***Inicio e inmersión al lenguaje Dart***

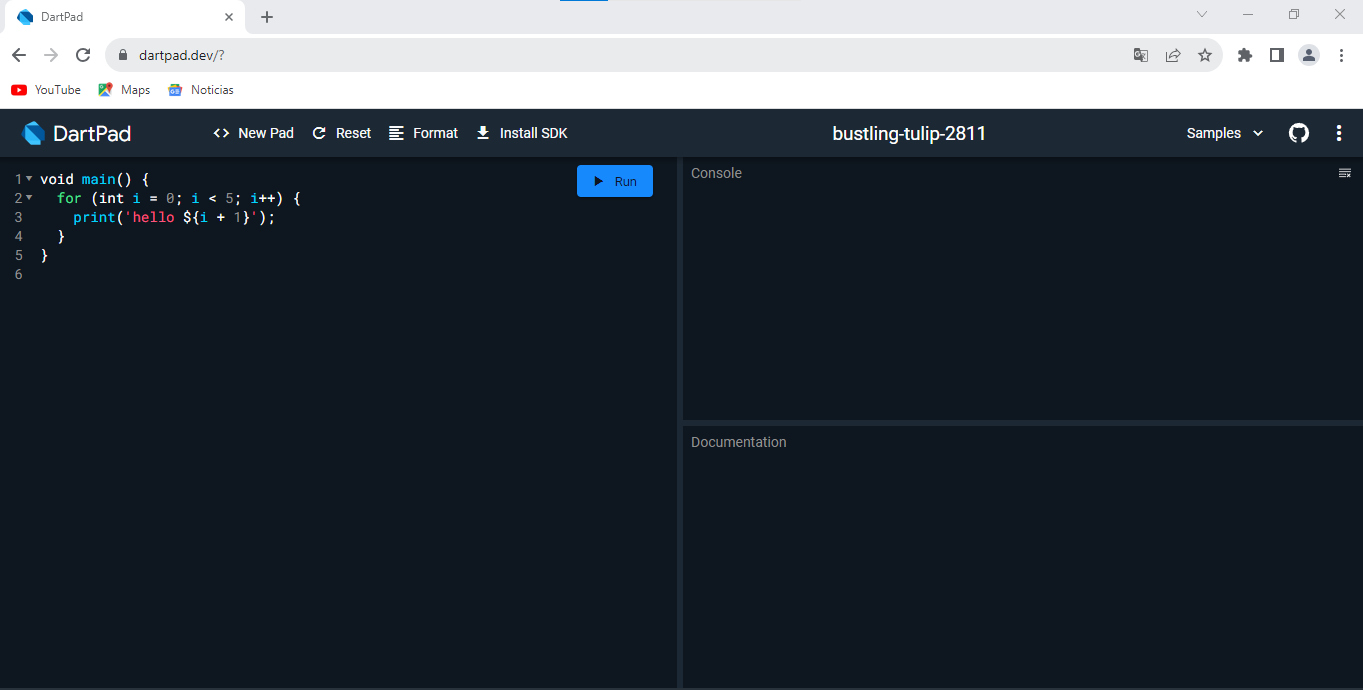
***(***[***https://dart.dev/***](https://dart.dev/) ***)***

******

Herramienta en línea de inicio e inmersión al lenguaje Dart:

* Dartpad, permite experimentar con el lenguaje Dart desde el navegador sin necesidad de instalar nada.
* Enlace:

<https://dartpad.dev/>



Tipos de datos en lenguaje Dart:

| En Dart podemos identificar los siguientes tipos de datos:   1. Enteros: **int** 2. De punto punto flotante: **double** 3. Cadenas de texto: **String** 4. Booleanos: **bool** 5. Tipos dinámicos: **dynamic** |
| --- |

Ejercicios ilustrativos:

