Trabajando con Listas

Hasta ahora hemos trabajado en base a los ejemplos para ir comprendiendo esta herramienta. Este es el primer caso en el que vamos a programar desde cero.

El enunciado que queremos resolver es el siguiente: "Queremos jugar a la primitiva pero estamos cansados de jugar con nuestros números de siempre que no salen nunca, así que vamos a consultar al Oráculo **gatuno**".

Para ello vamos a trabajar con un nuevo elemento llamado **Lista** y que se encuentra en el mismo lugar que las variables. La lista representa un conjunto de valores, números o palabras, a las cuales podemos acceder a través de su posición. En un lenguaje de programación común se conoce como **Vector (Array).**

Al hacer clic sobre nueva lista podemos darle el nombre y acceder al conjunto de operaciones ya definidas. Llámalo **Ciudades.**

En la pantalla de los objetos podemos ver la lista vacía junto al gato, ya definido.

Vamos a ver un pequeño ejemplo para ver las instrucciones que disponemos sobre la lista.

1. Comenzaremos inicializando la lista de ciudades.

2. A continuación insertaremos tres elementos en la lista:Valencia, Alicante y Castellón.

3. Ahora insertaremos 3 ciudades más en las posiciones primera, ultima y cualquiera

Ejecuta desde la bandera verde y compruébalo. 

4. Vamos a reemplazar el ultimo elemento de la lista, **"Bilbao"** por **"Madrid"**

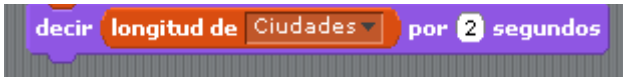
Ejecuta desde la bandera verde y compruébalo. 

5. Ahora borramos el ultimo elemento que hemos modificado anteriormente.

Observad que ahora tenemos 5 elementos en la lista.

6. Vamos a decir que el gato diga cuantos elementos tiene la lista, para ello arrastra:

y sobre la palabra hola la longitud de Ciudades, obteniendo:



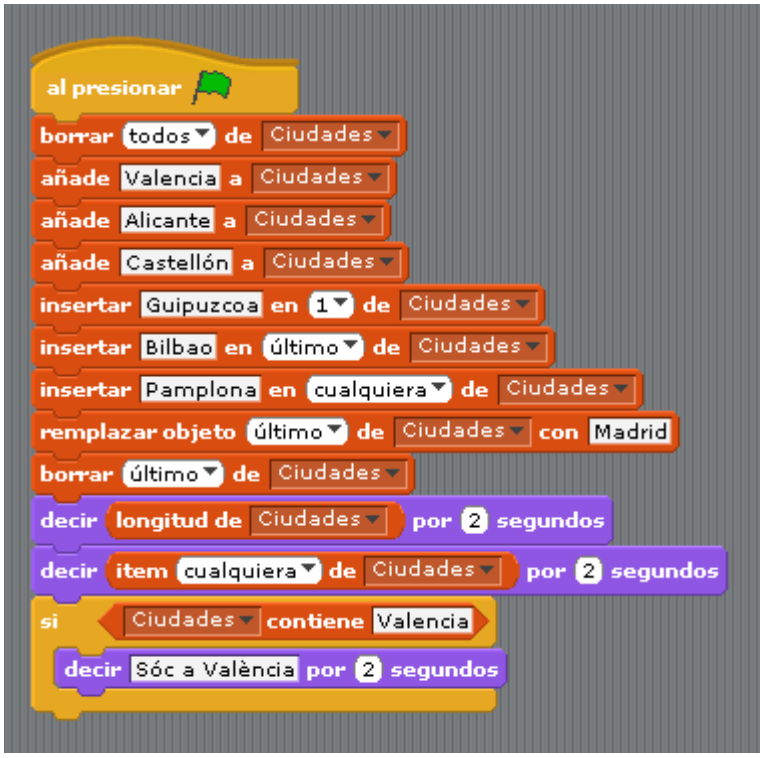
Ejecuta desde la bandera verde y compruébalo. 

7. Ahora haremos que el gato nombre una de las ciudades de la lista, la que quiera, es decir de forma aleatoria. Las posibilidades de esta instrucción son **primer y ultimo elemento**, ademas de **cualquiera.**

Ejecuta desde la bandera verde y compruébalo 

8. Por ultimo diremos que el gato diga una frase si una determinada ciudad se encuentra dentro de la lista, en este caso será Valencia.

El ejemplo completo es el siguiente:



Hasta aquí hemos visto las posibilidades que ofrece este nuevo elemento. Ahora volveremos a nuestro enunciado inicial y construiremos una primera versión del programa. Analizaremos el funcionamiento y lo modificaremos.

Versión Inicial.

