأن البريد الإلكتروني جميع عناصره متوفّر فيه كما هو واضح لدينا في المثال في الأسفل :

```
void main(){
  String patternEmail = r'^(([^\<@\>().\\]\\.\\.,;:\\s@\\"]+\\(.[^<@\>().\\]\\.\\.,;:\\s@\\"]+\\*)$';
  RegExp regExp = new RegExp(patternEmail);
  String email = "wael@gmail.net";
  print(regExp.hasMatch(email));
}

// Output
true
```

لحظة التحقق من البريد الإلكتروني

لو قمنا بادخال أي قيمة بشكل عشوائي وتحققنا منها سيقوم البرنامج بأرجاع قيمة
كما هو واضح لدينا في الأسفل :

```
String email = "wael" ;  
print(regExp.hasMatch(email)) ;
```

}

خرج البرنامج

AS OUTPUT DEBUG CONSOLE
AL
is PowerShell
ight (C) Microsoft Corporation. All rights reserved.

▼ DEBUG CONSOLE
false
Exited

مثال اخر لتحقق من ادخال رقم كما هو واضح لدينا في الأسفل :

```
String patternPhone = r'(^(?:[+0]9)?[0-9]{10,12}$)';  
RegExp regExp = new RegExp(patternPhone) ;  
String phone = "+96394223545" ;  
print(regExp.hasMatch(phone)) ;
```

ادخل رقم تتحقق منه

patternPhone

MS OUTPUT DEBUG CONSOLE
IAL
ws PowerShell
ight (C) Microsoft Corporation. All rights reserved.

▼ DEBUG CONSOLE
true
Exited

Private

نقوم باستخدامها داخل class من أجل ان تكون

Variable او Constructor او method خاص بلكلas نفسو ولا يمكن استخدامها بي أخرى عكس الخاص public هو page .

قمنا بتعريف كلاس بصفحة وتعريف متغيرين بداخله وعمل المتغير email من خلال وضع اشاش (_) قبل اسم المتغير واستدعاء instance من Private صفحة أخرى تنفيذ امر الطباعة للمتغير email سوف نلاحظ وجود خطأ كما هو واضح لدينا في الأسفل :

```
main.dart
lib > main.dart > main
1 import 'two.dart';
2 Run | Debug
3
4 User user = new User();
5
6 print(user.email); ← خط لا يمكن
7
8

two.dart
lib > two.dart > User > _email
1 class User {
2   var username = "wael";
3   var email = 'wael@gmail.com';
4 }

page one
page two
Private
```

لو قمنا بتمرير المتغير الخاص email public داخل method يقوم بتنفيذ امر الطباعة من دون أي مشاكل أي اننا نستطيع الوصول الوصول الى المتغير ال public من خلال Private method كما هو واضح لدينا في المثال في الأسفل :

```
main.dart
lib > main.dart > ...
1 import 'two.dart';
2 Run | Debug
3
4 User user = new User();
5
6 user.printName(); ← امر الطباعة
7
8
9
10

two.dart
lib > two.dart > User
1 class User {
2   var username = "wael";
3   var _email = 'wael@gmail.com';
4
5   printName(){
6     print(_email);
7   }
8 }

method public
variable Private
page one
page two
page two
Exit
DEBUG CONSOLE
wael@gmail.com ← خرج الطباعة
Exited
```

الامر `Private` ينطبق تماماً على `method` كما هو واضح لدينا في المثال في الأسفل :

The screenshot shows two Dart files side-by-side:

- main.dart:** Imports `'two.dart'`. In the `main` function, there is a call to `user.printName()`. A red box labeled "method Private" highlights this line, with a red arrow pointing to it from the text "رسالة خطأ CN" (CN error message).
- two.dart:** Contains a `User` class with a `username` variable and an `_printName` method. The `_printName` method is also highlighted with a red box and labeled "method Private".

وأيضاً كما ذكرنا ان الامر ينطبق تماماً على `Constructor` كما هو واضح لدينا في المثال في الأسفل :

The screenshot shows two Dart files side-by-side:

- main.dart:** Imports `'two.dart'`. In the `main` function, there is a line `User user = new User("wael");`. A red box labeled "ظهور خطأ" (Error appearance) highlights this line.
- two.dart:** Contains a `User` class with a constructor. The constructor is highlighted with a red box and labeled "Constructor Private".



