# Number

void main() {

// declare an integer

int num1 = 10;

// declare a double value

double num2 = 10.50;

// print the values

print(num1);

print(num2);

print(num.parse('12'));

print(num.parse('10.91'));

var a = -2;

print(a.abs());

}

|  |  |  |
| --- | --- | --- |
| Sr.No | Method | Description |
| 1 | abs | Returns the absolute value of the number. |
| 2 | ceil | Returns the least integer no smaller than the number. |
| 3 | compareTo | Compares this to other number. |
| 4 | floor | Returns the greatest integer not greater than the current number. |
| 5 | remainder | Returns the truncated remainder after dividing the two numbers. |
| 6 | round | Returns the integer closest to the current numbers. |
| 7 | truncate | Returns an integer after discarding any fractional digits. |

# String

void main() {

int n=1+1;

String str1 = "The sum of 1 and 1 is ${n}";

print(str1);

String str2 = "The sum of 2 and 2 is ${2+2}";

print(str2);

print("The length of the string is: ${str2.length}");

}

|  |  |
| --- | --- |
| Sr.No | Methods & Description |
| 1 | [toLowerCase()](https://www.tutorialspoint.com/dart_programming/dart_programming_string_tolowercase_method.htm)  Converts all characters in this string to lower case. |
| 2 | [toUpperCase()](https://www.tutorialspoint.com/dart_programming/dart_programming_string_toUppercase_method.htm)  Converts all characters in this string to upper case. |
| 3 | [trim()](https://www.tutorialspoint.com/dart_programming/dart_programming_string_trim_method.htm)  Returns the string without any leading and trailing whitespace. |
| 4 | [compareTo()](https://www.tutorialspoint.com/dart_programming/dart_programming_string_compareto_method.htm)  Compares this object to another. |
| 5 | [replaceAll()](https://www.tutorialspoint.com/dart_programming/dart_programming_string_replaceall_method.htm)  Replaces all substrings that match the specified pattern with a given value. |
| 6 | [split()](https://www.tutorialspoint.com/dart_programming/dart_programming_string_split_method.htm)  Splits the string at matches of the specified delimiter and returns a list of substrings. |
| 7 | [substring()](https://www.tutorialspoint.com/dart_programming/dart_programming_substring_method.htm)  Returns the substring of this string that extends from startIndex, inclusive, to endIndex, exclusive. |
| 8 | [toString()](https://www.tutorialspoint.com/dart_programming/dart_programming_tostring_method.htm)  Returns a string representation of this object. |
| 9 | [codeUnitAt()](https://www.tutorialspoint.com/dart_programming/dart_programming_string_codeunitat_method.htm)  Returns the 16-bit UTF-16 code unit at the given index. |

# List

void main() {

var lst = new List(3);

lst[0] = 12;

lst[1] = 13;

lst[2] = 11;

print(lst);

var num\_list = [1,2,3];

print(num\_list);

var lst1 = new List();

lst1.add(12);

lst1.add(13);

print(lst1);

}

|  |  |
| --- | --- |
| Sr.No | Methods & Description |
| 1 | [first](https://www.tutorialspoint.com/dart_programming/dart_programming_list_first_method.htm)  Returns the first element in the list. |
| 2 | [isEmpty](https://www.tutorialspoint.com/dart_programming/dart_programming_list_isempty_method.htm)  Returns true if the collection has no elements. |
| 3 | [isNotEmpty](https://www.tutorialspoint.com/dart_programming/dart_programming_list_isnotempty_method.htm)  Returns true if the collection has at least one element. |
| 4 | [length](https://www.tutorialspoint.com/dart_programming/dart_programming_list_length_method.htm)  Returns the size of the list. |
| 5 | [last](https://www.tutorialspoint.com/dart_programming/dart_programming_list_last_method.htm)  Returns the last element in the list. |
| 6 | [reversed](https://www.tutorialspoint.com/dart_programming/dart_programming_list_reversed_method.htm)  Returns an iterable object containing the lists values in the reverse order. |
| 7 | [Single](https://www.tutorialspoint.com/dart_programming/dart_programming_list_single_method.htm)  Checks if the list has only one element and returns it. |

# Map

void main() {

var details = {'Usrname':'tom','Password':'pass@123'};

details['Uid'] = 'U1oo1';

print(details);

var user = new Map();

user ['Usrname'] = 'admin';

user ['Password'] = 'admin@123';

print(user);

var usrMap = {"name": "Tom", 'Email': 'tom@xyz.com'};

usrMap.forEach((k,v) => print('${k}: ${v}'))

}

|  |  |
| --- | --- |
| Sr.No | Function Name & Description |
| 1 | [addAll()](https://www.tutorialspoint.com/dart_programming/dart_programming_map_function_addall.htm)  Adds all key-value pairs of other to this map. |
| 2 | [clear()](https://www.tutorialspoint.com/dart_programming/dart_programming_map_function_clear.htm)  Removes all pairs from the map. |
| 3 | [remove()](https://www.tutorialspoint.com/dart_programming/dart_programming_map_function_remove.htm)  Removes key and its associated value, if present, from the map. |
| 4 | [forEach()](https://www.tutorialspoint.com/dart_programming/dart_programming_map_function_foreach.htm)  Applies f to each key-value pair of the map. |