El programa pretende que genere aleatoriamente seis números para jugar a la primitiva. Por tanto la dificultad principal consiste en detectar si el numero ya existe en la lista y en ese caso generar otro numero antes de insertarlo en la lista. Comencemos:

A) **Programa Principal**, asociado a la *banderita* 

1. El programa principal debe *inicializar variables y lista.*

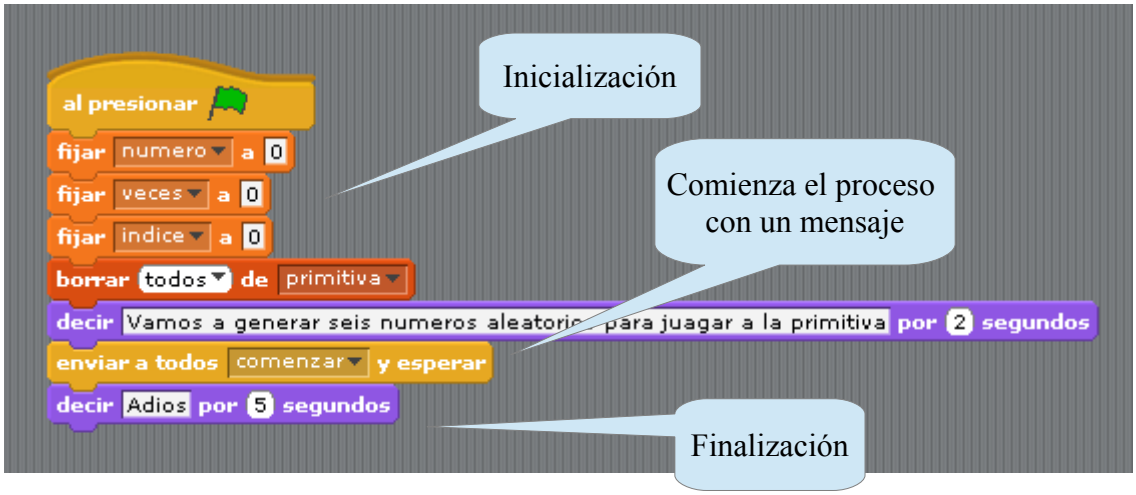
   Las variables que vamos a usar son:

   - **Numero**: Para guardar el numero generado
   - **Veces:** Para controlar cuantos números vamos a generar, en nuestro caso son 6.
   - **Indice**: Para controlar la posición de los elementos de la lista; Primero, segundo, tercero, etc..ultimo.

   Y la lista de números a guardar,a la que llamaremos **primitiva.**

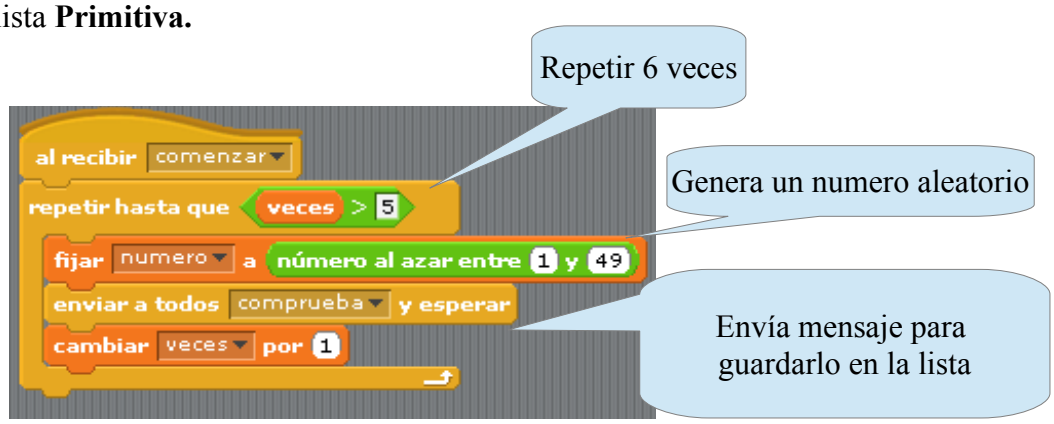
2. Debe comenzar el proceso, en este caso, enviara un mensaje llamado *"Comenzar"* para empezar a generar la lista de números.

