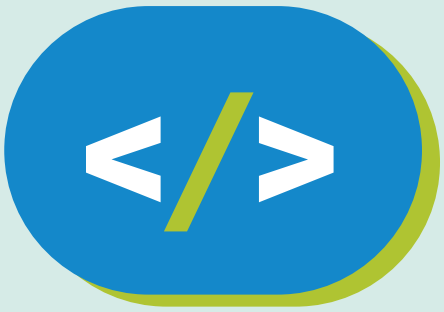


Kit de programación
Código Pi

¿Dónde está el tesoro?

APRENDER CONECTADOS

Ministerio de Educación,
Cultura, Ciencia y Tecnología
Presidencia de la Nación

Autoridades

Presidente de la Nación
Mauricio Macri

Jefe de Gabinete de Ministros
Marcos Peña

Ministro de Educación, Cultura, Ciencia y Tecnología
Alejandro Finocchiaro

Secretario de Gobierno de Cultura
Pablo Avelluto

Secretario de Gobierno de Ciencia, Tecnología e Innovación Productiva
Lino Barañao

Titular de la Unidad de Coordinación General del Ministerio de Educación, Cultura, Ciencia y Tecnología
Manuel Vidal

Secretaria de Innovación y Calidad Educativa
Mercedes Miguel

Subsecretario de Coordinación Administrativa
Javier Mezzamico

Directora Nacional de Innovación Educativa
María Florencia Ripani

ISBN en trámite

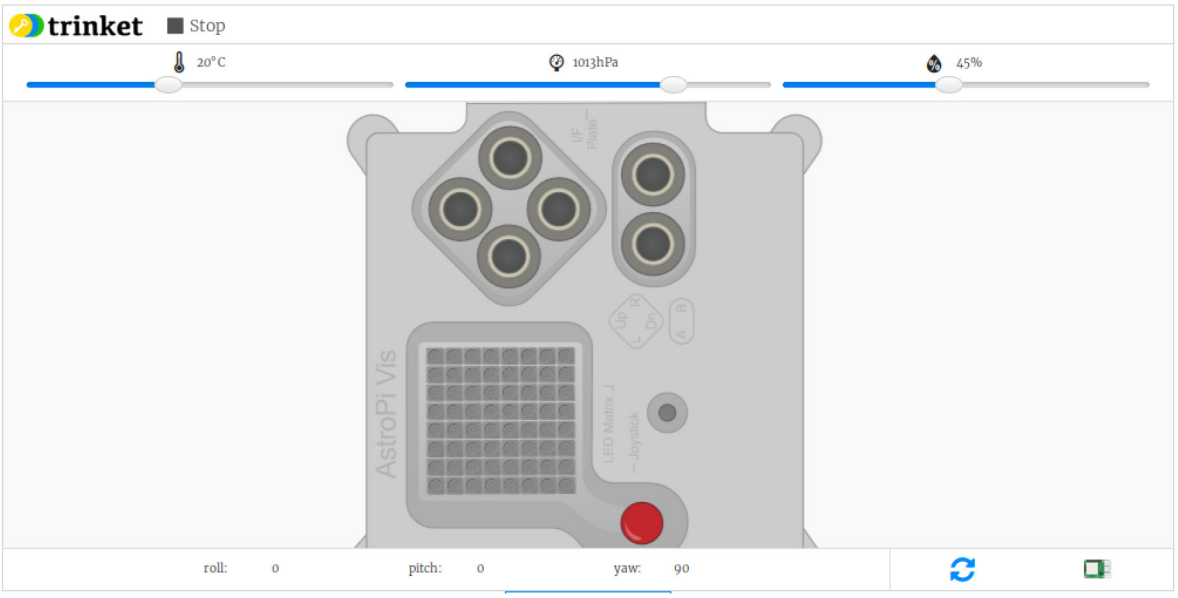
Índice

¿Dónde está el tesoro?