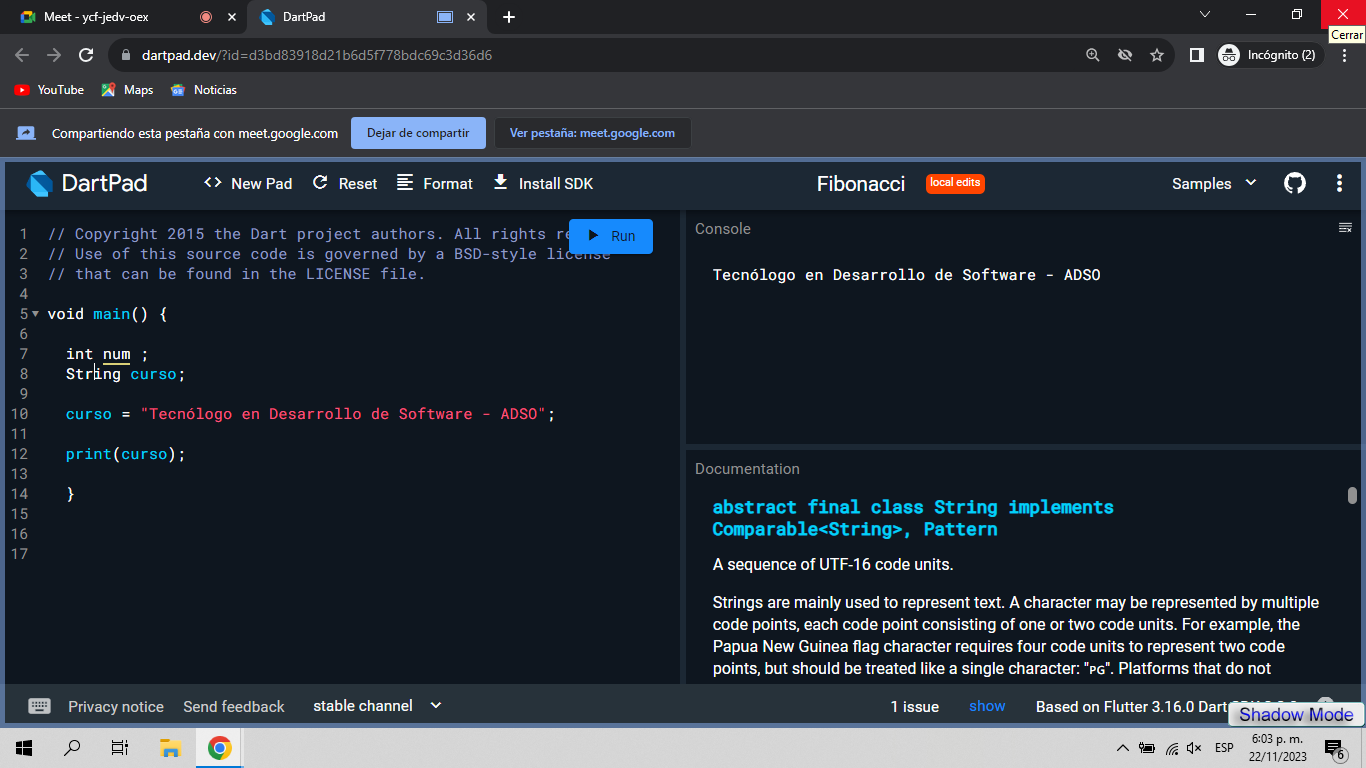
1. Uso de métodos de String.

* Declare la variable llamada curso

*curso = "Tecnólogo en Desarrollo de Software - ADSO";*

* Haga uso de otras funciones de la clase String.

Punto de partida:



* Obtenga el carácter de la posición No 5 de la cadena
* Imprima el tamaño de la cadena
* Indique si la cadena está vacía
* Utilice las funciones para imprimir todo el texto en mayúscula y en minúscula.

1. Declare dos variables numéricas y mediante un mensaje concatenado, imprima:

* La suma de los dos valores asignados a las variables.
* La resta
* La multiplicación
* La división

1. Declarar:

*String num1="23";*

*int num2=45;*

*int suma;*

* En la variable suma calcula la operación respectiva
* Imprima el mensaje correspondiente de la suma

Realice el mismo proceso para variables tipo double:

*String num3="5.9";*

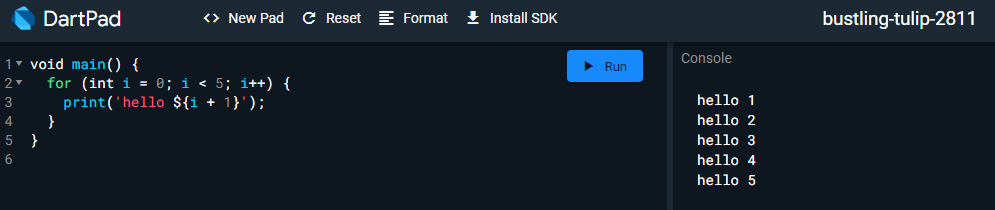
*int num4=3.5;*

*int sumaD;*

1. Interpolación de textos

* La interpolación de textos en Dart es una forma sencilla de combinar variables dentro de cadenas de caracteres en lugar de concatenar manualmente las partes de una cadena utilizando el operador de concatenación (+)
* Permite incrustar variables directamente en la cadena utilizando la sintaxis **$variable**
* Ejemplo:

| void main() {  for (int i = 0; i < 5; i++) {  print('hello ${i + 1}');  }  } |
| --- |

****

1. Punto de partida:

String nombre = 'José Mujica';

int edad = 88;

Imprima el siguiente mensaje usando interpolación:



1. Declare las variables requeridas e imprima el siguiente mensaje usando interpolación:



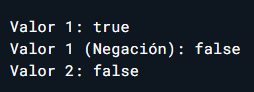
1. Tipo de dato lógico

Punto de partida:

bool valor1= true;

