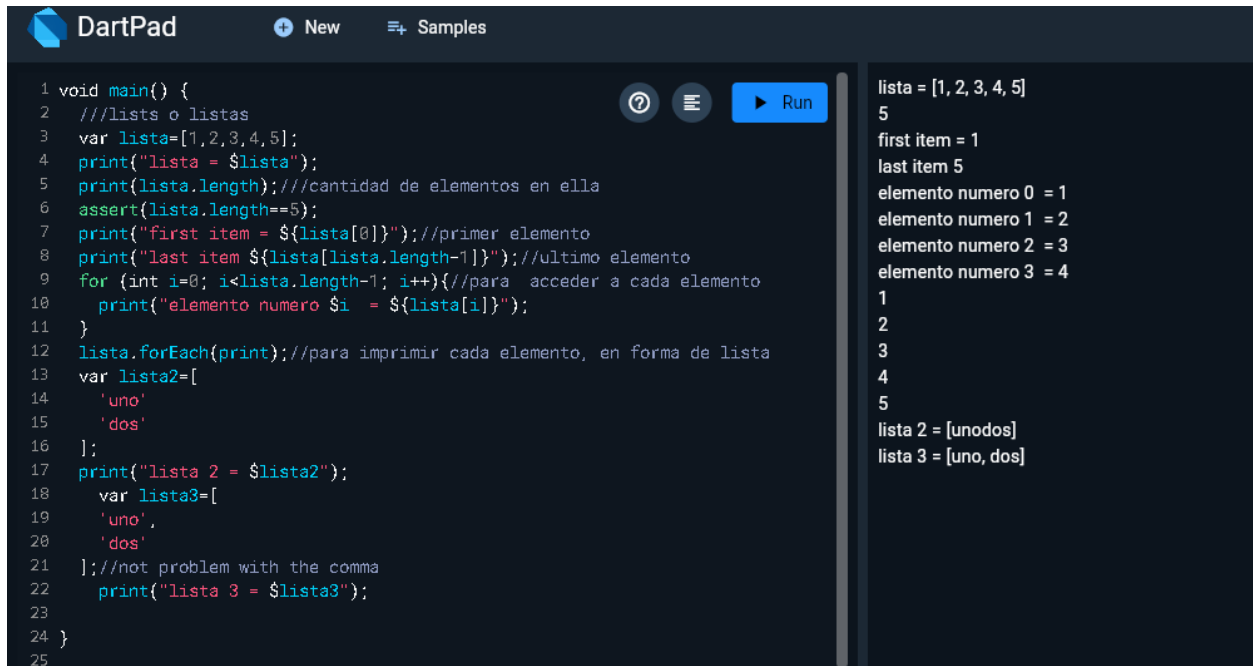


COLLECTIONS

Lists

Quizás la colección más común en casi todos los lenguajes de programación sea la matriz , o grupo ordenado de objetos. En Dart, las matrices son ListObjetos, por lo que la mayoría de las personas las llaman simplemente listas .

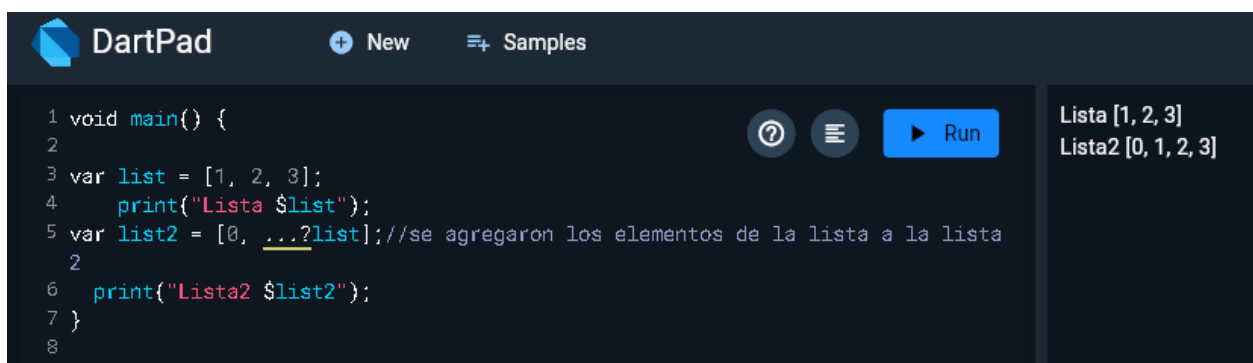


```
1 void main() {
2   ///lists o listas
3   var lista=[1,2,3,4,5];
4   print("lista = $lista");
5   print(lista.length);///cantidad de elementos en ella
6   assert(lista.length==5);
7   print("first item = ${lista[0]}");//primer elemento
8   print("last item ${lista[lista.length-1]}");//ultimo elemento
9   for (int i=0; i<lista.length-1; i++){//para acceder a cada elemento
10    print("elemento numero $i = ${lista[i]}");
11  }
12  lista.forEach(print);///para imprimir cada elemento, en forma de lista
13  var lista2=[
14    'uno'
15    'dos'
16  ];
17  print("lista 2 = $lista2");
18  var lista3=[
19    'uno',
20    'dos'
21  ];///not problem with the comma
22  print("lista 3 = $lista3");
23 }
24
25
```

Output:

```
lista = [1, 2, 3, 4, 5]
5
first item = 1
last item 5
elemento numero 0 = 1
elemento numero 1 = 2
elemento numero 2 = 3
elemento numero 3 = 4
1
2
3
4
5
lista 2 = [unodos]
lista 3 = [uno, dos]
```

