Hojas de trabajo de EdPy

Hojas de trabajo para estudiantes y hojas de actividades.

Nombre_____

Contenido

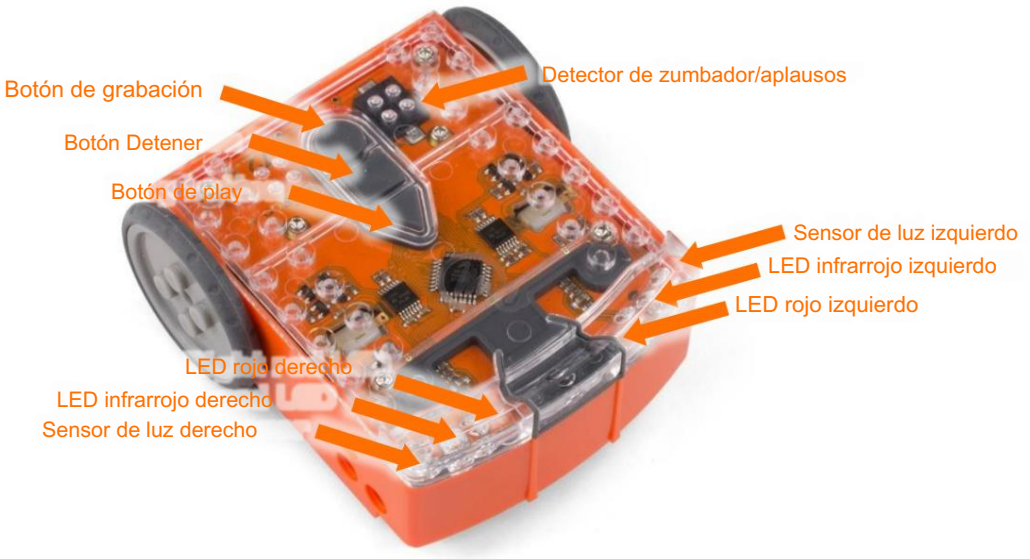
Nombre_____

Nombre_____

Lección 1: Hoja de trabajo 1.1 – Conoce a Edison

Edison es un pequeño robot programable. Edison usa sensores y motores para interactuar con el mundo. También puedes usar Edison con ladrillos LEGO para construir todo tipo de cosas.

Mire las imágenes a continuación para familiarizarse con los sensores, botones e interruptores de Edison.



Esta es la parte superior de Edison.

Botón de reproducción (triángulo) - Ejecutar programa
Botón de parada (cuadrado) – Detener programa
Botón de grabación (redondo) : 1 pulsación = descargar un programa, 3 pulsaciones = escanear código de barras

Esta es la parte inferior de Edison.

El sensor de seguimiento de línea de Edison se compone de dos partes: una luz LED roja y una luz sensor.

El sensor de seguimiento de línea también puede leer códigos de barras especiales que activan programas preestablecidos.

También utilizará el cable EdComm para descargar sus programas a Edison. Se conecta a la toma de auriculares de su computadora.

Este es el cable de programación EdComm.

Nombre_____

Lección 1: Hoja de trabajo 1.2: Programación de códigos de barras

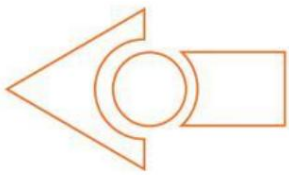
Edison viene con programas precargados que se activan al conducir sobre códigos de barras especiales.

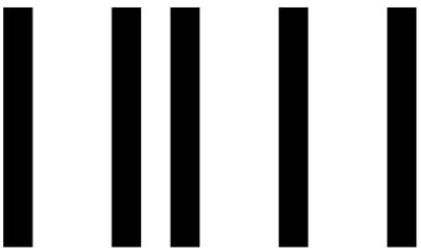
Lectura del código de barras

1. Coloque a Edison frente al código de barras en el lado derecho, arriba de la flecha 2. Presione el botón de grabación (redondo) tres veces 3. Edison avanzará y escaneará el código de barras 4. Presione el botón de reproducción (triángulo) para ejecutar el programa

Código de barras: conducción controlada por aplausos

Código de barras: evita obstáculos





Código de barras: seguimiento de línea

Código de barras: siga la antorcha

Su turno:

para cada código de barras, describa qué acciones realiza el robot y por qué lo hace.