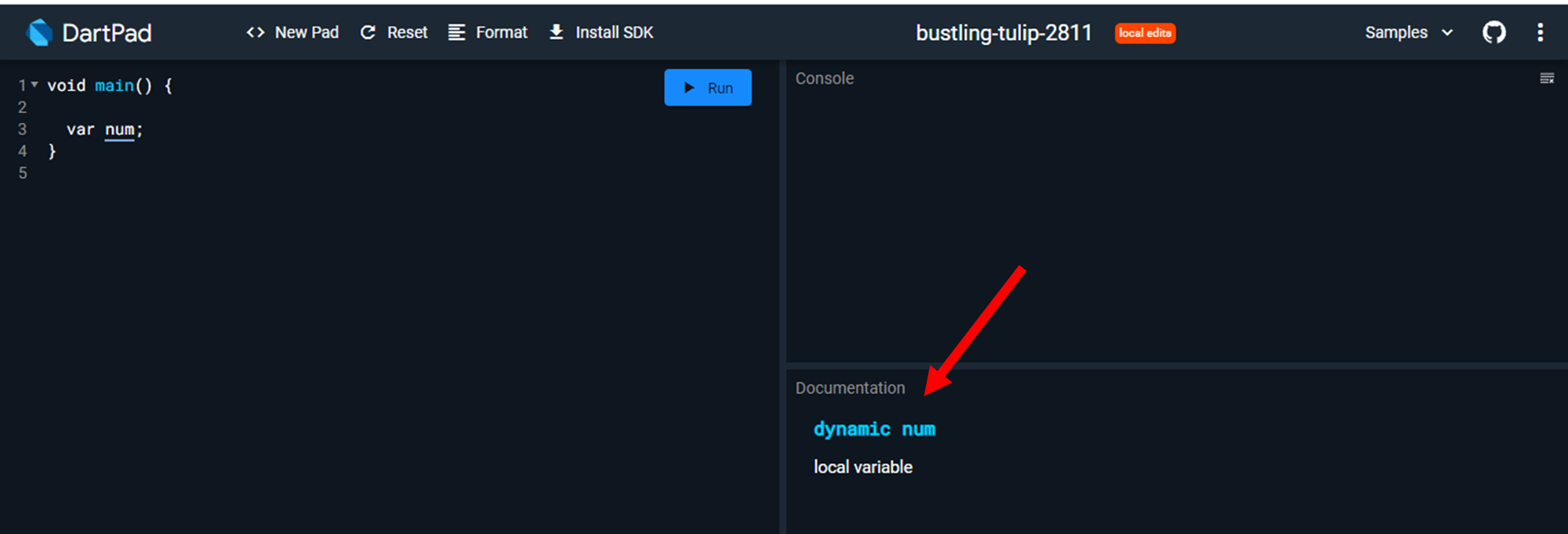
bool valor2= !valor1;

Imprima los valores de las variables usando *concatenación manual (+)* e *interpolación de textos*:



1. Dynamic (tipos de datos dinámicos)

* Las variables dinámicas son aquellas de las cuales no sabemos qué tipo de dato va a ser hasta que se le asignen a un valor.
* Toman el tipo de dato del valor asignado.
* Ejemplo:



Notemos que al ubicarnos sobre la variable, en la documentación se nos indica que corresponde al tipo **dynamic**.

* Ejercicio:
* Declare dos variables **Dynamic**
* Asigne un valor int y un valor double
* Calcule la suma de las dos variables
* Imprima usando *concatenación manual (+)* e *interpolación de textos*

1. Listas

* Una lista es una colección ordenada de elementos.
* Estos elementos pueden ser números, textos, objetos, o incluso otras listas.
* Una lista mantiene un orden. Es decir, cada elemento tiene una posición o índice para acceder, modificar y gestionar elementos basándose en su posición en la lista.
* Clases:
* Dinámicas: No especifican qué tipo de elementos contendrá la lista. Dart inferirá el tipo basándose en los elementos que se agreguen.

// Esta lista puede contener cualquier tipo de elemento.

var lista1= [];

lista1.add(true);

lista1.add(5);

lista1.add("Hola");

print(lista1);

List lista2 = [32,"a",21,true,5,1.4];

print(lista2);

dynamic lista3= [];

lista3.add(1);

lista3.add(2);

lista3.add(3);

print(lista3);

* Tipadas: Se establece explícitamente qué tipo de datos puede contener la lista.

List<String> listaFrutas = ["Uchuva", "Mora", "Pitahaya"];

print(listaFrutas);

// Evidentemente la siguiente instrucción generará un error:

// frutas.add(5);

* Listas constantes:

Dart nos ofrece la posibilidad de crear listas que no cambian después de su inicialización.

final listaCte = const [1, 2, 3];

print(listaCte);

// Esto generaría un error porque la lista es constante.

