**Core Library Classes**

# **int class**

An integer number.

The default implementation of int is 64-bit two's complement integers with operations that wrap to that range on overflow.

**Note:** When compiling to JavaScript, integers are restricted to values that can be represented exactly by double-precision floating point values. The available integer values include all integers between -2^53 and 2^53, and some integers with larger magnitude. That includes some integers larger than 2^63. The behavior of the operators and methods in the [int](https://api.flutter.dev/flutter/dart-core/int-class.html) class therefore sometimes differs between the Dart VM and Dart code compiled to JavaScript. For example, the bitwise operators truncate their operands to 32-bit integers when compiled to JavaScript.

Classes cannot extend, implement, or mix in int.

**See also:**

* [num](https://api.flutter.dev/flutter/dart-core/num-class.html) the super class for [int](https://api.flutter.dev/flutter/dart-core/int-class.html).
* [Numbers](https://dart.dev/guides/language/numbers) in [A tour of the Dart language](https://dart.dev/guides/language/language-tour).

Inheritance

* [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)>[num](https://api.flutter.dev/flutter/dart-core/num-class.html)>int

Available Extensions

* [IntToWasmInt](https://api.flutter.dev/flutter/dart-wasm/IntToWasmInt.html) //include in wasm library

Constructors

[int.fromEnvironment](https://api.flutter.dev/flutter/dart-core/int/int.fromEnvironment.html)([String](https://api.flutter.dev/flutter/dart-core/String-class.html) name, {[int](https://api.flutter.dev/flutter/dart-core/int-class.html) defaultValue = 0})

Returns the integer value of the given environment declaration name.

*const*

*factory*

Properties

[bitLength](https://api.flutter.dev/flutter/dart-core/int/bitLength.html) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

Returns the minimum number of bits required to store this integer.

[*hashCode*](https://api.flutter.dev/flutter/dart-core/num/hashCode.html) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

Returns a hash code for a numerical value.

[isEven](https://api.flutter.dev/flutter/dart-core/int/isEven.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Returns true if and only if this integer is even.

[*isFinite*](https://api.flutter.dev/flutter/dart-core/num/isFinite.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Whether this number is finite.

[*isInfinite*](https://api.flutter.dev/flutter/dart-core/num/isInfinite.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Whether this number is positive infinity or negative infinity.

[*isNaN*](https://api.flutter.dev/flutter/dart-core/num/isNaN.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Whether this number is a Not-a-Number value.

[*isNegative*](https://api.flutter.dev/flutter/dart-core/num/isNegative.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Whether this number is negative.

[isOdd](https://api.flutter.dev/flutter/dart-core/int/isOdd.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Returns true if and only if this integer is odd.

[*runtimeType*](https://api.flutter.dev/flutter/dart-core/Object/runtimeType.html) → [Type](https://api.flutter.dev/flutter/dart-core/Type-class.html)

A representation of the runtime type of the object.

[sign](https://api.flutter.dev/flutter/dart-core/int/sign.html) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

Returns the sign of this integer.

Methods

[abs](https://api.flutter.dev/flutter/dart-core/int/abs.html)() → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

Returns the absolute value of this integer.

[ceil](https://api.flutter.dev/flutter/dart-core/int/ceil.html)() → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

Returns this.

[ceilToDouble](https://api.flutter.dev/flutter/dart-core/int/ceilToDouble.html)() → [double](https://api.flutter.dev/flutter/dart-core/double-class.html)

Returns this.toDouble().

[*clamp*](https://api.flutter.dev/flutter/dart-core/num/clamp.html)([num](https://api.flutter.dev/flutter/dart-core/num-class.html) lowerLimit, [num](https://api.flutter.dev/flutter/dart-core/num-class.html) upperLimit) → [num](https://api.flutter.dev/flutter/dart-core/num-class.html)

Returns this [num](https://api.flutter.dev/flutter/dart-core/num-class.html) clamped to be in the range lowerLimit-upperLimit.

[*compareTo*](https://api.flutter.dev/flutter/dart-core/num/compareTo.html)([num](https://api.flutter.dev/flutter/dart-core/num-class.html) other) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

Compares this to other.

[floor](https://api.flutter.dev/flutter/dart-core/int/floor.html)() → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

Returns this.

