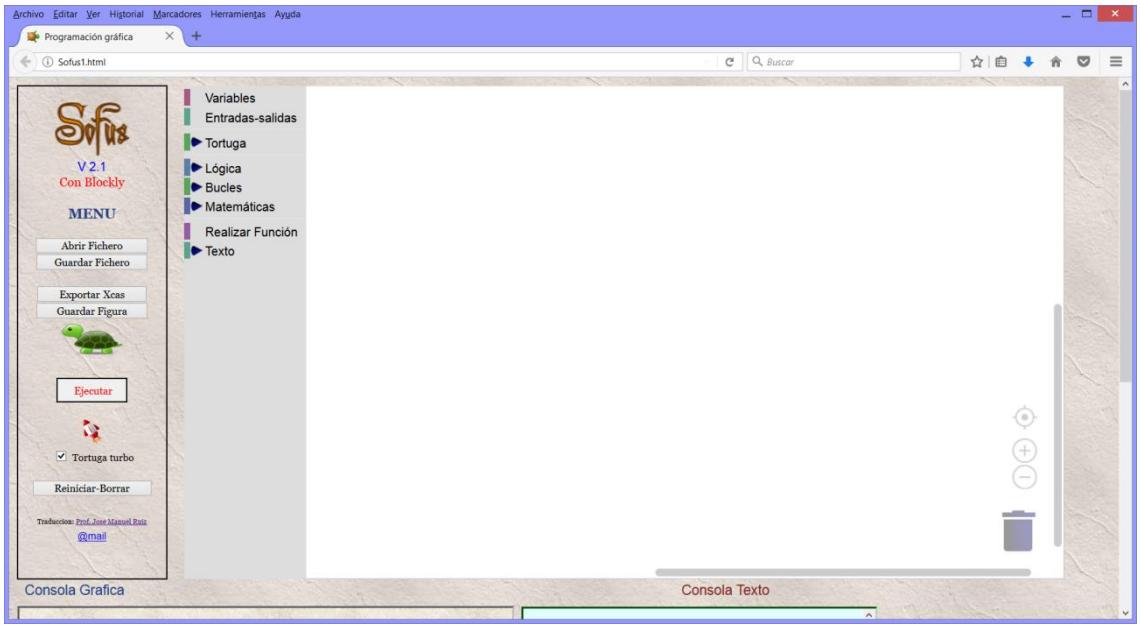
MANUAL DE SOFUS

UN ENTORNO DE PROGRAMACIÓN GRÁFICA BASADO EN BLOCKLY

(Versión 1.0)



Prof. José Manuel Ruiz Gutiérrez

Agosto 2017

Índice de contenidos

3.4.2. Longitud del texto.
3.4.3. Mayúsculas.
3.4.4. Invertir texto.
3.4.5. Búsqueda de la posición de un carácter en el texto.
3.4.6. Extraer un solo carácter de una cadena de texto.
3.4.7. Extraer sub cadenas de texto.
3.4.8. Recortar (eliminar) los espacios.
3.4.9. Imprimir texto.

3.5. Variables de tipo lista.
3.5.1. Creación de lista.
3.5.2. Creando listas a partir de cadenas de texto separadas por un carácter.
3.5.3. Manipulación de listas.
3.5.3.1. Verificar el tamaño de una lista.
3.5.3.2. Obtener elementos de una lista.
3.5.3.3. Búsqueda de elementos de una lista.

3.6. Vectores y matrices.
3.6.1. Punteros y vectores.
3.6.2. Matrices 2D.
3.6.2.1. Definición de matrices.
3.6.2.2. Operaciones con matrices.
3.6.2.3. Librería Matemáticas->Matrices->Matrices->Vectores.
3.6.3. Vectores y matrices tridimensionales 3D.

3.7. Instrucciones de control de flujo.
3.7.1. Instrucciones de tipo condicional.
3.7.1.1. Bloque Si.. hacer
3.7.1.2. Bloque de tipo Si-Si no
3.7.1.3. Bloques de varias condiciones.
3.7.2. Bucles de control.
3.7.2.1. Bloque repetir “n” veces.
3.7.2.2. Repetir por cada elemento de una lista.
3.7.2.3. Instrucción de Bucle tipo FOR NEXT.
3.7.2.4. Repetir mientras.
3.7.2.5. Repetir Hasta (Until).
3.7.2.6. Bucle contador (FOR NEXT).
3.7.2.7. Para cada elemento de una lista.
3.7.2.8. Bloque de terminación de bucle.
3.7.2.8.1. Continuar con la siguiente función.
3.7.2.8.2. Salirse del bucle.

3.8. Instrucciones de tipo matemático.
3.8.1. Constates matemáticas.
3.8.2. Funciones.
3.8.3. Operaciones matemáticas.
3.8.4. Números aleatorios.
3.8.5. Operaciones lógicas.

Sofus es una herramienta desarrollada por:

Autores:

- Alain Busser
- Patrice Debrabant
- Patrick Raffinat
- Florian Tobé

Depositada en: https://github.com/AlainBusser/Sofus

Versión del Documento V 1.0 Agosto 2017

Prof. José Manuel Ruiz Gutiérrez
j.m.r.gutierrez@gmail.com

1. ¿Qué es Sofus?

Sofus es una implementación de software realizada con carácter educativo en el IREM de la Réunion *Institut de recherche sur l'enseignement des mathématiques. Laboratoire d'informatique et de mathématiques*, (Instituto de investigación sobre la enseñanza de las matemáticas.Laboratorio de Informática y Matemáticas) perteneciente a la Universidad de la Isla de Reunion (Francia). Se puede descargar en el siguiente enlace Sofus_descarga en su versión francesa y en el siguiente enlace Sofus_español en la versión que he traducido y adaptado al español para su uso en la comunidad hispana.

La versión que voy a explicar esta basada en Blockly pero deriva, conceptualmente de otra anterior que implementaba un entorno de programación y algoritmia basada en un lenguaje de programación que permite escribir un programa en pseudocódigo.

La versión que voy a explicar esta basada en Blockly que como sabemos es una potente y útil aplicación que está siendo muy utilizada en diversas implementaciones de programación grafica como es el caso de Herramientas de Programación de Arduino, Diseño Gráfico para aplicaciones 3D, algebra y Geometría dinámica.

Existe una versión de Sofus para trabajar en línea que esta residente en un servidor de la Universidad… cuyo enlace es el siguiente.

¿Qué ventajas nos ofrece Sofus?

- Un entorno de programación gráfica ampliamente difundido y en constante desarrollo.
- Facilidad para la enseñanza de los rudimentos de la programación.
- Capacidad de generar código compatible con Phyton, JavaScript, y otros
- Facilidad de desarrollo de algoritmos de manera gráfica.
- Disponibilidad de una librería que permite el manejo de una “tortuga” que en un área grafica de la pantalla facilita el desarrollo de aplicaciones de salida gráfica.

Esta herramienta está orientada a los cursos de Informática o Asignaturas relacionadas con las TIC´s en las enseñanzas ESO, Bachillerato y en Primeros Cursos de estudios Universitarios en áreas de conocimiento relativas a la programación básica.

En la Web del IREM existen numerosos artículos y ejemplos de programas sobre Sofus, gran parte de este material lo usaré en el desarrollo de esta Guía de Usuario y Manual de Practicas.

Es importante decir que existen otras herramientas que he utilizado con mis alumnos y que igualmente son meritorias pero he seleccionado esta por su sencillez de utilización.

La programación algorítmica es una poderosa herramienta para que los estudiantes se familiaricen con la programación. Con el desarrollo de los lenguajes gráficos la

programación sintáctica (escribiendo líneas de texto con comandos y variables) ha sido relegada a un segundo plano. Ya no es necesario memorizar instrucciones escritas en lenguajes nemotécnicos y ordenadas en estricta formación. Ahora basta con abrir una caja de herramientas que contiene una colección de bloques gráficos que encajan unos en los otros y permiten establecer el flujo de los datos y otorgar de manera cómoda los métodos de comunicación (interfaces) con el teclado, la pantalla, el ratón, etc… Esta es la nueva era de la programación.

En el MIT se han desarrollado muchas aplicaciones basadas en esta idea de la "Programación Gráfica", ejemplo de estas es Scratch o Snap! Que han revolucionado de manera decisiva los conceptos de "Interface de Programación". Es cierto que los poderosos lenguajes actuales poseen entornos gráficos de programación, peor estos pertenecen a un estrato de "alto nivel" de cualificación que resulta complicado de usar en los niveles básicos de la educación.

1.1. Unas ideas básicas sobre Blockly

Blockly es una aplicación creada bajo los auspicios de Google for Education siendo una potente "biblioteca para la construcción de editores de programación visual". Blockly está disponible para descargar.

Blockly es una biblioteca que agrega un editor de código visual a las aplicaciones web y Android. El editor Blockly utiliza bloques gráficos entrelazados para representar conceptos de código como variables, expresiones lógicas, bucles y más. Permite a los usuarios aplicar principios de programación sin tener que preocuparse por la sintaxis o la intimidación de un cursor parpadeante en la línea de comandos.

1.1.1. Creación de una aplicación Blockly

Desde la perspectiva de un usuario, Blockly es una forma intuitiva y visual de crear código. Desde la perspectiva de un desarrollador, Blockly es esencialmente un cuadro de texto que contiene código generado por el usuario sintácticamente correcto. Blockly puede exportar bloques a muchos idiomas, incluyendo estas opciones populares:

JavaScript
Phyton
PHP
Lua

A continuación se muestra un desglose de alto nivel de lo que entra en la construcción de una aplicación Blockly:

- Integre el editor Blockly. El editor Blockly en su forma más simple consiste en una caja de herramientas para almacenar tipos de bloques y un espacio de trabajo para organizar bloques. Obtenga más información sobre la integración de Blockly en los documentos de Inicio para Web o Android.
- Crea los bloques de tu aplicación. Una vez que tienes Blockly en tu aplicación, debes crear bloques para que tus usuarios puedan codificar, y luego agregarlos a

tu caja de herramientas Blockly. Obtenga más información sobre cómo crear una vista general de bloques personalizados.

- Construir el resto de la aplicación. Por sí mismo, Blockly es sólo una manera de generar código. El corazón de su aplicación es decidir qué hacer con ese código.

1.1.2. Las fortalezas de Blockly y otras opciones

Blockly es uno de un creciente número de entornos de programación visual. Decidir cuál usar en su aplicación es un paso importante, así que aquí están algunas de las mayores fortalezas de Blockly para ayudarle a tomar la decisión:

Código exportable. Los usuarios pueden extraer sus programas basados en bloques a lenguajes de programación comunes y suavemente la transición a la programación basada en texto.

Fuente abierta. Todo acerca de Blockly está abierto: se puede bifurcar, hackear y usarlo en sus propios sitios y aplicaciones de Android.

Extensible. Modifique Blockly para adaptarse a sus necesidades añadiendo bloques personalizados para su API o eliminando bloques y funcionalidades innecesarios.

Muy capaz. Blockly no es un juguete. Puede implementar tareas de programación complejas como calcular la desviación estándar en un solo bloque.

Internacional. Blockly ha sido traducido a más de 40 idiomas, incluyendo versiones de derecha a izquierda para árabe y hebreo.

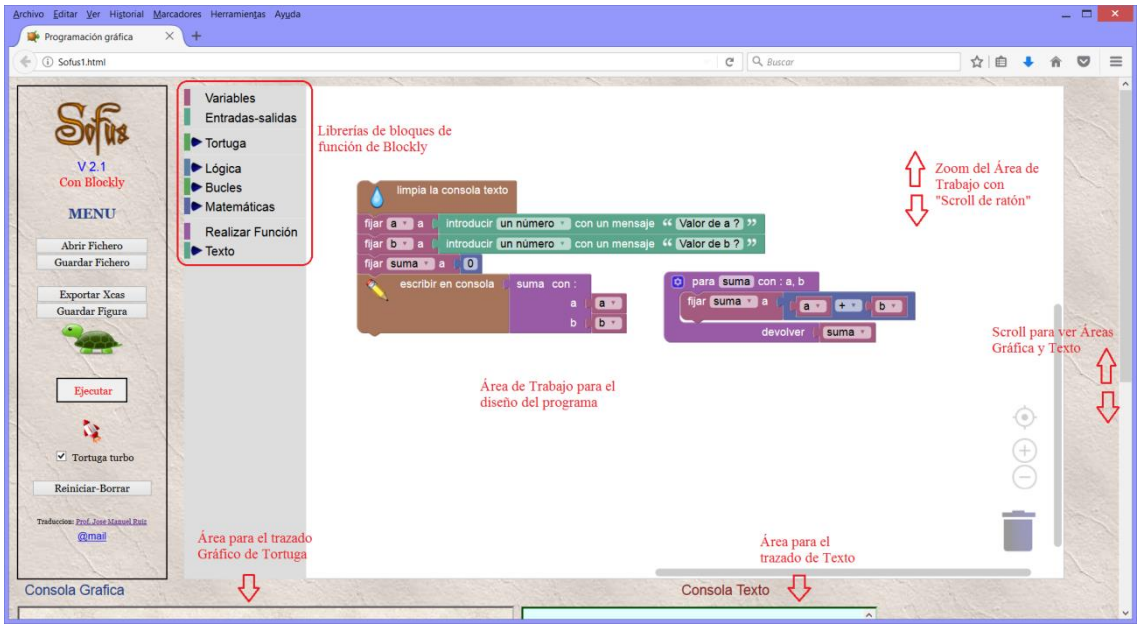
Incluso con todos esos aspectos positivos, Blockly no es la solución para cada aplicación. Éstos son algunos otros editores visuales que puede ser útil:

Scratch Blocks: Diseñado por la gente detrás del Scratch del MIT y construido sobre la base de código Blockly, Scratch Blocks ofrece un modelo de programación simplificado ideal para principiantes jóvenes.

Droplet: El editor de programación gráfica que las funciones lápiz código, su característica distintiva es la capacidad de convertir de código en bloques.

Snap: Un lenguaje de programación gráfico inspirado en Scratch, no es una biblioteca sino una aplicación completa con un entorno de ejecución integrado.

2. Explicación del entorno Sofus



2.1. Áreas de trabajo.

En la figura anterior se muestra las áreas de trabajo de la aplicación

2.1.1. **Menús**.

En la parte izquierda aparece recuadrado el menú de opciones de Sofus. Cada uno de los botones realiza las siguientes funciones:

Abrir fichero. Permite abrir un fichero anteriormente creado y guardado en el una carpeta.

Guardar Fichero: Permite guardar el trabajo que tengamos realizado en el área de edición poniendo un nombre al fichero que lleve la extensión *.sof. El fichero se guardara en la "Carpeta de Descargas" del Ordenador.

Exportar Xcas: Permite generar el texto correspondiente a la sintaxis del lenguaje Xcas. El texto lo escribe en la "Consola de Texto" desde la que es posible copiar y pegar en la ventana de edición de Xcas

Guardar Figura: Permite guardar la imagen que aparece en la "Consola Gráfica" de la construcción de los bloques del área de edición. Lo graba en formato *.svg"

Ejecutar: Permite ejecutar la aplicación construida en el área de edición.

Reiniciar-Borrar: Con eta opción se limpia el área de edición y limpiar la "Consola Gráfica" pero no la "Consola de Texto".

Opción Tortuga Turbo. Permite ralentizar el movimiento de la tortuga en su trazado.

Librerías de Bloques. En esta área aparecen n formato "árbol" las distintas librerías que incluyen los bloques clasificados por su tipo y naturaleza.

Aquí se muestran las categorías principales de los bloques de librería. En los que aparece el triángulo azul es porque se subdividen en otras categorías.

La librería "Tortuga" contiene todos los bloques de función que permiten seleccionar acciones en el trazado y aspecto de la tortuga que dibuja en la parte de "Consola Gráfica"

A lo largo de este manual se irán comentando cada una de las librerías a la vez que se utilizan en determinados ejemplos.

2.1.2. **Consola Gráfica.**

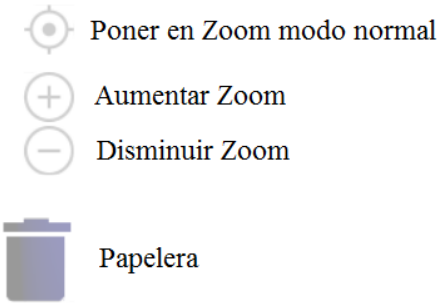
Esta área es en la que se ejecutan las aplicaciones que incluyen acciones de trazado grafico (usando la tortuga). Para ver completa esta área basta con hacer un scroll en la pantalla y desplazarse hacia abajo

2.1.3. **Consola Texto**.

En ella se escriben los mensajes y textos que se generan e la ejecución de una aplicación. Para ver completa esta área basta con hacer un scroll en la pantalla y desplazarse hacia abajo

2.1.4. **Área de Trabajo.**

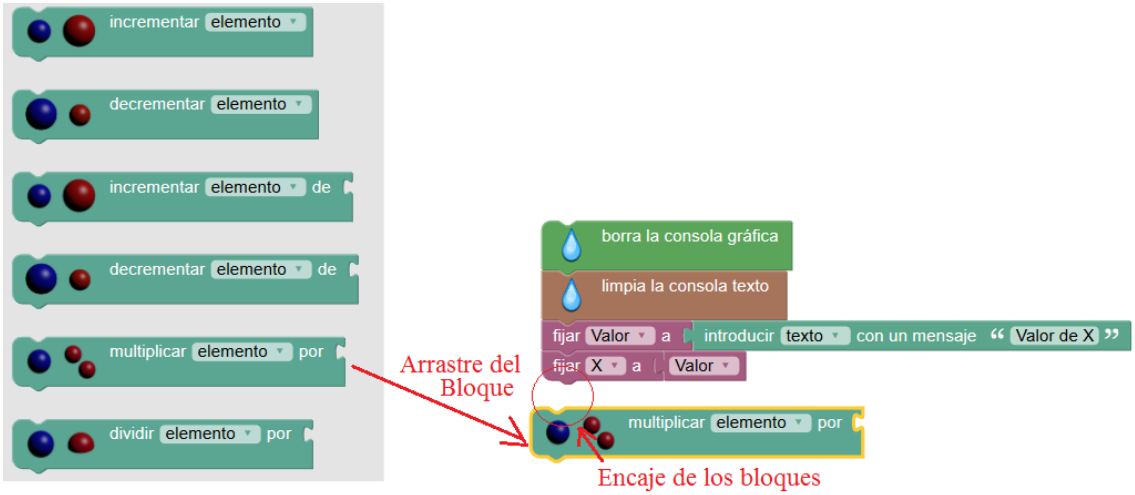
Esta área es la que permite el diseño de la aplicación a base de unir bloques que configuran el algoritmo. Como atributo importante diremos que para hacer zoom en el área basta con mover la rueda de Scroll del ratón. También se puede hacer pulsado en las áreas que se indican en la siguiente imagen. Para eliminar bloques se "Arrastrarán "y depositarán en la papelera. También es posible borrar elementos arrastrándoles al área de librerías y soltando en ella el bloque.



Borrado de un bloque arrastrado a la papelera

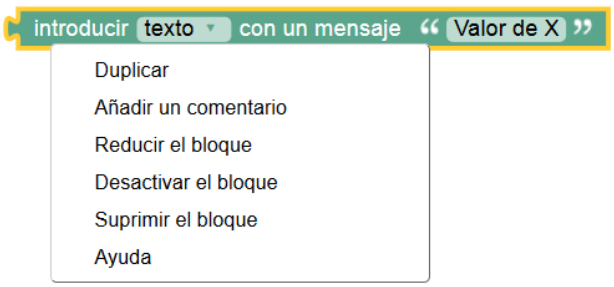
2.2. Operaciones básicas que se pueden realizar en el área de trabajo en el modo de edición.

2.2.1. **Incorporar Bloque**. Esta tarea es tan sencilla como buscar el bloque en las librerías, seleccionarlo con el ratón y mantenido pulsado el botón "derecho" del ratón se arrastra y suelta en la posición que queramos.



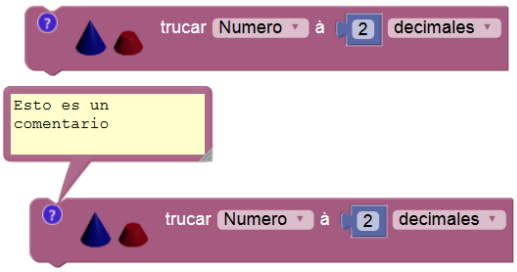
2.2.2. **Unir Bloques** para unir los bloques basta con aproximarlos y cuando su contorno se resalte significa que al soltar se queda "pegado" el bloque

2.2.3. **Menú Contextual de Bloques.**



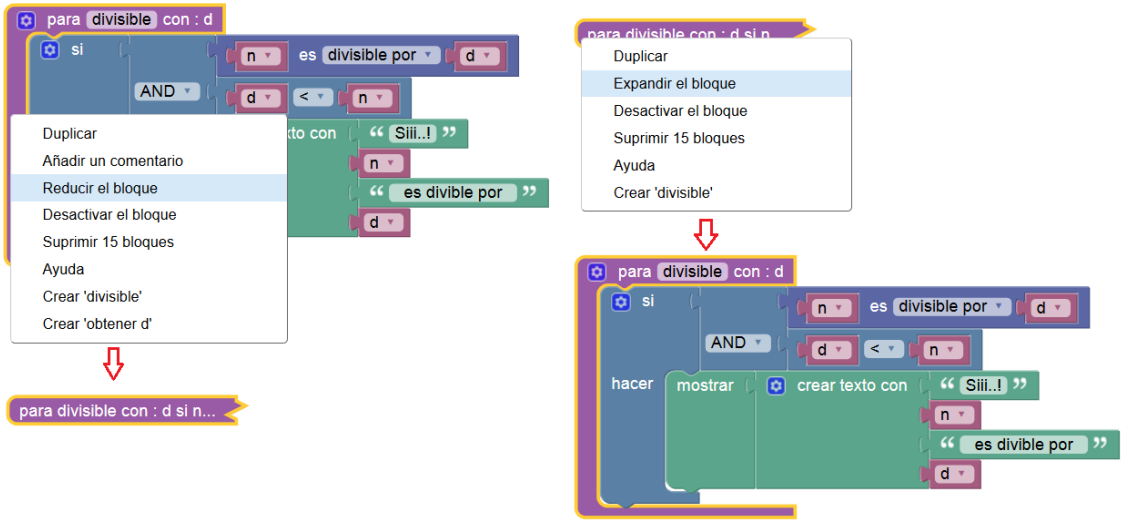
Cuando hacemos clic con el botón derecho estando sobre un bloque aparecerá el menú contextual de la figura. En algún caso aparece alguna opción más o también alguna opción inactiva.

2.2.4. **Duplicar**: Copia el bloque seleccionado



Añadir un comentario: Despliega una ventana sobre la que se escribe un texto. Al seleccionar esta opción aparece una pequeña interrogación que al pulsar sobre ella permite escribir en un "bocadillo" un texto a modo de comentario.

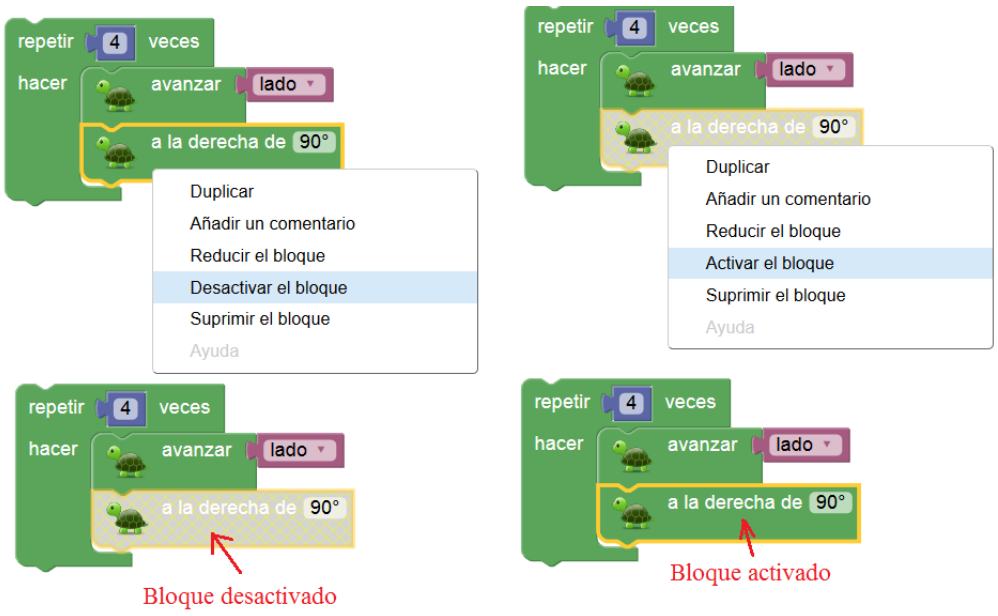
2.2.5. **Reducir el bloque**: Contrae el contenido del bloque con el fin de ocupar menos espacio



2.2.6. **Expandir Bloque:** Expande un bloque que este contraído.

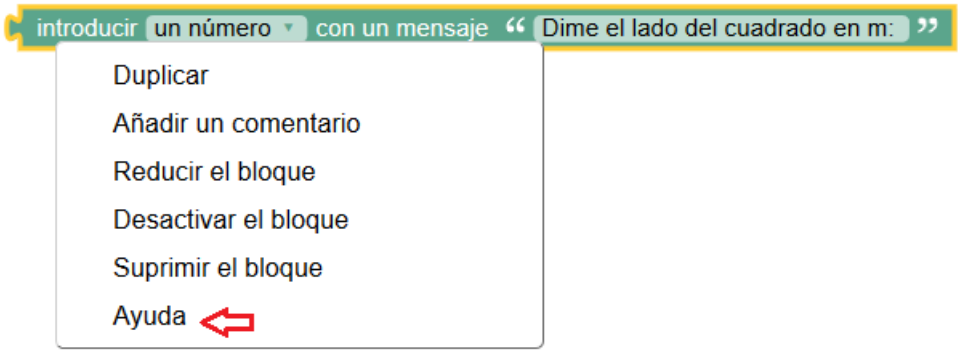
2.2.7. **Desactivar un Bloque:** Deja inoperativo un bloque en el diseño.

2.2.8. **Activar un bloque:** Devuelve operatividad a un bloque



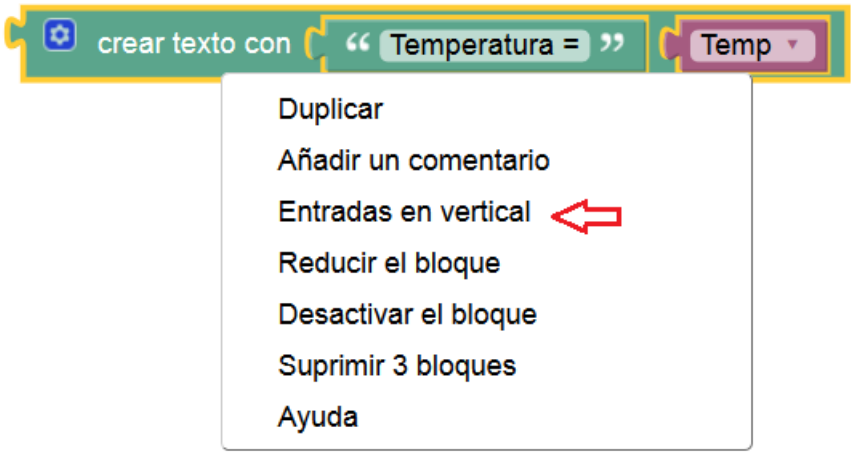
2.2.9. **Suprimir el bloque/es:** Suprime (borra) el o los bloques que se han marcado.

2.2.10. **Ayuda:** En los casos que la opción este activada se abrirá una pantalla con una página HTML de ayuda.

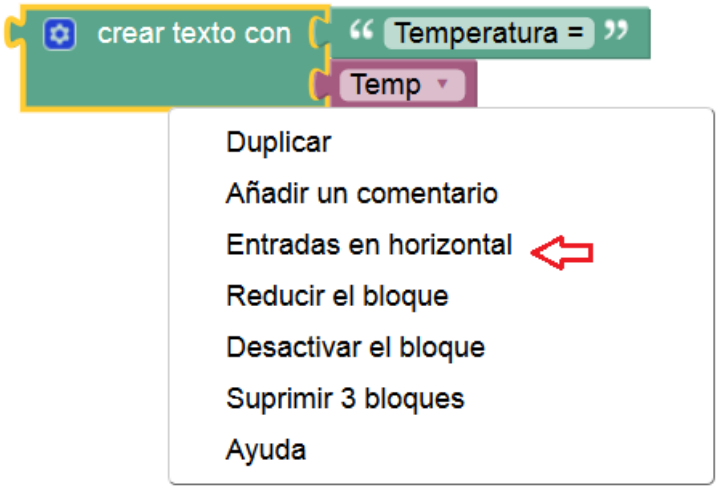


2.2.11. **Colocación de parámetros.**

Determinados bloques como el de “crear texto con” o “crear una lista con” tienen una opción en su menú contextual que nos permite colocar los parámetros en horizontal o en vertical.



