APRENDIENDO A PROGRAMAR CON PHOGO

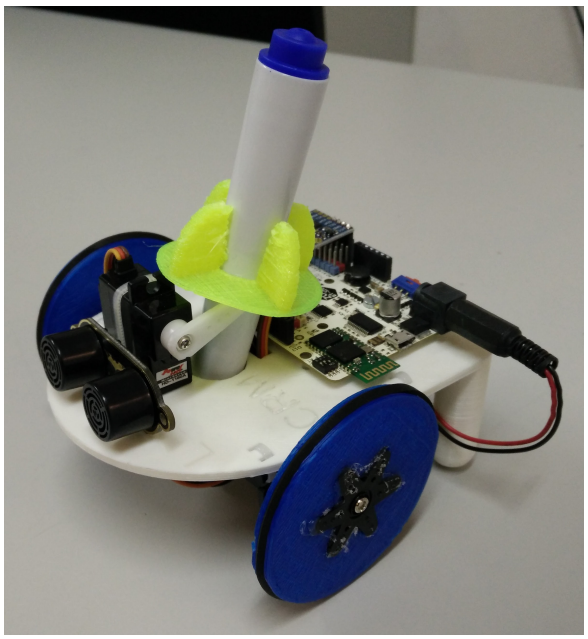
¿Alguna vez te has preguntado cómo es posible que Google sepa dónde encontrar cualquier cosa que busques en Internet? ¿Te has preguntado cómo funciona Minecraft? ¿Cómo sabe tu lavadora qué hacer para dejar tu ropa limpia? Son tres situaciones muy distintas pero las tres tienen en común la programación. Para que las tres funcionen, alguien ha programado un ordenador.

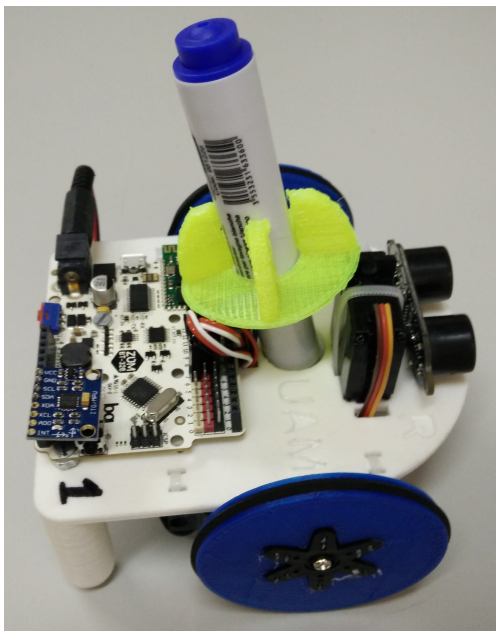
Programar es lo que hacemos los informáticos para dar instrucciones a un ordenador. Le decimos al ordenador qué tiene que hacer. Y el ordenador, si se lo hemos explicado bien, hace exactamente lo que le hemos dicho. Ahora vamos a aprender cómo hablarle al ordenador para que entienda.

Los ordenadores son como las personas en muchos aspectos. Por ejemplo, solo nos harán caso si hablamos un lenguaje que entiendan. Si a ti te hablan en chino, no harás caso, porque no entiendes chino. Si a un ordenador le hablas en español, no te entiende. Por eso nos toca aprender un nuevo lenguaje para hablar con el ordenador.

El lenguaje que vamos a aprender se llama Python. Se parece un poquito al inglés, pero solo un poco. Para que sea más divertido, al ordenador le hemos dado una tortuga. La tortuga es tan obediente como el ordenador. ¡Si sabemos cómo darle órdenes!

Aquí van dos foto de la tortuga. La tuya puede que tenga un color distinto, pero es igual en todo lo demás:





CÓMO HABLAR CON EL ORDENADOR

Para hablar con el ordenador vamos a usar un programa que se llama Idle. Ya deberías tenerlo abierto en tu ordenador. Si no está abierto, o se cerrara en algún momento, pide ayuda a un monitor para volver a abrirlo.

El programa tiene dos ventanas:

1. A la derecha, la **ventana de instrucciones**, donde tú escribirás lo que quieras decirle al ordenador.
2. A la izquierda, la **ventana de resultados**, donde el ordenador te dirá qué va haciendo.

¿Has visto que en la **ventana de instrucciones**, arriba del todo, aparece `from crm-uam import *`? A esa frase la llamamos el **saludo**. Es importante que aparezca en todos los programas que escribas que quieran hablar con la tortuga, porque es la manera que tenemos de decirle al ordenador que queremos hablar con la tortuga. Si no lo ponemos y le damos órdenes para la tortuga, el ordenador no nos entenderá y nos dirá que no nos entiende la **ventana de resultados**.