[floorToDouble](https://api.flutter.dev/flutter/dart-core/int/floorToDouble.html)() → [double](https://api.flutter.dev/flutter/dart-core/double-class.html)

Returns this.toDouble().

[gcd](https://api.flutter.dev/flutter/dart-core/int/gcd.html)([int](https://api.flutter.dev/flutter/dart-core/int-class.html) other) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

Returns the greatest common divisor of this integer and other.

[modInverse](https://api.flutter.dev/flutter/dart-core/int/modInverse.html)([int](https://api.flutter.dev/flutter/dart-core/int-class.html) modulus) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

Returns the modular multiplicative inverse of this integer modulo modulus.

[modPow](https://api.flutter.dev/flutter/dart-core/int/modPow.html)([int](https://api.flutter.dev/flutter/dart-core/int-class.html) exponent, [int](https://api.flutter.dev/flutter/dart-core/int-class.html) modulus) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

Returns this integer to the power of exponent modulo modulus.

[*noSuchMethod*](https://api.flutter.dev/flutter/dart-core/Object/noSuchMethod.html)([Invocation](https://api.flutter.dev/flutter/dart-core/Invocation-class.html) invocation) → dynamic

Invoked when a non-existent method or property is accessed.

[*remainder*](https://api.flutter.dev/flutter/dart-core/num/remainder.html)([num](https://api.flutter.dev/flutter/dart-core/num-class.html) other) → [num](https://api.flutter.dev/flutter/dart-core/num-class.html)

The remainder of the truncating division of this by other.

[round](https://api.flutter.dev/flutter/dart-core/int/round.html)() → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

Returns this.

[roundToDouble](https://api.flutter.dev/flutter/dart-core/int/roundToDouble.html)() → [double](https://api.flutter.dev/flutter/dart-core/double-class.html)

Returns this.toDouble().

[*toDouble*](https://api.flutter.dev/flutter/dart-core/num/toDouble.html)() → [double](https://api.flutter.dev/flutter/dart-core/double-class.html)

This number as a [double](https://api.flutter.dev/flutter/dart-core/double-class.html).

[*toInt*](https://api.flutter.dev/flutter/dart-core/num/toInt.html)() → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

Truncates this [num](https://api.flutter.dev/flutter/dart-core/num-class.html) to an integer and returns the result as an [int](https://api.flutter.dev/flutter/dart-core/int-class.html).

[toRadixString](https://api.flutter.dev/flutter/dart-core/int/toRadixString.html)([int](https://api.flutter.dev/flutter/dart-core/int-class.html) radix) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

Converts [this](https://api.flutter.dev/flutter/dart-core/int-class.html) to a string representation in the given radix.

[toSigned](https://api.flutter.dev/flutter/dart-core/int/toSigned.html)([int](https://api.flutter.dev/flutter/dart-core/int-class.html) width) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

Returns the least significant width bits of this integer, extending the highest retained bit to the sign. This is the same as truncating the value to fit in width bits using an signed 2-s complement representation. The returned value has the same bit value in all positions higher than width.

[toString](https://api.flutter.dev/flutter/dart-core/int/toString.html)() → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

Returns a string representation of this integer.

[*toStringAsExponential*](https://api.flutter.dev/flutter/dart-core/num/toStringAsExponential.html)([[int](https://api.flutter.dev/flutter/dart-core/int-class.html)? fractionDigits]) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

An exponential string-representation of this number.

[*toStringAsFixed*](https://api.flutter.dev/flutter/dart-core/num/toStringAsFixed.html)([int](https://api.flutter.dev/flutter/dart-core/int-class.html) fractionDigits) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

A decimal-point string-representation of this number.

[*toStringAsPrecision*](https://api.flutter.dev/flutter/dart-core/num/toStringAsPrecision.html)([int](https://api.flutter.dev/flutter/dart-core/int-class.html) precision) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

A string representation with precision significant digits.

[toUnsigned](https://api.flutter.dev/flutter/dart-core/int/toUnsigned.html)([int](https://api.flutter.dev/flutter/dart-core/int-class.html) width) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

Returns the least significant width bits of this integer as a non-negative number (i.e. unsigned representation). The returned value has zeros in all bit positions higher than width.

[truncate](https://api.flutter.dev/flutter/dart-core/int/truncate.html)() → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

Returns this.

[truncateToDouble](https://api.flutter.dev/flutter/dart-core/int/truncateToDouble.html)() → [double](https://api.flutter.dev/flutter/dart-core/double-class.html)

Returns this.toDouble().