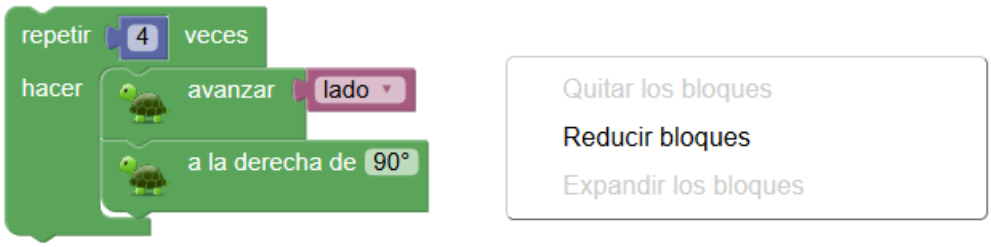
Colocación de los parámetros en horizontal



Colocación de los parámetros en vertical

2.3. Menú Contextual en el área de Edición.

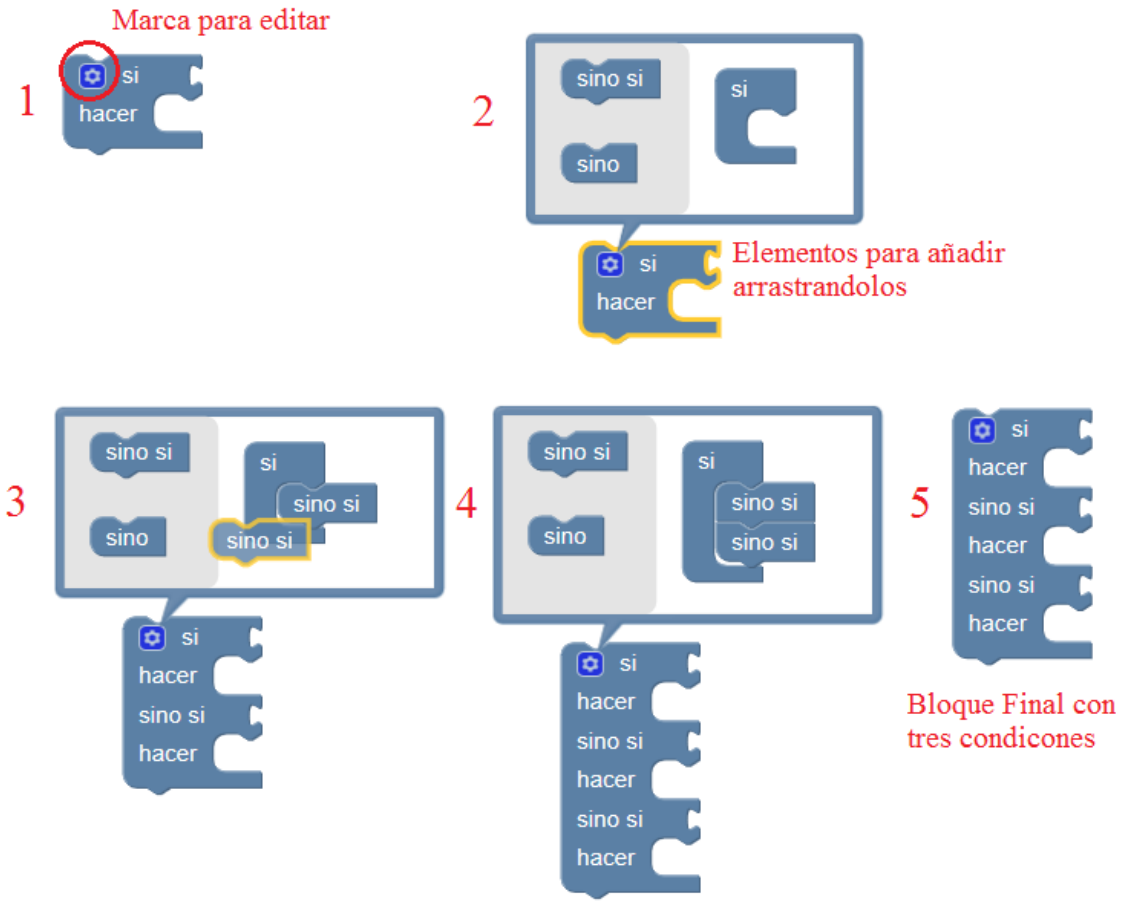
Si pulsamos el botón derecho del ratón estando en el área de edición sin tocar ningún bloque podremos optar al menú contextual de edición que presenta tres opciones.



2.3.1. **Reducir Bloques:** Contra todos los bloques que haya en el área.
2.3.2. **Expandir los Bloques:** Expande los bloques contraídos.

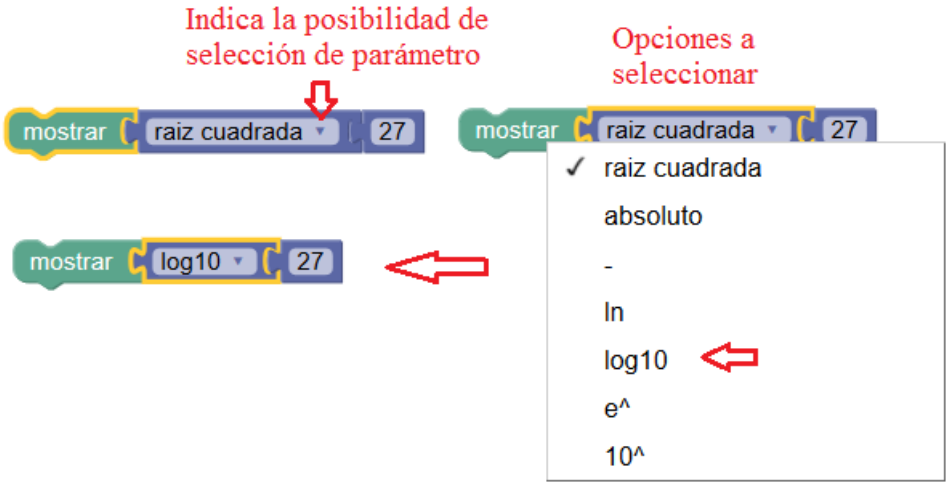
2.4. Añadir parámetros en un bloque.

Determinados bloque presentan una marca en la parte superior izquierda que nos indica que es posible editar y añadir parámetros al bloque. Tal es el caso, por ejemplo del bloque condicional, en el que vemos que al pulsar sobre la “Marca para editar” parámetros se despliega una pequeña ventana desde la que podemos añadir elementos nuevos en la función de comparación.



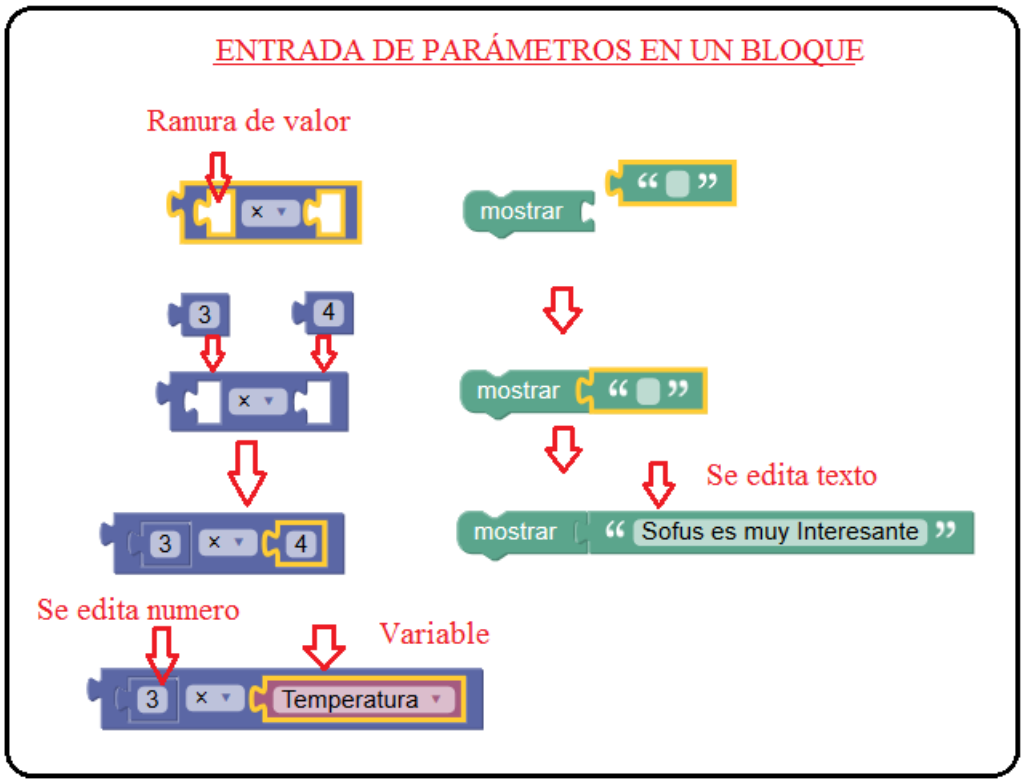
2.5. Selección de Opciones mediante menú desplegable en un bloque

Determinados bloque tienen la posibilidad de seleccionar parámetros u opciones pulsando sobre un área que despliega un menú de persiana del que podemos seleccionar una de ellas.



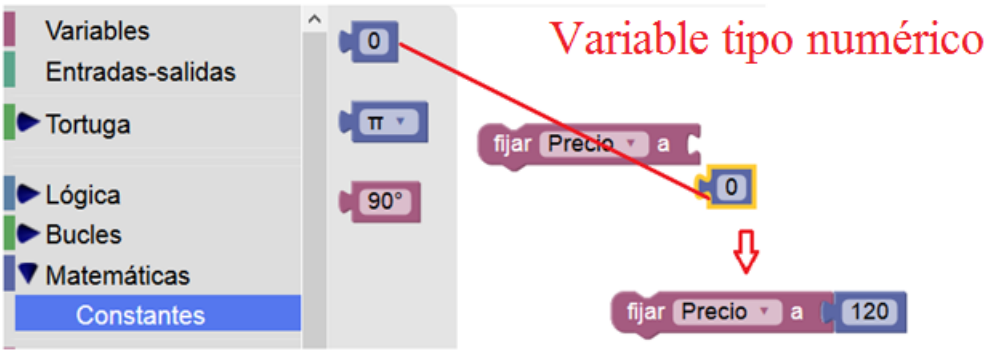
En la figura el bloque de cálculo permite modificar la operación a realizar con el dato (27)

2.6. Encaje de valores sobre un bloque.



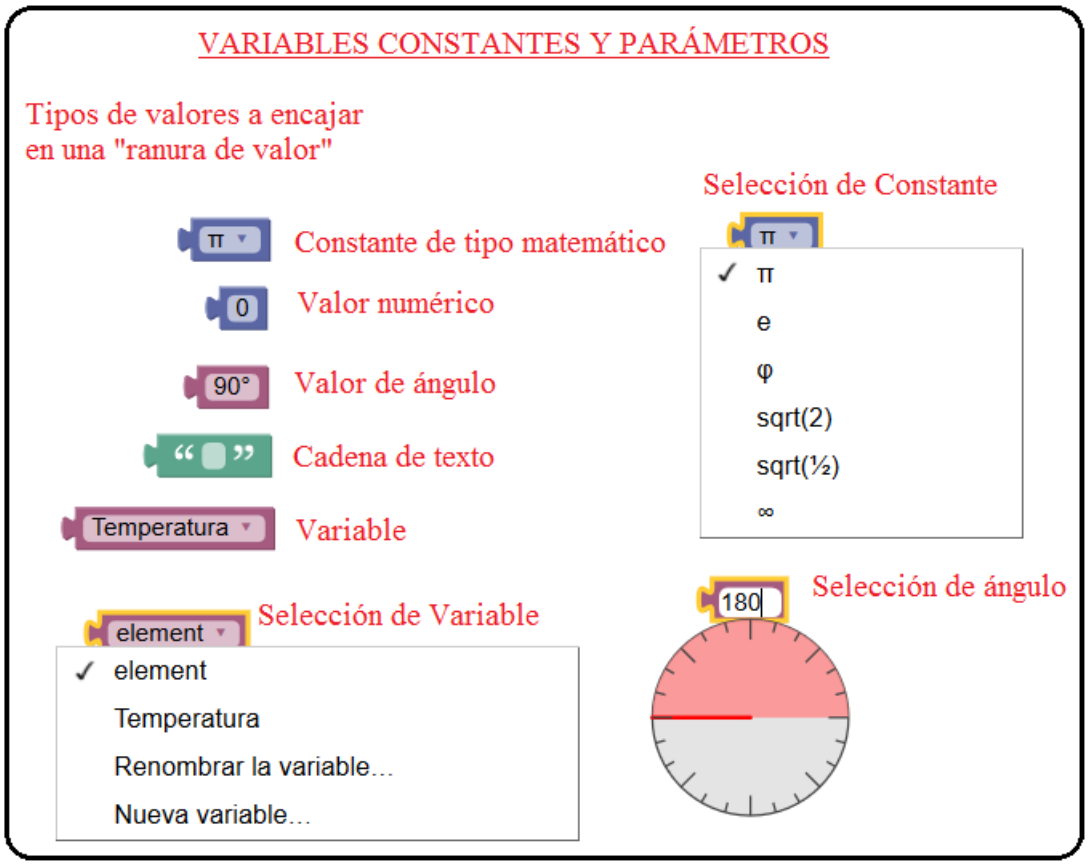
Determinados bloque tienen la posibilidad de conectar a ellos un valor que puede ser una constante (texto o valor numérico) para que se efectúen determinas operaciones también es posible encajar en estas ranuras de valor variables.

En determinados bloques solo se pueden encajar valores de un tipo por lo que si intentamos añadir un valor que no sea del tipo adecuado no se “pegara a la ranura”.



2.7. Tipos de parámetros y variables.

En la figura se muestran los tipos más usuales existiendo alguno más.



Los parámetros y las variables se pueden editar y modificar

3. Elementos básicos de Programación.

En este apartado vamos a repasar los principales conceptos de carácter general que son aplicables a los distintos programas que queramos diseñar.

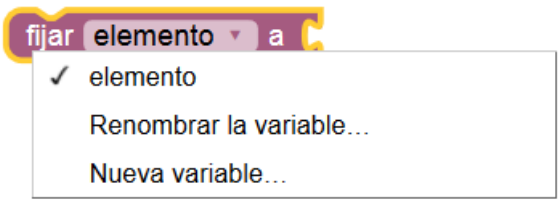
La realización de un algoritmo siempre lleva implícitas tareas comunes como son:

- Definición de las variables
- Ordenes de Entrada /salida de datos.
- Manejo y Control del trazador gráficos (Tortuga)
- Tratamiento de textos.
- Variables tipo "Lista"
- Variables tipo "Matriz" o "Vector"
- Operaciones matemáticas
- Instrucciones de Control de Flujo
- Realizar Funciones
- Cálculo Formal

3.1. Variables.

Las variables son elementos básicos en la definición de un algoritmo. Para definirlas con Sofus recurrimos a la pestaña "**Variables**" del menú de bloques y de allí extraemos el bloque "**fijar *elemento* a**" que nos permitirá seleccionando sobre el área desplegable "*elemento*" , "*renombrar la variable*"" -ponerle en nombre que queramos- o crear una nueva variable *"Nueva variable*.

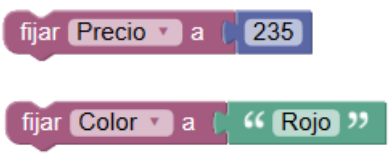
3.1.1. Definición de variable



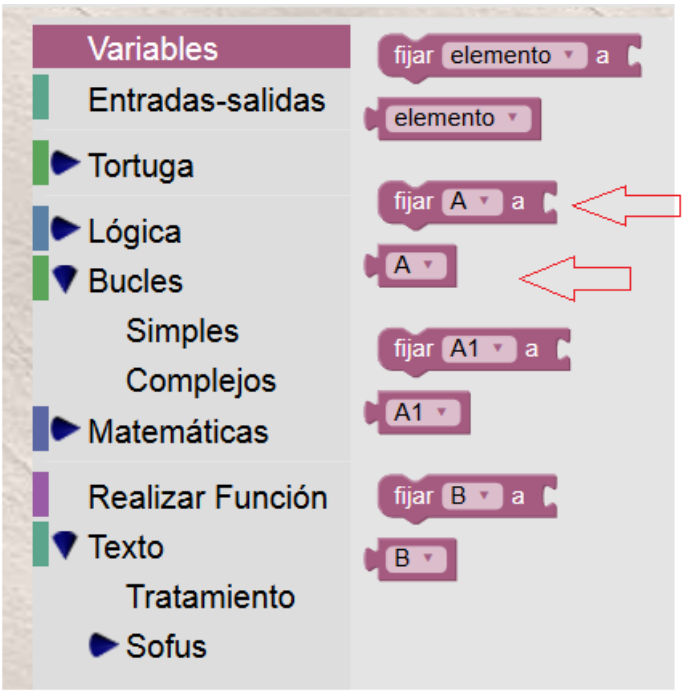
Al definir una variable esta se incorporara a la lista de bloques de la librería "Variables" con su nombre en dos bloques uno "**fijar** <*nombre de la variable*> **a**" y el otro con el nombre de la variable .

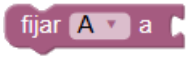
Las variables podrán ser de dos tipos "*texto*" o "*un numero*".

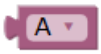
Una vez creada la variable se puede fijar ésta a un valor añadiéndole los datos a la conexión de la derecha.



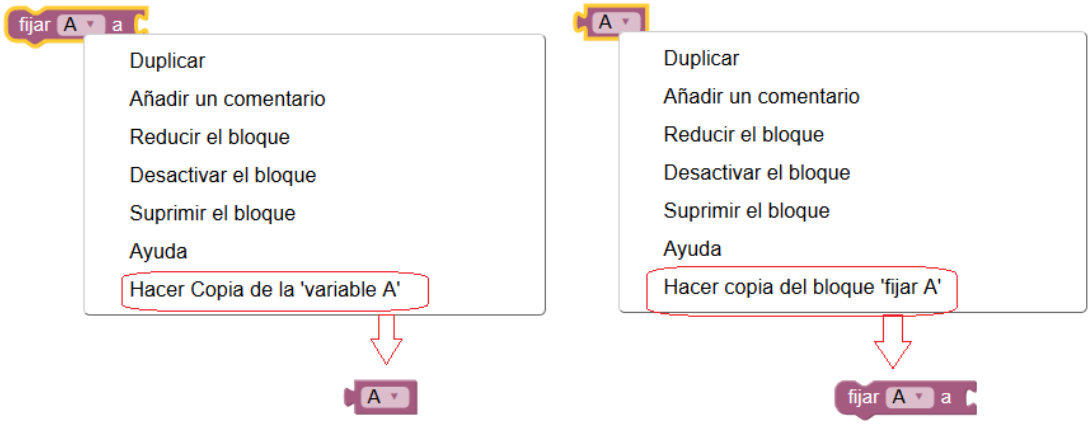
Una vez que tenesmos creada una variable aparecerá está en la pestaña de bloques correspondiente “Variables”

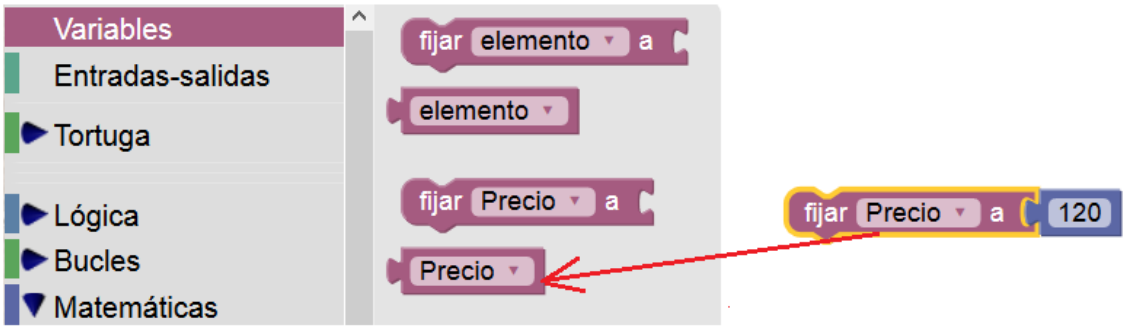


De esta pestaña podemos recoger y arrastrar las formas en las que se presenta la variable: “fijar” 

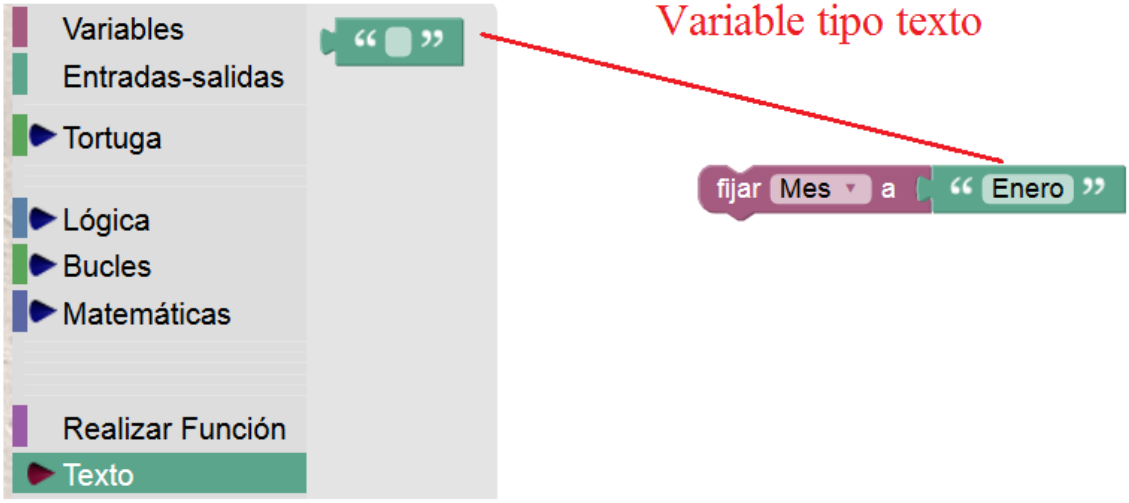
y como tal 

Si nos colocamos sobre estos bloques y pulsamos el botón derecho del ratón, aparecerán menús contextuales que permiten la generación del bloque homólogo

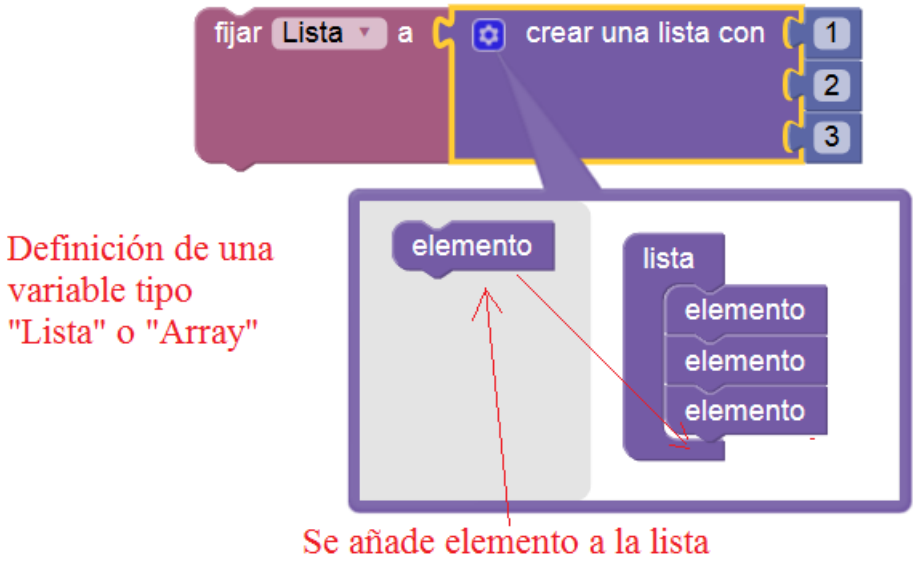




Ejemplo de designación de valor a la variable “Precio”



Ejemplo de designación de valor a la variable “Mes”



3.1.2. **Variables tipo lista**

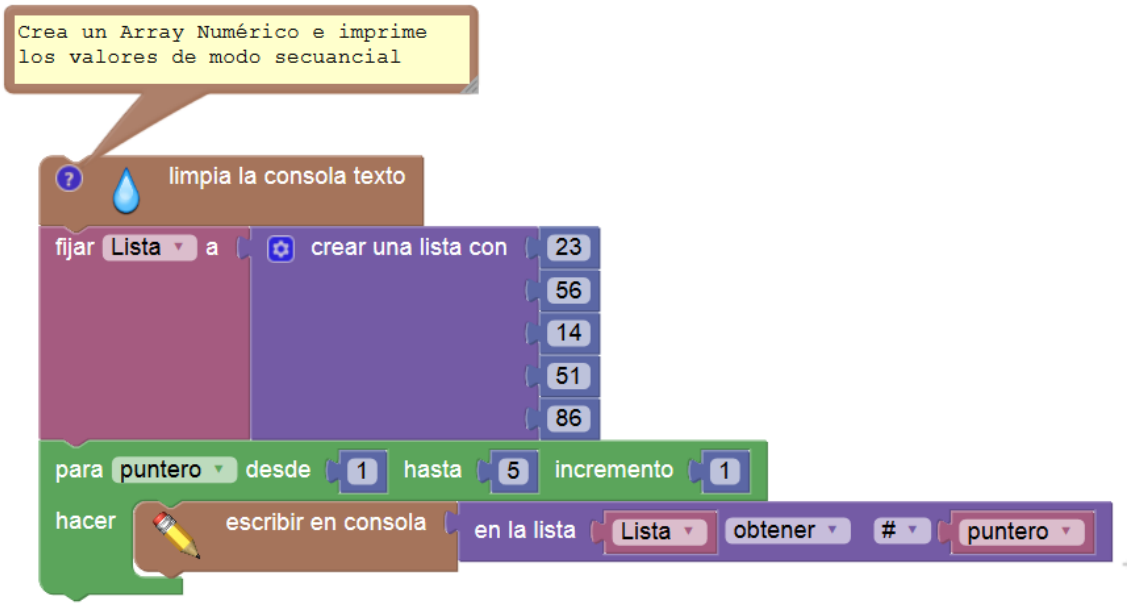
Es posible también asociar a una variable una lista, es decir crear una variable de tipo “Lista”. Estas variables son un conjunto de datos “**array**” que son tratados en función

de la posición que tengan dentro e la lista. En la imagen anterior hemos asignado a la variable “lista” tres valores de tipo numérico.

El bloque “**crear una lista con**” que se encuentra en la carpeta “**Matemáticas -> Lista -> Creación**”

En el siguiente ejemplo se muestra la manera de crear una lista de 5 elementos y mediante una instrucción e tipo bucle “**para**…” se escribe en la Consola de Texto el valor de cada uno de los elementos de la lista.

En la lista los elementos son: 1º=23, 2º=56, 3º=14, 4º=51 y 5º=86



3.2. Entrada y salida de datos.

Para la realización de un algoritmo siempre tendremos que recurrir a la entrada de datos y a la salida de resultados. Lo que equivale a leer y escribir datos.

La librería que vamos a usar para ello es la denominada “Entradas-salidas” que está formada por los bloques que figuran en la siguiente imagen. Se incluyen bloques para la escritura tanto en la ventana propia de Blokly como para la Consola de Texto de Sofus.

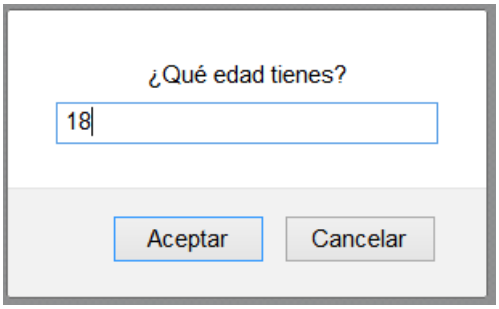
La salida de datos –trazado- en la ventana Consola Grafica se explicara posteriormente cuando hablemos de la “Tortuga”.

3.2.1. **Introducir texto o número.**

Para introducir un valor de texto o un valor numérico se utiliza el bloque

que al ejecutarse lo que hace es abrir una pequeña ventana con el texto que coloquemos para pedir el valor que se escribe en la caja de datos y después se pulsa aceptar.

Al ejecutarse aparece esta ventana:



Se pulsa Aceptar u la variable **Edad** adquiere el valor de 18.

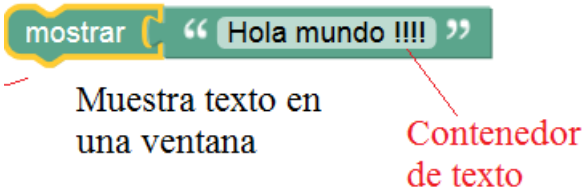
3.2.2. **Salida de datos**

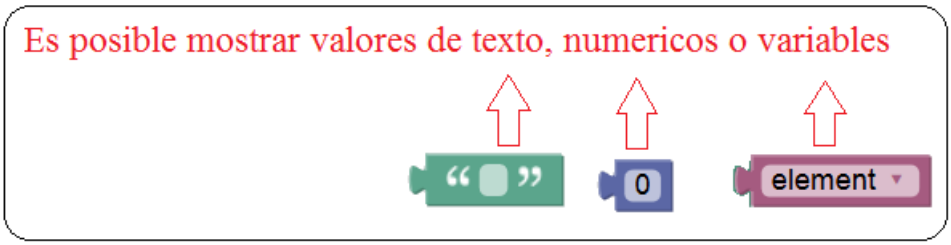
Los dos primeros bloques son 

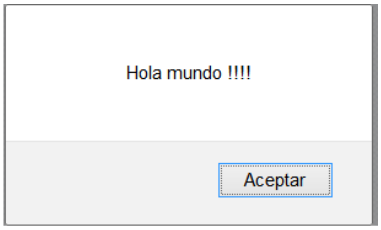
escribir un dato numérico y para escribir una cadena de texto 

3.2.2.1. Salida de datos en la ventana característica de Blocky

Con el bloque “mostrar” podemos enviar a la ventana de Blockly un valor

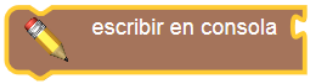






Ventana de Blockly

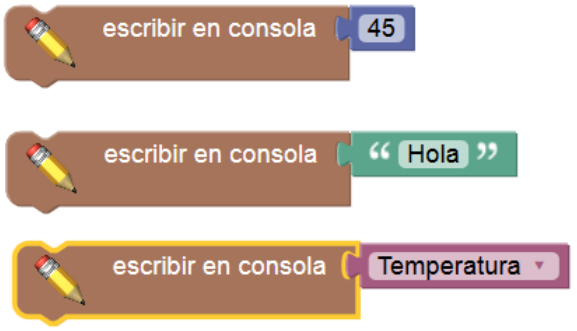
3.2.2.2. Salida de datos a través dela Consola de Texto

Con el bloque 

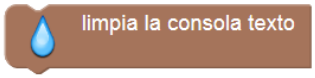
mostramos un texto en la Consola de Texto.