Agregar una lista a mi lista. EL signo de pregunta es para evitar errores en caso que la lista a agregar sea vacia.



```
1 void main() {
2
3   var list = [1, 2, 3];
4   print("Lista $list");
5   var list2 = [0, ...?list];///se agregaron los elementos de la lista a la lista
6   print("Lista2 $list2");
7 }
8
```

Output:

```
Lista [1, 2, 3]
Lista2 [0, 1, 2, 3]
```

For dentro de una lista



```
1 void main() {
2   //para implementar for dentro de una lista |
3
4   var list1 = [1, 2, 3];
5   var list2 = ['#0', for (var i in list1) '#$i'];
6   print("lista $list2");
7
8
9 }
10
```

lista [#0, #1, #2, #3]

Sets

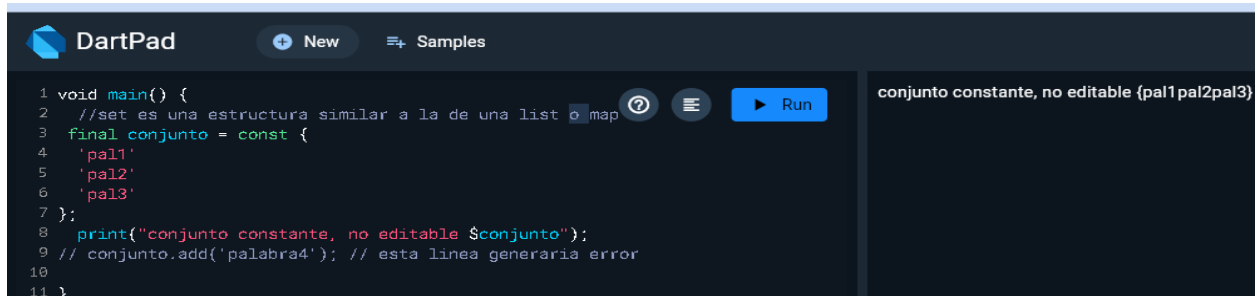
En Dart, un conjunto (Set) es una colección de elementos únicos y en desorden. Son similares a las listas pero los conjuntos no permiten elementos duplicados y tampoco tienen un orden específico como las listas.



```
1 void main() {
2   //set es una estructura similar a la de una list o map
3   var conjuntoConDatos={'palabra1', 'palabra2', 'palabra3'};
4   print("numero de elementos = ${conjuntoConDatos.length}");
5   conjuntoConDatos.add('palabra4');//agregamos otro elemento al set
6   var conjunto=<String>{};
7   print("aqui el set o conjunto esta vacio $conjunto");
8   conjunto.addAll(conjuntoConDatos);
9   print("aqui el set ya tiene los datos del primer set con sates $conjunto");
10
11
12   for (int i=0; i<conjunto.length;i++){//aqui recorremos el conjunto,
13     print("Elemento número $i = ${conjunto.elementAt(i)}");
14   }
15
16 |
17
18 }
19
```

numero de elementos = 3
aqui el set o conjunto esta vacio {}
aqui el set ya tiene los datos del primer set con sates {palabra1, palabra2, palabra3, palabra4}
Elemento número 0 = palabra1
Elemento número 1 = palabra2
Elemento número 2 = palabra3
Elemento número 3 = palabra4

O para crear un set constante lo creas de la siguiente forma



```
1 void main() {
2   //set es una estructura similar a la de una list o map
3   final conjunto = const {
4     'pal1',
5     'pal2',
6     'pal3'
7   };
8   print("conjunto constante, no editable $conjunto");
9   // conjunto.add('palabra4'); // esta linea generaria error
10
11 }
```

conjunto constante, no editable {pal1pal2pal3}

Maps

En general, un mapa es un objeto que asocia claves y valores. Tanto las claves como los valores pueden ser de cualquier tipo. Cada clave aparece solo una vez, pero se puede usar el mismo valor varias veces.

```
void main() {  
    //maps  
    //esta es la forma simple de crear un mapa  
    var map1={  
        "1": "Uno",  
        "2": "Dos",  
        "3": "Tres",  
        "elemento": "cuatro"  
    };  
    print("Map: $map1");  
  
    //forma dos de crear un map con un constructor  
    var map2=Map<int, String>(); //aquí lo estamos creando vacío y le estamos  
    //indicando que las llaves serán tipo int, y los valores tipo String  
  
    map2[1]="Uno"; //aquí le agregamos el primer elemento al map  
    map2[3]="Tres"; //segundo valor, donde le indico key=3, y value=Tres  
    print("mapa 2 $map2");  
}
```

Map: {1: Uno, 2: Dos, 3: Tres, elemento: cuatro}
mapa 2 {1: Uno, 3: Tres}

Para acceder a los elementos del mapa puede hacerlo de la siguiente forma. Si se busca una clave que no está en el mapa retornará null.