Operators

[*operator %*](https://api.flutter.dev/flutter/dart-core/num/operator_modulo.html)([num](https://api.flutter.dev/flutter/dart-core/num-class.html) other) → [num](https://api.flutter.dev/flutter/dart-core/num-class.html)

Euclidean modulo of this number by other.

[operator &](https://api.flutter.dev/flutter/dart-core/int/operator_bitwise_and.html)([int](https://api.flutter.dev/flutter/dart-core/int-class.html) other) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

Bit-wise and operator.

[*operator \**](https://api.flutter.dev/flutter/dart-core/num/operator_multiply.html)([num](https://api.flutter.dev/flutter/dart-core/num-class.html) other) → [num](https://api.flutter.dev/flutter/dart-core/num-class.html)

Multiplies this number by other.

[*operator +*](https://api.flutter.dev/flutter/dart-core/num/operator_plus.html)([num](https://api.flutter.dev/flutter/dart-core/num-class.html) other) → [num](https://api.flutter.dev/flutter/dart-core/num-class.html)

Adds other to this number.

[*operator -*](https://api.flutter.dev/flutter/dart-core/num/operator_minus.html)([num](https://api.flutter.dev/flutter/dart-core/num-class.html) other) → [num](https://api.flutter.dev/flutter/dart-core/num-class.html)

Subtracts other from this number.

[*operator /*](https://api.flutter.dev/flutter/dart-core/num/operator_divide.html)([num](https://api.flutter.dev/flutter/dart-core/num-class.html) other) → [double](https://api.flutter.dev/flutter/dart-core/double-class.html)

Divides this number by other.

[*operator <*](https://api.flutter.dev/flutter/dart-core/num/operator_less.html)([num](https://api.flutter.dev/flutter/dart-core/num-class.html) other) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Whether this number is numerically smaller than other.

[operator <<](https://api.flutter.dev/flutter/dart-core/int/operator_shift_left.html)([int](https://api.flutter.dev/flutter/dart-core/int-class.html) shiftAmount) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

Shift the bits of this integer to the left by shiftAmount.

[*operator <=*](https://api.flutter.dev/flutter/dart-core/num/operator_less_equal.html)([num](https://api.flutter.dev/flutter/dart-core/num-class.html) other) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Whether this number is numerically smaller than or equal to other.

[*operator ==*](https://api.flutter.dev/flutter/dart-core/num/operator_equals.html)([Object](https://api.flutter.dev/flutter/dart-core/Object-class.html) other) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Test whether this value is numerically equal to other.

[*operator >*](https://api.flutter.dev/flutter/dart-core/num/operator_greater.html)([num](https://api.flutter.dev/flutter/dart-core/num-class.html) other) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Whether this number is numerically greater than other.

[*operator >=*](https://api.flutter.dev/flutter/dart-core/num/operator_greater_equal.html)([num](https://api.flutter.dev/flutter/dart-core/num-class.html) other) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Whether this number is numerically greater than or equal to other.

[operator >>](https://api.flutter.dev/flutter/dart-core/int/operator_shift_right.html)([int](https://api.flutter.dev/flutter/dart-core/int-class.html) shiftAmount) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

Shift the bits of this integer to the right by shiftAmount.

[operator >>>](https://api.flutter.dev/flutter/dart-core/int/operator_triple_shift.html)([int](https://api.flutter.dev/flutter/dart-core/int-class.html) shiftAmount) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

Bitwise unsigned right shift by shiftAmount bits.

[operator ^](https://api.flutter.dev/flutter/dart-core/int/operator_bitwise_exclusive_or.html)([int](https://api.flutter.dev/flutter/dart-core/int-class.html) other) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

Bit-wise exclusive-or operator.

[operator unary-](https://api.flutter.dev/flutter/dart-core/int/operator_unary_minus.html)() → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

Return the negative value of this integer.

[operator |](https://api.flutter.dev/flutter/dart-core/int/operator_bitwise_or.html)([int](https://api.flutter.dev/flutter/dart-core/int-class.html) other) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

Bit-wise or operator.

[operator ~](https://api.flutter.dev/flutter/dart-core/int/operator_bitwise_negate.html)() → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

The bit-wise negate operator.

