# Datascript: Manual de referencia

# Bases

## Definir programa

Todo programa debe de tener la siguiente estructura. Donde el id representa el nombre del programa, sin embargo, program, end y main son keywords que tienen que respetarse.

```
program id:

main {
}
end
```

## Tipos de datos

### Int

```
program myProgram:
var i: int;

main {
  i = 0 - 10;
}
end
```

### Float

Para definir un float, agrega una f al final de la variable.

```
program myProgram:
var i: float;

main {
  i = 0.0f;
}
end
```

### Strings

```
program myProgram:
var i: string;

main {
  i = "HELLO";
}
end
```

## bool

true y false son los únicos valores booleanos, son sensibles a mayúsculas.

```
program myProgram:
var i: bool;

main {
  i = true;
  i = false;
}
end
```

## Arreglos

El tamaño del índice se define de 1 a N; Sin embargo, el índice de acceso para las variables estructuradas es base 0, de otra forma marcará el error de rango.

```
program myProgram:
var i[5]: int;

main {
  i[0] = 1;
  i[1] = 2;
  i[0] = i[1];
}
end
```

## Matrices

Para definir una matriz se define cada dimensión entre brackets. Al igual que el array, el del índice se define de 1 a N, pero el acceso es base 0.

```
program myProgram:
var i[5][5]: int;

main {
  i[0][1] = 1;
```

```
  i[1][1] = 2;
  i[0][1] = 1;
}
end
```

## Expresiones

Datascript maneja expresiones aritméticas y lógicas booleanas. Todas las expresiones pueden guardarse en variables, pero la mayoría de las funciones solo aceptan expresiones aritméticas como parámetros.

```
program myProgram:
var i: int;
var j: float;
var b: bool;

main {
  i = 10 + 2;
  j = 0 - 10.0f;
  b = 10 > j;
}
end
```

## Condiciones

Las condiciones solo aceptan valores booleanos o expresiones lógicas dentro de ellas para generar un control de flujo.

Un estatuto if puede ir solo o con su else correspondiente.

```
program myProgram:
var i: int;

main {
  if(1>2){
    i = 2;
  }else {
    i = 10;
  }

  if(true){
    i = 20;
  }
}
end
```

## Loops

Los loops solo aceptan valores booleanos o expresiones lógicas dentro de ellas para generar un control de flujo.

Datascript solo soporta por el momento while loops.

```
program myProgram:
var i: int;

main {
  if(1>2){
    i = 2;
  }else {
    i = 10;
  }

  if(true){
    i = 20;
  }
}
end
```

## Entrada de datos

Por medio de la función nativa readline, un usuario puede definir el valor de la función establecida en el segundo parámetro de la función. Como primer parámetro readline debe de llevar una expresión de tipo string.

```
program myProgram:
var i: int;

main {
  readline("N:", n);
}
end
```

## Salida de datos

La función print te permite generar una salida de datos, print acepta cualquier expresión aritmética o cualquier tipo de variable.

```
program myProgram:
var i: int;

main {
  print(10);
  print(10+1);
  print("HOLA");
```

```
}
end
```

## Funciones

Para definir una nueva función, declárala fuera del main con la siguiente estructura: function id(parámetros){ }, donde los parámetros se definen como variables, pero sin el "var" y sin punto y coma entre tipos. Además de los parametros una función puede definir variables locales.

Las funciones de datascript no soportan el estatuto return.

```
program myProgram:
var i: int;
  function myFun(cont:int, temp: int, anterior: int){
    var x: string;
  }

main {
}
end
```

## Llamadas a función

Para llamar una función, simplemente usa su identificador unico en el main o en una función, seguido de parentesis y los parametros necesarios. Cabe notar que si los parametros no son los mismos y del mismo tipo, el programa acabará con error.

```
program myProgram:
var i,j,k: int;
  function myFun(cont:int, temp: int, anterior: int){
    var x: string;
  }

main {
  i= 0;
  j= 0;
  k = 0;
  myFun(i,j,k);
}
end
```

# API nativa

## Estadistica básica

Datascript soporta 6 operaciones estadísticas básicas, max, min, mean, variance, stdev y range. Todas estas funciones nativas requieren de un vector o matriz de enteros o flotantes como parámetros y devuelven un número flotante.

Para introducir un vector o matriz, pon como parámetro el parámetro sin especificar sus dimensiones, por ejemplo: a1[3][3] => a1;

```
program myProgram:
  var a1[3][3], b:int;
  main {
    a1[0][0] = 20;
    a1[0][1] = 10;
    a1[1][0] = 30;
    b = max(a1);
    b = min(a1);
    b = mean(a1);
    b = variance(a1);
    b = stdev(a1);
    b = range(a1);
  }
  end
```

## Distribuciones

Datascript tiene funciones nativas para calcular la función de probabilidad y la función de probabilidad acumulada de tres diferentes distribuciones: normal, binomial y uniforme.

La función dBinomialPdf(X,P,N) regresa el valor de su función de probabilidad para X dado P probabilidad por intento y N intentos. La función dBinomialCdf(X,P,N) regresa el valor de su función de probabilidad acumulada para X dado P probabilidad por intento y N intentos.

```
program myProgram:
  var a1[3], b:int;
  main {
    a1[0] = 20;
    a1[2] = 10;
    a1[1] = 30;

    b = dBinomialPdf(2, 4, 0.2f);
    print("PDF: " + b);
    b = dBinomialCdf(2 ,4, 0.2f);
    print("CDF: " + b);
  }
  end
```

La función dUniformPdf(X,A,B) regresa el valor de su función de probabilidad para X dentro de los rangos establecidos por A y B. La función dUniformCdf(X,A,B) regresa el valor de su función de probabilidad acumulada para X dentro de los rangos establecidos por A y B.

```
program myProgram:
  var a1[3], b:int;
  main {
    a1[0] = 20;
    a1[2] = 10;
    a1[1] = 30;

    b = dUniformPdf(10.4f, 7,13);
    print("PDF: " + b);
    b = dUniformCdf(10.4f ,7,13);
    print("CDF: " + b);
  }
  end
```

La función dUniformPdf(X,vector) regresa el valor de su función de probabilidad normal para un valor X dentro del vector pasado como segundo parámetro. La función dUniformCdf(X,vector) regresa el valor de su función de probabilidad acumulada normal para un valor X dentro del vector pasado como segundo parámetro.

```
program myProgram:
  var a1[3], b:int;
  main {
    a1[0] = 20;
    a1[2] = 10;
    a1[1] = 30;

    b = dNormPdf(20,a1);
    print(b);
    b = dNormPdf(20,a1);
    print(b);
  }
  end
```

## Gráficas

Datascript soporta dos tipos gráficos, de línea y de barra, a partir de las funciones linePlot y barPlot respectivamente. Ambas funciones toman como parámetro dos vectores, uno de etiquetas para las coordenadas x y otro de datos enteros o flotantes. Los límites de las tablas se establecen a partir de los datos introducidos.

```
program myProgram:
  var series[3], b:int;
  var labels[3]: string;
```

```
main {
  labels[0] = "primero";
  labels[2] = "segundo";
  labels[1] = "tercero";

  series[0] = 10;
  series[1] = 13;
  series[2] = 4;
  linePlot(series,labels, "index.html");
}
end
```