# Function

void main() {

print(factorial(6));

}

factorial(number) {

if (number <= 0) {

// termination case

return 1;

} else {

return (number \* factorial(number - 1));

// function invokes itself

}

}

void main() {

printMsg();

print(test());

}

printMsg()=>

print("hello");

int test()=>123;

# Class

void main() {

Car c= new Car();

c.disp();

}

class Car {

// field

String engine = "E1001";

// function

void disp() {

print(engine);

}

}

void main() {

Car c = new Car('E1001');

}

class Car {

Car(String engine) {

print(engine);

}

}

void main() {

Car c1 = new Car.namedConst('E1001');

Car c2 = new Car();

}

class Car {

Car() {

print("Non-parameterized constructor invoked");

}

Car.namedConst(String engine) {

print("The engine is : ${engine}");

}

}

class Student {

String name;

int age;

String get stud\_name {

return name;

}

void set stud\_name(String name) {

this.name = name;

}

void set stud\_age(int age) {

if(age<= 0) {

print("Age should be greater than 5");

} else {

this.age = age;

}

}

int get stud\_age {

return age;

}

}

void main() {

Student s1 = new Student();

s1.stud\_name = 'MARK';

s1.stud\_age = 0;

print(s1.stud\_name);

print(s1.stud\_age);

}

void main() {

var obj = new Circle();

obj.cal\_area();

}

class Shape {

void cal\_area() {

print("calling calc area defined in the Shape class");

}

}

class Circle extends Shape {}

void main() {

var obj = new Leaf();

obj.str = "hello";

print(obj.str);

}

class Root {

String str;

}

class Child extends Root {}

class Leaf extends Child {}

void main() {

Child c = new Child();

c.m1(12);

}

class Parent {

void m1(int a){ print("value of a ${a}");}

}

class Child extends Parent {

@override

void m1(int b) {

print("value of b ${b}");

}

}

class StaticMem {

static int num;

static disp() {

print("The value of num is ${StaticMem.num}") ;

}

}

void main() {

StaticMem.num = 12;

// initialize the static variable }

StaticMem.disp();

// invoke the static method

}

void main() {

Child c = new Child();

c.m1(12);

}

class Parent {

String msg = "message variable from the parent class";

void m1(int a){ print("value of a ${a}");}

}

class Child extends Parent {

@override

void m1(int b) {

print("value of b ${b}");

super.m1(13);

print("${super.msg}") ;

}

}

# Object

class Student {

void test\_method() {

print("This is a test method");

}

void test\_method1() {

print("This is a test method1");

}

}

void main() {

new Student()

..test\_method()

..test\_method1();

}

# Interface

void main() {

ConsolePrinter cp= new ConsolePrinter();

cp.print\_data();

}

class Printer {

void print\_data() {

print("\_\_\_\_\_\_\_\_\_\_Printing Data\_\_\_\_\_\_\_\_\_\_");

}

}

class ConsolePrinter implements Printer {

void print\_data() {

print("\_\_\_\_\_\_\_\_\_\_Printing to Console\_\_\_\_\_\_\_\_\_\_");

}

}

void main() {

Calculator c = new Calculator();

print("The gross total : ${c.ret\_tot()}");

print("Discount :${c.ret\_dis()}");

}

class Calculate\_Total {

int ret\_tot() {}

}

class Calculate\_Discount {

int ret\_dis() {}

}

class Calculator implements Calculate\_Total,Calculate\_Discount {

int ret\_tot() {

return 1000;

}

int ret\_dis() {

return 50;

}

}

# Generic

void main() {

List <String> logTypes = new List <String>();

logTypes.add("WARNING");

logTypes.add("ERROR");

logTypes.add("INFO");

// iterating across list

for (String type in logTypes) {

print(type);

}

}

void main() {

List <String> logTypes = new List <String>();

logTypes.add(1);

logTypes.add("ERROR");

logTypes.add("INFO");

//iterating across list

for (String type in logTypes) {

print(type);

}

}

void main() {

Set <int>numberSet = new Set<int>();

numberSet.add(100);

numberSet.add(20);

numberSet.add(5);

numberSet.add(60);

numberSet.add(70);

// numberSet.add("Tom");

compilation error;

print("Default implementation :${numberSet.runtimeType}");

for(var no in numberSet) {

print(no);

}

}

import 'dart:collection';

void main() {

Queue<int> queue = new Queue<int>();

print("Default implementation ${queue.runtimeType}");

queue.addLast(10);

queue.addLast(20);

queue.addLast(30);

queue.addLast(40);

queue.removeFirst();

for(int no in queue){

print(no);

}

}

void main() {

Map <String,String>m={'name':'Tom','Id':'E1001'};

print('Map :${m}');

}