[*operator ~/*](https://api.flutter.dev/flutter/dart-core/num/operator_truncate_divide.html)([num](https://api.flutter.dev/flutter/dart-core/num-class.html) other) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

Truncating division operator.

Static Methods

[parse](https://api.flutter.dev/flutter/dart-core/int/parse.html)([String](https://api.flutter.dev/flutter/dart-core/String-class.html) source, {[int](https://api.flutter.dev/flutter/dart-core/int-class.html)? radix, [int](https://api.flutter.dev/flutter/dart-core/int-class.html) onError([String](https://api.flutter.dev/flutter/dart-core/String-class.html) source)?}) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

Parse source as a, possibly signed, integer literal and return its value.

override

[tryParse](https://api.flutter.dev/flutter/dart-core/int/tryParse.html)([String](https://api.flutter.dev/flutter/dart-core/String-class.html) source, {[int](https://api.flutter.dev/flutter/dart-core/int-class.html)? radix}) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)?

Parse source as a, possibly signed, integer literal.

override

**Code :-**

void main() {

int p = 10; //+ve number

int n = -5; //-ve number

int z = 0;

print("-----Properties of int class-----");

print("p.bitLength : ${p.bitLength}"); //Returns the minimum number of bits required to store this integer.

print("p.hashCode : ${p.hashCode}"); //Returns a hash code for a numerical value.

print("p.isEven : ${p.isEven}"); //Returns true if and only if this integer is even.

print("p.isFinite : ${p.isFinite}"); //Whether this number is finite.

print("p.isInfinite : ${p.isInfinite}"); //Whether this number is positive infinity or negative infinity.

print("p.isNaN : ${p.isNaN}"); //Whether this number is a Not-a-Number value.

print("n.isNegative : ${n.isNegative}"); //Whether this number is negative.

print("p.isOdd : ${p.isOdd}"); //Returns true if and only if this integer is odd.

print("n.runtimeType : ${n.runtimeType}"); //A representation of the runtime type of the object.

print("p.sign : ${p.sign}"); //Returns the sign of this integer.

print("z.sign : ${z.sign}");

print("n.sign : ${n.sign}");

print("\n\n p = 10 n = -5 z = 0");

print("-----Methods of int class-----");

print("n.abs() : ${n.abs()}");

print("(10.57).ceil() : ${(10.57).ceil()}");

print("p.ceilToDouble() : ${p.ceilToDouble()}");

print("p.clamp(5, 9) : ${p.clamp(5, 9)}");

print("p.clamp(5, 15) : ${p.clamp(5, 15)}");

print("p.clamp(11, 15) : ${p.clamp(11, 15)}");

print("p.compareTo(20) : ${p.compareTo(20)}");

print("p.compareTo(10) : ${p.compareTo(10)}");

print("p.compareTo(5) : ${p.compareTo(5)}");

print("(10.57).floor() : ${(10.57).floor()}");

print("(10.57).floorToDouble() : ${(10.57).floorToDouble()}");

print("p.gcd(2) : ${p.gcd(2)}");

print("p.modInverse(1) : ${p.modInverse(1)}");

print("p.modPow(2, 5) : ${p.modPow(2, 5)}");

print("p.remainder(5) : ${p.remainder(5)}");

print("(10.57).round() : ${(10.57).round()}");

print("(10.57).roundToDouble() : ${(10.57).roundToDouble()}");

print("p.toDouble() : ${p.toDouble()}");

print("(10.57).toInt() : ${(10.57).toInt()}");

print("p.toRadixString(2) : ${p.toRadixString(2)}");

print("z.toRadixString(2) : ${z.toRadixString(2)}");

print("n.toRadixString(2) : ${n.toRadixString(2)}");

print("p.toRadixString(8) : ${p.toRadixString(8)}");

print("z.toRadixString(8) : ${z.toRadixString(8)}");

print("n.toRadixString(8) : ${n.toRadixString(8)}");

print("p.toRadixString(16) : ${p.toRadixString(16)}");

print("z.toRadixString(16) : ${z.toRadixString(16)}");

print("n.toRadixString(16) : ${n.toRadixString(16)}");

print("n.toSigned(n.bitLength + 1) : ${n.toSigned(n.bitLength + 1)}");

print("p.toString() : ${p.toString()}");

print("(10.57).toStringAsExponential(5) : ${(10.57).toStringAsExponential(5)}");

print("(10.58).toStringAsFixed(1) : ${(10.58).toStringAsFixed(1)}");