Podemos escribir el tipo de variable o dato que queramos.

Para mostrar una matriz en la Consola de Texto se utiliza el bloque 

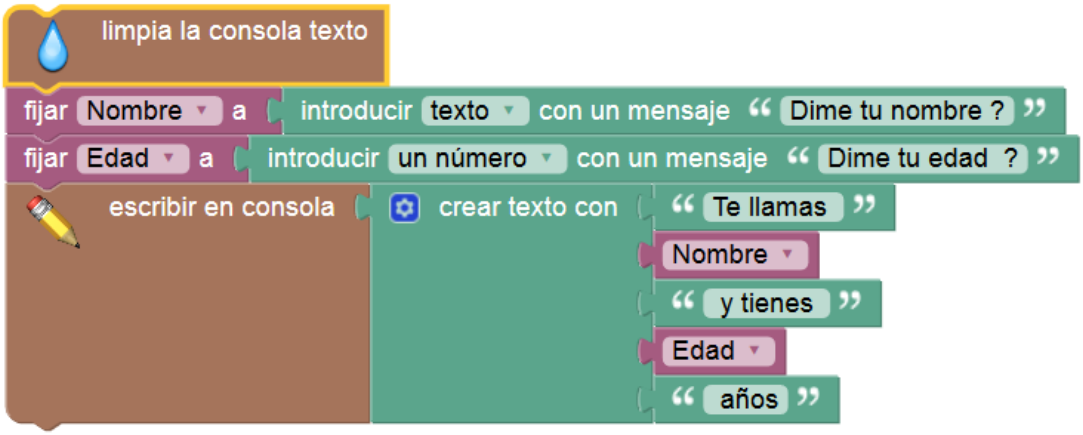


Para borrar el contenido de la Consola de texto se utiliza el bloque



3.2.2.3. Un ejemplo de entrada/salida de datos

En el siguiente ejemplo vemos algunas instrucciones de entrada y salida de texto

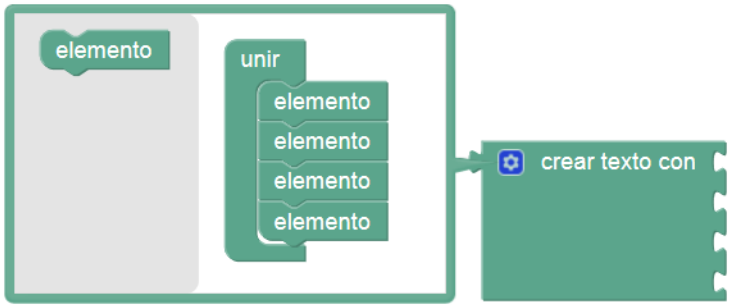


Como podemos ver las vaiables de entrad Nombre y Edad se nos piden mediente el bloque“**introducir texto con mensaje..”**

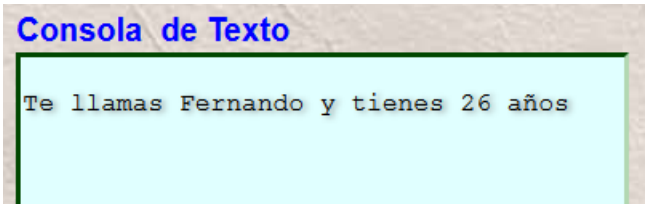
Seguidamente para escribir los valores se recurre el bloque “escribir en consola” al que se ha conecatdo un bloque “crear texto con” que lo que hace es crear una línea de textos con distintos tipos de valores:

“*Te llamas*” +**Nombre** + “*y tienes*” **+ Edad** +”*años*”

El bloque “**crear texto con**” se encuentra en la librería “**Texto -> Tratamiento**”

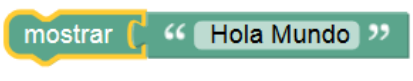


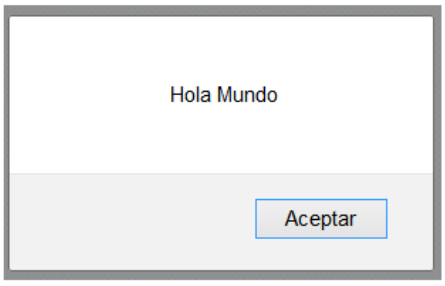
Cuando se ejecuta el programa anterior en la Consola de Texto apece el siguiente mensaje:

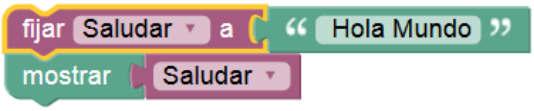


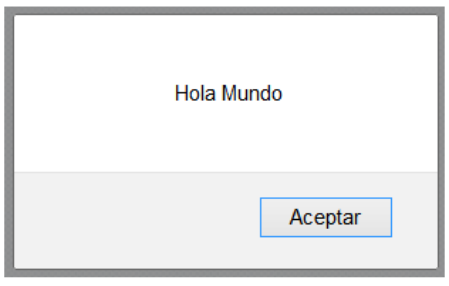
A continuación vemos otro sencillo ejemplo en el que se muestran dos formas de enviar un texto. En la primera se manda directamente “Hola Mundo” y en la segunda se hace a través de la variable “**Saludar**” que previamente la hemos cargado con el valor “Hola Mundo”

Dos maneras de saludar al "Mundo"





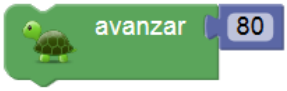


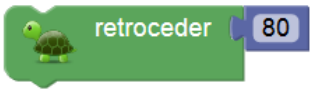


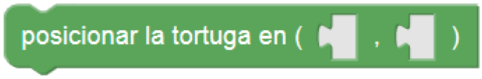
3.3. Manejo y Control de la Tortuga.

Bloques de función que están relacionados con la tortuga 

Desplazamiento.

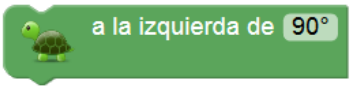


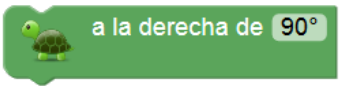


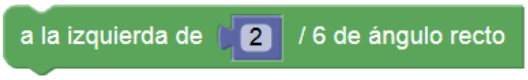


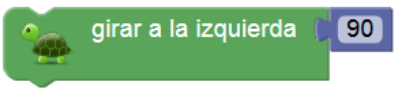
- Estas instrucciones permiten el desplazamiento de la tortuga en el área de la Consola Gráfica.
- Los parámetros son de tipo numérico pero también se pueden colocar variables, Por defecto aparecen con el valor "80". La unidad de desplazamiento es el pixel.
- Con la orden Posicionar podemos llevarnos la tortuga a una punto determinado de la Consola Gráfica

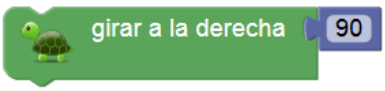
Rotaciones

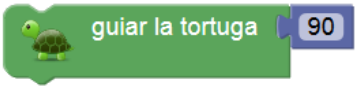


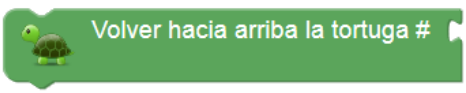






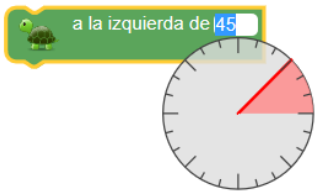




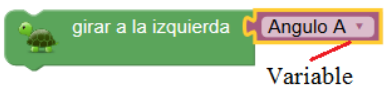


- Este grupo de instrucciones permiten el giro de la Tortuga en los términos que se especifica en el texto de la propia instrucción, teniendo en cuanta que el giro puede ser a la derecha o a la izquierda.

- En el caso de que el ángulo este dentro del propio bloque bastara con apuntar con el ratón sobre la ventana de ángulo para que aparezca

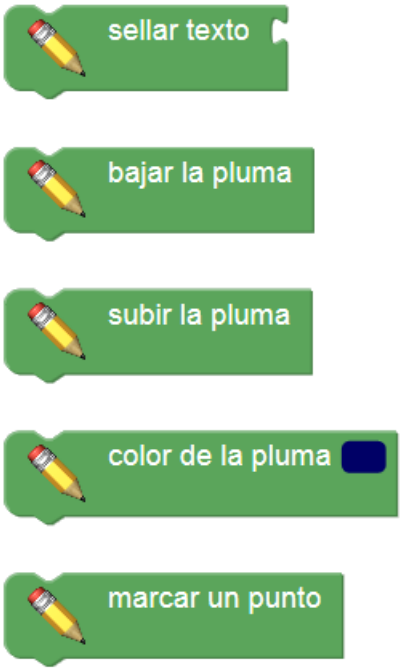


- El valor de la variable indica el angulo a girar.



- La tortuga puede girarse hacia arriba. El parámetro en este caso es el identificativo de la tortuga (un valor numérico)

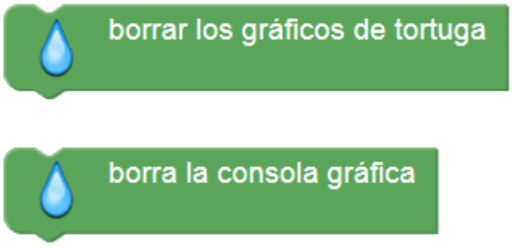
Diseño: Estilo



- En este aparado tenemos bloques que permiten cambiar el color de la pluma.
- La pluma puede subirse (no dibuja) o bajarse (dibuja).
- Es posible marcar un punto con el bloque correspondiente.
- Con la orden "Sellar texto" podemos escribir un texto en el área gráfica de la Consola Gráfica

---

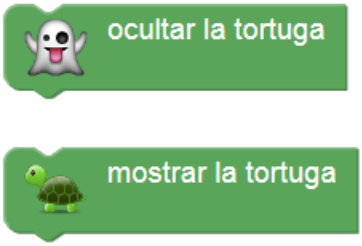
Diseño: Borrador.



- Con estas dos instrucciones se consigue borrar el contenido de la "Consola Gráfica".
- La instrucción "borra la consola.." reinicia y sitúa la tortuga en su origen, la de "borrar los gráficos.." solo borra lo dibujado y mantiene la tortuga en su posición

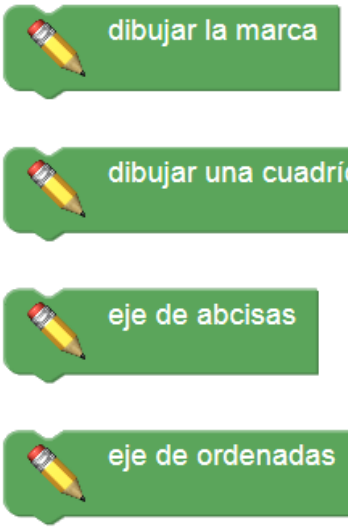
---

Diseño: Aspecto



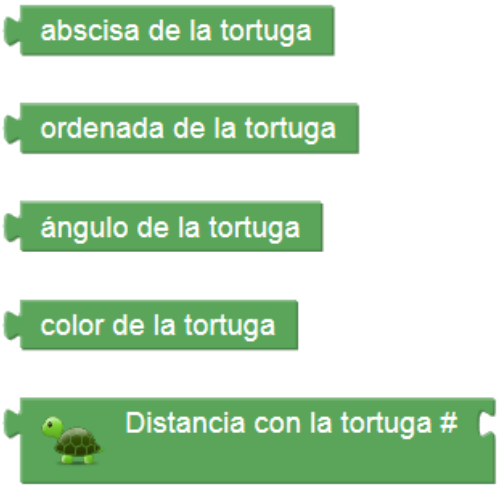
- Permiten estos bloques ocultar la tortuga o mostrarla

Diseño: Ejes



- En este apartado encontraremos bloques que permiten dibujar los ejes de coordenadas en la Consola Gráfica.
- Permite dibujar una cuadrícula y también.
- Dibujar la marca consiste en dibujar los dos ejes a la vez la marca.

Multitortugas: Datos



- Con estos bloques se pueden recoger los datos que se indica en cada uno de ellos relativos a la posición, ángulo y color de la tortuga.
- Se puede también recoger la distancia de la tortuga que se indica por su número al centro de la pantalla grafica

---

Multitortugas: Gestión

- "Con la tortuga" se pueden indicar acciones que solo se realizarán en la tortuga que se indica mediante su parámetro (número de tortuga).
- "Crear tortuga" permite crear una nueva tortuga con su número, a la que podremos acceder con la anterior instrucción.

3.4. Tratamiento de Texto

Ejemplos de partes de texto son:

- "cosa 1"
- "12 de marzo de 2010"
- El texto puede contener letras (que pueden ser minúsculas o mayúsculas), números, signos de puntuación, otros símbolos y espacios en blanco entre palabras.

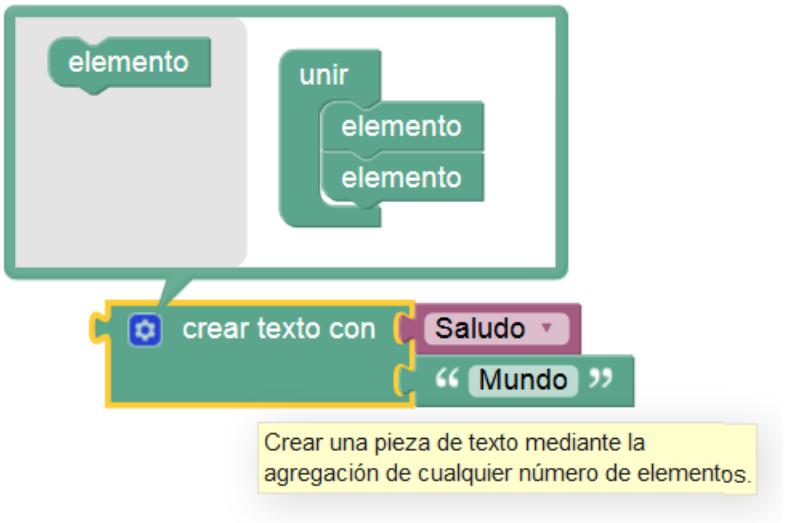
Vamos a repasar los bloques más importantes reaccionados con la manipulación de texto. La mayor parte de los textos serán para ampliar o explicar operaciones pero también se pueden mostrar a través de ellos variables.

3.4.1. **Creación de texto**

El siguiente bloque crea el fragmento de texto "hola" y lo almacena en la variable llamada *Saludo*.

El bloque "**crear texto con**" combina (concatena) el valor de la variable de *Saludo* y el nuevo texto "**Mundo**" para crear el texto "**HolaMundo**". Tenga en cuenta que no hay espacio entre ellos, ya que ninguno estaba en el texto original.

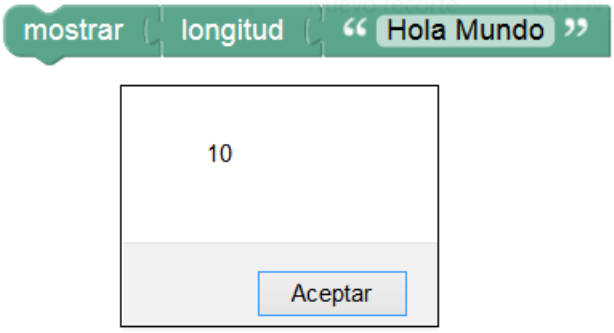
Para aumentar el número de entradas de texto, haga clic en el icono de rueda dentada, que cambia la vista a:



Se añaden entradas adicionales arrastrando un bloque de "elemento" de la caja de herramientas gris de la izquierda al bloque "unir".

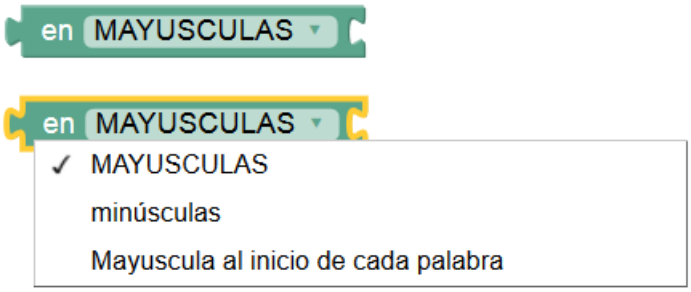
3.4.2. **Longitud del texto**

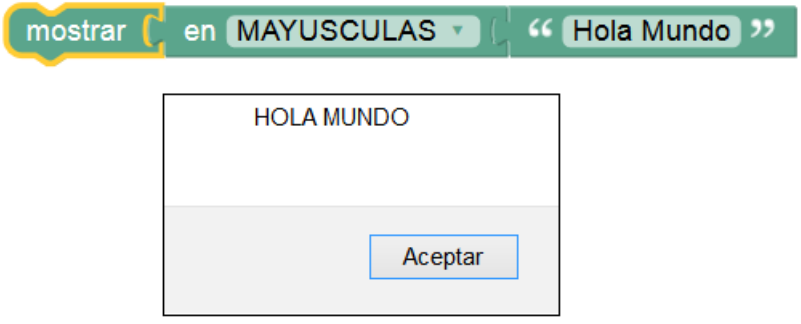
El bloque **longitud** cuenta el número de letras, números, etc., en cada texto. La longitud de "Hola Mundo" es 10.



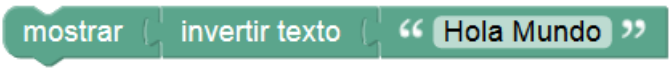
Al Ejecutar esta instrucción aparee la ventana mostrando el número de caracteres, incluido como tal el espacio

3.4.3. **Mayúsculas.**





3.4.4. **Invertir Texto**

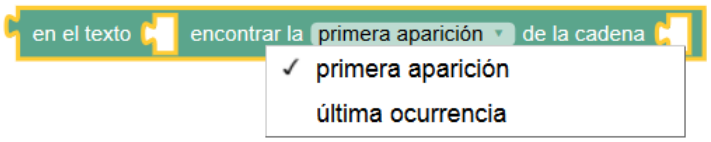


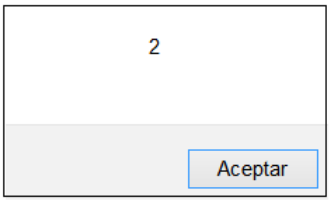
La instrucción al ejecutarse devuelve “odnuM aloH”

3.4.5. **Búsqueda de la posición de un carácter en el texto**

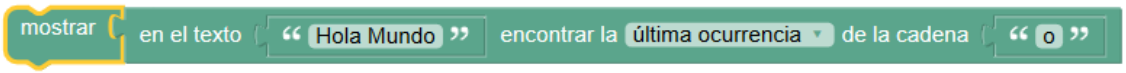
Estos bloques se pueden utilizar para buscar la posición en la que se encuentra un trozo de texto o simplemente un carácter en una cadena.

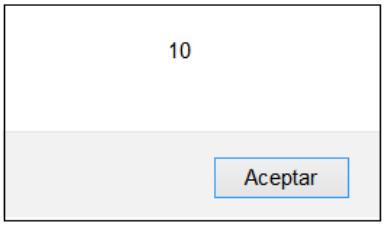
La instrucción tiene dos opciones encontrar la primer aparición o encontrar la última aparición





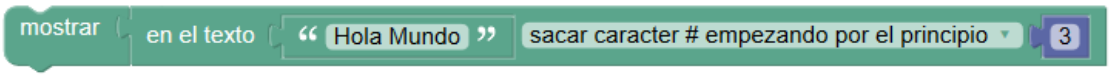
Por ejemplo, esto pide la primera aparición de "o" en "Hola Mundo". El resultado es 2.



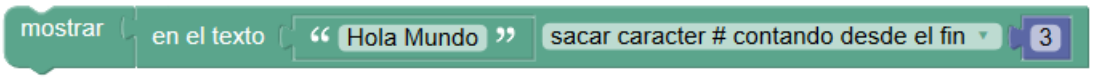


En este caso se pide la posición en la que aparece por última vez la letra “o” y nos devuelve la posición 10

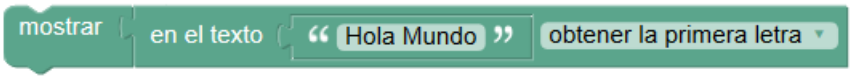
3.4.6. **Extraer un solo carácter de una cadena de texto**



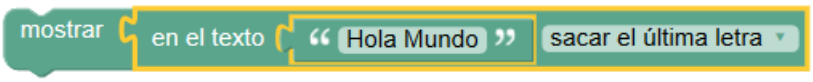
Extrae el carácter número 3 empezando a leer la cadena desde el principio: devuelve “l”



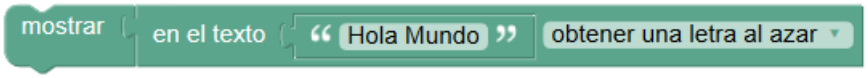
Extrae el carácter número 3 empezando a leer la cadena desde el final: devuelve “n”



Extrae el carácter primero de la cadena: devuelve “H”

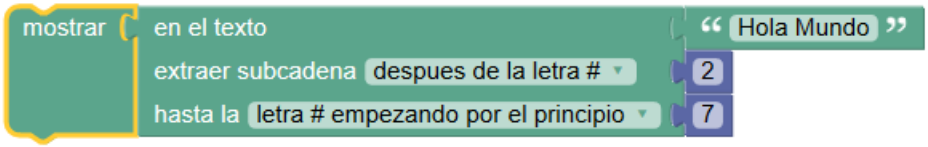


Extrae el último carácter de la cadena: devuelve “0”

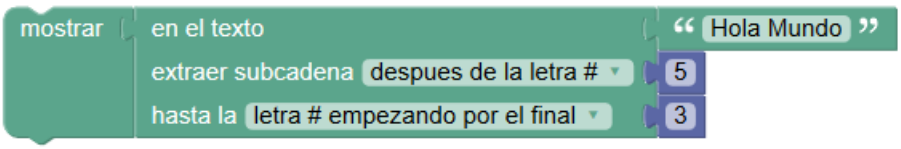


Extrae un carácter aleatorio

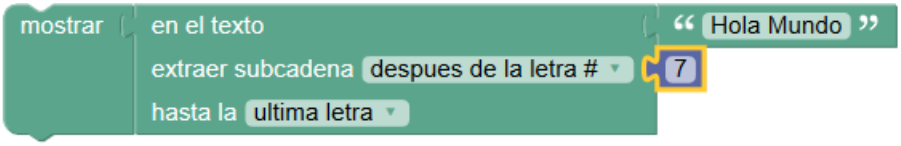
3.4.7. **Extraer sub cadenas de texto**



Extrae la subcadena que comienza en el carácter 2 y termina en el carácter 7: Devuelve “ola Mu”



Devuelve “Mu”



Devuelve “undo”

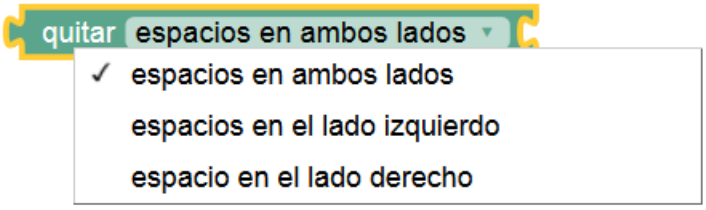
Existen otras posibilidades que omitimos y dejamos que el lector experimente.

3.4.8. **Recortar (eliminar) los espacios**

El siguiente bloque elimina caracteres de espacio de:

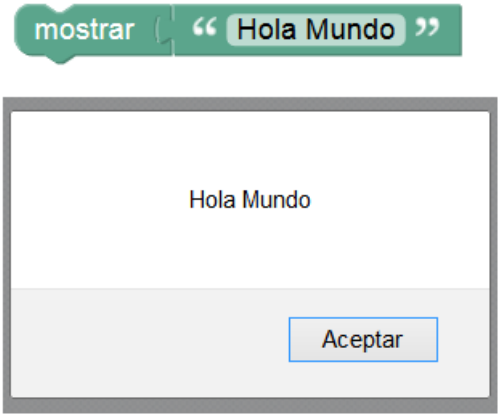
- El comienzo del texto
- El final del texto
- Ambos lados del texto

Los espacios en el centro del texto no se ven afectados.



3.4.9. **Imprimir texto**

El bloque **mostrar** hace que el valor de entrada se muestre en una ventana emergente, como se muestra:



Si el código se exporta como JavaScript, Python o Dart, se imprimirá en la consola (pantalla). En ningún caso se envía a la impresora, como su nombre podría sugerir.

3.5. Variables de tipo Lista

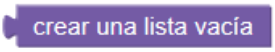
Una lista Blockly es una colección ordenada de elementos, como una lista de tareas pendientes o una lista de compras. Los elementos de una lista pueden ser de cualquier tipo y el mismo valor puede aparecer más de una vez en una lista.

Pasamos a comentar las operaciones que se pueden realizar con listas en el entorno Sofus.

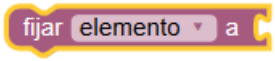
3.5.1. **Creación de lista**

Crear lista vacía

La lista más simple es la lista vacía, que se crea con el bloque de **creación de lista vacía** :



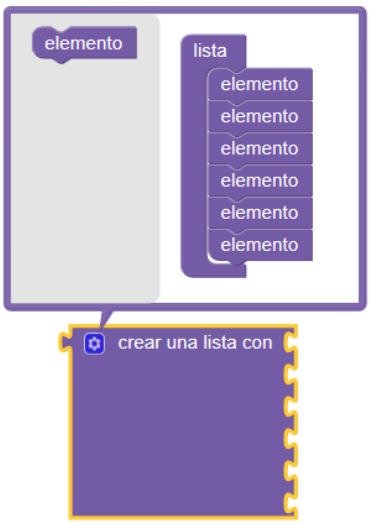
Crear lista con valores

La lista creada se llama **"Mi Lista"** y la hemos definido con la opción fijar poniéndole el nombre de **"Mi Lista"** 

.

Seleccionamos el bloque "crear una lista con" en el que hemos completado el número de entradas añadiéndole a las que tiene por defecto hasta llegar a 6 elementos. Se han retirado los parámetros de tipo numérico 

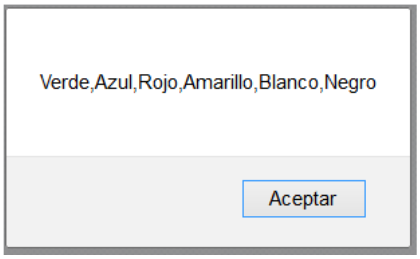
y se han colocado 6 valores de tipo texto con los 6 colores.



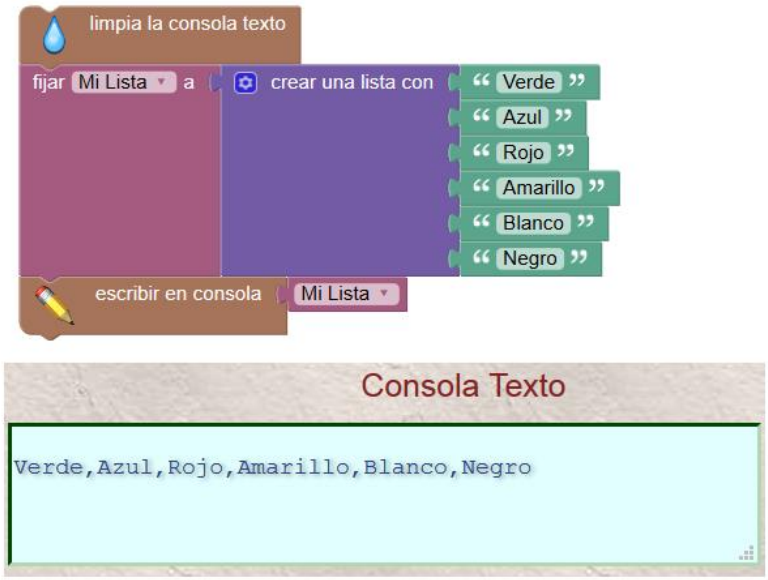
Se definen los colores, escribiéndoles en un bloque de tipo texto

Con los bloques de la siguiente figura se monta la aplicación que nos permitirá mostrar la lista en la caja de salida de texto.

Quedando de este modo la aplicación. Obsérvese que se han cambiado los elementos de la lista para ser mostrado en modo horizontal.

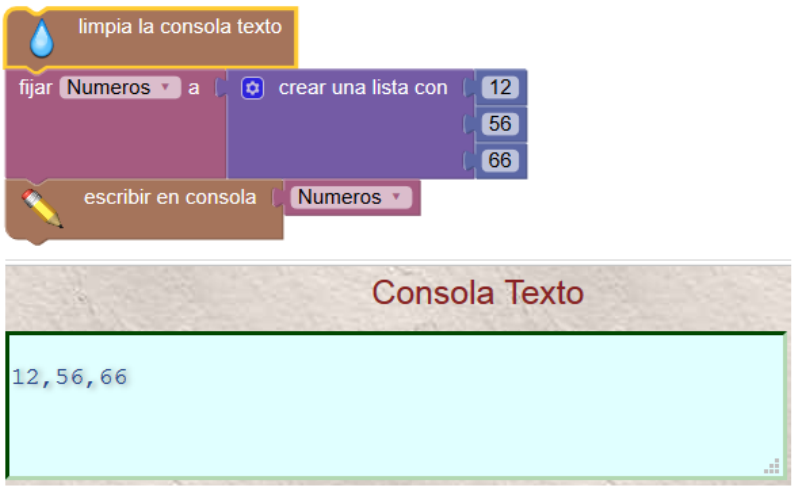


Es posible también mostrar la lista en la "**Consola de Texto**" del entorno para ello bastara con cambiar al siguiente montaje.