← → ↺ dartpad.dev

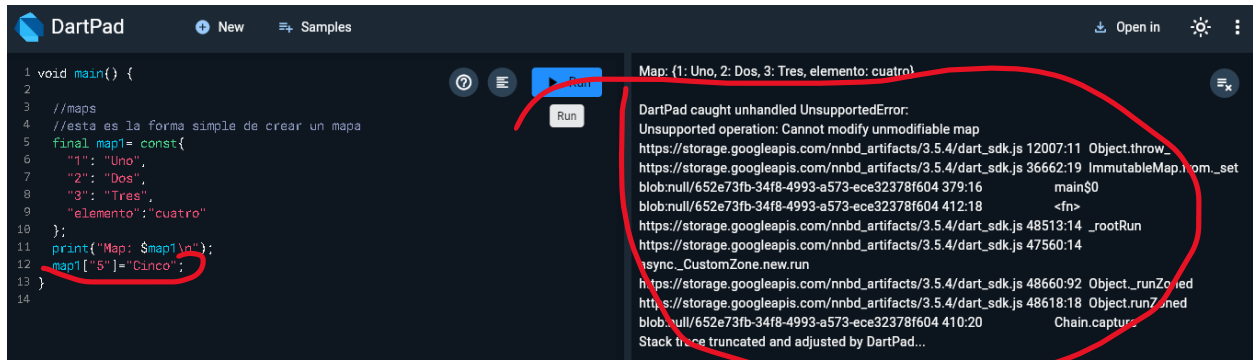
mode MindBox YouTube Maps Gmail Tecnológico de Zitá... Tec Zitácuaro DeepL Translate - El... Correo: Cruz García...

DartPad + New ≡ Samples

```
1 void main() {  
2  
3 //maps  
4 //esta es la forma simple de crear un mapa  
5 var map1={  
6     "1": "Uno",  
7     "2": "Dos",  
8     "3": "Tres",  
9     "elemento": "cuatro"  
10 };  
11 print("Map: $map1\n");  
12  
13 print("Valor perteneciente a la key 1= ${map1["1"]}");  
14 print("Valor perteneciente a la key 100= ${map1["100"]}");  
15 }  
16
```

Map: {1: Uno, 2: Dos, 3: Tres, elemento: cuatro}
Valor perteneciente a la key 1= Uno
Valor perteneciente a la key 100= null

Si quieres crear un mapa constante, es decir que más adelante no sea editable o no se le puedan agregar datos, créalo de la siguiente forma.



```
1 void main() {
2
3   //maps
4   //esta es la forma simple de crear un mapa
5   final map1= const{
6     "1": "Uno",
7     "2": "Dos",
8     "3": "Tres",
9     "elemento": "cuatro"
10  };
11  print("Map: $map1\n");
12  map1["5"]="Cinco";
13 }
14
```

Map: {1: Uno, 2: Dos, 3: Tres, elemento: cuatro}

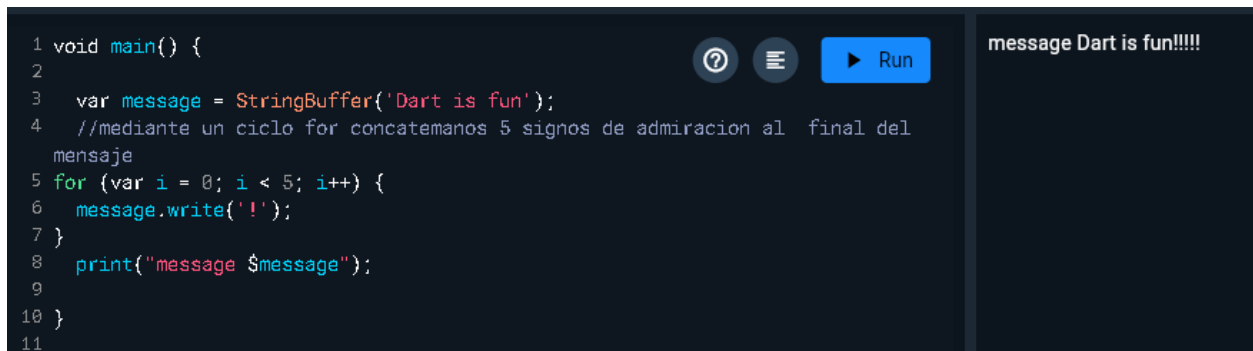
DartPad caught unhandled `UnsupportedError`:
Unsupported operation: Cannot modify unmodifiable map
https://storage.googleapis.com/nnbd_artifacts/3.5.4/dart_sdk.js 12007:11 Object.throw_...
https://storage.googleapis.com/nnbd_artifacts/3.5.4/dart_sdk.js 36662:19 ImmutableMap.from_set...
blob:null/652e73fb-34f8-4993-a573-ec32378f604 379:16 main\$0
blob:null/652e73fb-34f8-4993-a573-ec32378f604 412:18 <fn>
https://storage.googleapis.com/nnbd_artifacts/3.5.4/dart_sdk.js 48513:14 _rootRun
https://storage.googleapis.com/nnbd_artifacts/3.5.4/dart_sdk.js 47560:14
_sync_CustomZone.new.run
https://storage.googleapis.com/nnbd_artifacts/3.5.4/dart_sdk.js 48660:92 Object._runZoned
https://storage.googleapis.com/nnbd_artifacts/3.5.4/dart_sdk.js 48618:18 Object.runZoned
blob:null/652e73fb-34f8-4993-a573-ec32378f604 410:20 Chain.capture
Stack trace truncated and adjusted by DartPad...

La última agregación al mapa por ende retorna error, ya que se indico desde el inicio al mapa como constante.

Bucles

For

Como sabemos `i` representa el inicio del ciclo, la siguiente parte `i<5` representa el tope, e `i++` representa el incremento y lo que esta dentro de las llaves representa la acción a realizar.



```
1 void main() {
2
3   var message = StringBuffer('Dart is fun');
4   //mediante un ciclo for concatemos 5 signos de admiracion al final del
   mensaje
5   for (var i = 0; i < 5; i++) {
6     message.write('!');
7   }
8   print("message $message");
9
10 }
11
```

message Dart is fun!!!!