print("(10.58).toStringAsPrecision(1) : ${(10.58).toStringAsPrecision(1)}");

print("n.toUnsigned(5) : ${n.toUnsigned(5)}");

print("(10.58).truncate() : ${(10.58).truncate()}");

print("(10.58).truncateToDouble() : ${(10.58).truncateToDouble()}");

print("\n\n p = 10 n = -5 z = 0");

print("-----Applicable Operators in int class-----");

print("5 % 3 : ${5 % 3}"); // 2

print("-5 % 3 : ${-5 % 3}"); // 1

print("5 % -3 : ${5 % -3}"); // 2

print("-5 % -3 : ${-5 % -3}"); // 1

print("(2 & 1).toRadixString(2) : ${(2 & 1).toRadixString(2)}"); // 0010 & 0001 -> 0000

print((3 & 1).toRadixString(2)); // 0011 & 0001 -> 0001

print((10 & 2).toRadixString(2)); // 1010 & 0010 -> 0010

print("5 \* 3 : ${5 \* 3}");

print("-5 \* 3 : ${-5 \* 3}");

print("5 \* -3 : ${5 \* -3}");

print("-5 \* -3 : ${-5 \* -3}");

print("5 + 3 : ${5 + 3}");

print("-5 + 3 : ${-5 + 3}");

print("5 + -3 : ${5 + -3}");

print("-5 + -3 : ${-5 + -3}");

print("5 - 3 : ${5 - 3}");

print("-5 - 3 : ${-5 - 3}");

print("5 - -3 : ${5 - -3}");

print("-5 - -3 : ${-5 - -3}");

print("5 / 3 : ${5 / 3}");

print("-5 / 3 : ${-5 / 3}");

print("5 / -3 : ${5 / -3}");

print("-5 / -3 : ${-5 / -3}");

print("5 < 3 : ${5 < 3}");

print("-5 < 3 : ${-5 < 3}");

print("5 < -3 : ${5 < -3}");

print("-5 < -3 : ${-5 < -3}");

print("(3 << 1).toRadixString(2) : ${(3 << 1).toRadixString(2)}"); // 0011 -> 0110

print("(9 << 2).toRadixString(2) : ${(9 << 2).toRadixString(2)}"); // 1001 -> 100100

print("(10 << 3).toRadixString(2) : ${(10 << 3).toRadixString(2)}"); // 1010 -> 1010000

print("5 <= 3 : ${5 <= 3}");

print("-5 <= 3 : ${-5 <= 3}");

print("5 <= -3 : ${5 <= -3}");

print("-5 <= -3 : ${-5 <= -3}");

print("5 == 3 : ${5 == 3}");

print("5 > 3 : ${5 > 3}");

print("-5 > 3 : ${-5 > 3}");

print("5 > -3 : ${5 > -3}");

print("-5 > -3 : ${-5 > -3}");

print("5 >= 3 : ${5 >= 3}");

print("-5 >= 3 : ${-5 >= 3}");

print("5 >= -3 : ${5 >= -3}");

print("-5 >= -3 : ${-5 >= -3}");

print( "(3 >> 1).toRadixString(2) : ${(3 >> 1).toRadixString(2)}"); // 0011 -> 0001

print("(9 >> 2).toRadixString(2) : ${(9 >> 2).toRadixString(2)}"); // 1001 -> 0010

print("(10 >> 3).toRadixString(2) : ${(10 >> 3).toRadixString(2)}"); // 1010 -> 0001

print("(6 >> 2).toRadixString : ${(6 >> 2).toRadixString}"); // 111...1010 -> 111...1110 == -2

print("(85 >> 3).toRadixString : ${(85 >> 3).toRadixString}"); // 111...10101011 -> 111...11110101 == -11

print("(3 >>> 1).toRadixString(2) : ${(3 >>> 1).toRadixString(2)}"); // 0011 -> 0001

print("(9 >>> 2).toRadixString(2) : ${(9 >>> 2).toRadixString(2)}"); // 1001 -> 0010

print("(9) >>> 2).toRadixString(2) : ${(9 >>> 2).toRadixString(2)}"); // 111...1011 -> 001...1110 (> 0)

print("(2 ^ 1).toRadixString(2) : ${(2 ^ 1).toRadixString(2)}"); // 0010 ^ 0001 -> 0011