Recuerda que el ordenador habla un idioma raro. Si aparece algo en la ventana de resultados que no entiendes y la tortuga no hace lo que quieres, pregunta a un monitor, que sabe cómo entender al ordenador.

Hemos aprendido ya muchas cosas:

- Que en el programa hay dos ventas:
  1. A la derecha, la **venta de instrucciones**, donde le contamos al ordenador qué tiene que hacer.
  2. A la izquierda, la **venta de resultados**, donde el ordenador nos cuenta qué tal le va cumpliendo nuestras órdenes.
- La tortuga necesita un saludo cada vez: `from crm-uam import *`
- Si tenemos algún problema, podemos pedir ayuda a los monitores, en cualquier momento

APRENDIENDO A MOVERSE

Vamos a empezar con algo fácil. Vamos a hacer que la tortuga se mueva. Prueba a escribir **`forward()`** en la **ventana de instrucciones**, debajo del **saludo**. **`forward()`** en inglés significa "hacia delante" (así que la tortuga debería moverse hacia delante). Cuando lo hayas escrito, deberías tener algo así en tu **ventana de instrucciones**:

```
from crm-uam import *

forward()
```

Una vez hemos terminado de escribir tenemos que decirle al ordenador que ya hemos terminado y queremos que empiece a hacer lo que hemos puesto. Para eso, pulsa F5 (una tecla que está arriba en tu teclado). Deberías ver en la **ventana de resultados** una tortuga dibujada y al ordenador diciéndole a la tortuga que tiene que ir hacia delante (FD en el lenguaje de la tortuga). Cuando la tortuga termine de andar, le dirá que todo ha ido bien. Le dirá OK al ordenador.

¿Se ha movido la tortuga? Si no lo ha hecho, pide ayuda a alguno de los monitores. Si lo ha hecho… ¡has escrito tu primer programa!

La tortuga, como toda tortuga que se precie, no sólo puede moverse hacia delante. Borra **`forward()`** y prueba a escribir **`back()`**. Deberías tener algo así en tu **ventana de instrucciones**:

```
from crm-uam import *

back()
```

Fíjate que seguimos poniendo el **saludo**. Es importante que lo pongamos cada vez. ¿Qué ha hecho la tortuga? ¿Y qué pasa cuando borras **`back()`** y escribes **`right()`** o **`left()`**? ¡Pruébalo! Y recuerda pulsar F5 cada vez que quieras que el ordenador haga lo que hayas escrito en la **ventana de instrucciones**.

Si la tortuga solo pudiera girar 90º grados sería una tortuga mala. Como la nuestra no es una tortuga mala, puede girar tantos grados como quieras. ¿Cómo? Prueba a escribir en tu **ventana de instrucciones** lo siguientes:

```
from crm-uam import *

right(45)

left(180)

right(270)
```

Y fíjate en lo que pasa…

forward() y **back()** también pueden escribirse con un número dentro. Prueba a que tu **ventana de instrucciones** tenga el siguiente código:

```
from crm-uam import *

forward(5)

back(20)

forward(15)
```

Y observa qué ocurre…

Si has llegado hasta aquí, ahora sabrás cómo mover a la tortuga:

- Si escribes **forward()** avanzará.
- Si escribes **back()**, retrocederá.
- Si escribes **right()** girará a la derecha
- Con **left()** girará a la izquierda.

También sabrás que puedes escribir las ordenes con más información:

- Si escribes **right(grados)** la tortuga girará hacia la derecha tantos grados como hayas dicho.
- Si no pones nada, girará 90º.
- Lo mismo con **left()** (pero girando a la izquierda).
- **foward(número)** avanzan tantos centímetros como le hayas dicho.
- Si no escribes un número avanzará 10 cm.
- Igual para **back()** (solo que yendo hacia atrás).

APRENDIENDO A DIBUJAR

Que la tortuga se mueva es divertido, pero más divertido es que pinte. Si te fijas en tu tortuga, tiene un rotulador. Podemos pedirle a la tortuga que utilice el rotulador para ir pintando allí por donde vaya. Vamos a probarlo.

Asegúrate de que tu tortuga está encima de un papel. Escribe, justo después del saludo, **`pen_down()`** y luego algo que haga que la tortuga se mueva. **`pen_down()`** significa "rotulador abajo" y es la manera de decirle a la tortuga que vaya pintando.

Con esto, ¿sabrías cómo dibujar un cuadrado? ¡Inténtalo!

