Hello World

```
echo "Hello, world!"
```

Nim is an efficient, expressive, and elegant language created by Andreas Rumpf and released in 2008. It is a compiled and statically typed programming language. Nim combines concepts from languages such as Python, Ada, and Modula.

Basic Type

```
42,-4        # int  - Integer
42, 4        # uint - Integer (not negative)
true, false  # bool - Boolean
"42"         # string - String
'a','4'      # char - Character
42.0         # float - Floating-point
```

Comments

```
# This is a comment
echo "Hello!" # This is a comment

#[ This is a multiline
 comment
]#
```

Variable & Constant

```
var myVar1:string = "My variable" # Variable declared as string.
var myVar2 = "My variable" # Variable implicitly declared as string.
let myConstant = "Constant value" # Constant. Immutable value.
var salary:float = 9_000.00  # Float variable.
                             # _ is ignored. This is useful for long numbers.
const PI = 3.14 # Constant. Immutable value. Constants are computed at compile time.
```

Import & Include

```
import os # Imports a lib.
import strutils
import json, othermodule  # Imports a lib & a module.
include "folder/file.nim" # Includes a file. Useful to
include "file1.nimf".     # split a big file in parts.
```

Operator

```
5 + 2    # Addition
5 - 2    # Subtraction
5 * 2    # Multiplication
5 / 2    # Division
# Integer division    # Integer modulo operation (remainder)
5 div 2 # returns: 2  5 mod 2 # returns: 1
```

Advanced Type 1/2

```
# Table.
# Dictionary or Hash Structure.
import tables
var dateOfBirth: Table[string, uint]
dateOfBirth["Mike"] = 1995
dateOfBirth["John"] = 1961
echo dateOfBirth["Mike"]
1995
# Array.
# Array or matrix Structure.
var myArray: array[0 .. 2, bool] = [false,true,false]
echo myArray[1]
true
```

Advanced Type 2/2

```
# Sequence.
# List of values Structure.

let mySequence: seq[string] = @["abc", "def"]
echo mySequence[1]
def
# Tuple.
# Useful to return multiple values from a
# func or proc.
let pos: tuple[x, y, z: int] = (x: 100, y: 50, z: 30)
echo pos                    echo pos.x    echo pos[1]
(x: 100, y: 50, z: 30)      100           50

# Tuple results can be unpacked:
let (a,b,_) = pos # Variables a, b created.
                  # a = 100, b = 50.
                  # Variable z ignored.
```

String Operations 1/2

```
# Concatenation
echo "con" & "cat"
concat

# Quotation marks inside a string
echo """Nim is an "expressive" language"""
Nim is an "expressive" language"

# String interpolation
import strformat
let nimVersion = $NimVersion
echo fmt"""Using Nim version: {nimVersion}"""
Using Nim version: 1.6.8

echo fmt"""Result of 4 x 2 = {4*2}"""
Result of 4 x 2 = 8

# Remove part of a string
echo "Nim-lang".strip[0 .. 2]
Nim

# Multiline string
let multiline = """line1
  line2"""
echo multiline
line1
  line2
```

String Operations 2/2

```
import strutils
echo "A_B_C".normalize()
abc

echo "A_B_C".normalize().toUpper()
ABC

echo " ABC ".strip().len()
3

echo len(" ABC ") == " ABC ".len()
true

echo "ABC".toLower()
abc

echo "a b c".split()
@["a", "b", "c"]

echo ["a", "b", "c"].join(",")
a,b,c

echo "*".repeat(20)
********************

echo "   a".replace(" ","*")
***a
```

Boolean Operator

```
# and
echo true and true  # it returns true only if
true                # both values are true.
# or
echo true or false  # it returns true if at least one
true                # of the two values is true.
# xor
echo false xor true # it returns true if only one
true                # of the two values is true.
# not
echo not false      # it returns the inverse of the
true                # value.
```