في حال كان لدينا كلاس معين في برنامجنا واردنا ان نقوم بتوريثه من اكثر من كلاس يتحقق ذلك من خلال مفهوم .Mixin

نستخدم `with` التي تعبر عن mixin كما هو واضح لدينا في المثال في الأسفل في

: class three

```
1 class One {  
2     var username = "Wael" ;  
3 }  
  
4 class Two {  
5     var password = "wsafsfdsf" ;  
6 }  
  
7 class Three with Two , One | {  
8 }  
9  
10
```

class one

class two

وراثة الممتلكات من اكثر من كلاس

Mixin

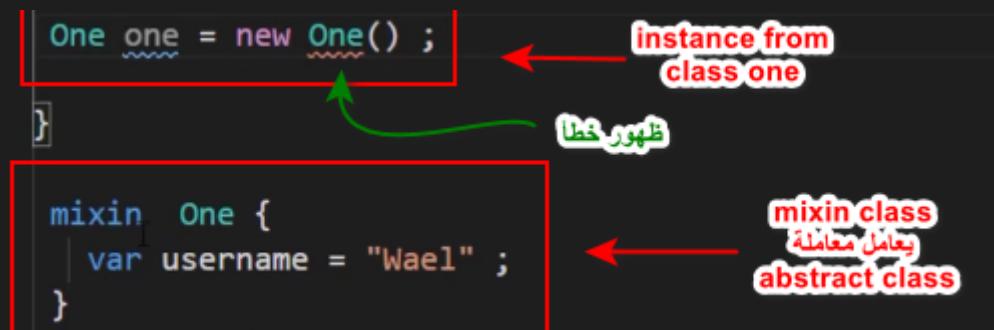
و عند انشاء instance من class three نلاحظ ظهور محتويات class one & two كما هو واضح في المثال في الأسفل :

```
1 Three three = new Three() ;  
2  
3 print(three.) ;  
4  
5         ↗ password String  
6         ↗ username  
7         ↗ hashCode  
8         ↗ runtimeType  
9         ↗ var userna ↗ noSuchMethod(...)  
10        ↗ ↗ toString()  
11         ↗ class Two {  
12             ↗ var password = "wsafsfdsf" ;  
13 }
```

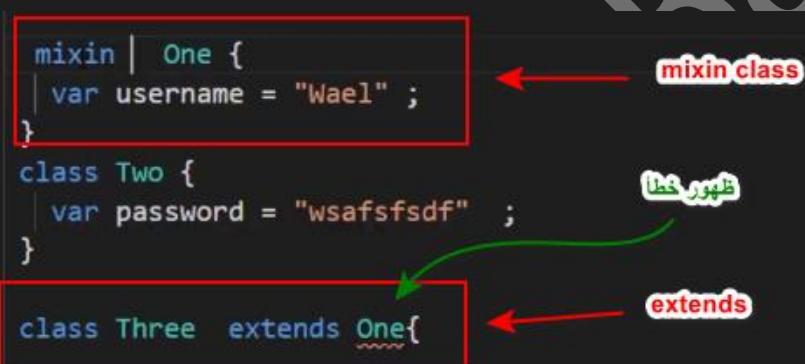
instance from class three

محتويات class one & two

و mixin يعامل معاملة abstract class أي اننا لا نستطيع انشاء instance منه كما هو واضح لدينا في المثال في الأسفل :



تستخدم mixin فقط مع with في حال استخدامها مع extends سيقوم البرنامج بأظهار خطأ كما هو واضح لدينا في المثال في الأسفل :



Null Safety

سابقاً عند تعریف متغير من دون اسناد قيمة له عند تشغيل البرنامج يقوم الحاسوب بأرجاع قيمة Null من دون أي مشاكل الان بعد ان أصبحت لغة Dart تدعم Null Safety عند طباعة متغير قيمته تساوي null سيقوم البرنامج بأظهار رسالة خطأ كما هو واضح لدينا في المثال في الأسفل :

A screenshot of Visual Studio Code showing a code editor and a message dialog. The code editor has the following content:

```
String name ; ← Null
print(name) ;
}
```

A red arrow points from the word "Null" in the code to a message box titled "Visual Studio Code" which says "Build errors exist in your project." Another red arrow points from the message box back to the word "Null" in the code. A red box highlights the word "Null" in the code. A red annotation at the bottom left of the code editor says "ظهور خطأ لأن قيمة المتغير تساوي Null".

تساعدنا على اكتشاف الخطأ قبل وقوعه وأيضاً تقدم أداء أفضل في التطبيقات.

في اردا ترك المتغير من دون قيمة يمكننا استخدام late او إشارة (?) و question mark و Late

A screenshot of Visual Studio Code showing a code editor with the following content:

```
late String name ; ←
name = "wael" ;
print(name.length) ;
}
```

A red arrow points from the word "late" in the code to a red annotation at the bottom left which says "المتغير سيأخذ قيمة فيما بعد".

A screenshot of Visual Studio Code showing the output of the code execution in the terminal:

```
PROBLEMS TERMINAL DEBUG CONSOLE Filter (e.)
Connecting to VM Service at http://127.0.0.1:65486/xWzEVdrPLLo=/
4
Exited
```

The END