Introducción

En este proyecto usaremos el joystick y la pantalla LED del Sense HAT para jugar a un juego de memoria. El Sense HAT mostrará una moneda de oro, debés recordar dónde estaba y usar el joystick para encontrar el tesoro oculto.

Escribiremos código en el lenguaje de programación Python, que ya aprendimos en el módulo de Python.



Para jugar al juego presioná Run y mirá dónde aparece el punto amarillo - ¡Ese es nuestro tesoro! Luego, usá las flechas en el teclado para mover el punto blanco a donde creés que está escondido el tesoro. Cuando llegues ahí, pulsá return. Deberías ver un punto verde si has acertado, o un punto rojo si has errado. Tenés 10 intentos para encontrar el tesoro, y un puntaje máximo de 10.

Cuando usás el emulador de Sense HAT debes usar las teclas de tu teclado, pero en tu Sense HAT usarás el joystick.

Paso 1: Escondiendo el tesoro

Primero, vamos a mostrar una moneda amarilla en un pixel al azar, y luego esconderla.

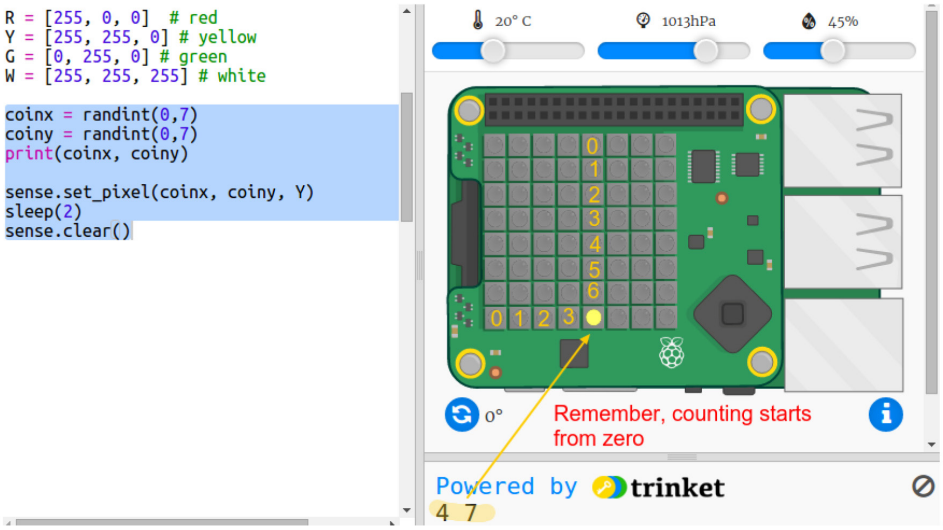
Actividad

• Abrí el Trinket de ejemplo de Where's the Treasure?: jumpto.cc/treasure-go.

• Mirá el código que ha sido incluido. Este configura el Sense HAT y las librerías necesarias, y además incluye algo de código de ayuda para que puedas llegar a la parte interesante más rápido:

```
#!/bin/python3

from sense_hat import *
from time import sleep
from random import randint

sense = SenseHat()
sense.clear()

# Just return the actions we are interested in
def wait_for_move():
  while True:
    e = sense.stick.wait_for_event()
    if e.action != ACTION_RELEASED:
      return e

R = [255, 0, 0]  # red
Y = [255, 255, 0] # yellow
G = [0, 255, 0] # green
W = [255, 255, 255] # white
```

• Vamos a mostrar una moneda amarilla en un lugar aleatorio para luego esconderla. Las variables `coinx` y `coiny` son las coordenadas x e y de la moneda. Andá al fondo del programa y agregá el siguiente código:



