Poder Tortuga

01—Poder Tortuga

Introducción:
Este trimestre vamos a aprender un lenguaje de programación llamado Python. La persona que creó Python lo llamó en honor a su programa de televisión favorito: El Circo Ambulante de los Monty Python. Python es usado por un montón de programadores para cosas muy diferentes.

Python es un lenguaje muy poderoso e inteligente que se utiliza en YouTube, NASA, CERN y muchos otros sitios. Si tu club tiene una Raspberry Pi, puedes usar Python para programarla. Mucha gente adora Python por que es fácil de leer (al menos comparado con otros lenguajes de programación). Poder leer código es una habiliad muy importante, tan importante como poder escribirlo.

Paso 0: Abrir el Editor de Python

Si ya tienes Python en tu ordenador, es hora de empezar.

- [ ] En Windows, busca IDLE en el menu Inicio.
- [ ] En Mac OS X, abre la Terminal.app y escribe “idle” y dale a la tecla enter.
- [ ] En Linux, abre una consola o Terminal, y escribe “idle” y dale a la tecla enter.

Si no tienes Python instalado, no pasa nada! Puedes descargarte la última versión de Python desde http://www.python.org/. La versión exacta no importa mucho, pero nosotros vamos a usar la versión 3, así que la versión debería empezar por 3 (no 2).

Cuando IDLE aranca verás una ventana de salida llamada “Python Shell”. Necesitamos abrir una nueva ventana para escribir código dentro. Ves a “Archivo -> Nueva Ventana” y ya estarás listo para el Paso 1. Asegúrate de que ambas ventanas están visibles.

Paso 1: Hola, Mundo!

- Abre IDLE, el editor que viene con Python. Todo nuestro código se escribirá con este editor. Cuando lo abras, verás una ventana donde aparecerán los errores y los resultados.
- En caso que no lo hayas hecho ya, ves a “Archivo -> Nueva Ventana”. Una ventana vacía aparecerá, con el título de ‘*Sin Título*’ en la barra de título.
  Ahora deberías tener dos ventanas abiertas, una para escribir el programa y otra para mostrar el resultado. Asegúrate de escribir en la ventana correcta!
- Escribe el siguiente código en la nueva ventana:

```
print("Hola, Mundo!")
```

- Asegúrate que no estás en la ventana de la Shell de Python, y guarda tu código.

Lo puedes hacer seleccionando Archivo -> Guardar. Cuando te pidan un nombre de archivo, escribe `hola.py`, y guarda el archivo en tu escritorio. Luego selecciona Ejecutar -> Ejecutar Módulo.

Felicidades, ya has escrito tu primer programa de Python :) (PD: Puedes hacer que imprima por pantalla cualquier cosa que quieras, ¿por qué no lo cambias para que te diga hola a ti? Cámbialo para que diga tu nombre)

Súper Pista:

- [ ] En Windows y en Ubuntu, utiliza Ctrl-N para crear una nueva ventana en tu terminal, utiliza ctrl-S para guardar tu archivo una vez hayas terminado de editarlo, y pulsa F5 para ejecutar tu programa. En algunos ordenadores a lo mejor has de pulsar también la tecla de función Fn.
- [ ] En Mac OS X, pulsa cmd-N para crear una nueva ventana en tu terminal, utiliza ctrl-S para guardar tu archivo y pulsa la tecla función (fn) y la tecla F5 a la vez para ejecutar tu programa.

Paso 2: Hola, Tortuga!

Lo siguiente que vamos a hacer es divertirnos un poco con las tortugas. Una tortuga es un robot muy chiquitito que dibuja en tu pantalla. Podemos hacer que se mueva usando comandos de Python.