Como recorrer una lista constante o no constante



```
1 void main() {
2
3   const iterable = ['elemen1', 'ele2', 'ele3'];
4   for (final element in iterable) {
5     print(element);
6   }
7
8 }
9
```

elemen1
ele2
ele3



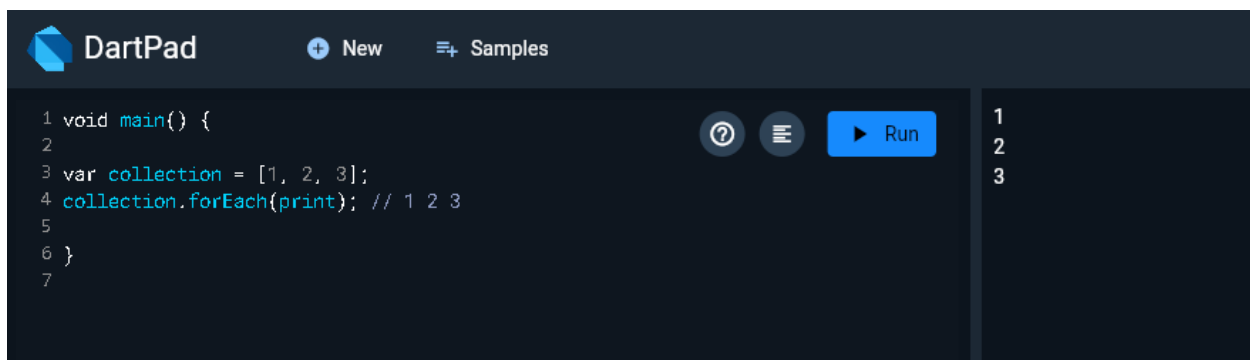
The screenshot shows the DartPad web interface. The code editor contains the following Dart code:

```
1 void main() {  
2  
3   var iterable = ['elemen1', 'ele2', 'ele3'];  
4   for (final element in iterable) {  
5     print(element);  
6   }  
7  
8 }  
9
```

On the right side, there is a 'Run' button and a console output area displaying the results of the program:

```
elemen1  
ele2  
ele3
```

Las colecciones también tienen un método llamado `foreach`



The screenshot shows the DartPad web interface. The code editor contains the following Dart code:

```
1 void main() {  
2  
3   var collection = [1, 2, 3];  
4   collection.forEach(print); // 1 2 3  
5  
6 }  
7
```

On the right side, there is a 'Run' button and a console output area displaying the results of the program:

```
1  
2  
3
```

Dentro de este `for` antes de imprimir cada elemento estoy condicionando que si es menor que 3 entonces se brinque a la siguiente iteración y ya no haga lo que iba a hacer(imprimir el elemento).



The screenshot shows the DartPad web interface. The code editor contains the following Dart code:

```
1 void main() {  
2  
3   var lista = [1, 2, 3,4,5,6,7];  
4   for (int i = 0; i < lista.length; i++) {  
5     var candidate = lista[i];  
6  
7     if (candidate < 3) {  
8       continue;  
9     }  
10    print("$candidate");  
11  
12  
13  }  
14  
15  
16  
17 }
```

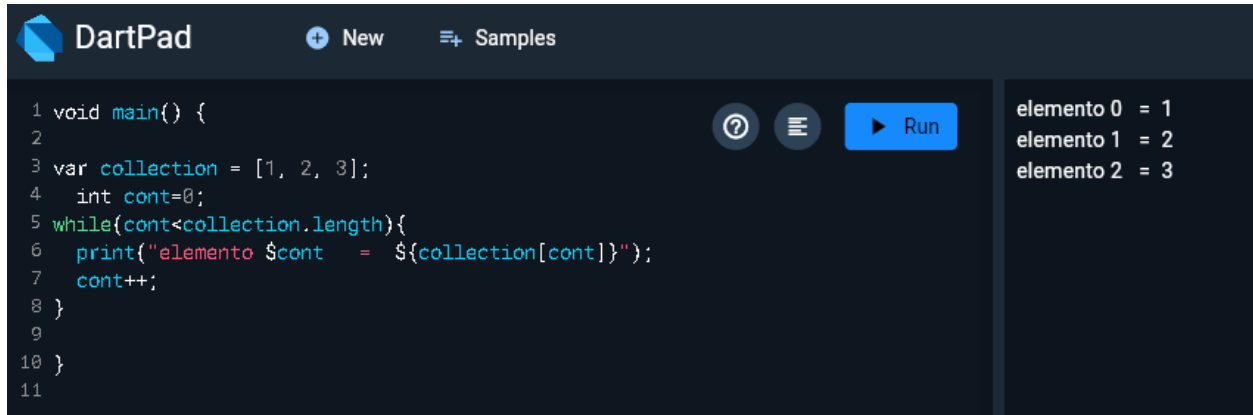
On the right side, there is a 'Run' button and a console output area displaying the results of the program:

```
3  
4  
5  
6  
7
```

While (mientras)

Un while-bucle evalúa la condición antes del ejecutar el bucle.

Como por ejemplo, aquí evaluamos que la variables cont sea menor a la longitud de la colección o lista para poder realizar lo que esta dentro de las llaves, de lo contrario se termina el bucle.