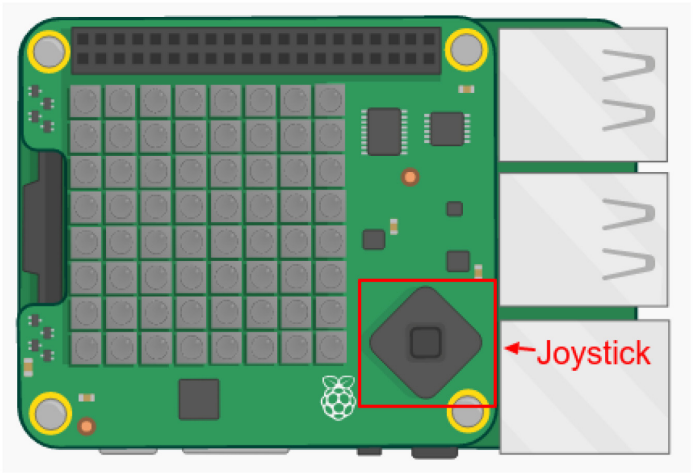
Asegurate de usar una letra Y mayúscula.

• Ejecutá el código varias veces para corroborar que la moneda aparece y desaparece en ubicaciones al azar.

Paso 2: Buscando el tesoro

Ahora vamos a mostrar al jugador como un pixel blanco. Debés usar el joystick de tu Sense HAT para navegar adonde crees que está el tesoro.

El Sense HAT tiene un mini joystick. Podemos ver una imagen de él en el emulador:



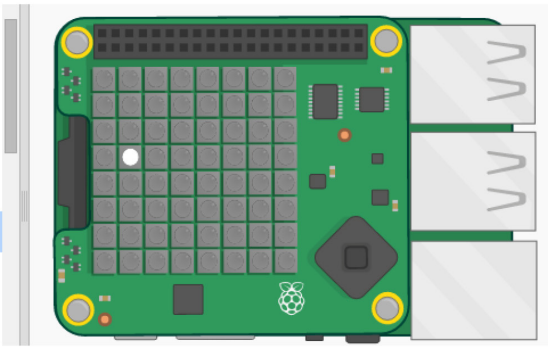
En el emulador podés usar las flechas del teclado como las direcciones del joystick, y Enter (Return) para la pulsación del botón.

Actividad

Ahora vamos a agregar un pixel que el jugador puede mover para buscar el tesoro. El jugador es un pixel blanco.

• Mostramos la ubicación del jugador con un pixel blanco:

```
W = [255, 255, 255] # white

coinx = randint(0,7)
coiny = randint(0,7)
print(coinx, coiny)

sense.set_pixel(coinx, coiny, Y)
sleep(2)
sense.clear()

x = randint(0,7)
y = randint(0,7)
sense.set_pixel(x, y, W)
```



`x` e `y` son las coordenadas del jugador.

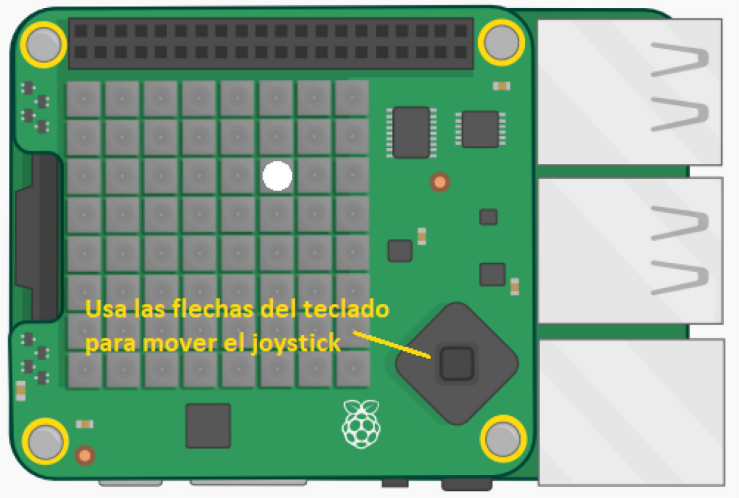
• Vamos a mover el pixel blanco usando el joystick. Cada vez que el jugador presiona una dirección en el joystick debemos borrar el pixel actual y dibujar uno en la nueva ubicación. Empecemos por permitir al jugador moverse en la dirección y (arriba y abajo):

```
sense.set_pixel(x, y, W)

while True:
    e = wait_for_move()
    sense.clear()

    if e.direction ==  DIRECTION_UP:
        y = y - 1
    elif e.direction ==  DIRECTION_DOWN:
        y = y + 1
    sense.set_pixel(x, y, W)
```

espera a que se mueva el joystick

dibuja al jugador en una nueva ubicación

• Probá tu programa pulsando las flechas arriba y abajo del teclado.