3. Debe indicar el final del proceso con algún mensaje, p.e **Decir Adios durante 2 segundos.**



B) **Bloque Comenzar**, asociado a *Recibir Comenzar.*

Este bloque se encarga de generar los números aleatorios y enviar un mensaje para que lo inserten en la lista **Primitiva.**

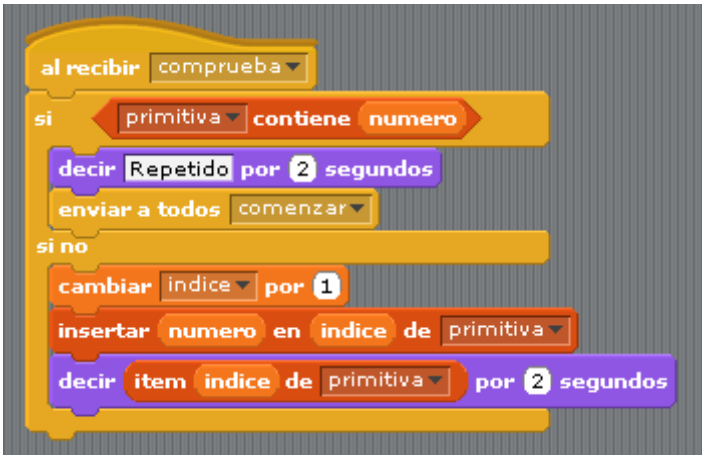


Cuando se genera un numero aleatorio(numero), se envía el mensaje para que se guarde en la lista (y se espera)y se incrementa veces(indica el número de veces que hemos de ejecutar este bloque) para indicar que este proceso lo hemos de hacer una vez menos, ya que hemos guardado un numero en la lista.

C) **Bloque Comprueba**, asociado a *Recibir Comprueba.*

Este bloque debe comprobar que si el numero que genero el bloque anterior (Comenzar) esta en la lista. En caso negativo, incrementa el indice (hay un nuevo elemento en la lista), inserta el numero en la lista y lo muestra por pantalla.

En caso de que ya exista el numero, lo indica por pantalla y llama de nuevo al bloque anterior (Comenzar)para que genere un nuevo numero aleatorio.



Comprobamos ahora la ejecución de todo, pulsando en la 

Podemos observar que el proceso finaliza correctamente. Pero estamos seguros que el proceso funciona bien cuando se repiten números.

Vamos a comprobarlo pero cambiando el rango de números de **1 a 49** por **1 a 7**. Ejecutamos de nuevo el programa. Ahora las repeticiones son muy normales.

Observamos que los bloques *Comenzar y Comprueba* finalizan pero **NO** el bloque principal, que sigue activo.

¿A que se debe esto?. Cada vez que nos hemos encontrado con una repetición se vuelve a llamar al bloque *Comenzar*, que se vuelve a ejecutar.

Este bloque tiene un *repetir hasta que veces>5*, variable que solo este bloque va a modificar. Hasta aquí no vemos problema alguno.

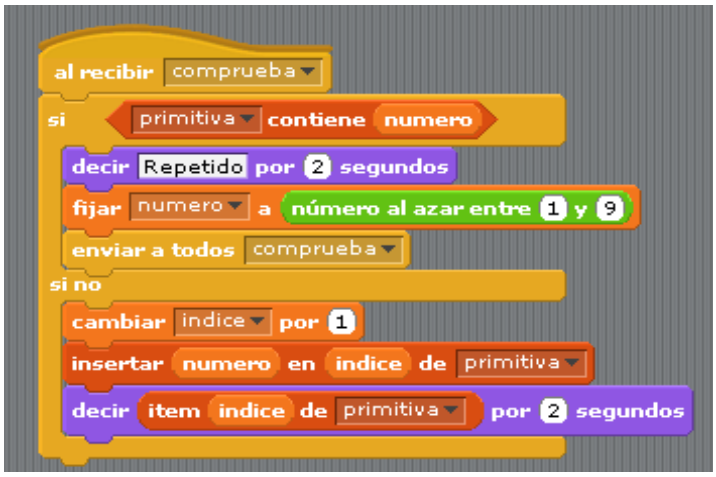
Pero si observamos el bloque principal este esta esperando a que todos los bloques comenzar hayan finalizado y esto puede significar que se haya quedado en un bucle infinito porque hayan muchas instancias del bloque *Comenzar*, que llamo el bloque *Comprueba*.

Esto nos lleva a realizar otra versión diferente a la inicial.

Versión modificada I

Si deseamos mantener la estructura del bloque principal, que parece lógica deberemos introducir cambios en alguno de los otros dos bloques. Y según hemos deducido el bloque a cambiar en principio será el *Comprueba*, ya que era el que debe de encargarse del proceso de los números repetidos.

La idea ahora es la siguiente: Cuando detecte que el numero esta repetido el mismo bloque va a generar el numero aleatorio y se va a llamar a si mismo para poder seguir el proceso, hasta que no se produzca el numero repetido. De esta forma no realizamos ninguna llamada al bloque previo. Para poder ver bien las repeticiones vamos a reducir aun mas el rango, de 1 a 7. en los dos bloques *Comenzar y Comprueba.*



¿Que sucede ahora? Por un lado la ejecución del bloque principal acaba antes que los otros dos bloques y esto no debería suceder y ademas este bloque se queda en un bucle continuo.

El bloque Comenzar

esta esperando a que comprueba finalice.