Puede que en algunos dibujos, quieras que la tortuga se mueva por algún sitio sin dejar rastro. Que deje de pintar. Para eso escribe **`pen_up()`** que significa "rotulador arriba". Al subirlo deja de apoyarse en el papel, y ya no se ve.

¿Podrías dibujar un cuadrado, pero ahora con líneas discontinuas? Algo así:

Con esto puedes dibujar muchas cosas distintas. ¡A ver qué se te ocurre! Dibuja todas ellas. Mira que ha hecho el resto, y si hay algo te gusta, ¡intenta dibujarlo tú!

La tortuga ya no solo se mueve, ¡ahora también dibuja!

- Si escribes **`pen_down()`** empezará a dibujar.
- Si escribes **`pen_up()`** dejará de hacerlo.

Acuérdate de poner la tortuga sobre papel, si va a pintar.

APRENDIENDO A REPETIR

Nota: Esta parte puede ser un poco complicada. Léela solo si quieres. Pregunta si tienes dudas.

¿Te acuerdas del cuadrado que has hecho al principio? Debes haber escrito en la **ventana de instrucciones** algo parecido a esto:

```
from crm-uam import *

forward()

right()

forward()

right()

forward()

right()

forward()

right()
```

Te has fijado en que las líneas se repiten, ¿verdad? Hay una manera de escribir esto mismo, pero de una manera más corta. Le podemos decir al ordenador que él repita cosas por nosotros.

Existe una palabra para ello: **`while`**. En inglés signfica “mientras” y la vamos a usar para decirle al ordenador que siga haciendo algo mientras algo ocurra. Vamos a ver mejor un ejemplo:

```
from crm-uam import *

veces = 1

while veces <= 4:

    forward()

    rigth()

    veces = veces + 1
```

Como ves, es más corto que lo de antes ¡y también hace un cuadrado! El único problema que tiene es que todavía no lo entendemos… ¡pero por poco tiempo! Pruébalo. Para lograr que las últimas líneas salgan desplazadas a la derecha, utiliza la tecla del tabulador, que ese esa con una flecha a la derecha y a la izquierda que está a la izquierda del teclado, encima de la de mayúsculas.

La línea que dice `veces = 1` le está diciendo al ordenador que recuerde que cuando aparezca la palabra `veces` él tiene que pensar en un 1. Así, cuando lee `while veces <= 4` el ordenador leerá `while 1 <= 4`, es decir, "mientras 1 sea menor o igual que 4". Como el 1 es menor que el 4, ejecutará lo que hay dentro del while, es decir, las líneas que están debajo que están metidas a la derecha. Avanzará, girará y le dirá al ordenador que `veces` ahora será `veces + 1`, es decir, `1 + 1`, es decir, ahora le decimos que `veces` es `2`.

Cuando termina, el ordenador vuelve al `while` para mirar si todavía sigue siendo verdad. `while veces <= 4` es "mientras veces sea menor o igual a 4" que como `veces` es `2`, sigue siendo verdad. De nuevo vuelve a avanzar, girar y `veces` ahora vale `3`. `3` es menor que `4`, así que otra vez: avanzar, girar y ahora veces es `4`. Vuelve a comprobar y como veces vale `4` todavía lo hace una vez más: avanza, girar y ahora `veces` es `5`. Cuando vuelve al `while`, ¡esta vez ya no es verdad! Como `5` no es mayor o igual que `4`, ya no hace más lo que aparece dentro.

Piensa en otras cosas que has escrito antes, y cómo podrías utilizar el `while` para ahórrate escribir cosas.

Si queremos repetir algo, no hace falta que lo escribamos varias veces:

- Si escribimos `while condición` la tortuga hará lo que pone dentro mientras la condición sea verdad.
- Para que funcione, tenemos que escribir `:` después de la condición, y todo lo que queremos que haga debajo, poniendo un tabulador delante de cada línea.

Pero el ordenador tiene una manera más chula de repetir. Con lo que hemos escrito antes, hemos dibujado un cuadrado. Y sería muy útil poder enseñarle a la tortuga que si le pedimos un cuadrado, queremos que haga eso, ¿no? ¡Pues se puede!

¿Qué haces tú cuando no conoces una palabra que es nueva? Le pides a alguien que te la defina, o miras su definición en un diccionario. Utilizas palabras que ya sabes para aprender palabras nuevas. Pues la tortuga funciona igual.

Vamos a definirle qué es un cuadrado. Para ello, usamos la palabra **`def`**, que es una manera corta de decir "definition", que en inglés significa "definición".