The screenshot shows the DartPad web interface. The code editor contains the following Dart code:

```
1 void main() {  
2  
3   var collection = [1, 2, 3];  
4   int cont=0;  
5   while(cont<collection.length){  
6     print("elemento $cont = ${collection[cont]}");  
7     cont++;  
8   }  
9  
10 }  
11
```

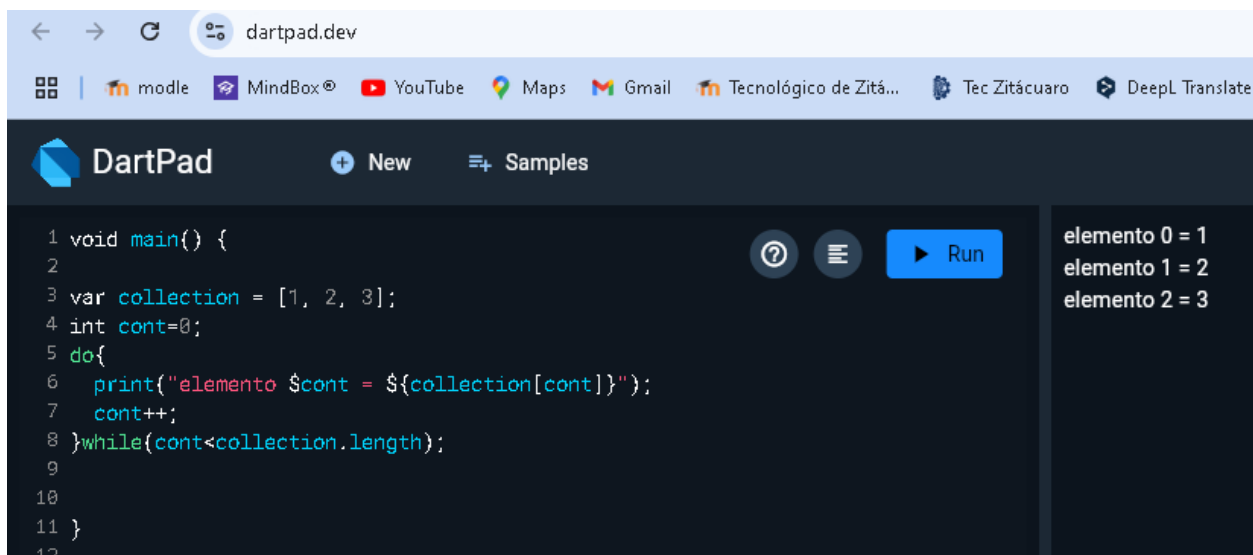
On the right side, the output of the program is displayed:

```
elemento 0 = 1  
elemento 1 = 2  
elemento 2 = 3
```

Do while

Un do while evalúa la condición después del bucle:

En este ciclo deberíamos tener precaución, ya que puede botar errores por la forma de implementación, como por ejemplo si le doy valor inicial a cont como 100 y esa posición en la colección no existe. Sin embargo podemos utilizar un break para romper el ciclo si este fuera el caso.



The screenshot shows the DartPad web interface. The code editor contains the following Dart code:

```
1 void main() {  
2  
3   var collection = [1, 2, 3];  
4   int cont=0;  
5   do{  
6     print("elemento $cont = ${collection[cont]}");  
7     cont++;  
8   }while(cont<collection.length);  
9  
10  
11 }  
12
```

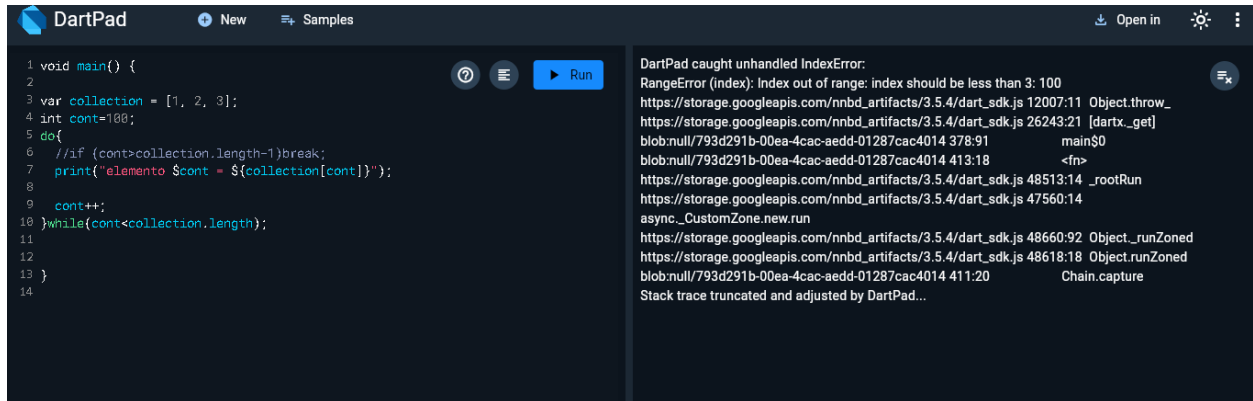
On the right side, the output of the program is displayed:

```
elemento 0 = 1  
elemento 1 = 2  
elemento 2 = 3
```

Break

Usando break para romper un ciclo.