• ¿Qué sucede cuando llegamos al borde superior y pulsamos arriba?

```
    sense.set_pixel(x, y, W)
```

Powered by trinket

```
2 3
```

ValueError: Y position must be between 0 and 7 on line 49 in main.py

• Si la posición en y es menor a 0 o mayor a 7, recibiremos un error cuando intentemos encender el pixel.

• Agreguemos una comprobación que nos asegure que el pixel sigue en la pantalla:

```
if e.direction ==  DIRECTION_UP and y > 0:
  y = y - 1

elif e.direction ==  DIRECTION_DOWN and y < 7:
  y = y + 1
```

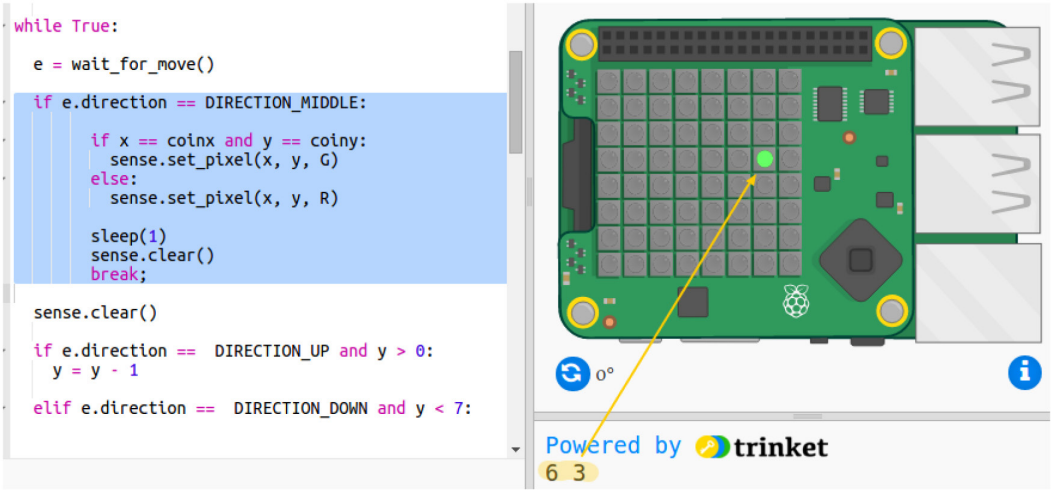
• Ahora agreguemos movimiento en la dirección x. Agrega el código resaltado:

```
    if e.direction ==  DIRECTION_UP and y > 0:
      y = y - 1

    elif e.direction ==  DIRECTION_DOWN and y < 7:
      y = y + 1

    elif e.direction ==  DIRECTION_LEFT and x > 0:
      x = x - 1

    elif e.direction ==  DIRECTION_RIGHT and x < 7:
      x = x + 1

    sense.set_pixel(x, y, W)
```

• Una vez que te encuentres en la posición donde crees que está el tesoro debés pulsar el botón del joystick. En el emulador debés pulsar la tecla Enter (Return)

Si el jugador se encuentra en la misma posición que el tesoro, entonces lo ha encontrado. El pixel debe iluminarse verde por un segundo.

Si el jugador eligió una ubicación errónea entonces el pixel debe iluminarse rojo por un segundo.



`break` quiere decir que no debemos esperar más eventos luego de que el jugador eligió una ubicación, así que podemos detener el bucle.