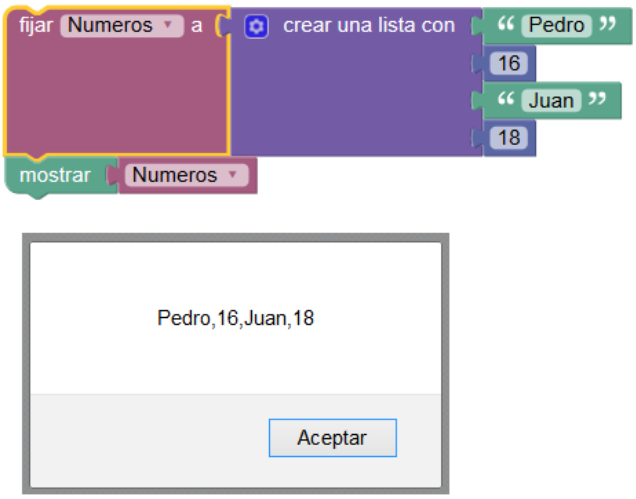


En el montaje se han utilizado dos bloques específicos de Sofus: "**limpia la consola texto**" y "**escribir en consola**" para, por un lado limpiar la consola (borrar lo que pudiera tener del anterior programa) y el segundo con el que se escribe la lista "**Mi Lista**"

En la siguiente imagen se muestra una lista de números con los valores [12,56,66] que también se muestra en la Consola de texto

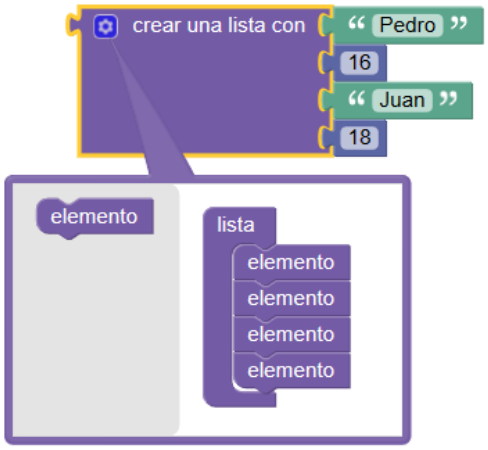


Es menos común, pero posible, crear una lista con valores de diferentes tipos, texto y números.



Cambiando el número de entradas

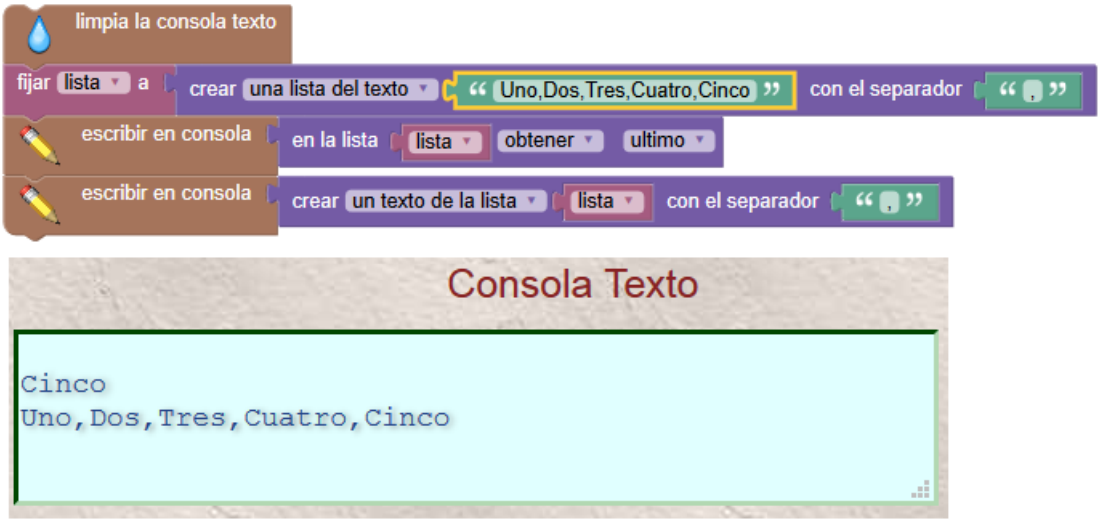
Para cambiar el número de entradas, haga clic en el icono de rueda dentada. Esto abre una nueva ventana:



3.5.2. Creando listas a partir de cadenas de textos separadas por un carácter.

En este caso vamos a estudiar el bloque “crear una lista del texto” que tiene dos posibilidades, la anterior y “crear un texto de la lista”

Vamos a confeccionar una lista separando sus elementos con el separador “,”.

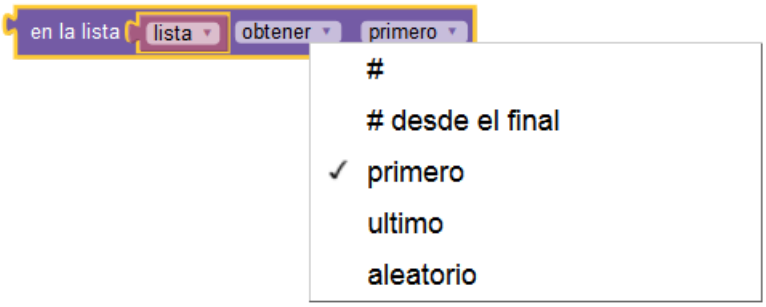


La lista la llamamos “**lista**” y está formada por:

Lista=[Uno,Dos,Tres,Cuatro,Cinco]

La variable lista se conecta al bloque “crear *una lista del texto*” y se rellena el texto expresado anteriormente separando por comas cada elemento

Después mediante los bloques “escribir en consola” en el primer bloque lo que hacemos es imprimir en la “Consola de Texto” el elemento “5” de la lista (es decir el último), aunque podríamos haber puesto el número de las opciones que se muestran en la siguiente imagen..



Es posible, así mismo, partiendo de una lista convertir esta en una cadena de texto mediante la opción “crear *un texto de la lista*”

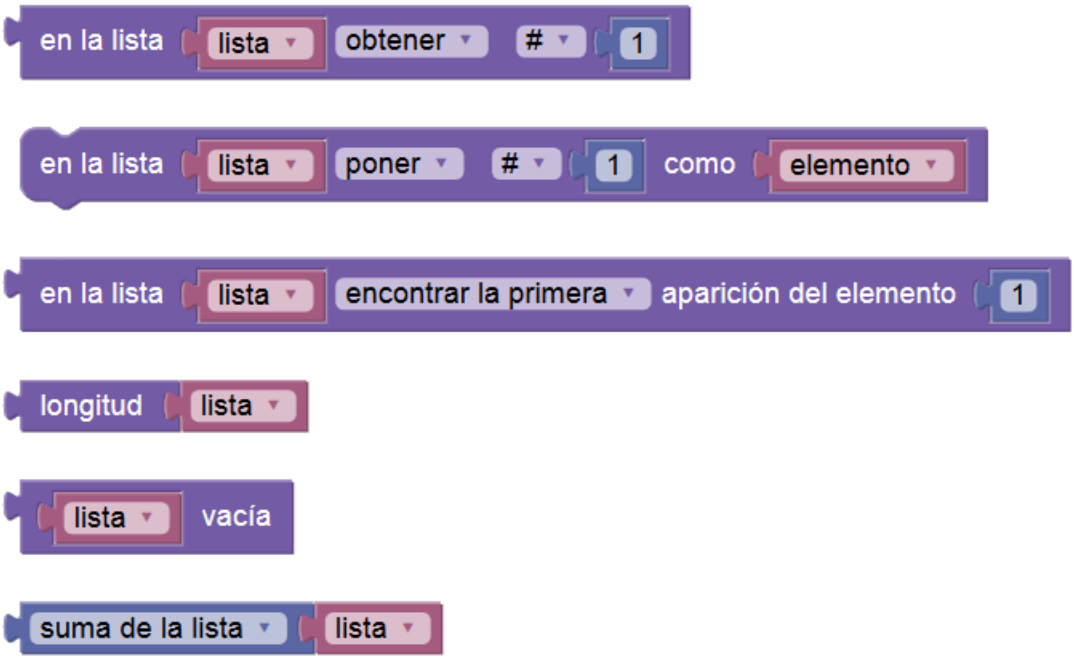
Crear una lista con un elemento repetido

Es posible crear una lista que contenga todos elementos iguales, tato de texto como numérica.

3.5.3. **Manipulación de las listas**

Vamos a comentar los bloques que se encuentran en la librería:

Listas-> Manipulación



Vamos a contar con la lista siguiente para ponerla como ejemplo de las operaciones que siguen en lo que se relaciona con la manipulación. Se trata de la lista llamada Colores que contiene 6 elementos cada uno de los cuales es un color.

Colores=[Verde,Azul,Rojo,Amarillo,Blanco,Negro]

3.5.3.1. Verificar el tamaño de una lista.

Esta instrucción "**longitud**" devuelve un numero al ejecutarse que es el número de elementos de la lista a la que está conectado –colores- En la "**Consola de Texto**" se escribe "**6**"

Con el bloque "**Lista vacia**" lo que se averigua es si una lista contien algún elemento o esta vacia.

Si la lista esta vacía la instrucción devuelve **true** (Cierto)
Si la lista contiene algún elemento devuelve **false** (Falso)

3.5.3.2. Obtener elementos de una lista.

Con la instrucción **"En la lista**…**aparición del elemento**" podemos extraer un elemento cualquiera de la lista

- primero
- último
- aleatorio
- con una posición empezando a contar desde el principio
- con una posición empezando a contar desde el final de la lista

Escribe en la **"Consola de Texto": Verde**

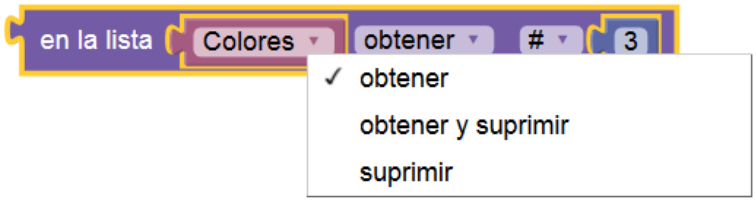
Escribe en la **"Consola de Texto": Negro**

Escribe en la **“Consola de Texto”: cualquiera de los colores**

Escribe en la **“Consola de Texto”: Rojo**

Escribe en la **“Consola de Texto”: Blanco**

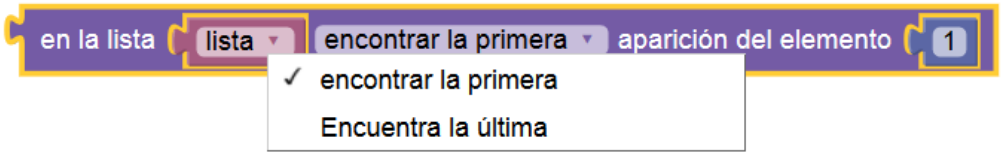
Es posible mediante la selección del área “**obtener**” tres formas de intervenir en la operación



- **Obtener**: Lee el elemento (es la que hemos comentado en los ejemplos)
- **Obtener y suprimir**: lee el elemento y lo elimina
- **Suprimir:** Suprime directamente el elemento.

3.5.3.3. Búsqueda de elementos en una lista

Vamos a manejar las instrucción **“en lista…encontrar… aparición de elemento”**



Con la instrucción buscamos si existe un elemento –número o texto- e una lista bien que busquemos la primera vez que aparece o la última, dependiendo de lo que seleccionemos. La instrucción devuelve la posición del elemento buscado. En el siguiente ejemplo la instrucción imprime en la “**Consola de Texto**” “**4**” que es la posición del elemento “Amarillo”.

3.6. Vectores y Matrices.

3.6.1. **Puntos y vectores**

Un punto es una matriz que tiene dos coordenadas. Un vector es una matriz que tiene dos coordenadas (o tres si uno está en el espacio). En otras palabras, en términos de la matriz, Sofus no sabe la diferencia entre puntos y vectores. Esto permite hacer cálculos entre ellos.

Ejemplo1

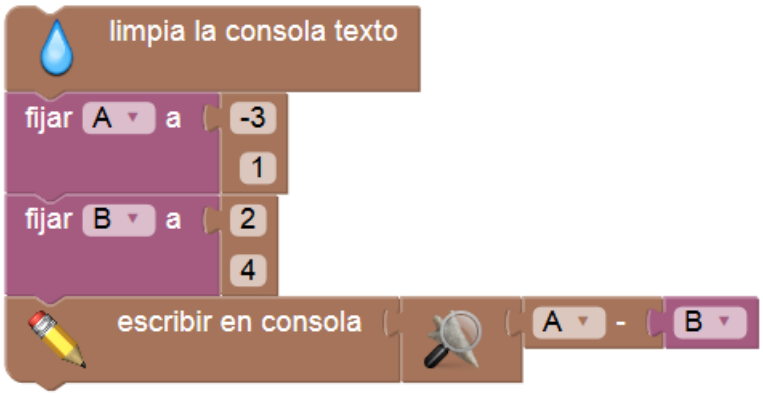
En el siguiente ejemplo definimos dos puntos en el plano:

$$A=[-3, 1] ; B=[2, 4]$$

y al restar las coordenadas obtenemos el valor de las coordenadas del vector

$$V_{AB}=[-5,-3]$$

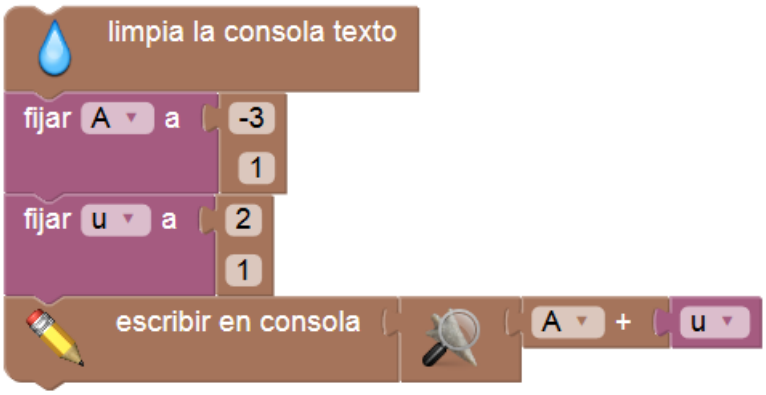
Para el cálculo de las coordenadas del vector de A (punto) B (), se resta de A a B:



Para poder mostrar vectores o matrices es preciso utilizar el bloque  .

Ejemplo 2

Ahora, para calcular la imagen de A (un punto) por el vector unidad u, u se añade a A:



En este caso las coordenadas de la imagen del punto A con respecto al vector unitario u es [-1,2]

Múltiples operaciones con vectores

Es posible sumar, restar vectores, sino también multiplicar el escalar o real, e incluso calcular su ángulo. La adición, substracción y multiplicación por un verdadero también trabajan con matrices distintas de los vectores.

Ejemplo 3

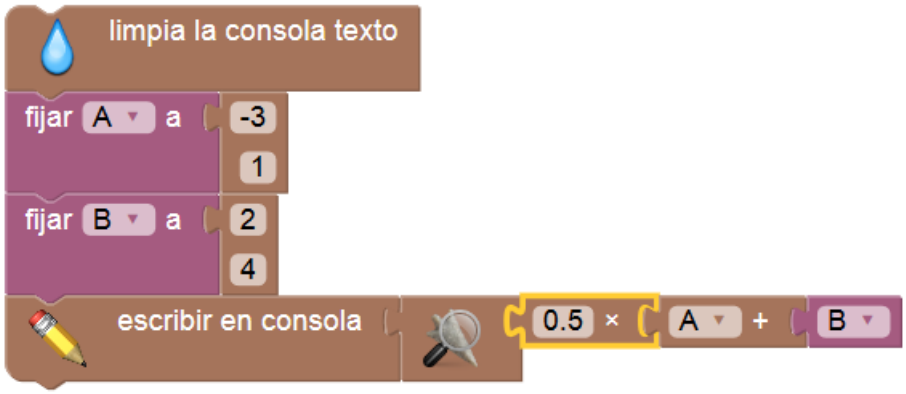
Calcular el punto medio entre dos puntos.

Para las coordenadas del punto medio de AB, sumamos A y B y el resultado se multiplica por 0,5:

A=[-3, 1] ; B=[2, 4]

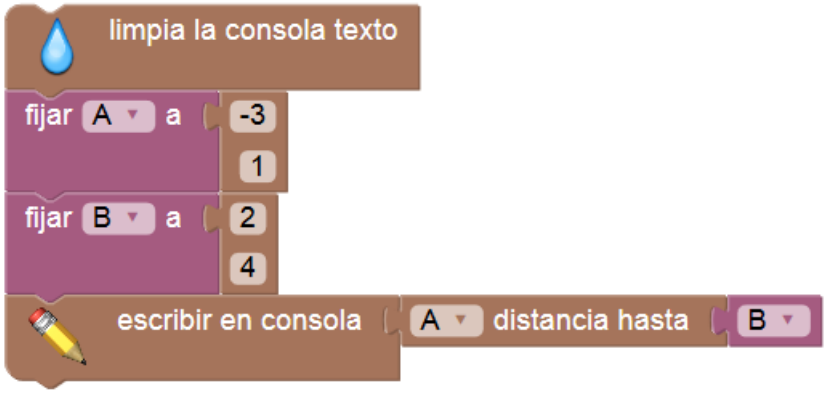
Devuelve la coordenada del punto medio

Pm=[-0.5, 2.5]



Distancia entre dos puntos

En este caso se trata de calcular la distancia entre dos puntos, téngase en cuenta que se trata del valor del módulo del vector V_{AB} en este caso la distancia calculada es 5.83

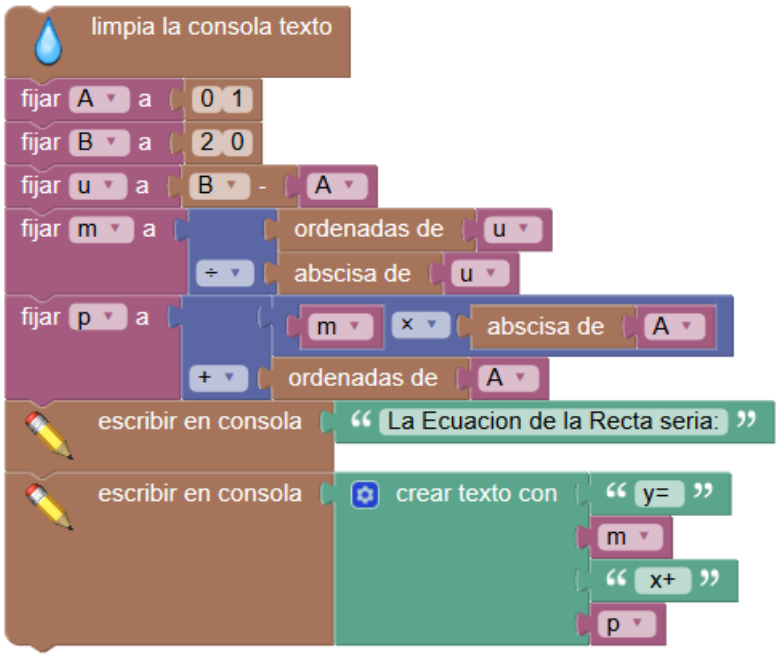


Ejemplo4.

Recta que pasa por dos puntos A y B

A=[-3, 1]; B=[2, 4];

La ecuación a la que queremos llegar es del **tipo y=m*x+p** **u** es el vector $\mathbf{V_{AB}}$ cuyo cociente de componentes es justo el valor da la pendiente **m** y p la ordenada en el origen es



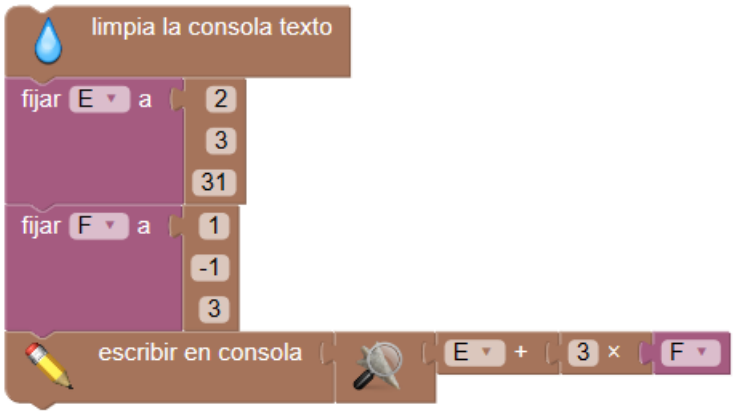
Ejemplo 5

Resolución de un sistema de ecuaciones mediante determinantes

Vamos a resolver el sistema

- 2x + 3y = 31
- x - y = 3

Vemos que si multiplicamos la segunda ecuación por 3, aparece -3y que se ira con 3y si a continuación se suman las ecuaciones. Para realizar esta multiplicación por 3, puede representar la ecuación ax + by = c por el vector de coordenadas (a, b, c) y triples este vector:

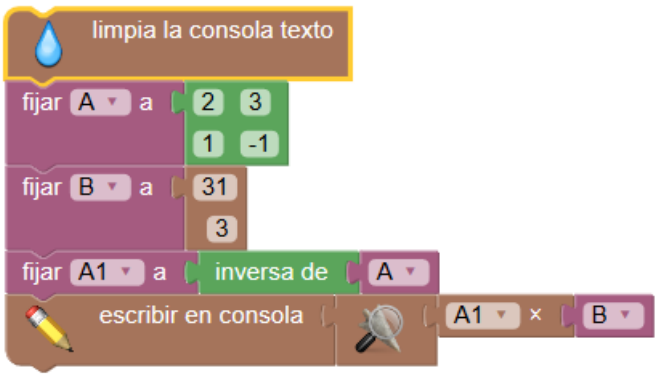


El vector calculado es el resultado de la suma de la primera ecuación con 3multiplicado por la segunda ecuación: **[5, 0, 40]** es decir **5x=40**

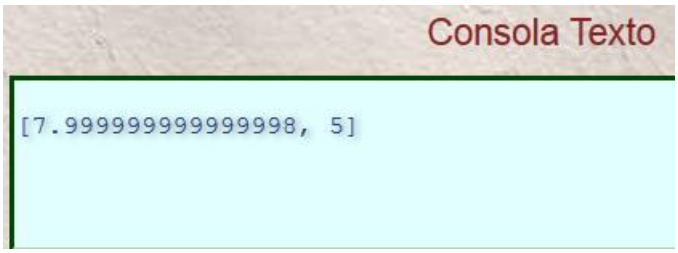
Tomamos nota de este resultado que se nos muestra en la Consola de Texto y ahora es fácil resolver esta ecuación. 5x=40. Que equivale **x= 8** y teniendo en cuenta la segunda ecuación **y=5**

Otra forma de resolver el sistema es mediante sus determinantes.

La pantalla, con el vector de codificación seleccionada aquí, toma nota de 5x = 40, que es fácil de resolver. Pero podemos resolver todo el sistema a la vez, con matrices, imitando la resolución Ax = b, la solución x = b / a se obtiene multiplicando ambos lados por la inversa de una: AX = B puede ser resuelto de forma análoga multiplicando ambos lados por la inversa de a, incluso si a es una matriz. Una precaución, sin embargo: La multiplicación de matrices no es conmutativa, se deja uno debe llevar a cabo la multiplicación por A1 (inversa de A):



El resultado que se escribe en la Consola de Texto es

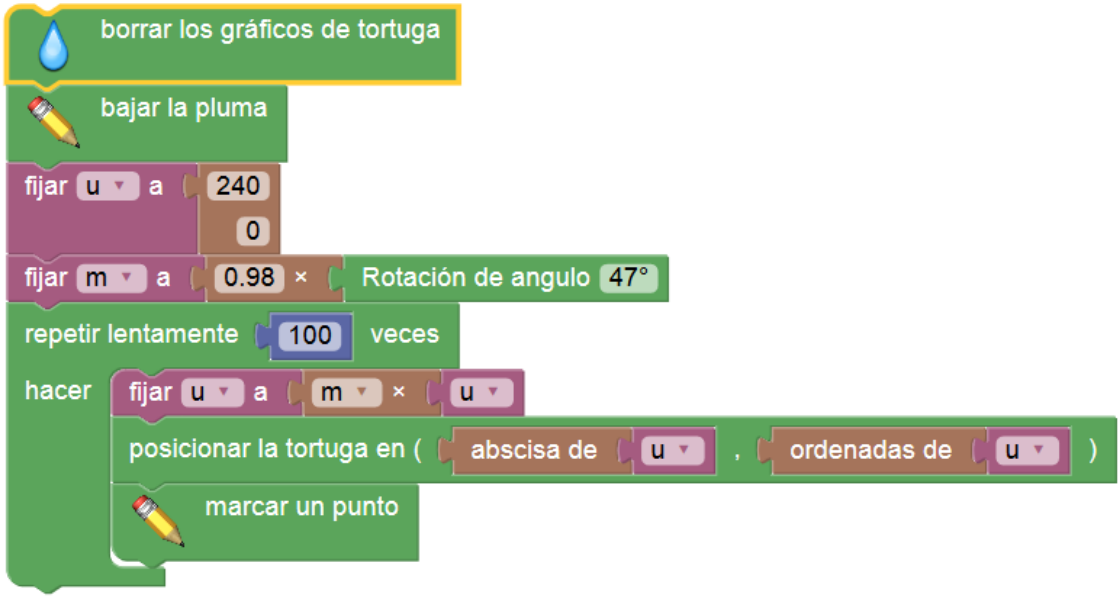


Obsérvese que en lugar de 8 como debería dar da algo menos pero este error es por problemas del tratamiento de los datos en Sofus.

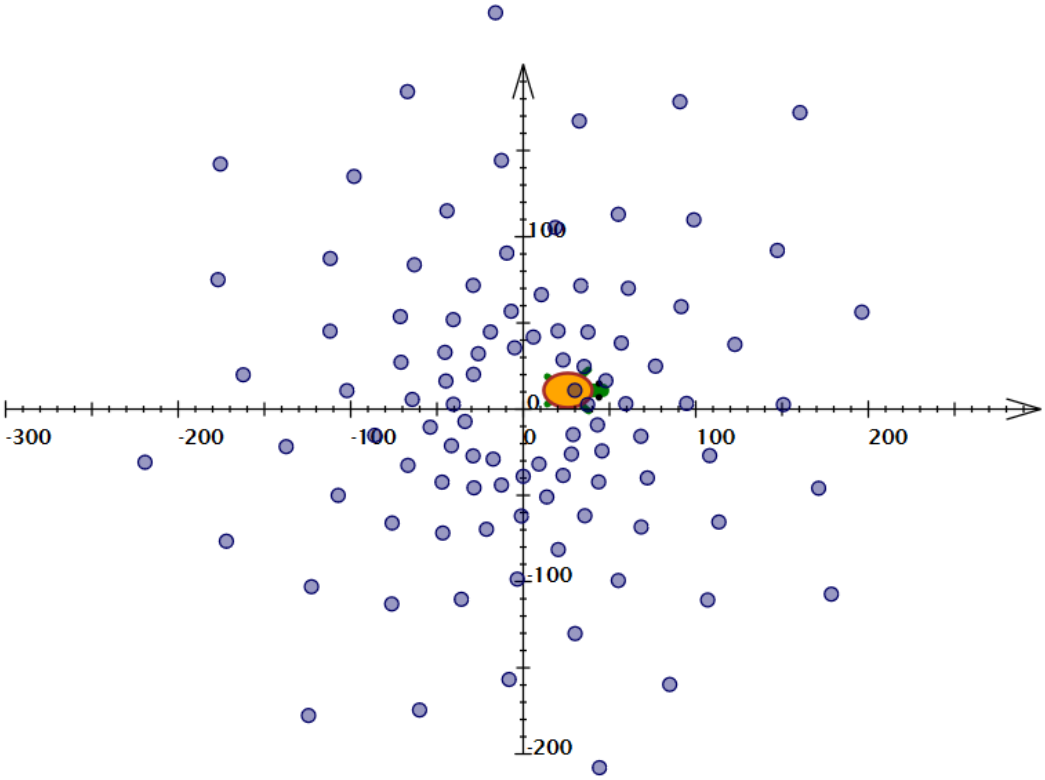
Ejemplo 6

Similitud directa

Una similitud directa es el producto de una constante (la relación de similitud) por una matriz de rotación. Tales matrices se pueden fijar directamente:



Aquí, la iteración de la similitud, creando una nube de puntos que Sofus tortuga puede dibujar:



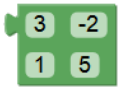
3.6.2. **Matrices 2D.**

El tratamiento de matrices y sus determinantes es muy importante en el cálculo matemático. En este apartado vamos a ver los principales bloques de Sofus que abordan este campo.

3.6.2.1. Definición de matrices

Para la construcción de una matriz de utilizan los bloques de la librería

Matemáticas -> Matrices ->Matrices ->Construcción



Creación de una matriz 2x2

Matriz nula

Creación de una matriz diagonal


Matriz aleatoria

Matriz diagonal

Matriz de rotación

Matriz de identidad

Con estos bloques podremos definir matrices de tamaño 2x2

3.6.2.2. **Operaciones con matrices**.

A continuación se muestran los bloques de función de la librería de Operaciones con matrices.

Matemáticas -> Matrices ->Matrices ->Operaciones

A modo de recuerdo.