Sin break

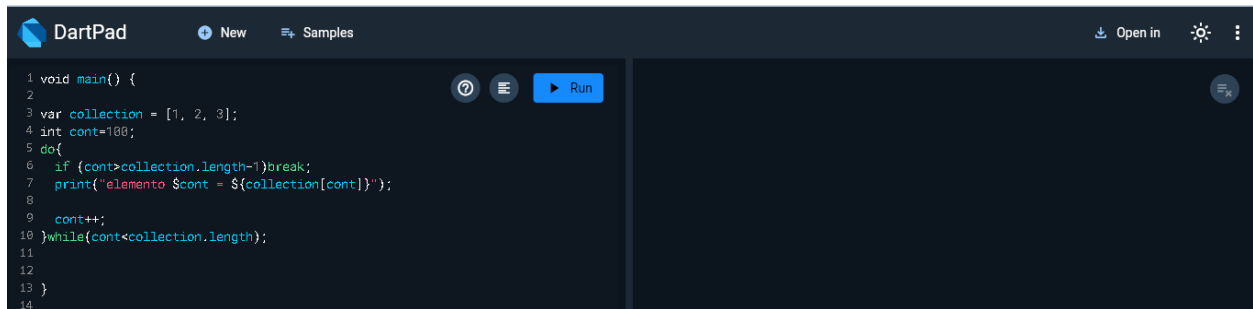


The screenshot shows the DartPad interface with a Dart program that enters an infinite loop. The code defines a list `collection` with values `[1, 2, 3]` and a counter `cont` set to 100. It uses a `do` loop that prints the element at index `cont` and increments `cont`, but it never reaches a `break` statement. The console on the right displays a `DartPad caught unhandled IndexError: RangeError (index): Index out of range: index should be less than 3: 100`, indicating the program has run out of memory or time due to the infinite loop.

```
1 void main() {  
2  
3   var collection = [1, 2, 3];  
4   int cont=100;  
5   do{  
6     //if {cont>collection.length-1}break;  
7     print("elemento $cont = ${collection[cont]}");  
8  
9     cont++;  
10  }while(cont<collection.length);  
11  
12  
13 }  
14
```

DartPad caught unhandled IndexError:
RangeError (index): Index out of range: index should be less than 3: 100
https://storage.googleapis.com/nnbd_artifacts/3.5.4/dart_sdk.js 12007:11 Object.throw_
https://storage.googleapis.com/nnbd_artifacts/3.5.4/dart_sdk.js 26243:21 [dartx._get]
blob:null/793d291b-00ea-4cac-aedd-01287cac4014 378:91 main\$0
blob:null/793d291b-00ea-4cac-aedd-01287cac4014 413:18 <fn>
https://storage.googleapis.com/nnbd_artifacts/3.5.4/dart_sdk.js 48513:14 _rootRun
https://storage.googleapis.com/nnbd_artifacts/3.5.4/dart_sdk.js 47560:14
async_CustomZone.new.run
https://storage.googleapis.com/nnbd_artifacts/3.5.4/dart_sdk.js 48660:92 Object.runZoned
https://storage.googleapis.com/nnbd_artifacts/3.5.4/dart_sdk.js 48618:18 Object.runZoned
blob:null/793d291b-00ea-4cac-aedd-01287cac4014 411:20 Chain.capture
Stack trace truncated and adjusted by DartPad...

Pero aquí todo bien porque se implemento break con una condición.



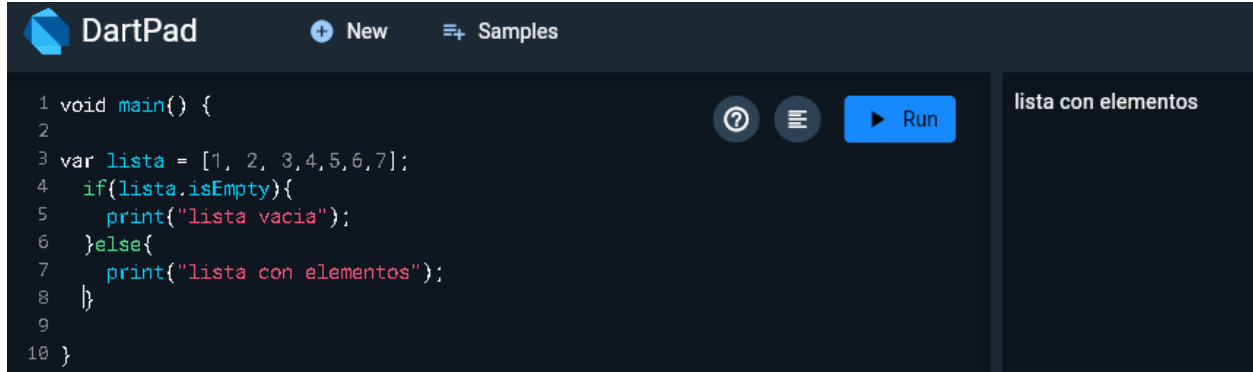
The screenshot shows the DartPad interface with the same Dart program as before, but with a `break` statement added to the `do` loop when the counter `cont` exceeds the length of the `collection` list. The program runs successfully, printing the elements of the list in order: "elemento 1 = 1", "elemento 2 = 2", and "elemento 3 = 3".

```
1 void main() {  
2  
3   var collection = [1, 2, 3];  
4   int cont=100;  
5   do{  
6     if {cont>collection.length-1}break;  
7     print("elemento $cont = ${collection[cont]}");  
8  
9     cont++;  
10  }while(cont<collection.length);  
11  
12  
13 }  
14
```

Branches

If


Ejemplo1: La condición entre paréntesis después if debe ser una expresión que evalúe un valor booleano .



```
1 void main() {  
2  
3   var lista = [1, 2, 3,4,5,6,7];  
4   if(lista.isEmpty){  
5     print("lista vacia");  
6   }else{  
7     print("lista con elementos");  
8   }  
9  
10 }
```