- Abre una nueva ventana en tu terminal (Desde el menú Archivo) y escribe lo siguiente (forward significa ade:

```
from turtle import *

forward(100)
```

- Guarda tu programa como mitortuga.py y elije la opción Ejecutar -> Ejecutar Módulo. ¿Ves cómo la tortuga se mueve 100 píxeles hacia adelante? La tortuga tiene pegado un lápiz y dibuja líneas al moverse alrededor de la pantalla.
  Súper Pista: Cuando guardes programas de Python en un archivo siempre deberías darle un nombre que acabe en '.py'.
- Vamos a hacer que la tortuga se mueva por la pantalla! Puedes usar `backward(distancia)` para que la tortuga vaya hacia atrás, para que gire puedes usar `right(ángulo)` y `left(ángulo)`. Por ejemplo `backward(20)` le dice a la tortuga que se mueva hacia atrás 20 píxeles, y `right(90)` le dice la tortuga que gire hacia la derecha 90

grados. Si le das más de una instrucción, éstas se ejecutarán en orden.

```
from turtle import *

speed(11)
shape("turtle")

forward(100)
right(120)
forward(100)
left(90)
backward(100)
left(90)
forward(50)
```

Ángulos y Grados

Juega un poco dibujando tus propias formas utilizando las instrucciones `forward` (adelante), `backward` (atrás), `left` (izquierda), `right` (derecha). Recuerda, con `forward` y `backward` haces que la tortuga se se desplace un número de pixeles, pero con `left` y `right` giras ángulos. Vamos a ver cómo una tortuga gira a la derecha.

```
      Norte
        0
        |
Oeste   |   Este
 270 -----+----- 90
        |
        |
       180
      Sur
```

Cuando la tortuga está mirando al Norte, girar 90 grados a la derecha

hace que mire al Este, girando 180 grados hace que mire al Sur, y girando 270 grados hace que mire al Oeste. Si giras 360 grados, acabas donde empezaste.

Y si giras a la izquierda?

```
      Norte
        0
        |
Oeste   |   Este
 90 -----+----- 270
        |
        |
       180
      Sur
```

Cuando la tortuga está mirando hacia el Norte, girar 90 grados a la izquierda hace que se ponga mirando al Oeste. Girando 180 hace que se ponga mirando al Sur y girando 270 grados hace que se ponga mirando al Este. Si giras la tortuga 360 grados, acabarás en la misma posición que habías empezado. Una vuelta completa son 360 grados.

¿Qué hace el código al principio de nuestro programa?

- `from turtle import *` le dice a Python que queremos usar la libreria turtle, una colección de código que podemos usar para dibujar en nuestra pantalla. Usar una librería nos puede ahorrar mucho tiempo.
- `speed()` asigna la velocidad de la tortuga, coge un valor entre 1 y 11. 11 es el más rápido y 1 el más lento.
- `shape()` Estamos utilizando la forma “turtle” (tortuga), pero tambien podemos usar valores como “arrow” (flecha), “circle” (círculo), “square” (cuadrado), “triangle” (triángulo) o “classic” (clásico).

Vamos a poner estas tres instruciones al principio de todos nuestros programas durante esta lección. Si quieres, puedes cambiar la forma de la tortuga o hacerla ir tan rápido o despacio como quieras.

Paso 3: Dibujando Formas!

Vamos a hacer un cuadrado diciéndole a la tortuga que se mueva siguiendo la forma de un cuadrado.

- Abre un nuevo archivo en IDLE, y escribe el siguiente código:

```
from turtle import *

speed(11)
shape("turtle")

forward(100)
right(90)
forward(100)
right(90)
forward(100)
right(90)
forward(100)
right(90)
```

Guarda tu programa y elige Ejecutar -> Nuevo Módulo. ¿Ves un cuadrado? La tortuga gira 90 grados cuatro veces, y acaba mirando en la misma dirección que había empezado. Girar 90, 90, 90 y 90 hace que la tortuga gire 360 grados en total (4x9 = 36)
¿Y si queremos hacer un triángulo? Un triángulo tiene tres esquinas, así que tenemos que girar tres veces. Si queremos acabar mirando en la misma dirección, necesitaremos que la tortuga gire un total de 360 grados, como en el caso del cuadrado. Así que giraremos 120 grados, luego 120 grados más y

una última vez 120 grados. (120 + 120 + 120 = 360)

- Para dibujar un triángulo, edita tu código para que sea como lo que ponemos a continuación:

```
from turtle import *

speed(11)
shape("turtle")

forward(100)
right(120)
forward(100)
right(120)
forward(100)
right(120)
```

- Ejecuta tu código. ¿Tienes un triángulo?