// listaConst.add(4);

* Algunas operaciones:

1. Agregar elementos:

Se usa el método add() para añadir un nuevo elemento al final de la lista.

Ejemplo:

var colores = ['rojo', 'azul'];

colores.add('verde');

1. Acceder a elementos: Simplemente utiliza el índice del elemento.

Ejemplo:

print(colores[0]);

1. Eliminar elementos:

Se usa el método remove() que permite eliminar un elemento específico y el metodo removeAt() que elimina un elemento en un índice determinado.

Ejemplo:

colores.remove('verde'); // Elimina 'verde' de la lista.

colores.removeAt(1); // Elimina el segundo elemento

* Recorrer listas:
* Construcciones de listas

Dart proporciona un enfoque particular para la creación de listas más complejas a partir de operaciones o de condiciones específicas, lo cual aporta una gran flexibilidad al código.

1. Listas creadas con condiciones:

***Ejemplo 1***

void main() {

List<int> edades = [25,17,30,16,1];

List<int> mayoresDeEdad = [];

for (var e in edades)

{

if (e >= 18) {

mayoresDeEdad.add(e);

}

}

print('Edades mayores que 18:');

print(mayoresDeEdad);

for (var edad in mayoresDeEdad) {

print('$edad años');

}

}

***Ejemplo 2***

void main() {

List<int> listaNumeros = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

List<int> listaNumerosPares = [];

for (var numero in listaNumeros) {

if (numero % 2 == 0) {

listaNumerosPares.add(numero);

}

}

print('Números pares:');

print(listaNumerosPares);

}

***Ejemplo 3***

void main() {

List<int> listaNumerosImpares = [

for (int i = 1; i <= 20; i++) if (i % 2 != 0) i

];

// Imprimir la lista

print('Números Impares:');

print(listaNumerosImpares);

}

***Ejemplo 4***

//Crearemos una lista que contenga los cuadrados de los

//números del 1 al 10 únicamente si el cuadrado del número es menor o igual a 50:

void main() {

List<int> listaCuadradosMenorIgualA50 = [

for (int i = 1; i <= 10; i++) if (i \* i <= 50) i \* i

];

// Imprimir la lista

print('Cuadrados Menor o Igual a 50:');

print(listaCuadradosMenorIgualA50);

}

***Ejemplo 5***

//Crear una lista de strings que contenga nombres de frutas

//y sea usada como punto de partida para crear otra lista con las frutas

//que tengan más de 5 caracteres y que comiencen con la letra "a":

void main() {

List<String> listaFrutas = [

'manzana',

'pera',

'banana',

'uva',

'arándano',

'sandía',

'aguacate',

'piña'

];

List<String> listaFrutasFiltradas = [

for (String fruta in listaFrutas)

if (fruta.length > 5 && fruta.startsWith('a')) fruta

];

// Imprimir la lista

print('Frutas con más de 5 caracteres y que comienzan con vocal "a":');

print(listaFrutasFiltradas);

}

1. Listas creadas con operaciones:

***Ejemplo 1***

var listaPares = [for (int i = 0; i <= 10; i+= 2) i];

print('Lista de números pares:\n $listaPares');

***Ejemplo 2***

void main() {

//--------------------------------------------------

// Lista tipada

//-------------------------------------------------

List<String> listaSimbolosUnicode =

[

for (int i = 48; i <= 90; i++) String.fromCharCode(i)

];

// Imprimir la lista

print('Lista Unicode:');

print(listaSimbolosUnicode);

//--------------------------------------------------

// Lista dinámica

//-------------------------------------------------

var simbolosUnicode = [

for (int i = 48; i <= 90; i++) String.fromCharCode(i)];

print(simbolosUnicode);

}

**Ejercicio**:

**List listaEdad** = [32,54,21,34,5,5];

* Declare una lista con datos predefinidos
* Imprima los datos contenidos en la lista
* Imprima el tamaño de la lista
* Imprima el elemento de la posición No 3
* Ordene la lista (método **sort**) y vuelva a imprimir la lista ordenada
* Imprima una sublista de tres elementos (<https://api.flutter.dev/flutter/dart-core/List/sublist.html>)
* Declare una lista que contenga diferentes tipos de datos: lógicos, enteros, reales. Posteriormente, adicione (add) un valor a su preferencia.
* Declare una lista que solo permita almacenar enteros:

**List** <TipoDato> **nombreLista**

| **Sublist:**   * Es similar al substring de las cadenas. * Funciona matemáticamente con intervalos [*cerrado* y *abierto)*     Recuerde: Un intervalo cerrado incluye sus puntos finales. Al contrario, un intervalo abierto no incluye sus puntos finales. |
| --- |