Se define como **traza de una matriz** a la suma de los elementos de su diagonal principal

Ejemplos de Traza de matrices

$$M1=\begin{bmatrix}1 & -3\\ 5 & 2\end{bmatrix} \quad \text{Tr } M1 = 1 + 2 = 3$$

$$M2=\begin{bmatrix}2 & -3 & 4\\ 0 & -3 & 2\\ 3 & 1 & 5\end{bmatrix} \quad \text{Tr } M2 = 2 - 3 + 5 = 4$$

La traspuesta de una matriz es la matriz resultante de cambiar

La traspuesta de una matriz A consiste en intercambiar las filas por las columnas y se denota por A^T.

Así, la traspuesta de

$$A=\begin{pmatrix}3 & -1 & 4\\ 2 & 5 & -7\\ 4 & 0 & 9\end{pmatrix} \text{ es } A^T=\begin{pmatrix}3 & 2 & 4\\ -1 & 5 & 0\\ 4 & -7 & 9\end{pmatrix}.$$

Calculo del **determinante de una matriz**

De tamaño 2x2

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11} a_{22} - a_{12} a_{21}$$

De tamaño 3x3 (regla se Sarrus)

$$\begin{vmatrix} 1 & 2 & 3 \\ 1 & 1 & -1 \\ 2 & 0 & 5 \end{vmatrix} = 1\cdot 1\cdot 5 + 2\cdot(-1)\cdot 2 + 3\cdot 1\cdot 0 - 3\cdot 1\cdot 2 - 2\cdot 1\cdot 5 - 1\cdot 0\cdot(-1) =$$

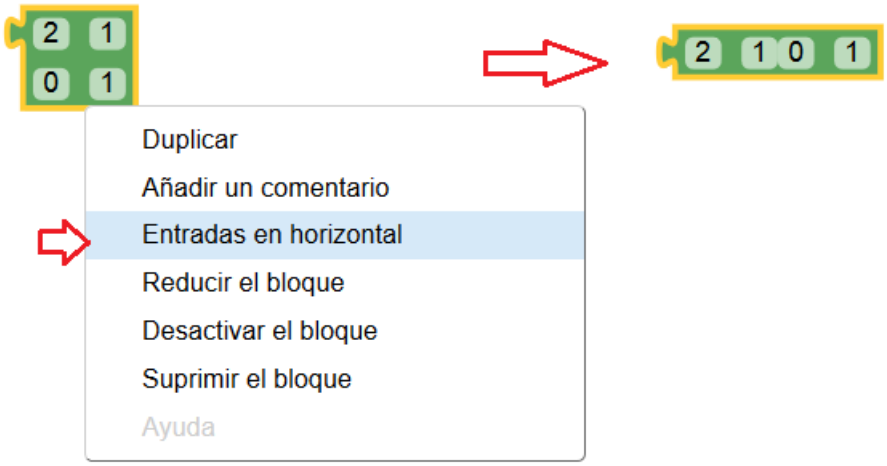
$$= 5 - 4 + 0 - 6 - 10 + 0 = -15$$

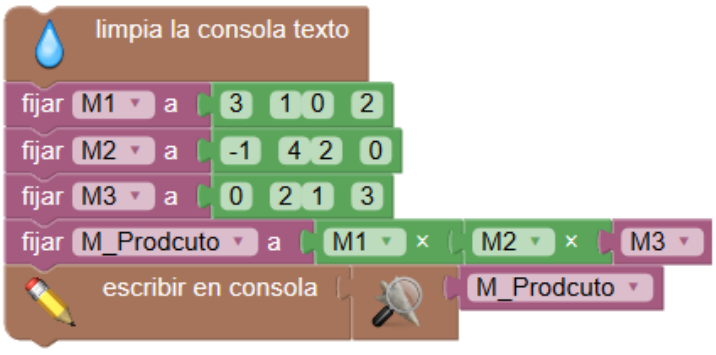
Veamos algunos ejemplos.

Ejemplo1.

Producto de tres matrices M1, M2 y M3

Se han colocado las matrices en formato vertical para que ocupen menos espacio.





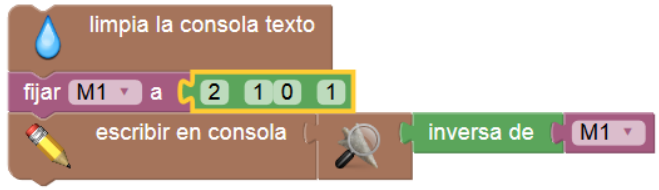

```
[12, 34]
[0, 8]
```

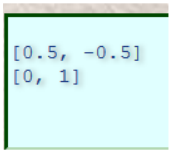
M1 [3,1] M2 [-1,4] M3 [0,2]
[0,2] [2,0] [1,3]

Resultado

Ejemplo 2.

Matriz Inversa.



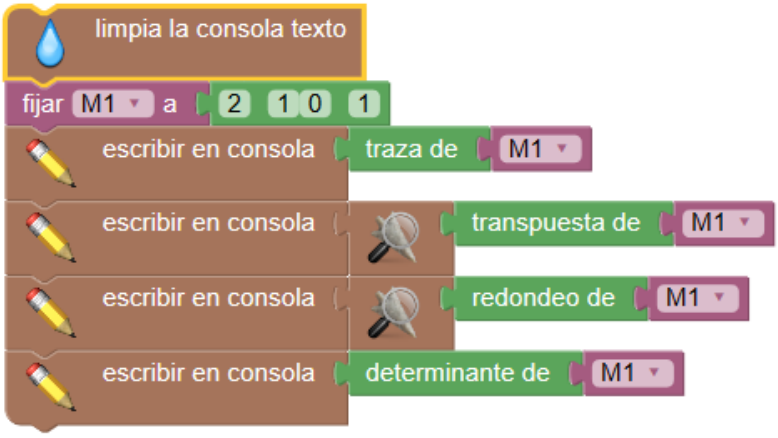


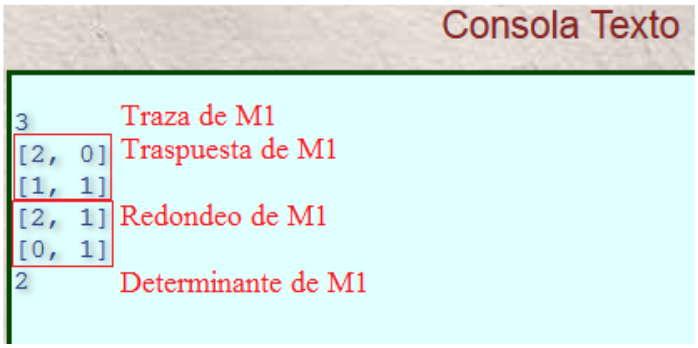
M1 [2,1]
[0,1]

Resultado

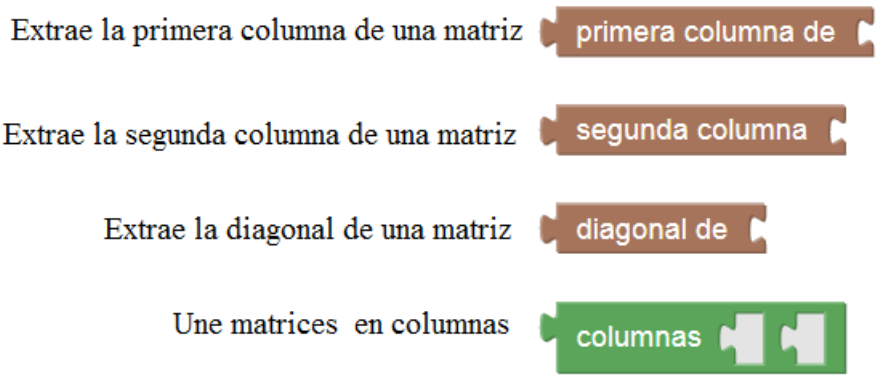
Ejemplo 3

Varias operaciones.

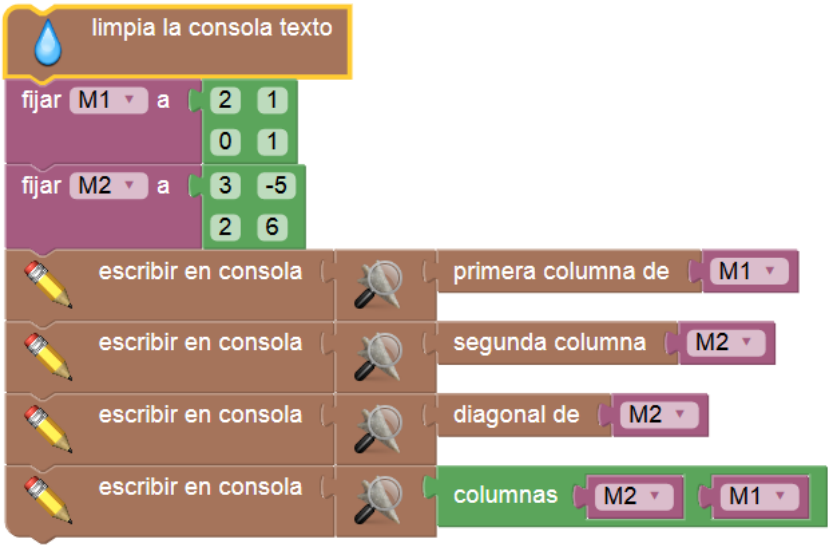




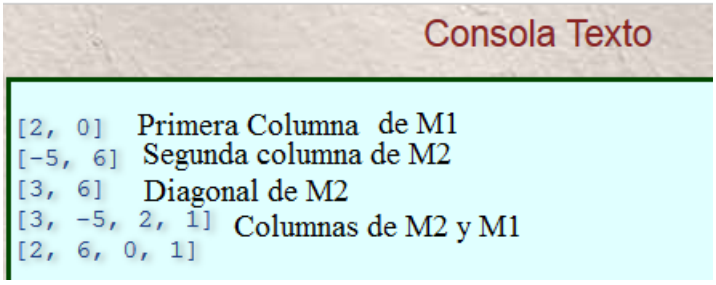
3.6.2.3. Librería Matemáticas -> Matrices -> Matrices ->Vectores



Vemos un ejemplo a continuación en el que se han colocado todos los bloques



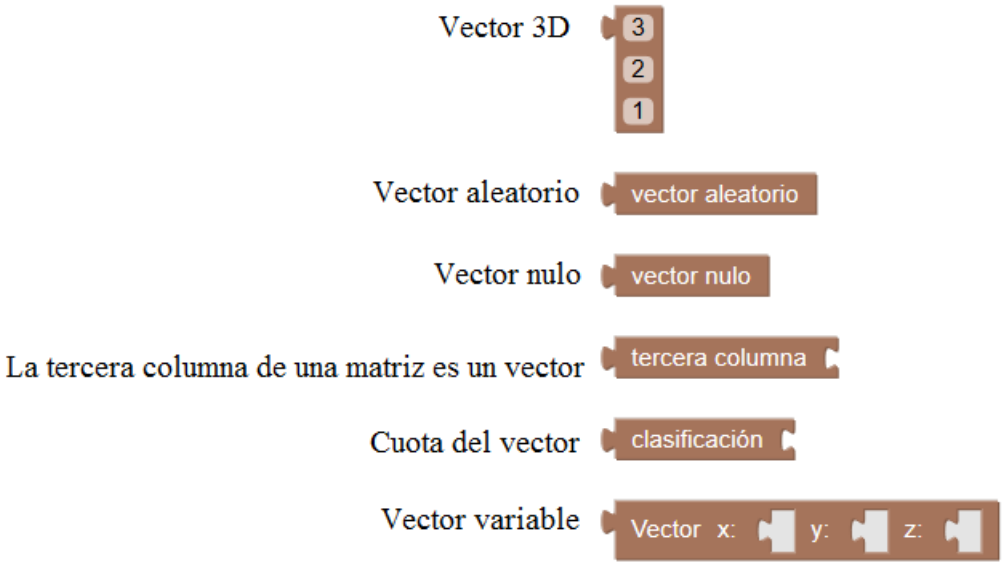
Esta sería la salía de resultados

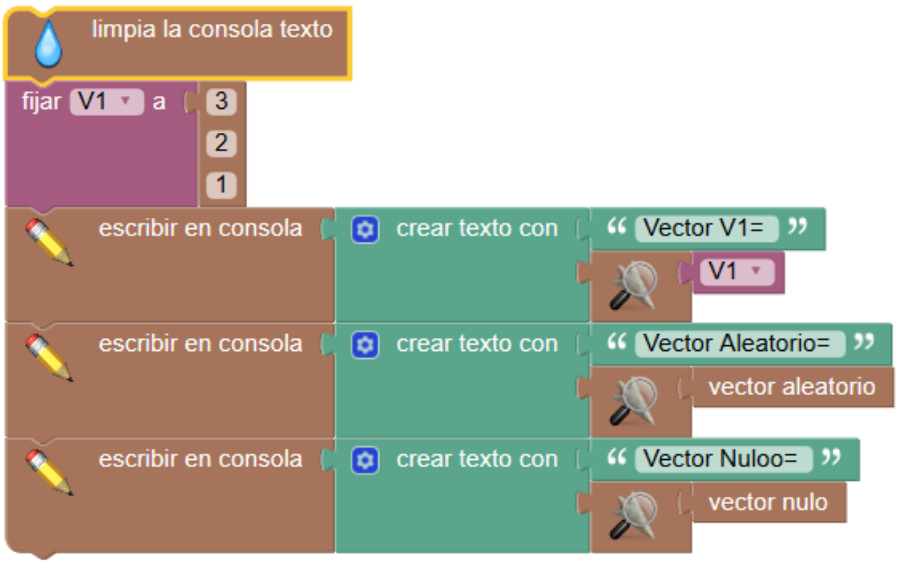


3.6.3. **Vectores y matrices tridimensionales 3D.**

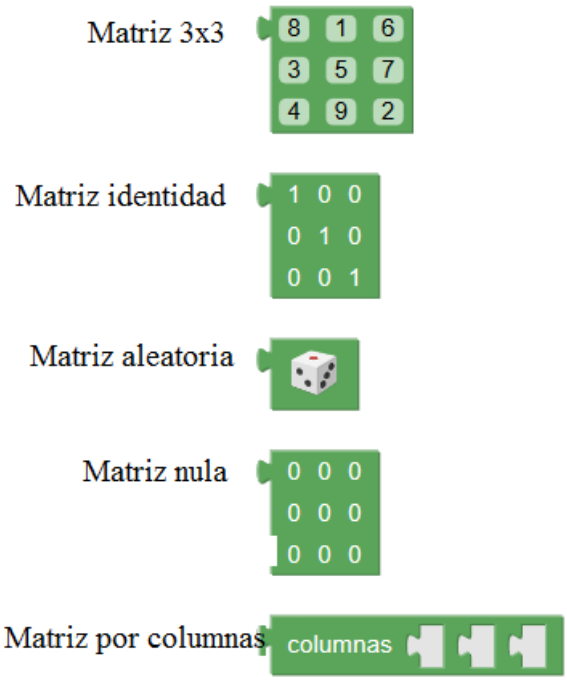
Vamos a estudiar los bloques de función de las matrices y vectores de formato 3D

Librería Vectores





Librerías matrices.



3.7. Instrucciones de control de flujo.

Vamos a estudiar un conjunto de instrucciones muy poderosas en el desarrollo de algoritmos.

3.7.1. Instrucciones de tipo Condicional

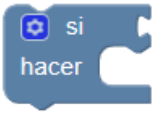
Las sentencias condicionales son fundamentales para la programación de computadoras. Ellos hacen posible expresar declaraciones como:

- Si hay un camino a la izquierda, gire a la izquierda.
- Si la puntuación = 100, imprime "¡Eres Campeón!".

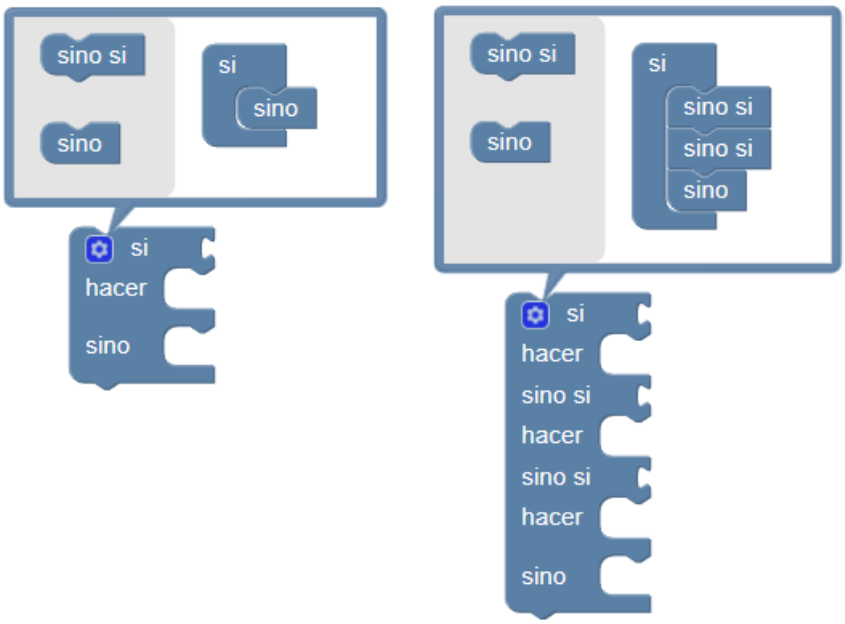
Para el uso de estas estructuras de tipo condicional en nuestros programas podemos hacer uso de los bloques que se explican a continuación. Estos bloques se encentran en la librería llamada “Lógica->Condiciones” en el entorno Sofus.

3.7.1.1. Bloque si..hacer

La sentencia condicional más simple es un bloque **si.. hacer** , como se muestra:



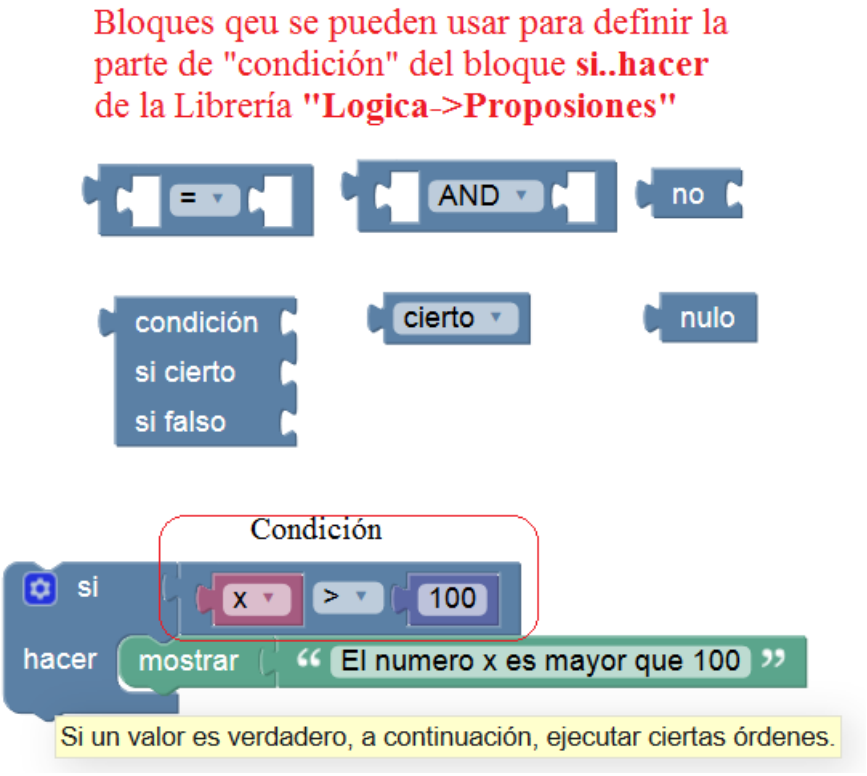
Esta es la expresión más sencilla del bloque . Pulsando en el pequeño engranaje que figura en el bloque podremos construir bloques condicionales derivados con más condiciones.



Veamos algunos ejemplos de uso de este bloque.

Se trata de establecer la condición de que un valor sea mayor que una constante (*x>100*) si se cumple la condición entonces se muestra un mensaje.

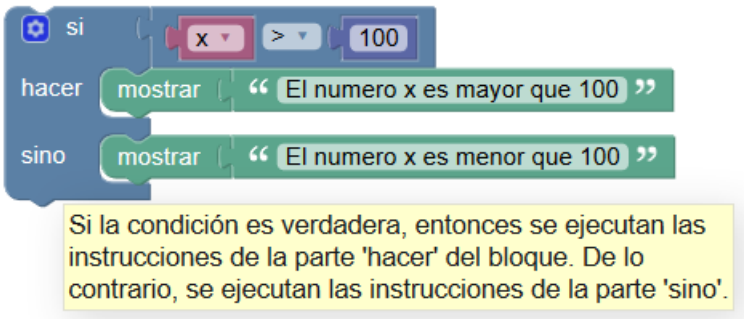
La parte condicional de la instrucción **si.. hacer** puede ser cualquier condición que se establecerá usando los bloques de la librería "*Lógica->Proposiciones*"



Cuando se ejecuta, esta comparará el valor de la variable **x** con 100. Si es mayor, se imprimirá "El numero x es mayor que 100". De lo contrario, no pasa nada.

3.7.1.2. Bloques de tipo Si-Si no

También es posible especificar que algo debería suceder si la condición *no* es verdadera, como se muestra en este ejemplo:

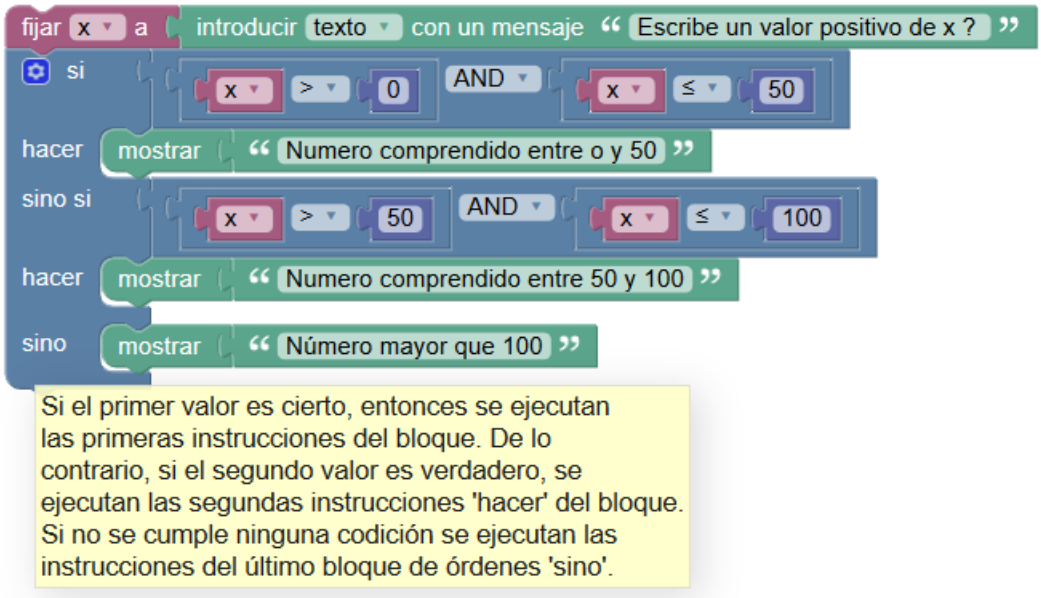


Cuando se ejecuta, este comparará el valor de la variable **x** con 100. Si es mayor, "El numero x es mayor que 100" Será impreso. De lo contrario, se imprimirá “El número es menor que 100”.

Un bloque **si** puede tener cero o uno **más** secciones pero no más de uno.

3.7.1.3. Bloques de varias condiciones

También es posible probar condiciones múltiples con un solo bloque **si** agrega cláusulas **sino si** :



El bloque primero si comprueba si *0< **x** <= 50* En este caso se imprime el texto “*Número comprendido entre 0 y 50*”

El bloque segundo sino si comprueba *50 < x <= 100* En este caso se imprime *"Número comprendido entre 50 y 100"*

Finalmente la parte sino se ejecutara cuando ninguna de las anteriores condiciones se cumpla. En este caso se imprime "*Número mayor que 100*"

Las condiciones se evalúan de arriba a abajo hasta que uno esté satisfecho, o hasta que no queden más condiciones.

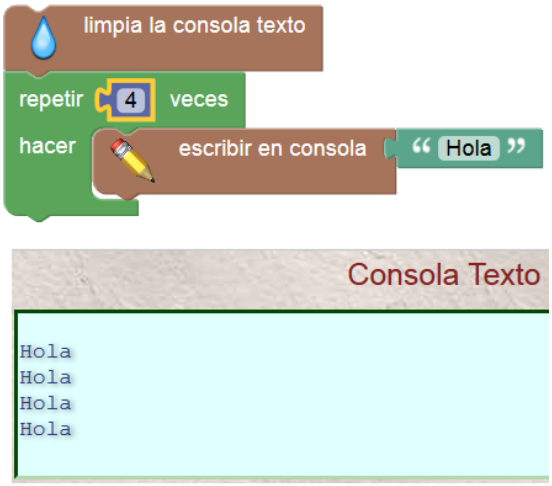
3.7.2. **Bucles de control**

La categoría **Control** contiene bloques que controlan si se ejecutan otros bloques colocados en su **cuerpo** . (Por ejemplo, en el siguiente bloque "repeat", el cuerpo contiene el bloque "print" y su entrada.) El cuerpo se ejecuta y, en algunos casos, el valor de una variable utilizada dentro del cuerpo. Estas estructuras se llaman **bucles** ya que el cuerpo se repite (posiblemente) varias veces, una reminiscencia de una cuerda que contiene bucles. Cada paso a través del bucle se llama **iteración** . Para obtener más información, consulte https://en.wikipedia.org/wiki/Control_flow#Loops .

Los bloques de control están la librería "**Bucles -> Simples**"

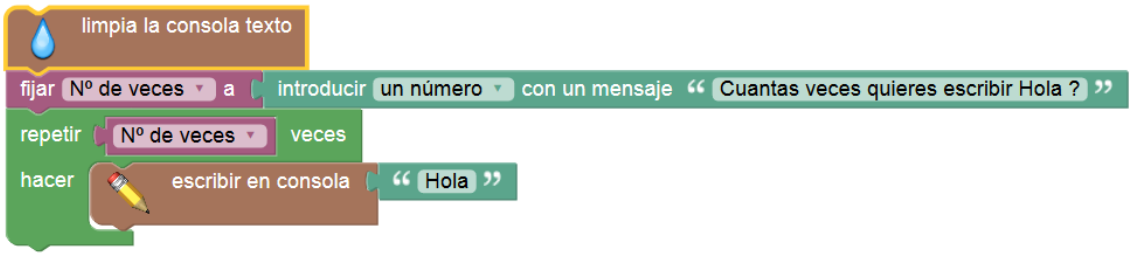
3.7.2.1. **Bloque repetir "n" veces**

Este es el bloque más simple de "repetición" ejecuta el código en su cuerpo el número especificado de veces. Por ejemplo, el siguiente bloque imprimirá "¡Hola!" cuatro veces.



El parámetro “número de veces” puede ser un valor numérico o una variable numérica

En el siguiente ejemplo se nos pregunta el número de veces que queremos escribir “Hola” y la respuesta la asociamos a la variable “**Nº de veces**” que es la que colocamos en el bloque “**repetir**”.



Con este programa escribimos el número de veces que queramos el texto “Hola”

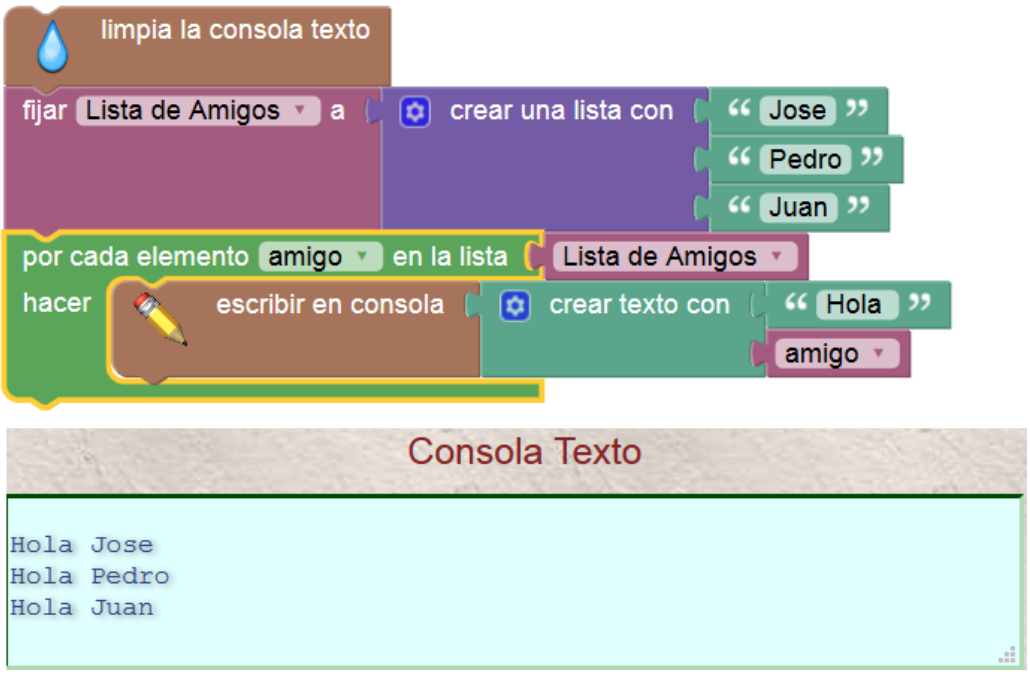
3.7.2.2. Repetir por cada elemento de una lista

Este bloque lo que hace es realizar las instrucciones que le indiquemos en su lado de “hacer” por cada uno de los elementos de una “lista” de valores, cargando cada vez que se repite el valor del elemento de la lista mediante el “*puntero variable*” **i** que puede ser nombrado con otro nombre.