print("(3 ^ 1).toRadixString(2) : ${(3 ^ 1).toRadixString(2)}"); // 0011 ^ 0001 -> 0010

print("(10 ^ 2).toRadixString(2) : ${(10 ^ 2).toRadixString(2)}"); // 1010 ^ 0010 -> 1000

print("-(p) : ${-(p)}");

print("-(z) : ${-(z)}");

print("(2 | 1).toRadixString(2) : ${(2 | 1).toRadixString(2)}"); // 0010 | 0001 -> 0011

print("(3 | 1).toRadixString(2) : ${(3 | 1).toRadixString(2)}"); // 0011 | 0001 -> 0011

print("(10 | 2).toRadixString(2) : ${(10 | 2).toRadixString(2)}"); // 1010 | 0010 -> 1010

print("~(p) : ${~(p)}");

print("~(z) : ${~(z)}");

print("~(n) : ${~(n)}");

print("5 ~/ 3 : ${5 ~/ 3}");

print("-5 ~/ 3 : ${-5 ~/ 3}");

print("5 ~/ -3 : ${5 ~/ -3}");

print("-5 ~/ -3 : ${-5 ~/ -3}");

print("\n\n p = 10 n = -5 z = 0");

print("-----Static Methods in int class-----");

print("parse(String source, {int? radix, int onError(String source)?})");

print("var value = int.tryParse(text); ");

print("tryParse(String source, {int? radix})");

print(int.tryParse('2021')); // 2021

print(int.tryParse('1f')); // null

// From binary (base 2) value.

print(int.tryParse('1100', radix: 2)); // 12

print(int.tryParse('00011111', radix: 2)); // 31

print(int.tryParse('011111100101', radix: 2)); // 2021

// From octal (base 8) value.

print(int.tryParse('14', radix: 8)); // 12

print(int.tryParse('37', radix: 8)); // 31

print(int.tryParse('3745', radix: 8)); // 2021

// From hexadecimal (base 16) value.

print(int.tryParse('c', radix: 16)); // 12

print(int.tryParse('1f', radix: 16)); // 31

print(int.tryParse('7e5', radix: 16)); // 2021

// From base 35 value.

print(int.tryParse('y1', radix: 35)); // 1191 == 34 \* 35 + 1

print(int.tryParse('z1', radix: 35)); // null

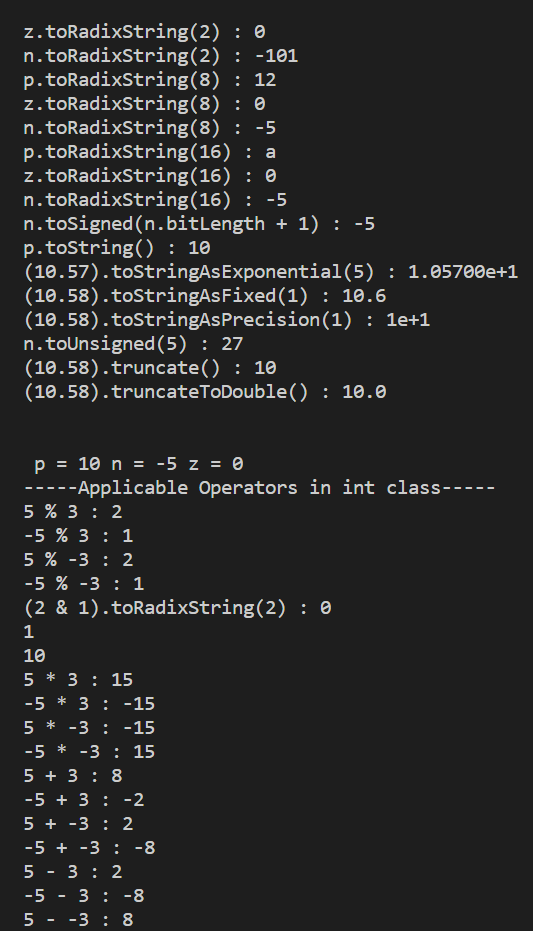
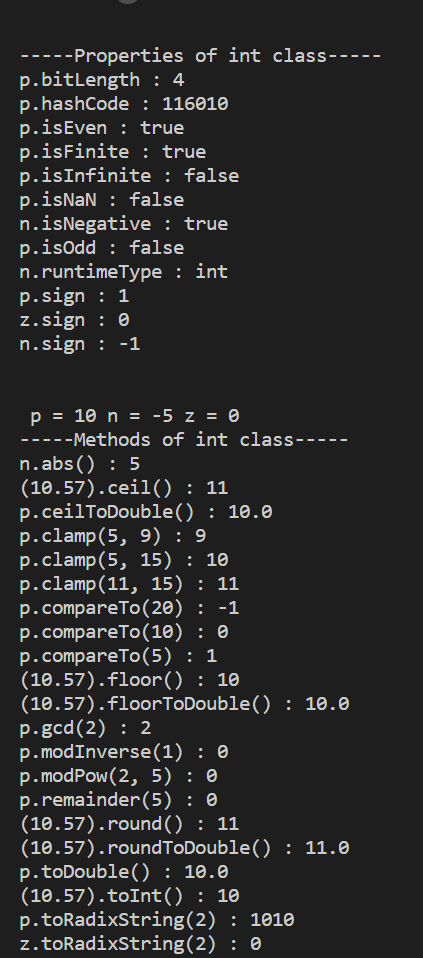
// From base 36 value.