Si ponemos **`def nombre():`** la tortuga mirará que hay debajo y se aprenderá que cuando le escribas **`nombre()`** quiere que hagas lo que pusiste antes debajo. Siguiendo con el ejemplo del cuadrado, si queremos que la tortuga aprenda lo que es un cuadrado, se lo definimos:

```
from crm-uam import *

def cuadrado():

    veces = 1

    while veces <= 4:

        forward()

        rigth()

        veces = veces + 1
```

Prueba a escribir esto y mira qué pasa… ¿Pasa algo cuando le das a F5? Pues no. Porque hemos definido que es un cuadrado, ¡pero no lo hemos utilizado! Escribe al final del todo **`cuadrado()`**…

Es importante que sepas que la tortuga tiene muy mala memoria y olvida las definiciones cada vez. Si quieres usar una definición, tienes que recordársela, debajo del **saludo** y antes de utilizarla. Si no lo haces, no sabrá de qué estás hablando… Si borras todo para hacer algo nuevo, recuerda no borrar las definiciones, para que todavía te sirvan.

La tortuga, como nosotros, puede aprender palabras nuevas si se las definimos:

- Si escribimos **`def nombre()`** la tortuga recordará que cuando pones nombre, quieres que haga lo que pone debajo.
- Para que funcione, tenemos que escribir **`:`** después del nombre y lo que queremos que haga debajo, poniendo un tabulador delante de cada línea.

APRENDIENDO A USAR LOS OJOS

Nota: Esta parte puede ser un poco complicada. Léela solo si quieres. Pregunta si tienes dudas.

Si te has fijado, la tortuga tiene unos ojos en la parte de delante. No son unos ojos como los nuestros, pero le sirven para ver si hay algo delante de ella y así poder esquivarlo.

Si escribimos **`obstacle()`** en la ventana de instrucciones, la tortuga nos dirá a qué distancia se encuentra el objeto que esté delante de ella. Pon la tortuga delante de una obstáculo y prueba esto:

```
from crm-uam import *

obstacle()
```

En la **ventana de resultados**, verás que sale un número. Son los centímetros a los que está el obstáculo. Prueba a separar la tortuga del obstáculo y repite. ¿Qué le pasa al número? ¿Cambia o se queda igual?

Aunque es útil que la tortuga nos diga a qué distancia están los obstáculos, sería más útil que le pudiéramos decir a la tortuga que hiciera cosas distintas si hay un obstáculo y si no lo hay. Por ejemplo, si no hay un obstáculo, que avance, y si hay un obstáculo, que dé la vuelta y se aleje del obstáculo.

Para eso podemos escribir lo que se llaman condiciones. Una condición es cuando le decimos a la tortuga que haga una cosa u otra dependiendo de si algo ocurre. Si te fijas, hemos usado mucho la palabras “si” cuando lo explicábamos. En inglés, “si” se traduce por **`if`** y ¡es exactamente lo que necesitamos!

Vamos a decirle a la tortuga que avance, pero solo si no hay ningún obstáculo. Si hay un obstáculo, que se dé la vuelta y luego avance. Escribe esto en tu **ventana de instrucciones**:

```
from crm-uam import *

if obstacle() < 10:

    right(180)

forward()
```

Cuando decimos que **`if obstacle() < 10:`** estamos diciendo que "si hay un obstáculo a menos de 10 cm" haga lo que pone justo dentro (para eso ponemos los dos puntos, para que mire lo que hay dentro). ¿Y qué es dentro? Fíjate que la siguiente orden, **`right(180)`**, no empieza a la misma altura que las demás, sino que tiene espacios delante. Eso es porque está dentro del **`if`**. Eso significa que sólo deberá girar 180º si hay un obstáculo a menos de 10 cm. La tortuga después avanzará, haya o no haya delante un obstáculo. Si lo hay habrá girado antes de hacerlo. Pruébalo, poniendo la tortuga delante de una pared y con espacio delante, para ver qué pasa.

Resulta que la tortuga tiene ojos. Y los puede usar para ver si hay algo delante:

- Si escribimos **`obstacle(),`** la tortuga nos dirá a cuántos centímetros hay un obstáculo delante.

Además, también podemos pedirle que tome decisiones:

- Si escribimos **`if condición`** la tortuga hará lo que pone dentro solo si la condición es verdad.
- Para que funcione, tenemos que escribir **`:`** después de la condición, y todo lo que queremos que haga debajo separado con cuatro espacios.

AHORA, A EXPLORAR

Ya sabes todo lo que puedes pedirle a la tortuga que haga, ¡e incluso cómo lograr que aprenda cosas nuevas! Ahora, te toca ponerlo a prueba, y hacer todo lo que te apetezca. Dibujar una estrella, enseñar a la tortuga a salir de un laberinto, escribir tu nombre… ¡las opciones son infinitas y ya eres un programador!