Paso 3: Contando el puntaje

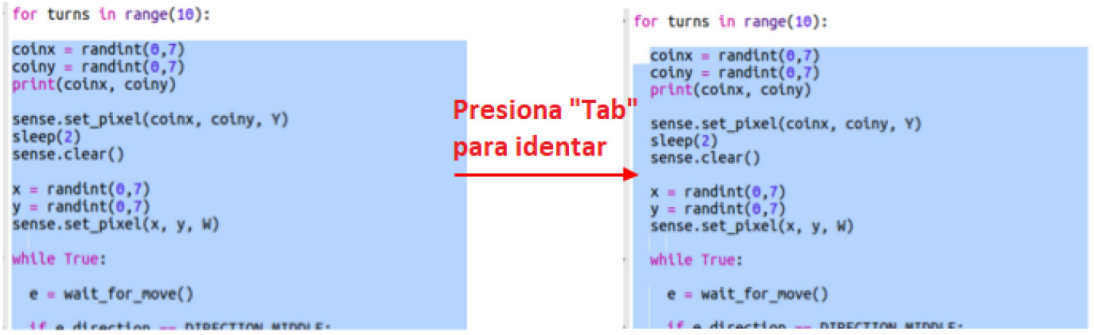
Por el momento sólo tenés un intento para encontrar el tesoro. Agreguemos 10 intentos y mantengamos un puntaje.

Actividad

• Ahora necesitás un bucle `for` para permitir que el jugador tenga 10 intentos para encontrar el tesoro:

```
R = [255, 0, 0]  # red
Y = [255, 255, 0] # yellow
G = [0, 255, 0] # green
W = [255, 255, 255] # white

for turns in range(10):

coinx = randint(0,7)
coiny = randint(0,7)
print(coinx, coiny)
```

• En Python, el código debe estar indentado para estar dentro de un bucle. ¡No necesitás hacerlo una líne a la vez! Seleccioná todo el código después del bucle `for`, luego pulsá la tecla tab para que todo el texto sea indentado automáticamente:



Asegurate que todo el código luego del `for` esté indentado, hasta el final del programa.

• Agregá una variable de puntaje que empiece en cero:

```
R = [255, 0, 0]  # red
Y = [255, 255, 0] # yellow
G = [0, 255, 0] # green
W = [255, 255, 255] # white

score = 0

for turns in range(10):
```

• Luego debés agregar uno al puntaje cuando el jugado selecciona la ubicación correcta:

```
        if x == coinx and y == coiny:
            sense.set_pixel(x, y, G)
            score += 1
        else:
            sense.set_pixel(x, y, R)

        sleep(1)
        sense.clear()
```

• Finalmente, mostrá el puntaje al final:

```
    elif e.direction ==  DIRECTION_RIGHT and x < 7:
        x = x + 1

    sense.set_pixel(x, y, W)

sense.show_message("Score: " + str(score))
```

Sin indentar

Asegurate que la última línea no tenga indentación, ya que debe ejecutarse luego de las 10 iteraciónes del bucle `for` (es decir, luego de que el juego haya terminado).

• Ahora jugá el juego. ¿Podés hacer 10 puntos de 10?

Desafío: Personalizá el juego

Intentá usar otros colores, o mostrar un mensaje distinto de acuerdo al puntaje.
Desafío: Hacelo más difícil

¿Encontrás el juego muy fácil? ¿Por qué no hacerlo más difícil?

Podés mostrar la moneda por menos tiempo. `sleep(2)` muestra la moneda dos segundos. ¿Y qué tal `sleep(0.5)`?

¿O qué tal confundir al jugador, mostrando todos los pixeles en amarillo antes de elegir la ubicación? Podés usar `sense.clear(Y)` para llenar la pantalla con monedas amarillas luego de mostrar al jugador dónde está escondida. También necesitarás usar `sleep(1)`, o los segundos que quieras mostrar la pantalla amarilla.