Elije un color

¿Cual es tu color favorito? puedes cambiar el color de las líneas usando la función `pencolor` (Python utiliza inglés americano), y tambien puedes cambiar el tamaño de la línea usando `pensize`:

- Edita el código de antes para que sea como el del siguiente ejemplo:

```
from turtle import *

speed(11)
shape("turtle")

pensize(10)
pencolor("red")
forward(100)
right(120)
pencolor("blue")
forward(100)
```

```
right(120)
pencolor("green")
forward(100)
right(120)
```

- Ejecuta tu codigo, ¿qué se dibuja en la pantalla?
  Este código dibuja un triángulo de líneas anchas en tres colores diferentes.
- Intenta cambia los colores en tu código, ejecútalo y mira a ver qué sucede.
  La tortuga conoce cientos de colores diferentes, no sólo rojo, azul y verde. Intenta usar tu color favorito!
  Puedes especificar los colores en hex (hexadecimal) tal y como hicimos en CSS. En lugar de usar `pencolor("red")` podrías usar `pencolor("#FF0000")`.
  ¿Qué color es el #FF4F00?

Paso 4: Repítete a tí mismo (con un bucle)

El último programa usaba las mismas instrucciones una y otra vez. En lugar de escribirlos, vamos a pedirle al ordenador que los repita por nosotros. Cuando estudiabas Scratch aprendiste el concepto de *iteración* cuando usaste los bloques `Para Siempre` y `Repitir` / `Repitir Hasta Que`. En Python los bucles for se usan cuando tienes un trozo de codigo que quieres repetir un número de veces. En este caso, queremos repetir el código (indentado, que está metido hacia adentro) 4 veces (por que un cuadrado tiene 4 lados).

- Abre un nuevo archivo y escribe lo siguiente:

```
from turtle import *

speed(11)
shape("turtle")
```

```
for count in range(4):
    forward(100)
    right(90)
```

- Guarda el programa y ejecuta el programa: Ejecutar -> Ejecutar Módulo.
  Fíjate que el programa está *indentado*, o desplazado a la derecha dentro del bucle. Python utiliza espacios en blanco para saber qué instrucciones hay que repetir. Puedes utilizar el tabulador (tecla Tab) para indentar tu código.
- Vamos a ver qué ocurre si sólo indentamos `forward`, para ello, edita tu programa para que diga:

```
from turtle import *

speed(11)
shape("turtle")

for count in range(4):
    forward(100)
right(90)
```

- Fíjate que `forward` está indentado y `right` no lo está, ¿verdad? ¿Qué crees que hace el programa? prueba de ejecutarlo para descubrirlo.
  ¿Te ha salido una línea recta? Python repetirá la instrucción `forward` cuatro veces y *luego* girará a la derecha. Python utiliza espacios en blanco para agrupar instrucciones, de la misma manera que en Scratch teníamos los bloques. ¿Te acuerdas? Python se quejará si el código no está debidamente tabulado.
- Vamos a dejar el programa como lo teníamos al principio y que nos dibuje un cuadrado, pero en lugar de usar números en el código vamos a asignar variables. Nos ayudará a ver qué hace realmente el programa, y además nos ayudará a que no nos repitamos una y otra vez.

Cambia el programa para que contenga lo siguiente:

```
from turtle import *

speed(11)
shape("turtle")

lados = 4
distancia = 100
angulo = 90
for count in range(lados):
    forward(distancia)
    right(angulo)
```

- Guarda el programa y ejecuta el programa: Ejecutar -> Ejecutar Módulo.

Reto: Dibujar otras formas

¿Puedes dibujar las siguientes formas cambiando sólo los valores?