Pero desde comenzar se realizan llamadas a *Comprueba* de forma independiente a esta. Y por tanto siguen ejecuciones independientes.

De la misma forma el bloque principal esta esperando a *Comenzar* y cuando este finaliza el

bloque *Comprueba* aún no acabo.

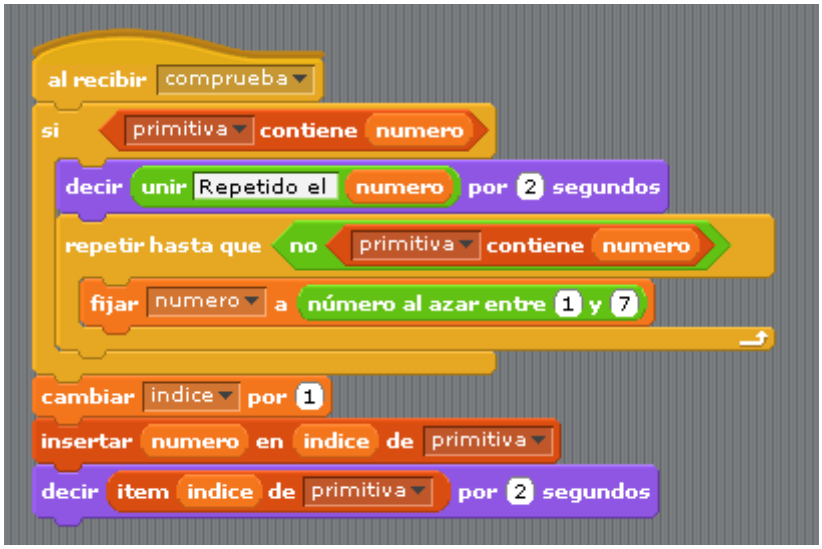
Obviamente la solución pasa porque el propio bloque *Comprueba* gestione de forma independiente, sin llamadas, la problemática con los números repetidos.

Versión modificada II

Vamos a cambiar el bloque **Si..Sino por un Si.**

Este primer Si debe dar solución a los números repetidos y a continuación insertaremos el numero en la lista.

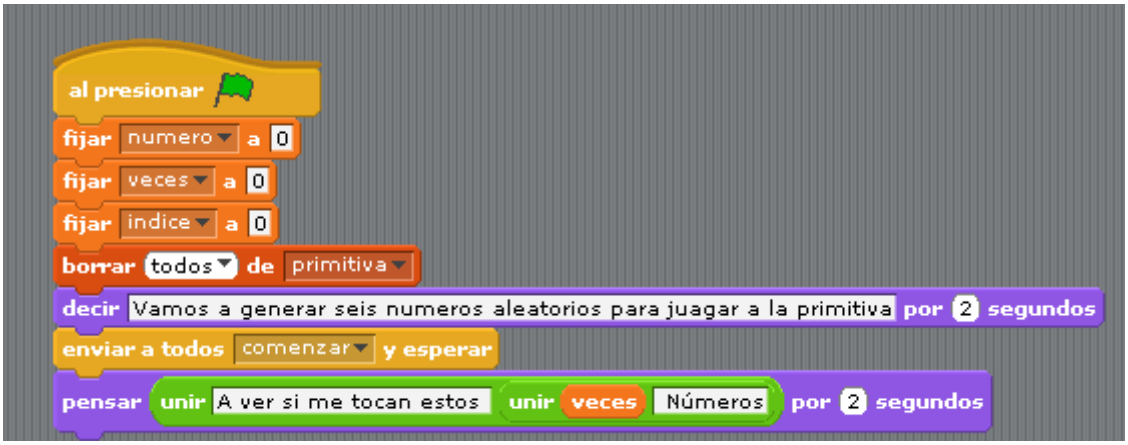
La solución pasa ahora por repetir que genere el numero aleatorio hasta que el número no este en la lista. Esta solución no compromete a otros bloques con llamadas ni a si mismo.



De paso hemos modificado *decir* no solo para que indique que hay un numero repetido sino ademas que nos diga el número concreto que se repitió. Para ello se ha empleado un bloque *unir* que sirve para unir varias frases.

El bloque comenzar no lo modificamos.

En el bloque principal se ha modificado la ultima frase, de forma similar, para despedirse de otra forma. El resto del bloque es el mismo.



Esta forma de trabajo de ir refinando versiones y suponiendo casos concretos (el mejor caso:numero no repetido y el peor caso:numero repetido), como aquí eran los números repetidos, es el procedimiento normal de comprobar si el programa que hemos realizado es correcto o no.

Comprueba su funcionamiento y observa que se generan de forma correcta. Cambia la generacion de numeros aleatorios, para establecer la original de **1 a 49** y no de 1 a 7.

Comprueba de nuevo el programa.