Relational Operator

```
echo 5 < 4      echo 5 <= 4     echo 5 == 4
false           false           false

echo 5 != 4     echo 5 > 4      echo 5 >= 4
true            true            true
```

Input from Keyboard

```
echo "Enter your name: "
let name = stdin.readLine()
echo "\nHello ",name
Hello Sergio
```

If Statement

```
echo "Enter your name: "
let name = stdin.readLine()
if name == "":
  echo "Have you forgotten your name?"
elif name == "name":
  echo "Enter your name!"
else:
  echo "Hello, ",name,"!"
```

Ternary Operator

```
import random
randomize()
let n = rand(10)
let evenOdd = n mod 2
echo "The number ",n," is "
echo if evenOdd == 0: "even" else: "odd" # ternary if
```

Case Statement

```
from std/strutils import parseInt
let validNumber ="Enter number from 0 to 9"
echo validNumber
let n = parseInt(readLine(stdin))
case n
  of 0..2, 4..7: echo n,"Number is in the set:{0,1,2,4,5,6,7}"
  of 3, 8: echo "The number is 3 or 8"
  else: echo validNumber
```

For Statement

```
echo "Counting from 1 to 5"
for i in countup(1, 5):
  stdout.write(i," ")
1 2 3 4 5
echo ""
for i in countup(1, 5): stdout.write(i," ")
1 2 3 4 5
echo ""
for i in 1 .. 5: stdout.write(i," ")
1 2 3 4 5
echo ""
for i in countup(1, 5, 2): stdout.write(i," ")
1 3 5
echo "\nCounting from 2 to -2"
for i in countdown(2, -2): stdout.write(i," ")
2 1 0 -1 -2
echo "\nIterate through each characters of the string"
let word = "Nim Lang"
for c in word: stdout.write(c)
Nim Lang
```

While Statement

```
var a = 1
while a*a < 10:
  stdout.write(" a = ",a)
  a = a + 1
echo "\nFinal value of a = ",a
a = 1 a = 2 a = 3
Final value of a = 4

let phrase = "\nEnter your name. <ENTER> to quit:"
echo phrase
var name = readLine(stdin)
while name != "":
  echo "Hello, ",name
  echo phrase
  name = readLine(stdin)
while true: # "while" using break
  echo phrase
  name = readLine(stdin)
  if name == "": break # break to exit from "while"
  echo "Hello, ",name
```

Procedure 1/3

```
# It returns the value of its last expression
proc helloWorld(): string =
  "Hello, world!"

echo helloWorld()
Hello, world!

# return: returns the procedure value
proc numSignal(n: int): int =
  if n == 0: return n
  return if n > 0: 1 else: -1

stdout.write(numSignal(-10),",",numSignal(2),",",numSignal(0))
-1, 1, 0
```

Procedure 2/3

```
# result is an implicit variable

proc sayHello(name: string): string =
  result = "Hello, "
  if name == "":
    result = "What's your name again?"
  else:
     result &= name

echo sayHello("Sergio")
Hello, Sergio
echo sayHello("")
What's your name again?
```

Procedure 3/3

```
# Declare an argument as var to allow changing its value within the procedure
proc isPossibleDivision(dividend: int, divisor: int, quotient: var float, remainder: var int): bool =
  if divisor == 0: return false
  quotient = dividend / divisor
  remainder = dividend mod divisor
  return true

var quotient: float = 0.0
var remainder: int = 0
stdout.write(isPossibleDivision(5, 2, quotient, remainder), ", ", quotient, ", ", remainder)
true, 2.5, 1

# Use a Tuple (or Seq) to return multiple values from a procedure
var divItems = (isPossible: false, quotient: 0.0, remainder: 0)
proc isPossibleDivision(dividend: int, divisor: int): (bool, float, int) =
  if divisor == 0: return (false, 0.00, 0)
  return (true, dividend / divisor, dividend mod divisor)

divItems = isPossibleDivision(5, 2)
stdout.write(divItems.isPossible, " ,", divItems.quotient, " ,", divItems.remainder)
true, 2.5, 1
```