lista con elementos

Ejemplo3

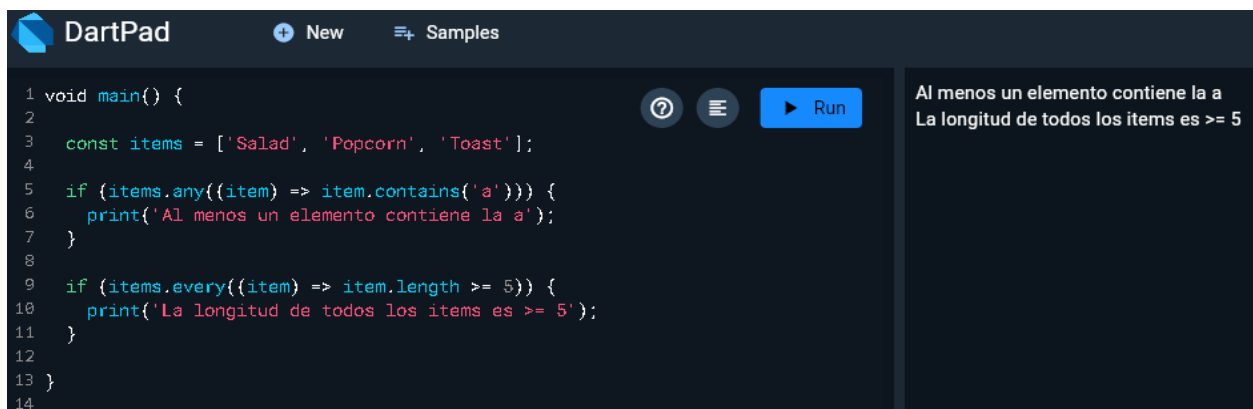


```
1 void main() {  
2  
3   var lista = [1, 2, 3,4,5,6,7];  
4   if(lista.elementAt(0)==2){  
5     print("en la posicion 0 el valor es 2");  
6   }else{  
7     print("en la posicion 0 el valor no es 2");  
8   }  
9  
10 }
```

en la posicion 0 el valor no es 2

Ejemplo2

Any verifica que al menos uno, mientras que every verifica que todos cumplan.



```
1 void main() {  
2  
3   const items = ['Salad', 'Popcorn', 'Toast'];  
4  
5   if (items.any((item) => item.contains('a'))) {  
6     print('Al menos un elemento contiene la a');  
7   }  
8  
9   if (items.every((item) => item.length >= 5)) {  
10    print('La longitud de todos los items es >= 5');  
11  }  
12  
13 }  
14
```

Al menos un elemento contiene la a
La longitud de todos los items es >= 5

Otra forma de usar if es con else if que evalúa la siguiente condición después que la anterior no se cumple.



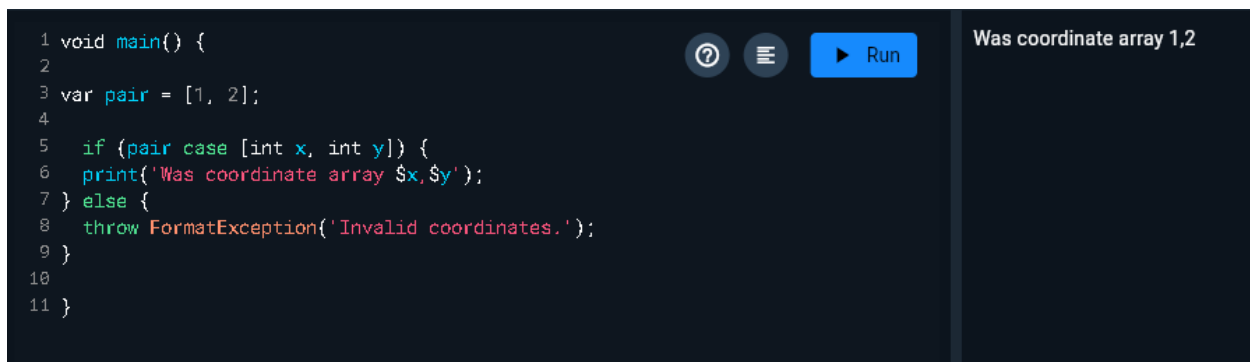
```
1 void main() {  
2  
3   var lista = [1, 2, 3, 4, 5, 6, 7];  
4   if(lista.elementAt(0)==2){  
5     print("en la posicion 0 el valor es 2");  
6   }else if (lista.elementAt(1)==2){  
7     print("en la posicion 1 el valor es 2");  
8   }else{  
9     print("ni en la posicion 0 ni 1 el valor es 2");  
10  
11 }
```

en la posicion 1 el valor es 2

If case

La declaración if-case proporciona una forma de comparar y desestructurar con un único patrón.

En este ejemplo si se cumple lo que está dentro del paréntesis se ejecuta la sentencia, en donde se da valor a las variables que dice el case para después usarlas.



```
1 void main() {  
2  
3   var pair = [1, 2];  
4  
5   if (pair case [int x, int y]) {  
6     print('Was coordinate array $x,$y');  
7   } else {  
8     throw FormatException('Invalid coordinates.');
```