En el ejemplo siguiente se crea una lista de mis amigos con el nombre:

Lista de Amigos= [Jose, Pedro, Juan]

Mediante esta instrucción de repetición lo que hacemos es sacar cada elemento de la lista y saludad a cada amigo *“Hola Jose”, “Hola Pedro”, “Hola Juan”*

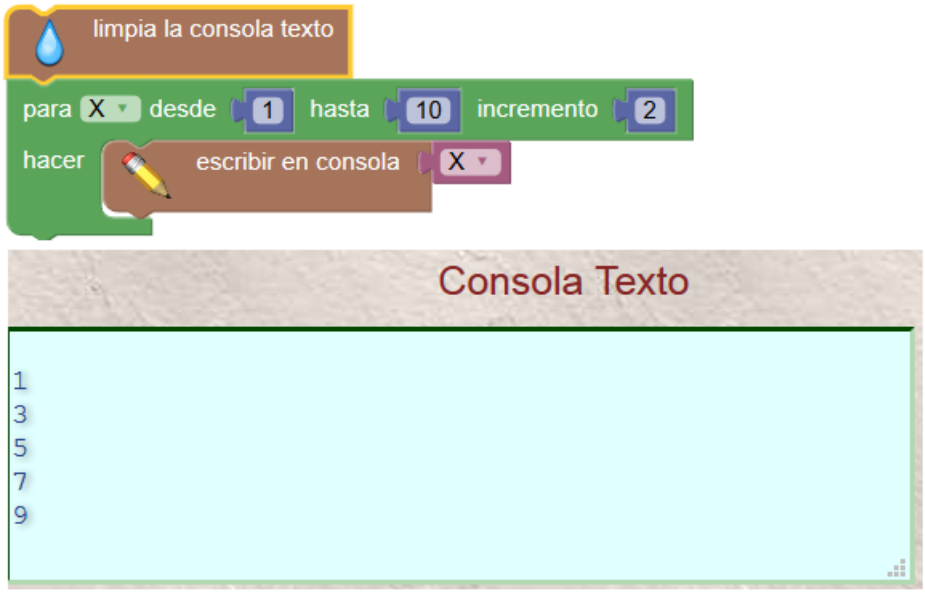


3.7.2.3. Instrucción de Bucle tipo FOR NEXT

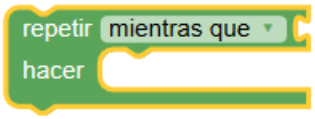
Esta instrucción es la equivalente a la clásica instrucción de programación **FOR NEXT.** Esta pensada para realizar una serie de repeticiones dentro de un bucle ejecutando las instrucciones que se colocan dentro de ella en la parte "**hacer**" de la instrucción.

La condición para se ejecute el bucle es en primer lugar definir un rango de valores por los que se va a mover el valor de la variable "**i**" también llamado *puntero* y la parte de "**incremento**" viene a expresar la forma de variar los valores siempre partiendo del primero valor "**desde**" al segundo valor "**hasta**"

En el ejemplo siguiente lo que hacemos es escribir los valores de "**X**" desde 1 a 10 incrementando de 2 en 2, es decir **1,3,5,7,9**



3.7.2.4. Repetir mientras

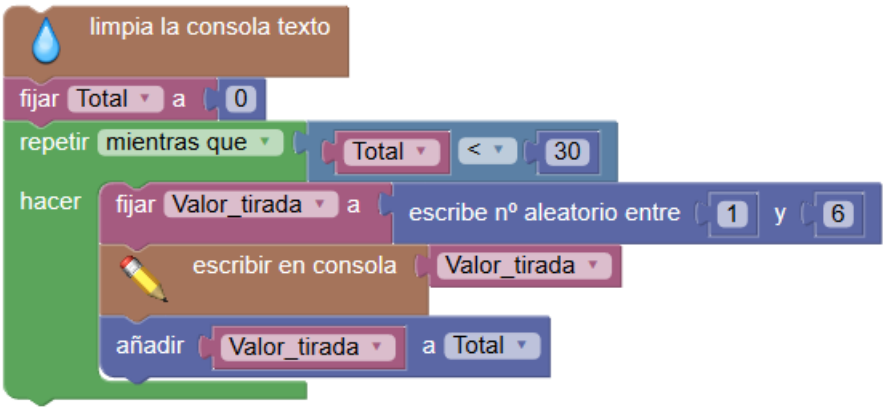


Vamos a poner un ejemplo para entender esta instrucción**.**

Un juego: Imagine un juego en el que un jugador lanza un dado y suma todos los valores que le van saliendo **siempre que el total sea menor que 30** (condición del bloque "**mientras que**". Los siguientes bloques permiten implementar este juego:

1. Definimos la variable llamada **Total** dándole el valor inicial de **"0"**
2. El ciclo comienza con la comprobación de que el **Total** *es menor que 30* (condición del bucle). Si es así, los bloques en el cuerpo se ejecutan.
3. Se genera un número aleatorio en el rango de 1 a 6 (simulando una tirada del dado) y se almacena en una variable llamada **Valor_tirada** .
4. Se imprime el valor **Valor_tirada**.
5. La variable **Total** se incrementa en el valor de **Valor_tirada** .
6. Al llegar al final del bucle, el control vuelve comprobarse si la condición

   Total >30.

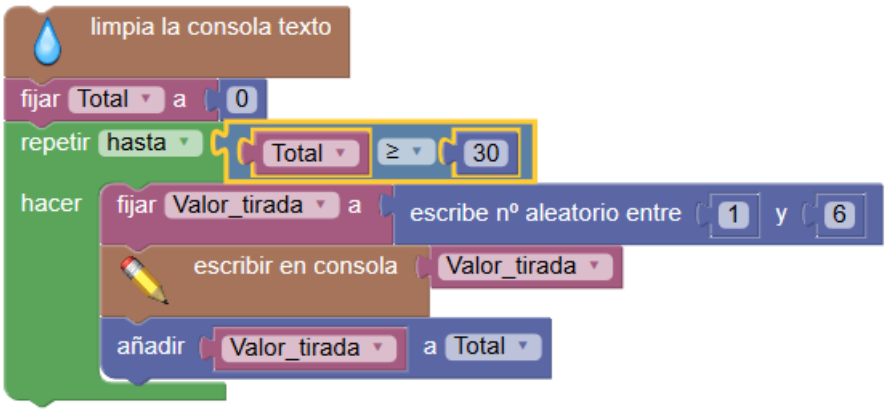


En nuestro ejemplo, el bucle terminaría después de que se hubieran impreso las distintas tiradas con números aleatorios en el rango de 1 a 6, y la variable **Total** contendría la suma de estos números, lo que se garantiza que será de al menos 30.

Para obtener más información, consulte https://en.wikipedia.org/wiki/While_loop .

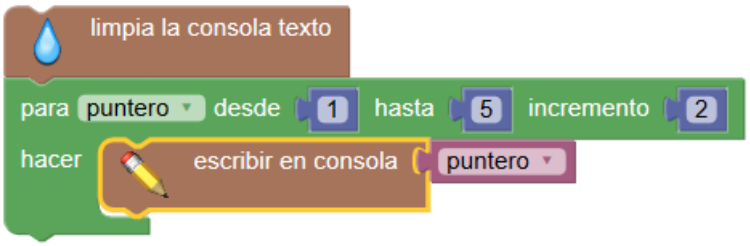
3.7.2.5. Repetir hasta (Until)

"Repita mientras" los bucles repiten sus cuerpos de instrucciones *mientras que* alguna condición es verdadera. Repetir-hasta es similar excepto que se repiten las instrucciones del cuerpo del bucle *hasta que una* cierta condición sea verdad. Los siguientes bloques son equivalentes al ejemplo anterior porque el bucle contiene hasta que **total** es mayor o igual a 30.



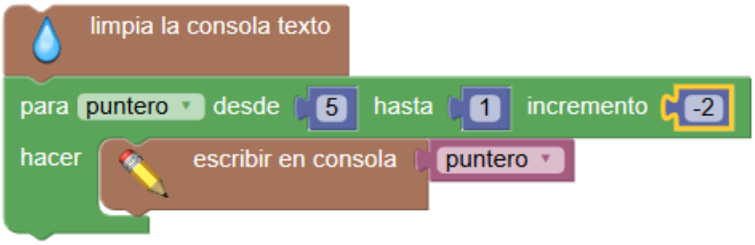
3.7.2.6.Bucle contador (FOR NEXT)

El **contador con** bloque (llamado bucle for en la mayoría de los lenguajes de programación) incrementa una variable desde el primer valor al último valor un incremento determinado (tercer valor), ejecutando las instrucciones que pongamos dentro del cuerpo del bucle. Por ejemplo, el siguiente programa imprime los números 1, 3 y 5. El rango de valores va desde 1 hasta 5 y el incremento es 2.



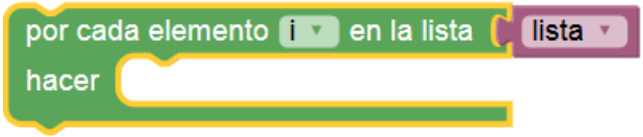
Imprimirá **1, 3, 5**

Incremento -: Como se muestra en la imagen siguiente se puede hacer que el recorrido sea con incrementos negativos estableciendo el rango desde un numero grande a otro más pequeño.

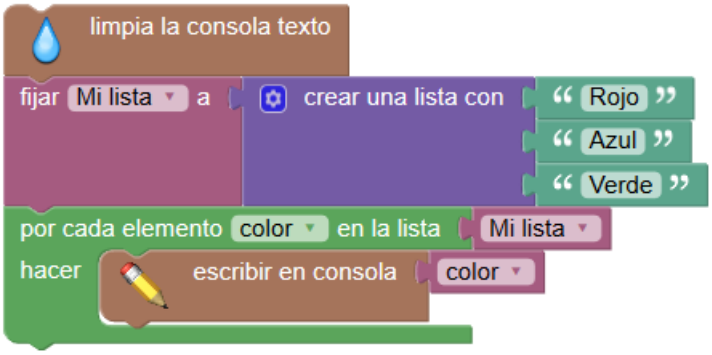


En este caso se imprimirían los valores **5, 3, 1**

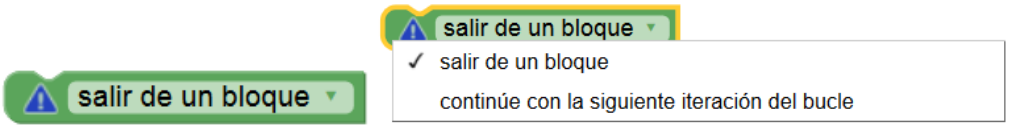
3.7.2.7.Para cada elemento de una lista



Esta instrucción (ver https://en.wikipedia.org/wiki/Foreach) es similar a las anteriores, excepto que en lugar de dar los valores de la variable de bucle en una secuencia numérica, utiliza los valores de una lista. El siguiente programa imprime cada elemento de la lista: "Mi lista", [Rojo, Verde, Azul].



3.7.2.8.Bloques de terminación de bucle

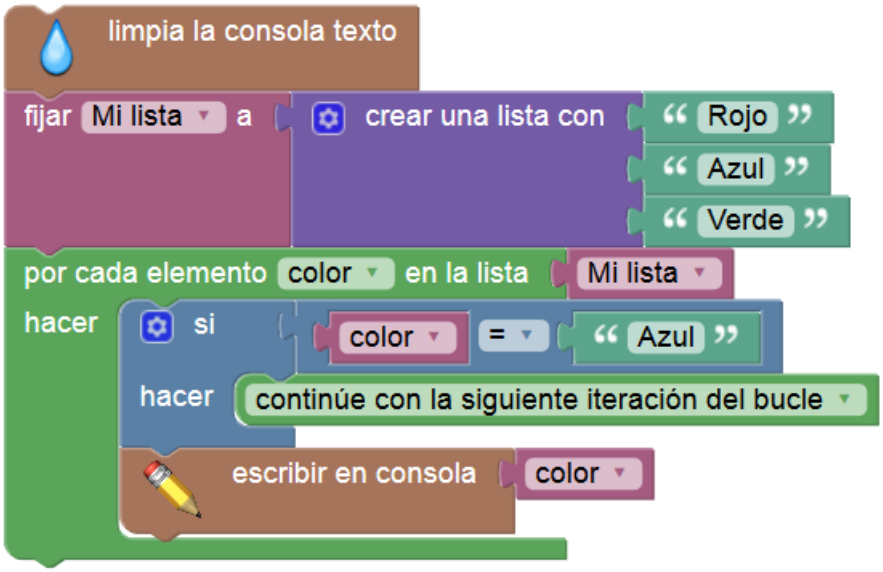


La mayoría de bucles se ejecutan hasta que se cumple la condición de terminación (en el caso de bloques de **repetición**) o hasta que todos los valores han sido tomados por la variable de bucle. Dos bloques raramente necesarios, pero ocasionalmente útiles, proporcionan medios adicionales para controlar el comportamiento del bucle. Aunque los siguientes ejemplos son para **cada** loops, pueden utilizarse con cualquier tipo de bucle.

3.7.2.8.1. Continuar con la siguiente iteración

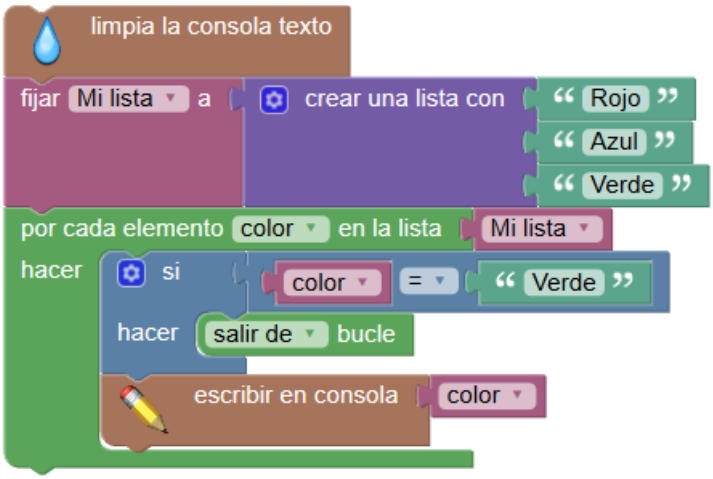
La **continuación con la siguiente iteración** (llamada continuar en la mayoría de los lenguajes de programación) hace que se salte el código restante en el cuerpo y se inicie la siguiente iteración (paso) del bucle.

El siguiente programa imprime "alfa" en la primera iteración del bucle. En la segunda iteración, se ejecuta el bloque de **continuación con siguiente iteración** , omitiendo la impresión de "beta". En la iteración final, se imprime "gamma".



3.7.2.8.2. Salirse del bucle

La **ruptura del bucle** proporciona una salida anticipada de un bucle . El siguiente programa imprime "Rojo" y “Azul” en tercera iteración y "rompe" del bucle cuando la variable de bucle es igual a "Verde". El tercer elemento de la lista nunca se alcanza.



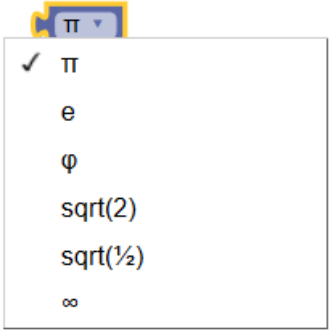
3.8. Instrucciones de tipo Matemático

En este grupo de bloques vamos a estudiar las operaciones matemáticas elementales dejando para después las operaciones que se integran en las librerías Sofus que son específicas de esta herramienta. Ahora veremos las librerías comunes que se incluyen en la aplicación estándar Blockly.

3.8.1. Constantes matemáticas.

- Valores numéricos 
- Constantes especiales 
- Ángulos 

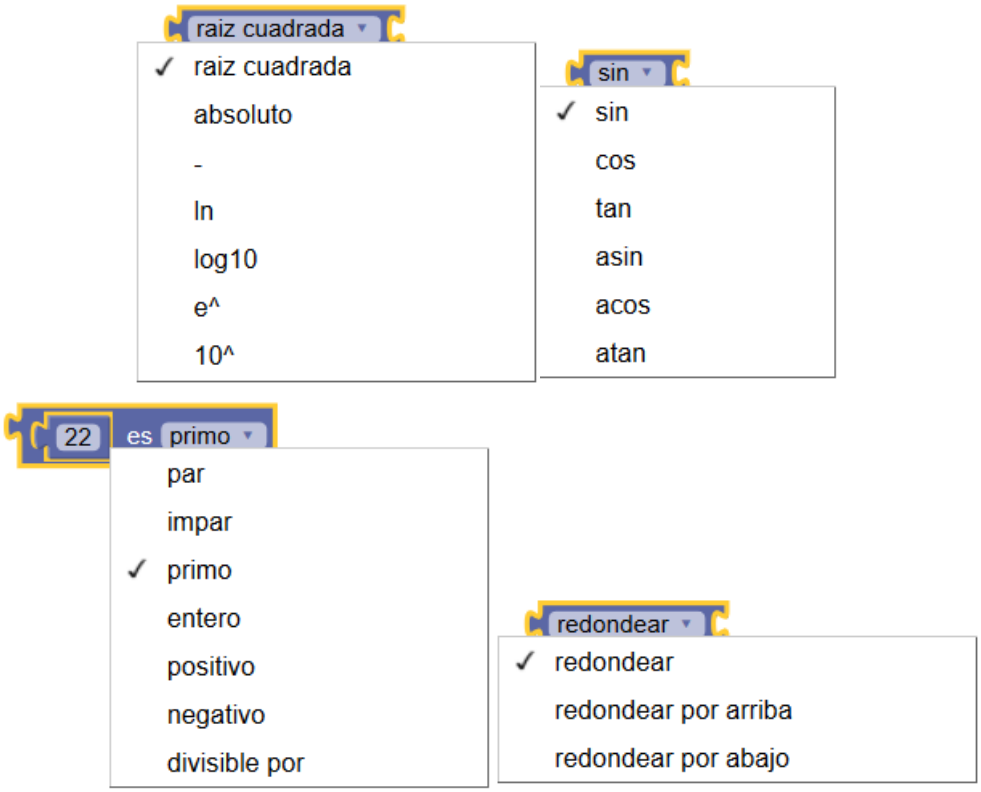
Entre las “Constantes especiales” tenemos las que se muestran en la imagen en la que se despliega el menú de selección.



3.8.2. Funciones

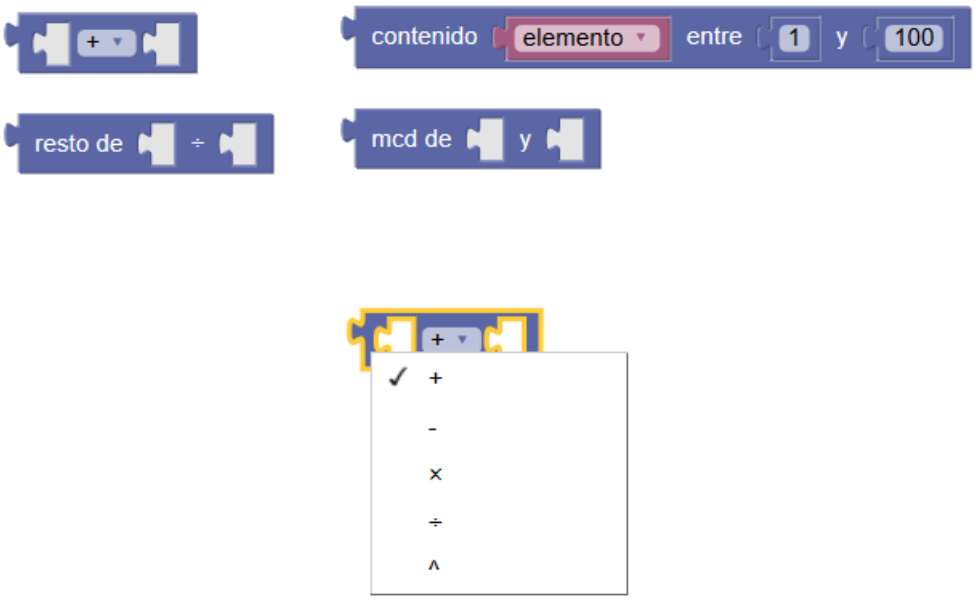
Las funciones que disponemos son las siguientes:

A continuación mostramos las distintas operaciones que aparecen en los menús de cada bloque cuando se pulsa sobre ellos:



3.8.3. Operaciones Matemáticas.

Aquí se recogen las operaciones mas comunes incluyendo el máximo común divisor "**mcd de...",** y la reducción de una numero a l valor máximo indciado en el rango "**contenido**.." así como la operación que devuelve el **"resto de.."** una división



Este bloque permite seleccionar entre las operaciones indicadas.

3.8.4. Números aleatorios.

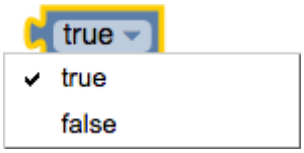
En esta librería encontramos dos bloques. El primero genera un numero aleatorio dentro de un rango que le indicamos en los parámetros del bloque

El segundo bloque genera el numero aleatorio entre 0 y 1 

3.8.5. Operaciones lógicas.

El álgebra booleana es un sistema matemático que tiene dos valores:

- **cierto**
- **falso**

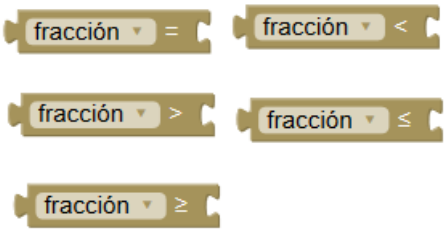


Para la construcción de la parte condicional de las instrucciones tipo **if** y los bloques de repetición tipo "**mientras que**" y "**hasta**".

Existen funciones en Blockly y Sofus que al ejecutarse devuelven "**true**" o "**false**" (cierto o falso) que son valores booleanos. Estas funciones están en la librería

"**Lógica -> Proposiciones**"

"**Lógica -> Test Fracciones**"

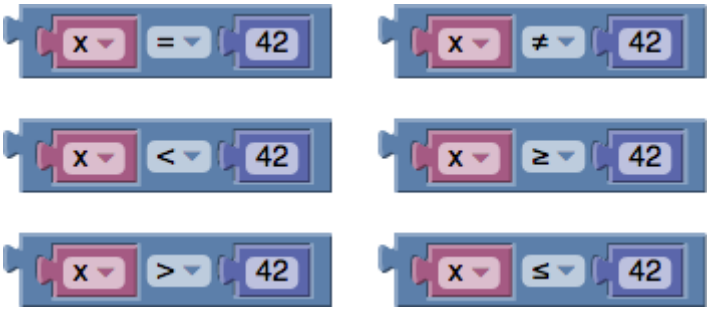


Conviene tener en cuenta:

- Si un bloque espera un valor booleano como entrada, normalmente interpreta una entrada ausente como **falsa** .
- Los valores no booleanos no se pueden conectar directamente donde se esperan valores booleanos, aunque es posible (pero no es aconsejable) almacenar un valor no booleano en una variable y luego conectarlo a la entrada. Ninguna de estas prácticas se recomienda, y su comportamiento podría cambiar en futuras versiones de Blockly.

3.8.5.1.Comparaciones

Hay seis operadores de comparación. Cada uno toma dos entradas (normalmente números) y devuelve verdadero o falso dependiendo de cómo las entradas se comparan entre sí.



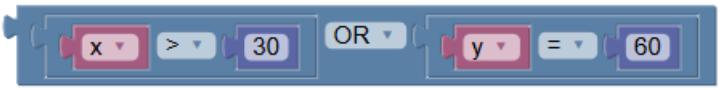
Los seis operadores son: igual, no iguales, menores que, menores o iguales, mayores que, mayores o iguales.

3.8.5.2.Operaciones lógicas

Vemos a continuación algunos ejemplos de operaciones lógicas.

El bloque **AND** devolverá **verdadero** sólo si ambas entradas son también verdaderas.

El bloque **OR** devolverá **true** si cualquiera de sus dos entradas es verdadera.



3.8.5.3.Función no

Aunque nosotros podemos seleccionar "Cierto" o "Falso", internamente el programa trabaja como "true" y "false"

El bloque **no** convierte su entrada booleana en su opuesto. Por ejemplo, el resultado de:

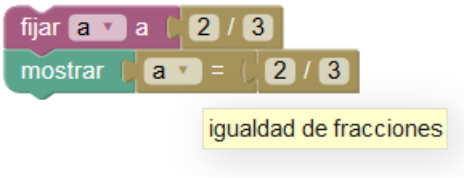
Es **true**.

Como se mencionó anteriormente, si no se proporciona ninguna entrada, se asume un valor de **true** , por lo que el siguiente bloque produce el valor **false** :

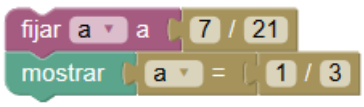
Sin embargo, no se recomienda dejar una entrada vacía.

3.8.5.4.Comparación de fracciones.

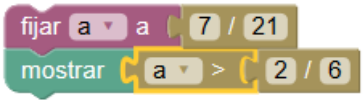
Se trata de utilizar los bloques de la librería "**Test Fracciones**"



Esta operación de comparación devuelve **"true"**



Este otro devolvería también **"true"**



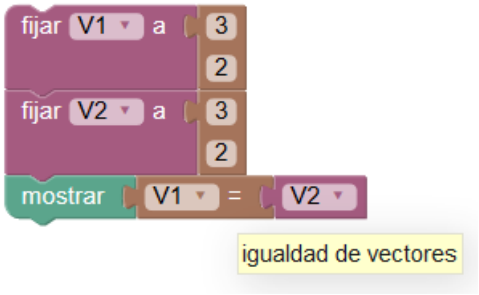
Esta comparación devuelve "**false**"

3.8.5.5.Comparación con vectores.

Es posible también averiguar si se cumplen algunas condiciones con relación a dos vectores. Tales bloques aparecen en la librería "**Lógica -> Vectores**"

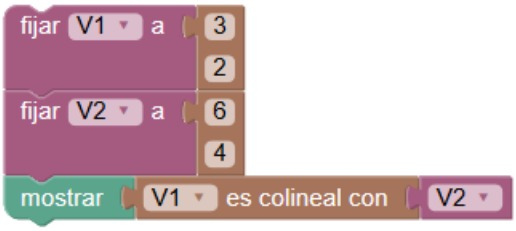
Se puede averiguar s dos vectores son **"iguales**", "**colineales o paralelos**" y "**perpendiculares**"

- **Igualdad de vectores**



Devolvería "**true**" -> Los vectores son iguales

- **Vectores colineales o paralelos.**



Devolvería "**true**" -> Los vectores son paralelos

- **Vectores perpendiculares**

Devolvería "**true**" -> Los vectores son perpendiculares

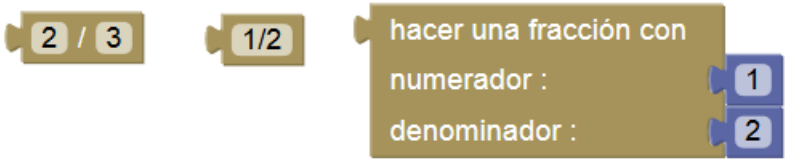
Es posible hacerlo también con vectores tridimensionales.

Aquí devolvería **"true"**

3.8.6. Trabajando con Fracciones.

3.8.6.1. Constantes.

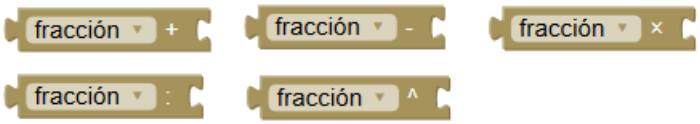
Las fracciones se pueden escribir de tres maneras:



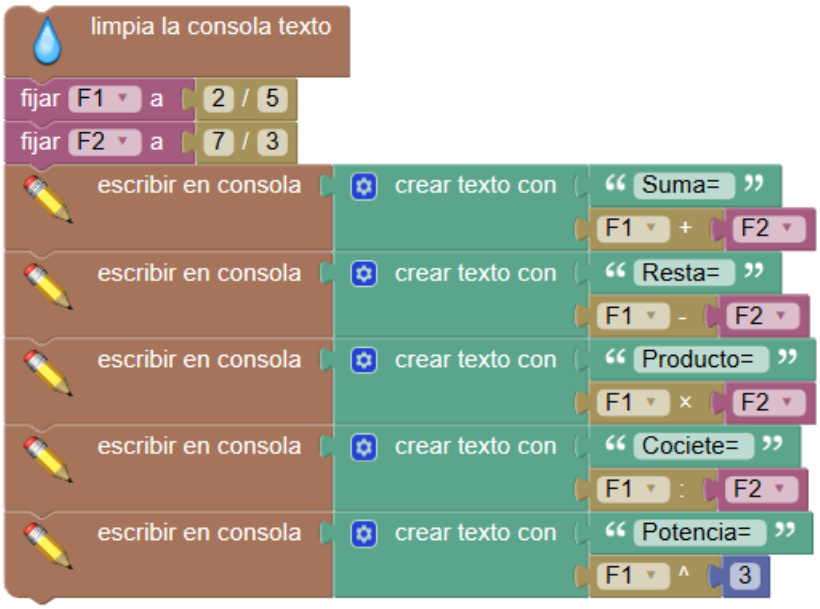
Las tres son equivalentes

3.8.6.2.Operaciones con fracciones

Con las fracciones se pueden realizar las operaciones que aparecen en la librería "**Matemáticas -> Fracciones -> Operaciones**"



En el siguiente ejemplo tenemos un ejemplo de operaciones con dos fracciones definidas F1=2/5 y F2=7/3

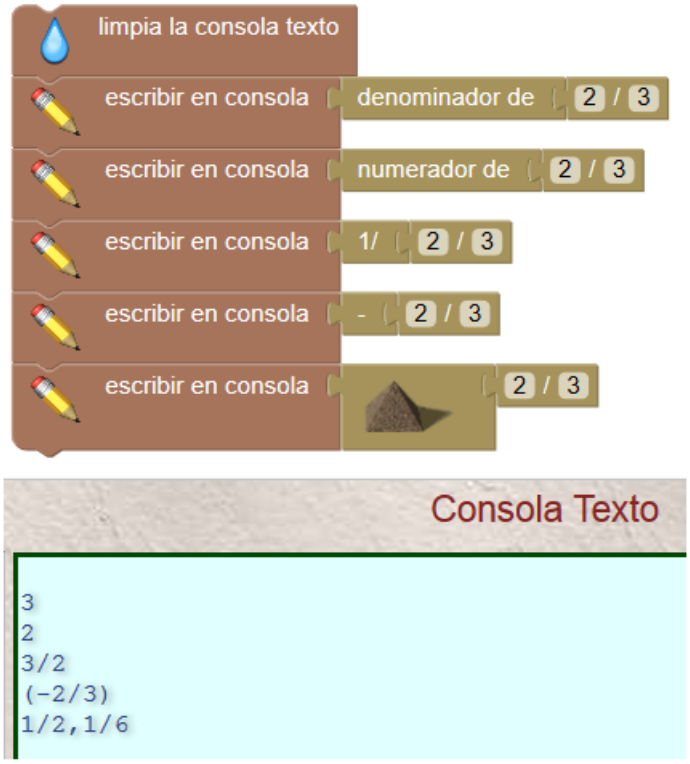


Este es el resultado en la Consola de Texto

Es posible realizar operaciones con fracciones con más de dos fracciones simplemente uniendo bloques de operación.