- [ ] ¿un triángulo? (tres lados)
- [ ] ¿un pentágono? (five lados)
- [ ] ¿un hexágono? (six lados)
- [ ] ¿un octágono? (eight lados)

Recuerda, un triángulo tiene tres lados, y tiene giros de 120 grados. Si giras 120 grados en cada esquina al final habrás girado 360 grados en total. Para un cuadrado gira 90 grados en las 4 esquinas, que también suman 360 grados.

Si giras seis veces, ¿cuantos grados deberás girar para que sumen 360? Prueba algunos números y mira a ver qué pasa.

Paso 5: Gira, Gira, Gira

En lugar de ser nosotros los que calculamos el ángulo, ¿por qué no dejas que sea el ordenador el que lo haga? Python te permite hacer operaciones con números como sumar, restar, multiplicar y dividir.

Podriamos reescribir `lados = 4 + 1` en lugar de 5, o `lados = 4 - 1` en lugar de 3. Para la multiplicación Python usa `*`, y para la división usamos la barra `/`.

Si necesitamos girar 360 grados en total, podemos calcular el ángulo que necesitaremos. Para un cuadrado, `360 / 4` es 90, para un triángulo, `360/3` es 120.

- Cambia tu prorama para calcular el ángulo de la siguiente manera.

```
from turtle import *

speed(11)
shape("turtle")

lados = 4
distancia = 20

angulo = 360/lados
for count in range(lados):
    forward(distancia)
    right(angulo)
```

- Ahora cambia el número de lados. Comprueba que Python lo haga bien. Intenta dibujar un hexágono (6 lados), o cualquier número de lados que quieras!

Paso 6: Forma Sólida

- Podemos pedirle a la tortuga que rellene las formas con un color usando las instrucciones `begin_fill()` y `end_fill()`. Cambia tu código y añade éstas instrucciones dentro:

```
from turtle import *

speed(11)
shape("turtle")

lados = 4
distancia = 20

fillcolor('red')
pencolor('red')
begin_fill()

angulo = 360/lados
for count in range(lados):
    forward(distancia)
    right(angulo)
end_fill()
```

  De la misma forma que hacíamos con `pencolor`, con `fillcolor` puedes darle un color a la tortuga para que rellene la forma que dibujes. Vamos a dibujar un cuadrado rojo con una línea de color rojo también.
  Utilizarás `begin_fill()` para decirle a la tortuga que vas a dibujar una figura que quieres que rellene y end_fill()` para decirle que puede dejar de rellenar con color.
- Intenta cambiar los colores, los lados, las longitudes y mira a ver cuantas formas puedes dibujar!

Paso 7: Levantas el Lápiz, Bajas el Lápiz

Si quieres mover la tortuga sin dejar ningún rastro detrás puedes usar la instrucción `penup()` y `pendown()` para hacer que la tortuga dibuje o no.

- En un nuevo archivo, intenta lo siguiente:

```
from turtle import *

speed(11)
shape("turtle")

pencolor('red')

for count in range(20):
    penup()
    forward(10)
    pendown()
    forward(20)
```

- Este código debería dibujar una línea de guiones a lo largo de la pantalla. Ejecútalo y verás!!

Home, origen en la pantalla.

Un último truco. La instrucción `home()` devuelve la tortuga a su posición inicial. Por contra, la instrucción `clear()` borra la pantalla completamente, y la instruccion `reset()` devuelve la tortuga al inicio y limpia la pantalla!

Paso 8: A Tope!

Puedes ir adelante `forward()`, atrás `backward()`, izquierda `left()`, y derecha `right()`, puedes iterar usando el bucle `for count in range(4)`, cambiar colores, cambiar la velocidad incluso cambiar las formas!

¿Puedes dibujar una casa?, ¿un gato?, ¿un pájaro?, ¿una serpiente?, ¿un perro?, ¿un león? Combina diferentes formas para ver qué otras muchas cosas puedes dibujar. ¿Puedes dibujar un robot?