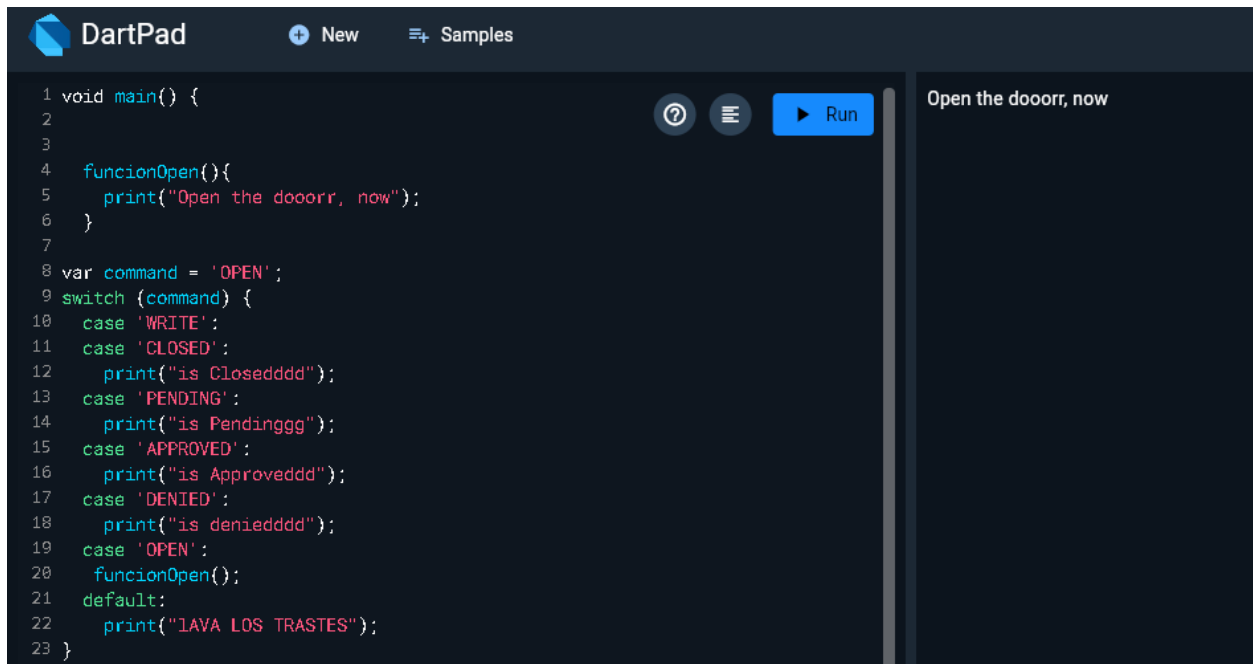
Was coordinate array 1,2

Switch

Una switch declaración evalúa una expresión de valor en relación con una serie de casos. Cada case cláusula es un patrón con el que se compara el valor. Puede utilizar cualquier tipo de patrón para un caso. Cuando el valor coincide con el patrón de un caso, se ejecuta el cuerpo del caso. Las cláusulas no vacías case saltan al final del cambio después de completarse. No requieren una break declaración. Otras formas válidas de finalizar una case cláusula no vacía son una declaración continue, throw o return.

Utilice una cláusula comodín default o para ejecutar código cuando no coincida ninguna cláusula.

Los casos vacíos como write permiten saltarse al siguiente case.

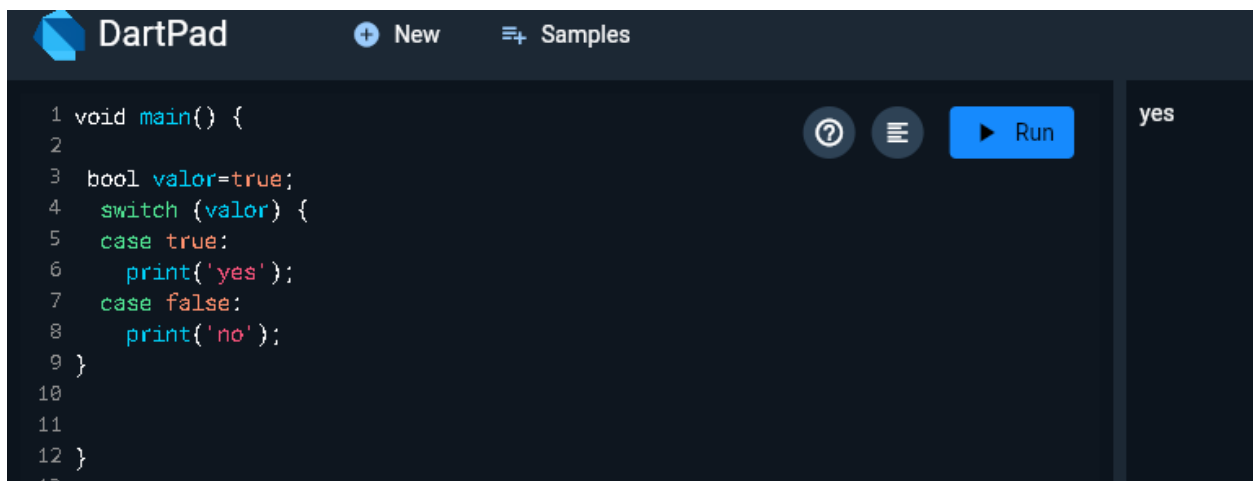


The screenshot shows the DartPad interface with a code editor on the left and a console on the right. The code defines a `main` function that calls `functionOpen` and then uses a `switch` statement to handle different command values. The console output shows the result of the first `print` statement.

```
1 void main() {  
2  
3  
4   functionOpen(){  
5     print("Open the dooorr, now");  
6   }  
7  
8   var command = 'OPEN';  
9   switch {command} {  
10    case 'WRITE':  
11    case 'CLOSED':  
12      print("is Closedddd");  
13    case 'PENDING':  
14      print("is Pendinggg");  
15    case 'APPROVED':  
16      print("is Approveddd");  
17    case 'DENIED':  
18      print("is deniedddd");  
19    case 'OPEN':  
20      functionOpen();  
21    default:  
22      print("LAVA LOS TRASTES");  
23  }  
24 }
```

Open the dooorr, now

Otro ejemplo



The screenshot shows the DartPad interface with a code editor on the left and a console on the right. The code defines a `main` function that sets a `valor` variable to `true` and then uses a `switch` statement to print 'yes' or 'no' based on the value of `valor`. The console output shows the result of the `print` statement.

```
1 void main() {  
2  
3   bool valor=true;  
4   switch {valor} {  
5     case true:  
6       print('yes');  
7     case false:  
8       print('no');  
9   }  
10  
11  
12 }  
13 }
```

yes