3.8.6.3.Funciones con fracciones.

A continuación vemos un sencillo ejemplo de los bloques que se incorporan en la librería "**Matemáticas -> Fracciones -> Funciones con fracciones**"



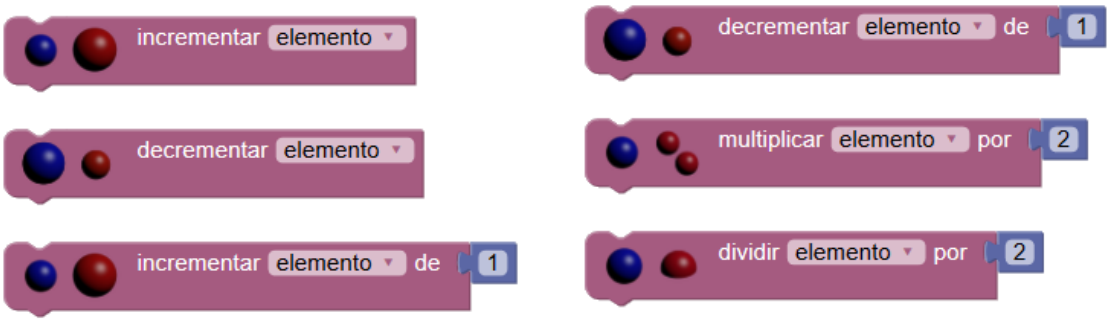
"denominador de"	Devuelve el denominador de la fracción
"numerador de"	Devuelve el numerador de la fracción
"1/"	Devuelve la fracción inversa
"-"	Devuelve la fracción cambiada de signo
"  "	Devuelve "fracciones egipcias"

3.9. Librerías "Matemáticas -> Sofus"

Esta librería significa una aportación importante a Blockly que permite afirmar la herramienta como un importe aporte al diseño de aplicaciones de tipo matemático.

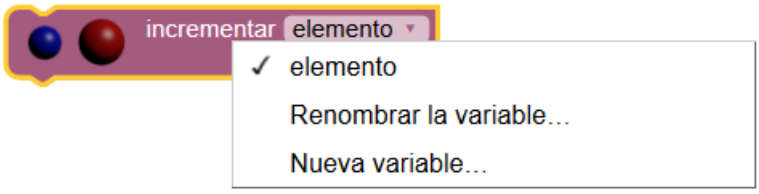
3.9.1. Trasformaciones.

Los bloques de función de esta librería afectan y cambian una variable numérica de la manera que el propio bloque indica en el texto que lo identifica.



Vemos hay dos tipos, el primero no presenta parámetro de entrada, es decir incrementa o decremento la variable. El otro tipo realiza la operación indicada pero con un parámetro.

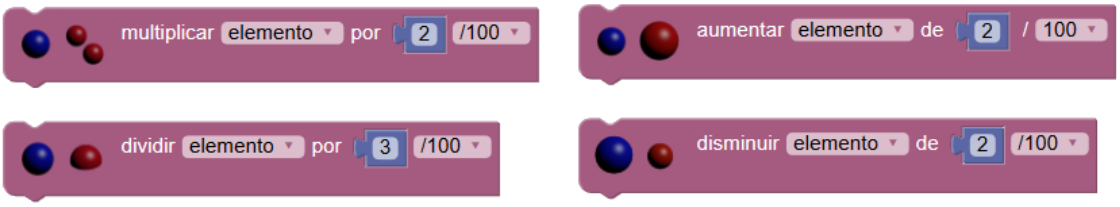
Pulsando en la parte "**elemento**" obtenemos una ventana de selección en la que podremos seleccionar una variable que ya tuviésemos definida o crea una "**Nueva variable**"

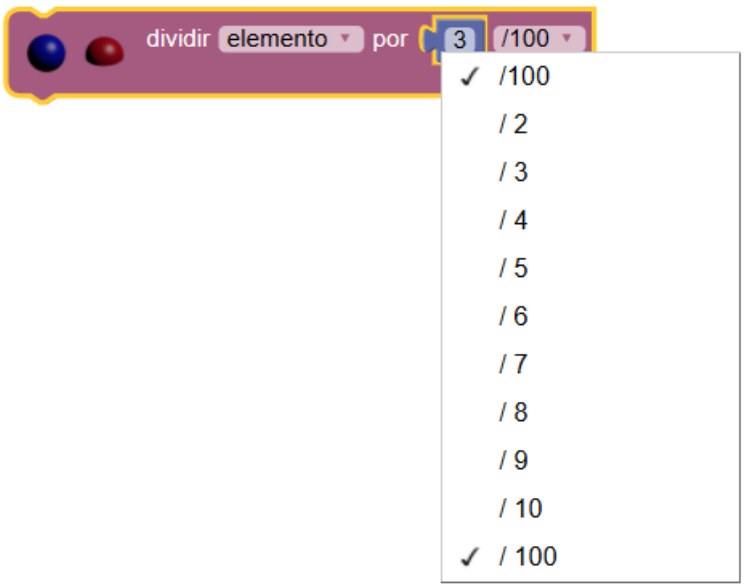


Vemos un sencillo ejemplo.

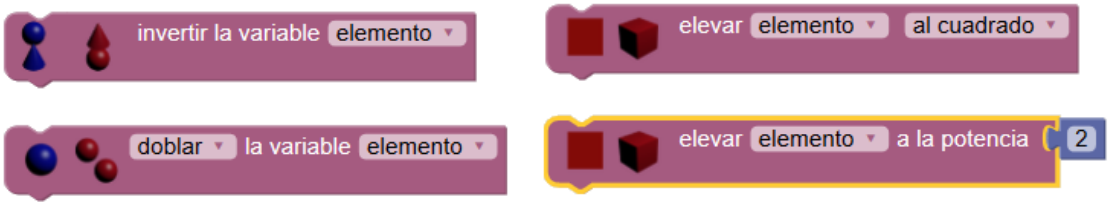
Estos bloques no admiten datos de tipo “texto”

3.9.2. Porcentajes.



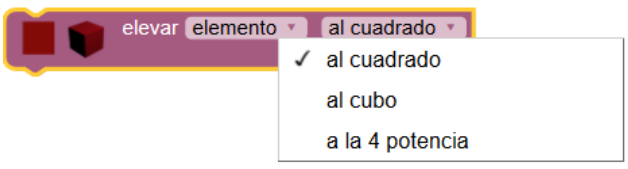


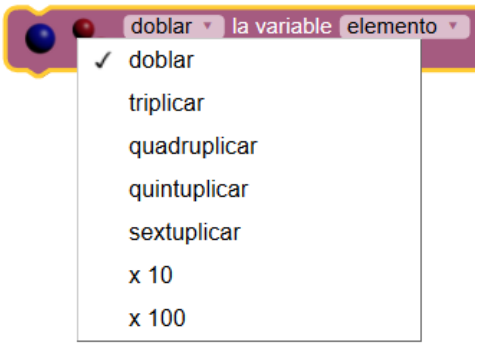
3.9.3. Funciones.



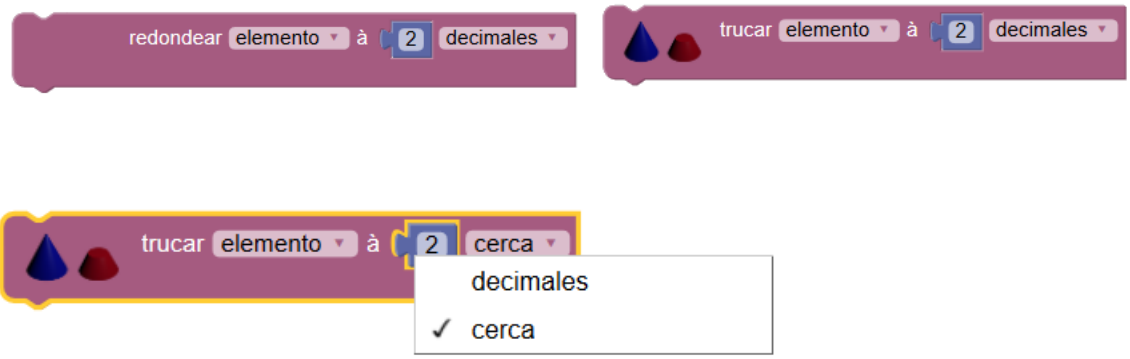
Estos bloques permiten implementar las funciones matemáticas que indican.

Tenemos que considerar que dos de ellos despliegan funciones.

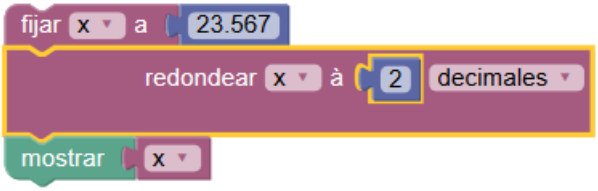




3.9.4. Redondear.



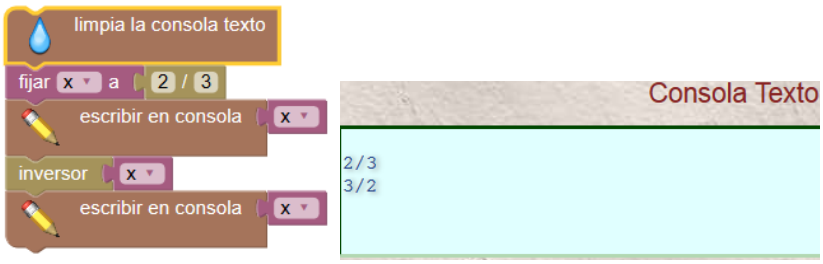
La función "redondear.." permite redondear al número de decimales marcados en el parámetro del bloque. Si introducimos x=23.567 la función



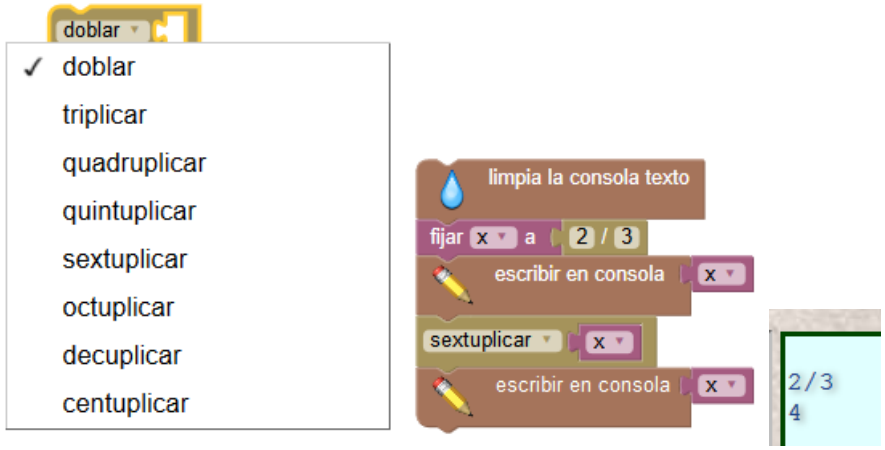
nos devolvería 23.57

3.9.5. Fracciones.

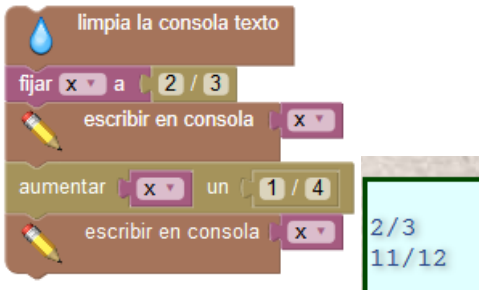
La función "**inversor**" invierte la fracción" tal como se ve en el ejemplo



A continuación vemos la función "**doblar**" que despliega las funciones que se muestran. En el ejemplo hemos puesto "**sextuplicar**"

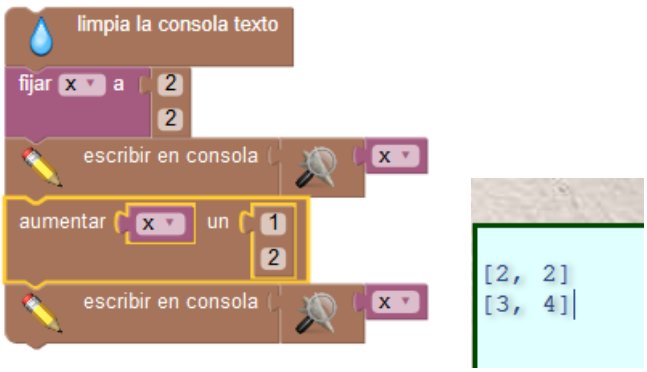


La función "**aumentar**" devuelva la suma de una variable en formato fracción otra fracción.



3.9.6. Vectores.

Vemos un ejemplo de uso de la función "Aumentar.. un.."



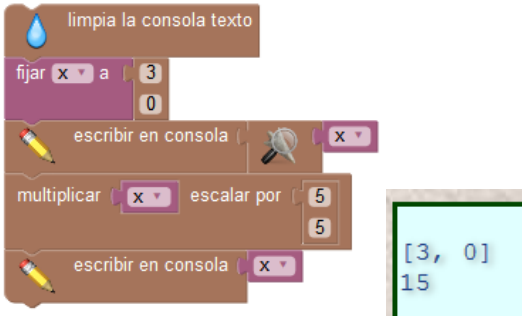
Veamos el ejemplo del producto escalar de dos vectores

$\vec{u} = (3, 0) \quad \widehat{uv} = 45°$

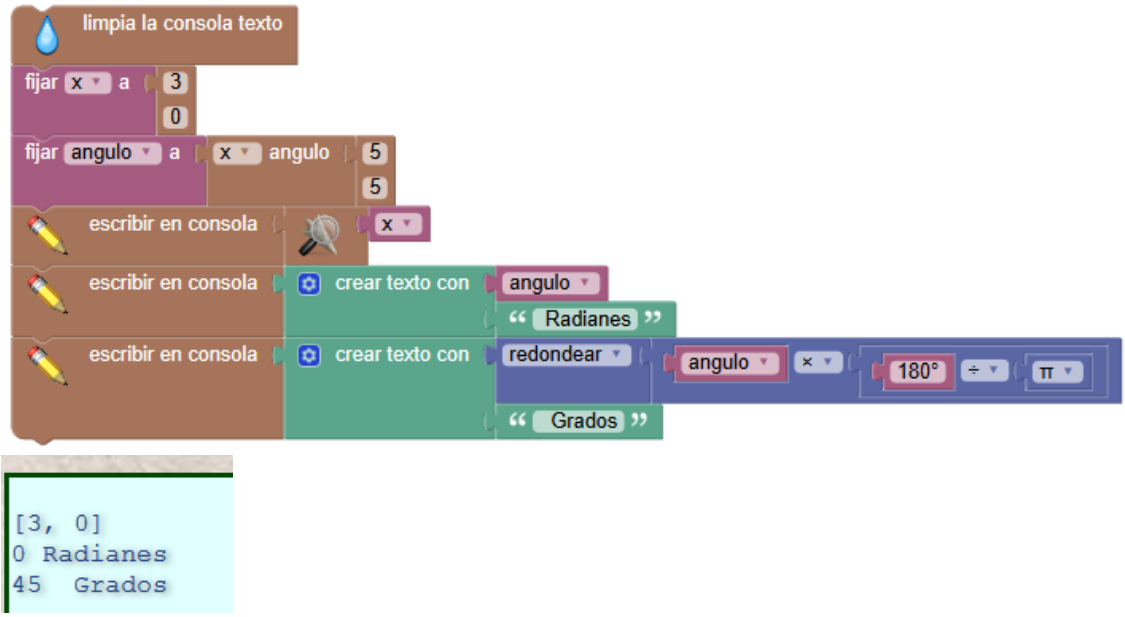
$\vec{v} = (5, 5)$

$\vec{u} \cdot \vec{v} = \sqrt{3^2 + 0^2} \cdot \sqrt{5^2 + 5^2} \cdot \cos 45°$

$\vec{u} \cdot \vec{v} = 3 \cdot 5 \cdot \sqrt{2} \cdot \frac{\sqrt{2}}{2} = 15$

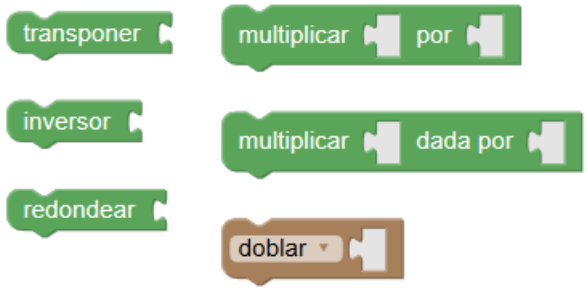


El ángulo que forman los dos vectores se puede comprobar con el siguiente algoritmo.

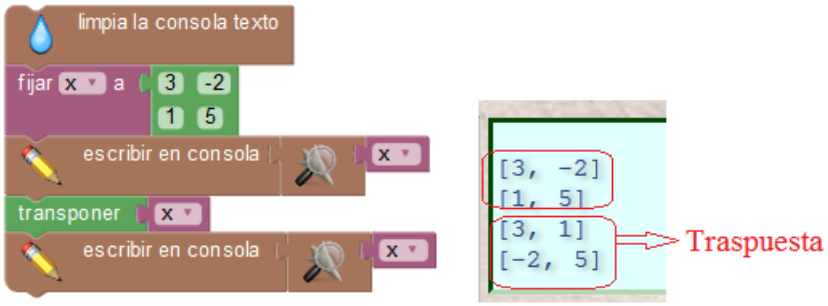


Podemos comprobar que la función "**angulo**" devuelve el ángulo en radianes, lo hemos convertido en grados para que también se vea de esta manera

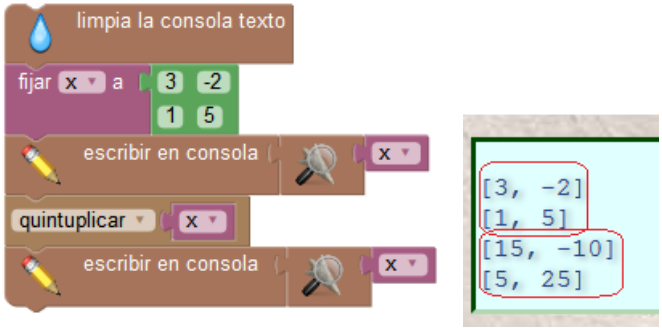
3.9.7. Matrices.



Calcular la matriz traspuesta de una matriz



Quintuplicar una matriz

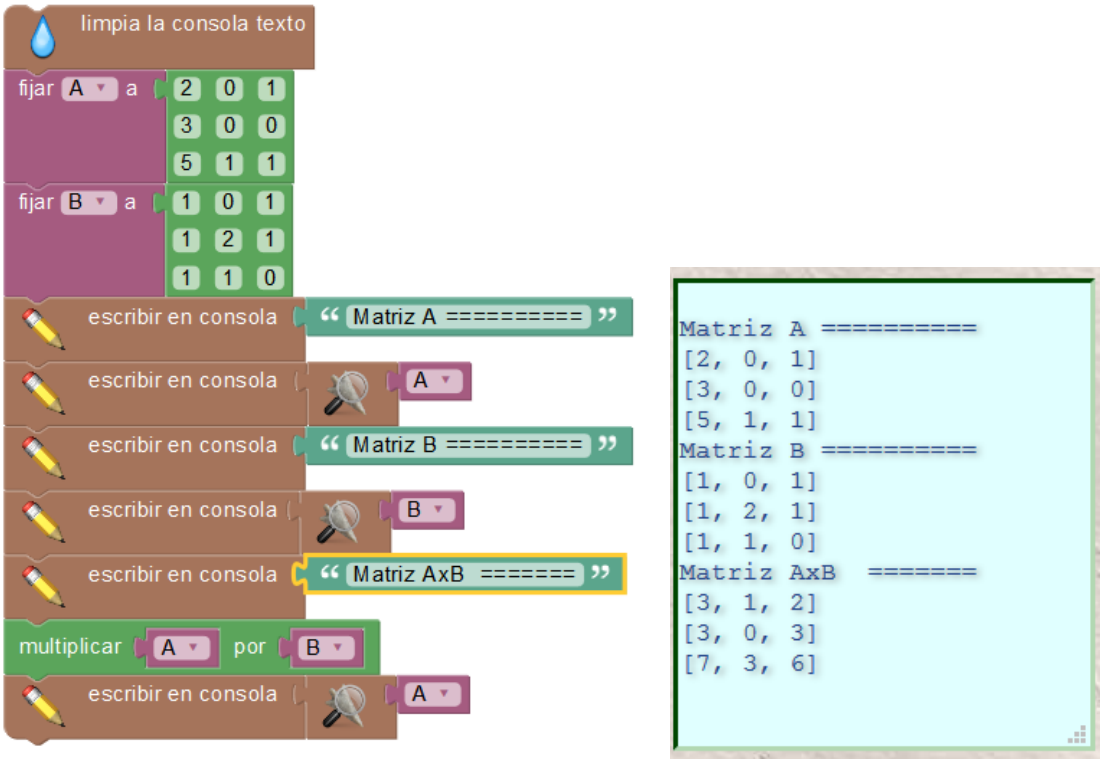


Producto de dos matrices.

Sean las matrices A y B

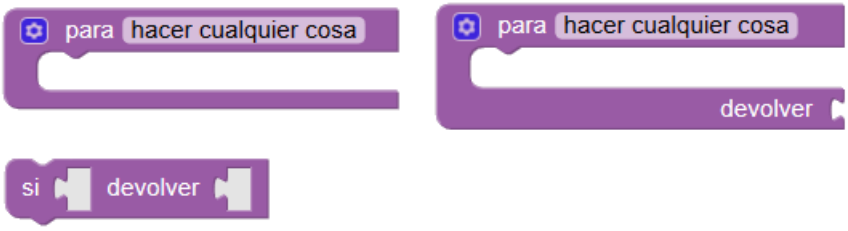
$$A \cdot B = \begin{pmatrix} 2 & 0 & 1 \\ 3 & 0 & 0 \\ 5 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 0 \end{pmatrix} =$$

$$A \cdot B = \begin{pmatrix} 2 \cdot 1 + 0 \cdot 1 + 1 \cdot 1 & 2 \cdot 0 + 0 \cdot 2 + 1 \cdot 1 & 2 \cdot 1 + 0 \cdot 1 + 1 \cdot 0 \\ 3 \cdot 1 + 0 \cdot 1 + 0 \cdot 1 & 3 \cdot 0 + 0 \cdot 2 + 0 \cdot 1 & 3 \cdot 1 + 0 \cdot 1 + 0 \cdot 0 \\ 5 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 & 5 \cdot 0 + 1 \cdot 2 + 1 \cdot 1 & 5 \cdot 1 + 1 \cdot 1 + 1 \cdot 0 \end{pmatrix} = \begin{pmatrix} 3 & 1 & 2 \\ 3 & 0 & 3 \\ 7 & 3 & 6 \end{pmatrix}$$



3.10. Realizar Funciones.

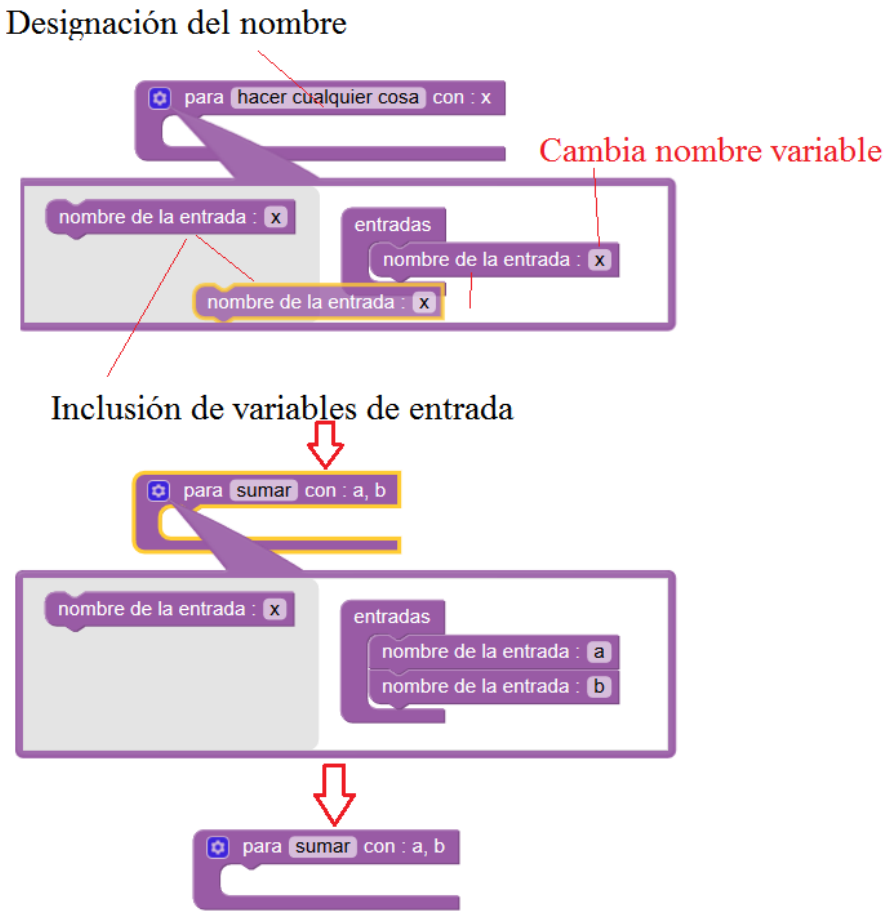
La realización de un algoritmo de un programa en ocasiones requiere el uso de funciones. Las funciones como su nombre nos indica son estructuras del programa que realizan una determinada función y a las que se puede invocar cuando lo necesitemos. La librería que Sofus nos ofrece para el tratamiento de las funciones es la misma que nos ofrece Blockly. Estos son los bloques que ofrece:



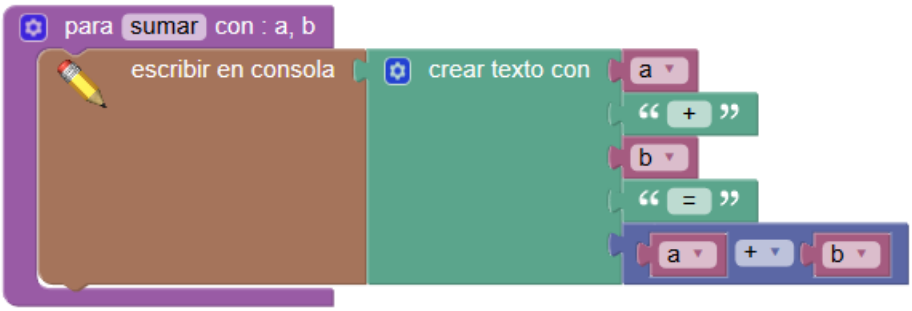
Para definir una función se usan cualquiera de los dos primeros bloques de tipo "**para**"

3.10.1. Función sin devolver valor:

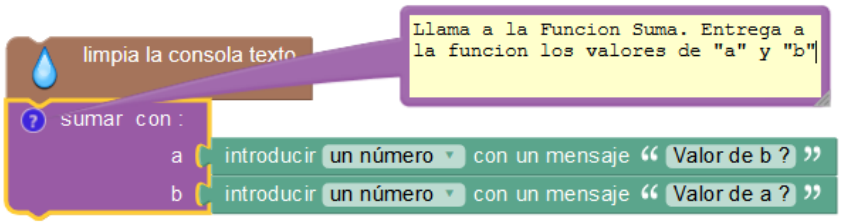
En el primer bloque se crea una función que no devuelve ningún valor "función sin salida".

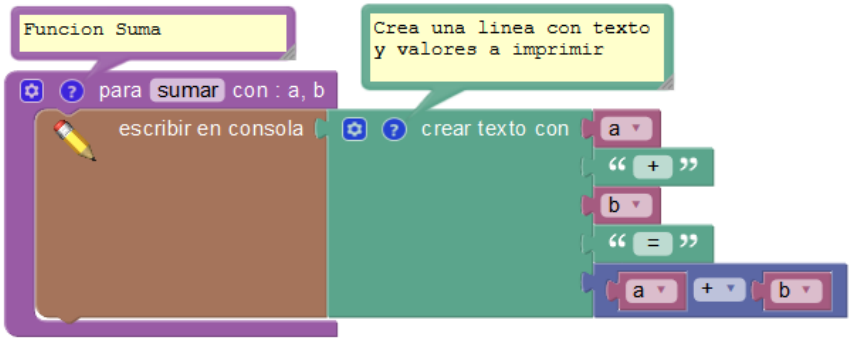


La función que queremos hacer muestra la suma de dos variables “a” y b”.