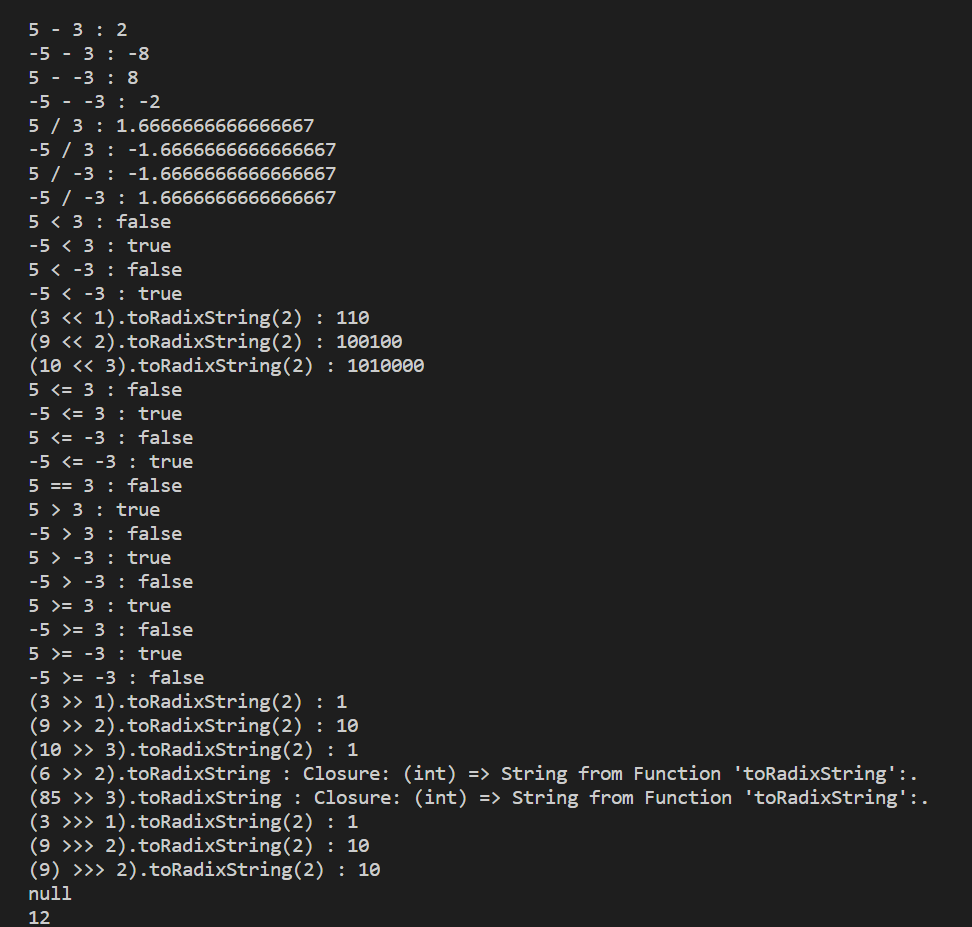
print(int.tryParse('y1', radix: 36)); // 1225 == 34 \* 36 + 1

print(int.tryParse('z1', radix: 36)); // 1261 == 35 \* 36 + 1

}

**Output :-**

****

****

****