El programa principal borra la Consola de Texto e invoca a la Función Suma pidiendo previamente los valores “a” y “b” al usuario para entregarlos a la función.

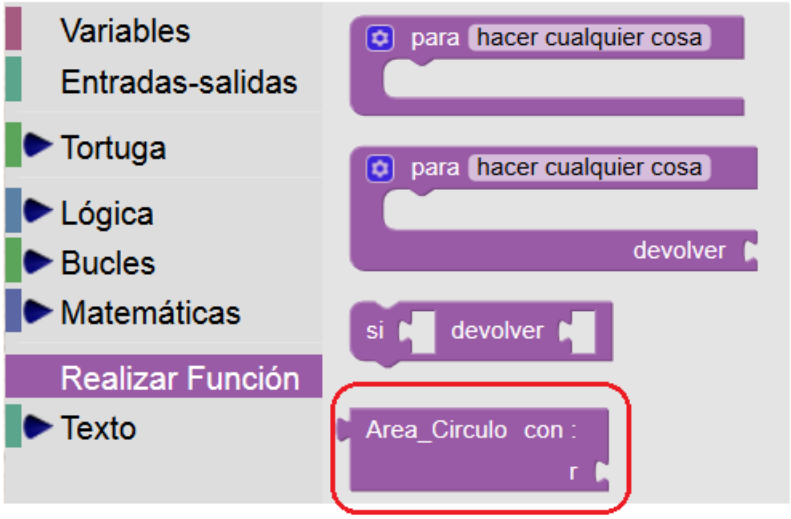




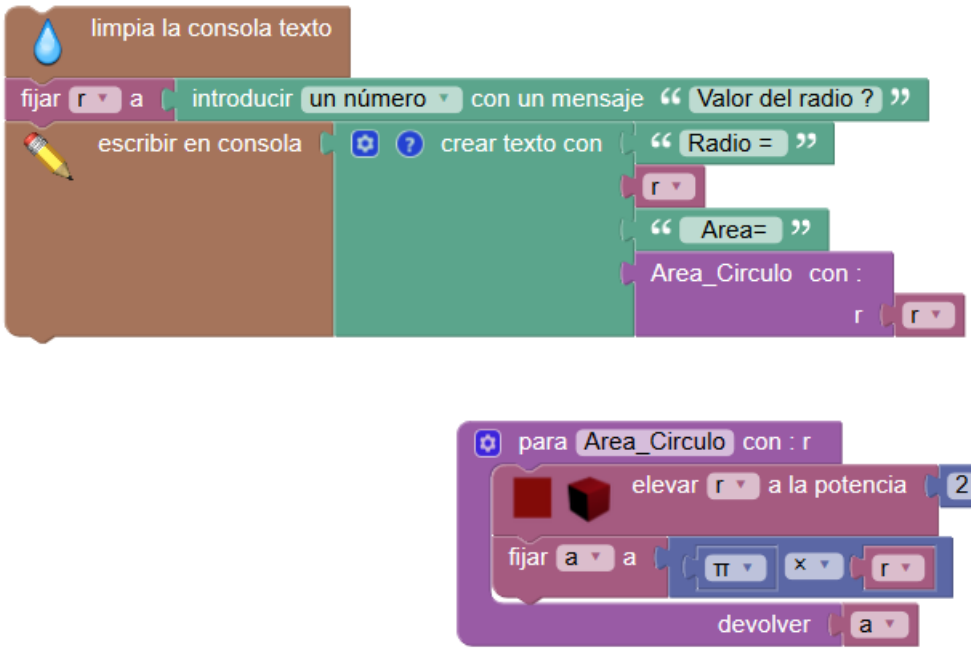
3.10.2. Función que devuelve valor.

En este caso definimos la función que calcula el área de un círculo "**Area_Circulo**" la función recogerá el valor del radio "**r**" , calcula el área "**a**" y devuelve este valor a través del bloque que la invoca.

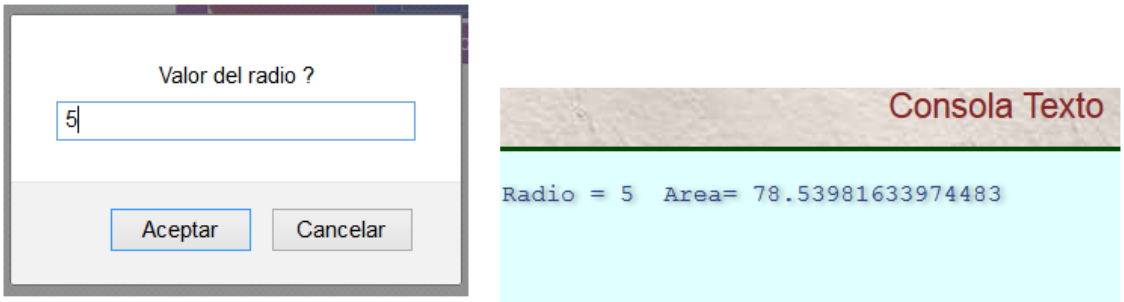
Siempre que se crea una función aparece en la pestaña "Realizar" la función creada.



Este es el programa que utiliza este tipo de función.

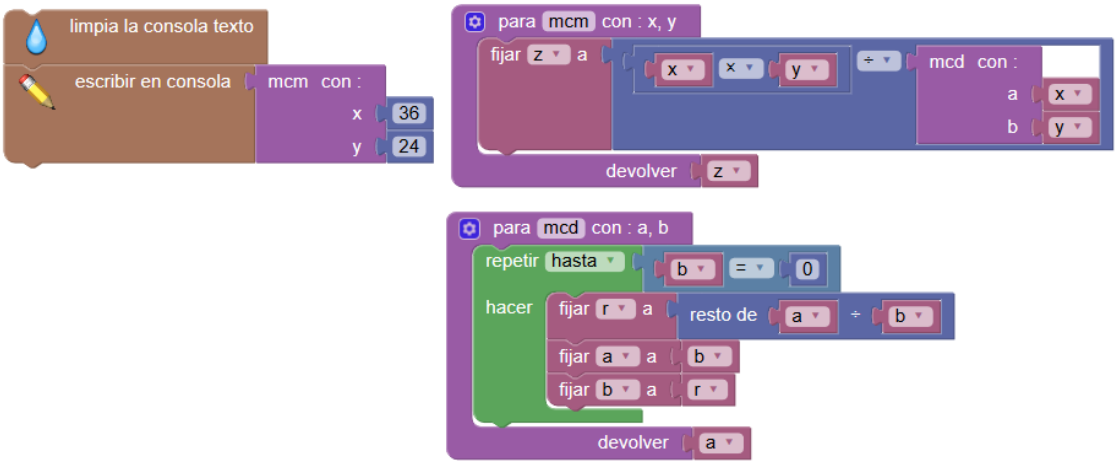


Este es el resultado que se muetra en la Consola de Texto cuando cuando ejecutamos el programa dando al radio el valor de 5



Es posible encadenar varias funciones.

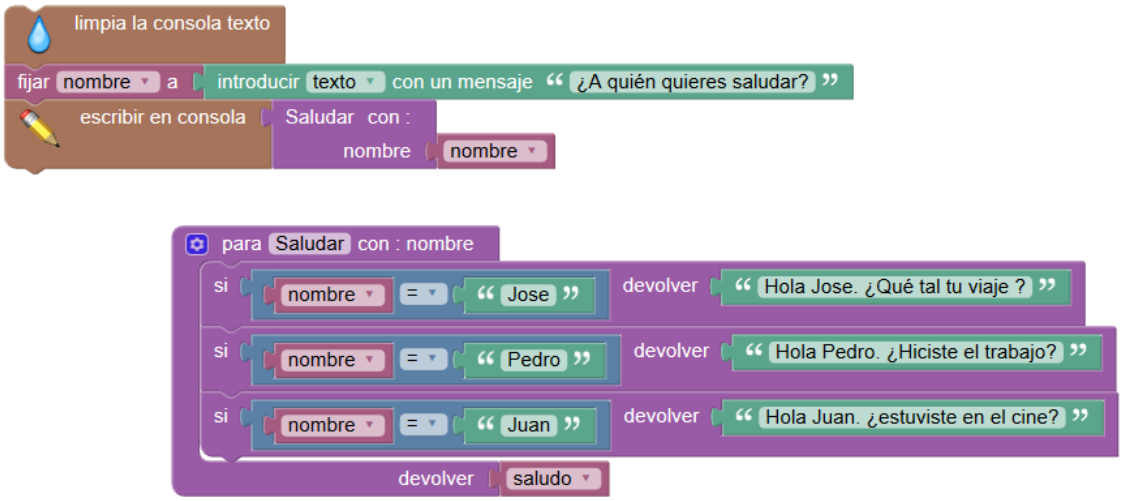
A continuación vemos como calcular el mcm calculando previamente el mcd.



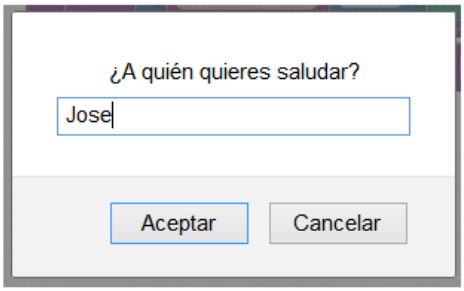
3.10.3. Función “si.. devolver”.

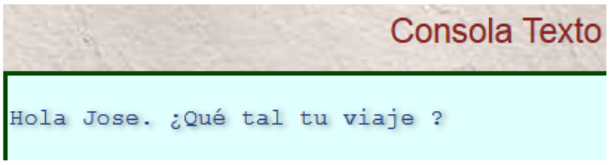
Esta función solo se puede utilizar dentro de un bloque de definición de función del que devuelve valor. Se trata de establecer unos condicionales en los cuales, si se cumple la condición, de devuelve un valor numérico o de texto.

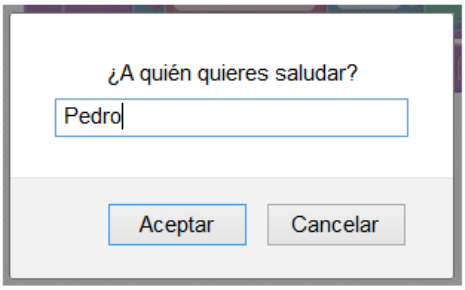
A continuación ponemos un ejemplo en el que se establece la función “**Saludar**” que recoge la variable “**nombre**” los condicionales “**si .. devolver**” recogen la condición del nombre y devuelven una cadena de texto que es un saludo.

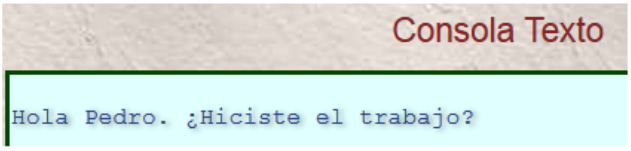


Vemos a continuación algunas pruebas.



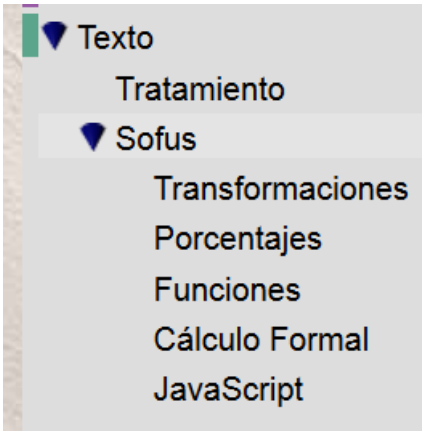






3.11. Librerías Texto-Sofus

Nos referimos ahora a las librerías específicas que incluyen Sofus y no Blockly. Estas librerías aunque pueden parecer similares a las librerías **Matemáticas -> Sofus** en este caso hablamos de las librerías **Texto -> Sofus**. Una de las diferencias es que en esta librería existen bloques que procesan cadenas de texto en las que es posible identificar letras de números. Veremos a continuación los contenidos.

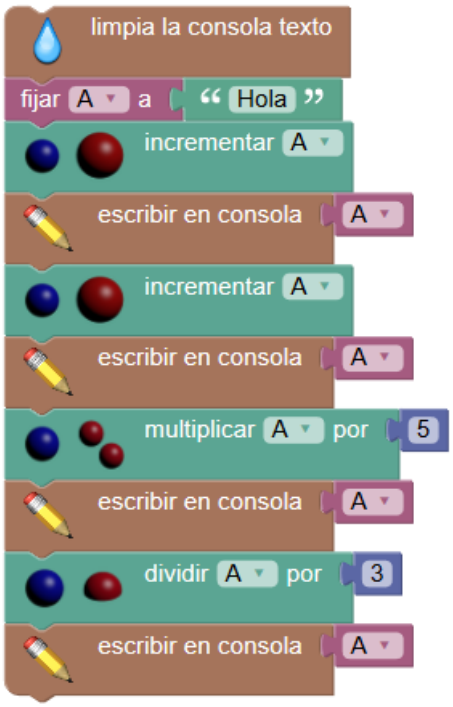


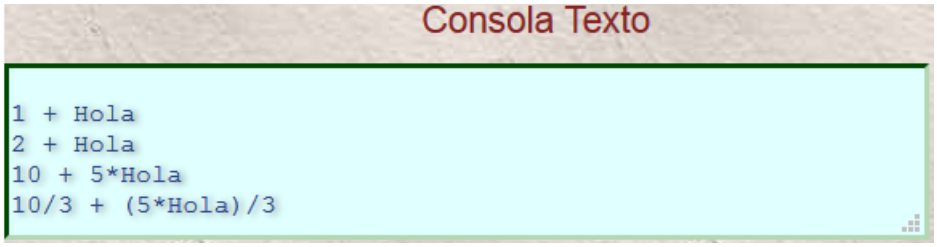
3.11.1. Transformaciones.

En este caso podemos partir de tres supuestos:

- Variable de tipo texto (sin valores numéricos)

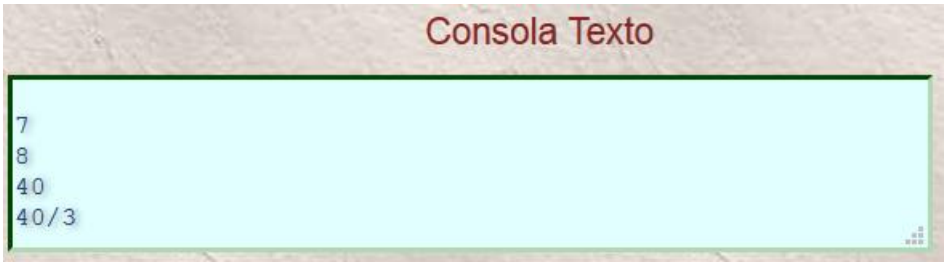
En este caso vemos que el tratamiento en lo que se referte a los bloques es incluir valores numéricos (incrementar o decrementar) o realizar cálculos sobre la cadena que en cada paso se va transformando.





- Variable de tipo numérico.

Si el valor de A es por ejemplo A=6 entonces ese mismo diagrama de bloques genera la siguiente salida.



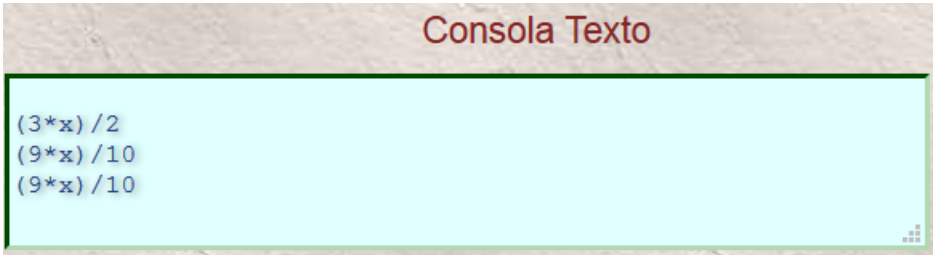
- Variables no admitidas

Si el valor de A= “12 hola” nos error

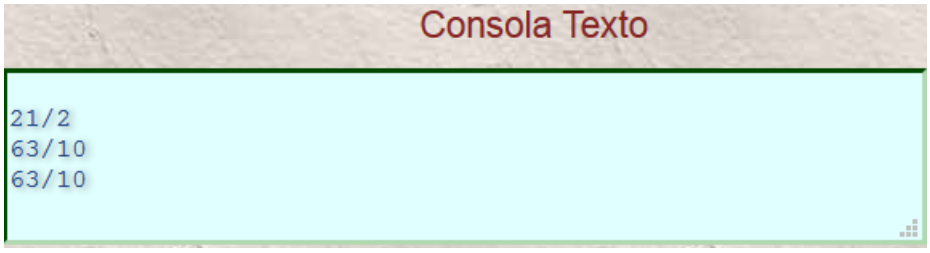
Si el valor de A= “12+ hola” lo acepta

Se pueden probar otras opciones…

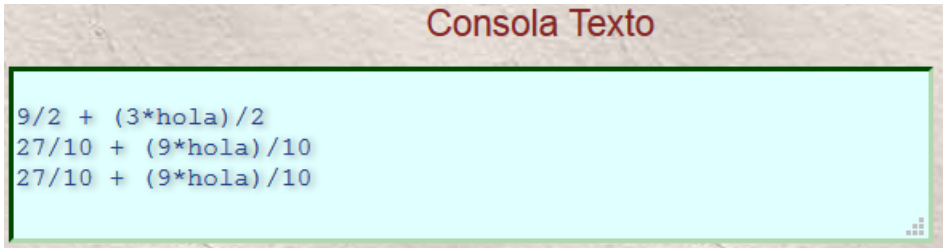
3.11.2. Porcentajes.



Si A= 7



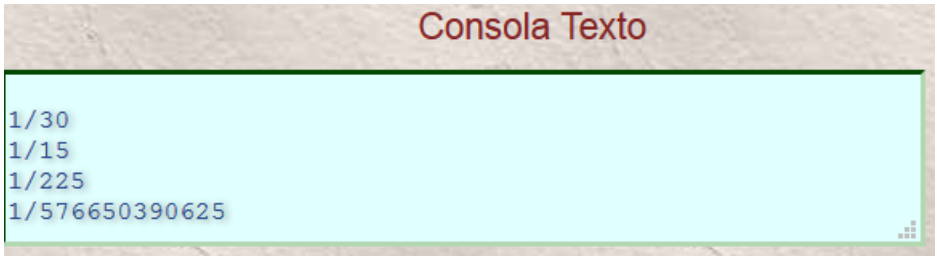
Si A= 3 + Hola



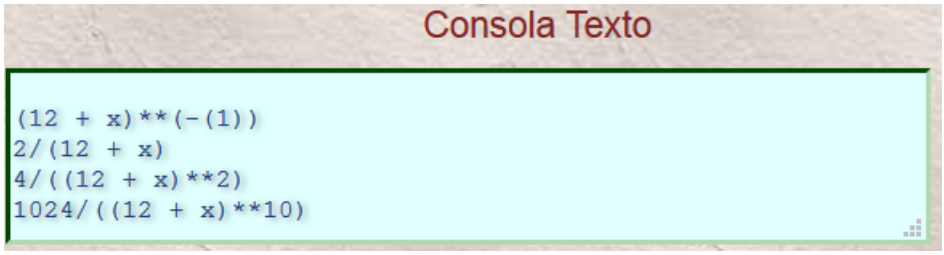
3.11.3. Funciones.

Las funciones afectan de la misma a la variable A. Veamos unos ejemplos

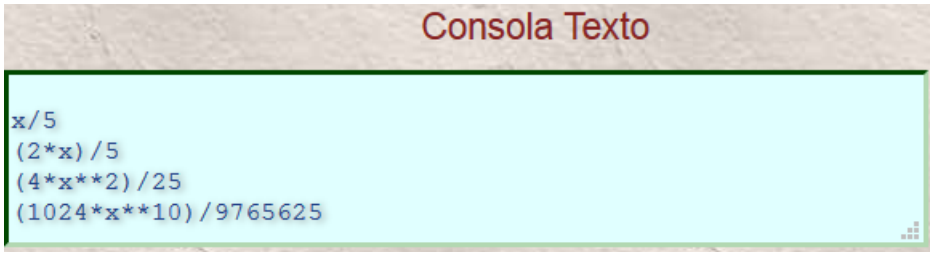
Cuando A=30 Los resultados son los siguientes:



Si A= 12 +x Los resultados son los siguientes:



Si A=5/x Los resultados son los siguientes:



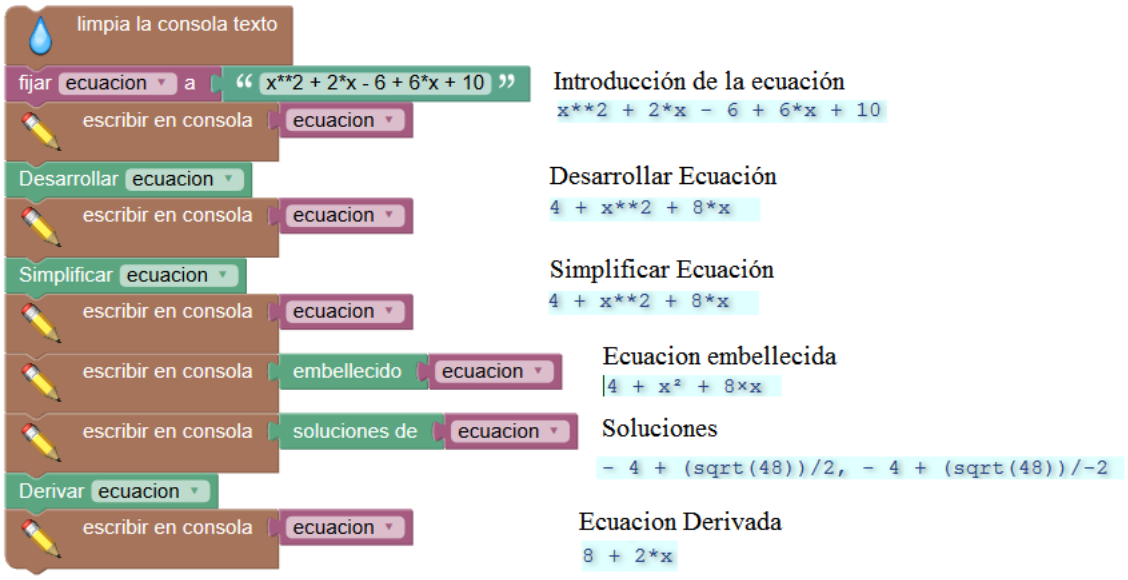
3.11.4. Cálculo Formal

En la librería **Text -> Sofus -> Cálculo** Formal encontramos una serie de bloques de función que pasamos a comentar y que revisten una gran importancia en el tratamiento de las ecuaciones.

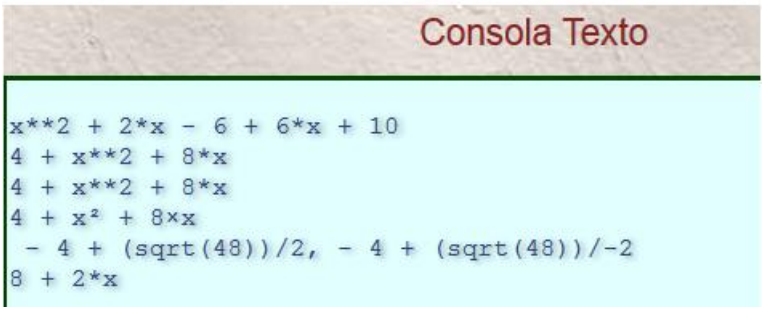
Es importante saber que para escribir por ejemplo

x^2 debemos escribir x**2

x^4 debemos escribir x**4



Esta aplicación devuelve:



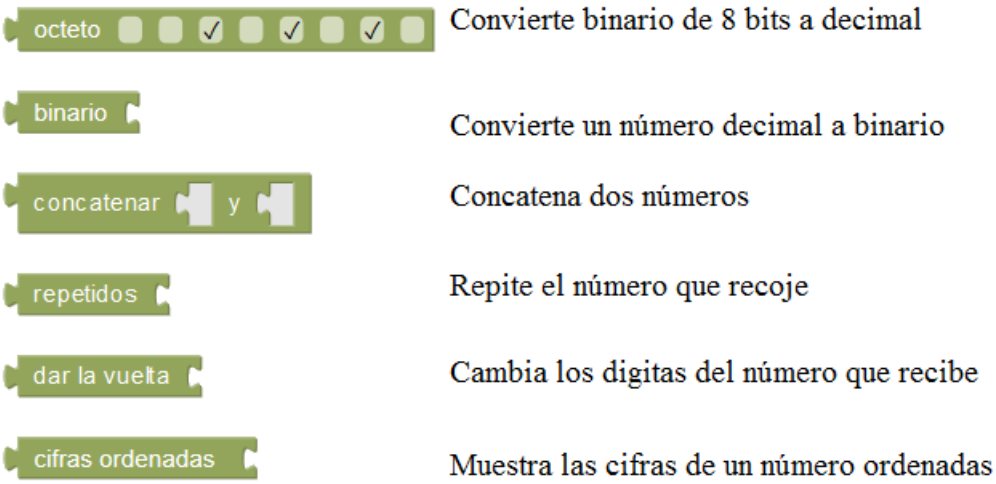
3.12. Librería Kaprekar

Esta librería se encuentra en la carpeta de librerías:

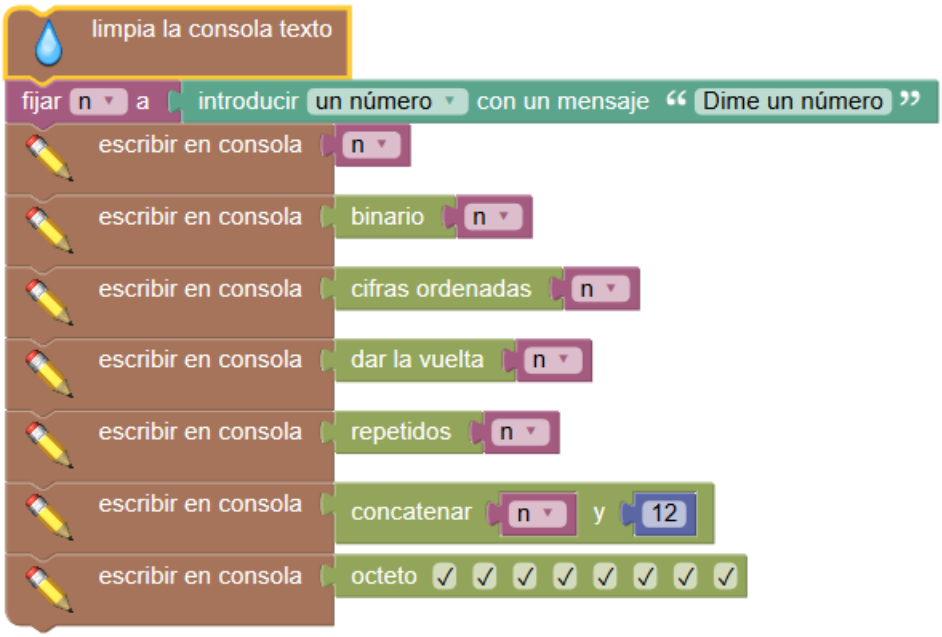
Lógica->Vampiros->Kaprekar

Sus funciones están relacionadas con los cambios de base Binario-decimal decimal-binario y otras más.

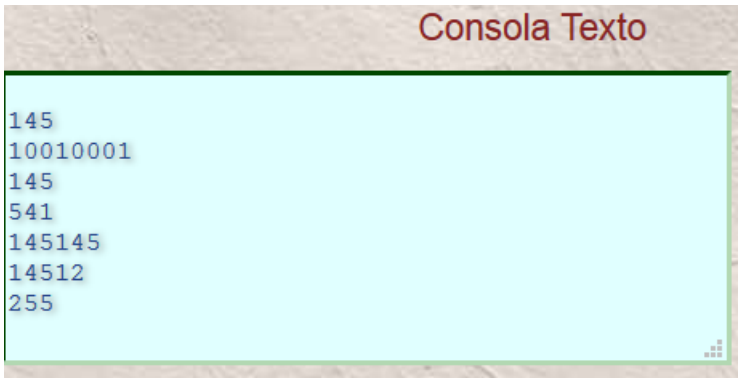
Seguidamente se muestran los bloques de esta librería y su función.



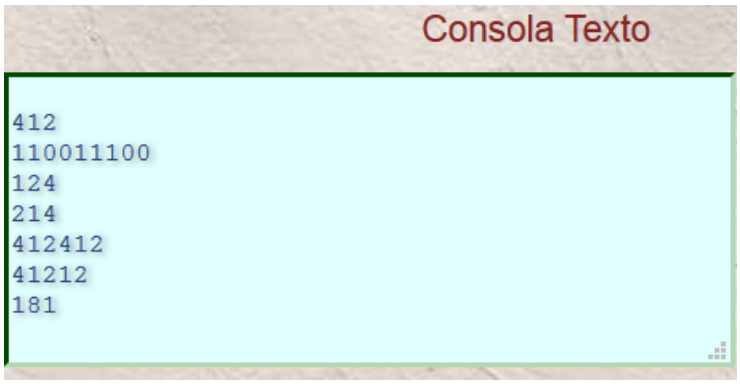
En el siguiente algoritmo usamos cada uno de los bloques para mostrar lo que hacen.



Hemos escrito el número 145.



Hemos escrito el número 412.



Sofus es una herramienta desarrollada por:

Autores:

- Alain Busser
- Patrice Debrabant
- Patrick Raffinat
- Florian Tobé

Depositada en: https://github.com/AlainBusser/Sofus

Versión del Documento V 1.0 Agosto 2017

Prof. José Manuel Ruiz Gutiérrez
j.m.r.gutierrez@gmail.com