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

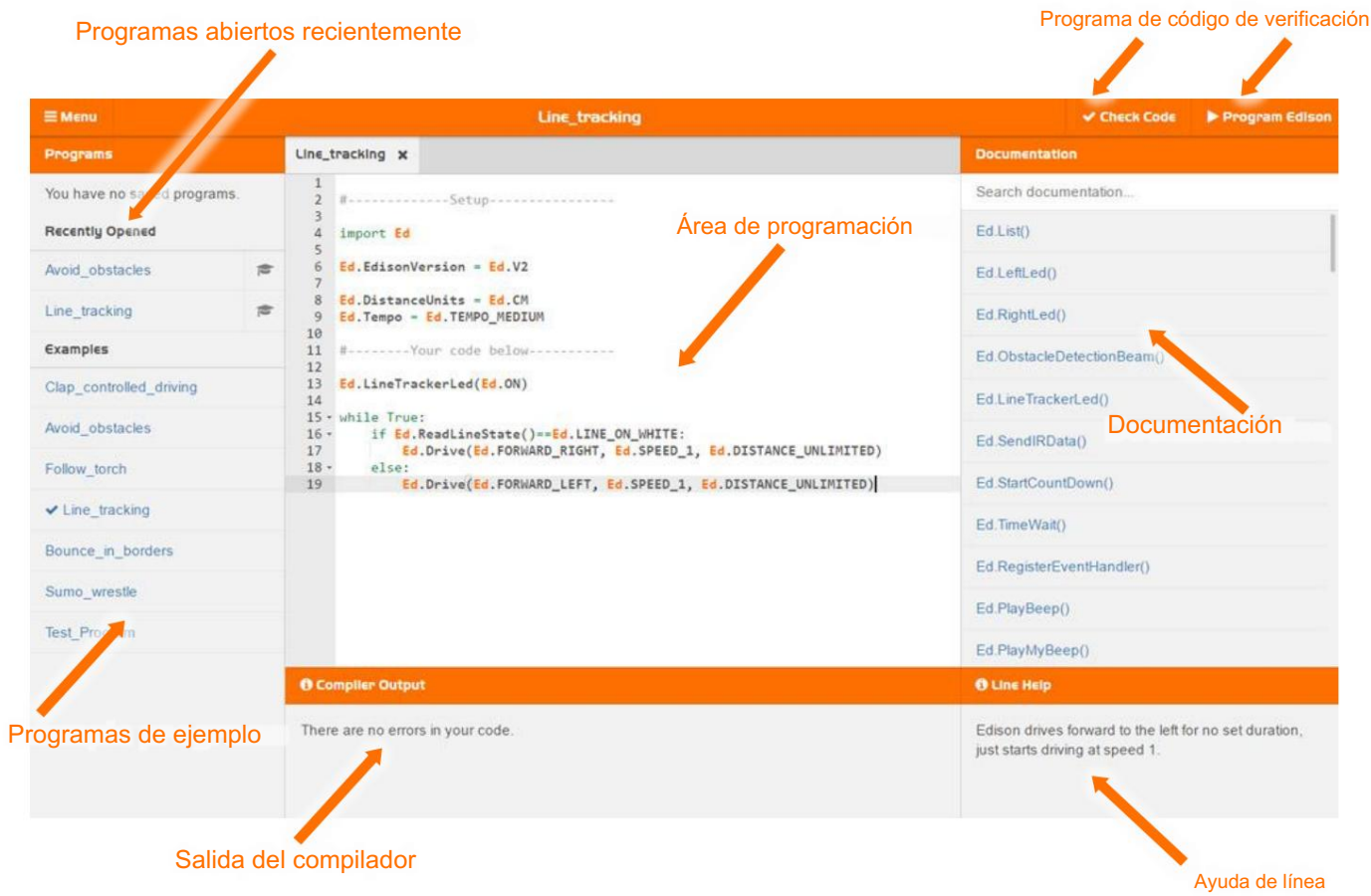
_____

_____

Nombre_____

Lección 1: Hoja de trabajo 1.3 – Conozca la aplicación EdPy En esta actividad, conocerá la aplicación EdPy, el software que usaremos para programar el robot Edison.

Para familiarizarse con la aplicación y la programación de EdPy, intente abrir algunos programas de ejemplo. Investigue cómo funcionan algunas de las funciones buscando en la ventana 'Documentación'. Todo lo que necesita saber sobre los comandos de la aplicación EdPy se puede encontrar en la sección de documentación. Además, intente usar la 'Ayuda de línea' para descubrir más sobre cada



función.

Tu turno:

1. ¿A qué debe cambiar la siguiente línea si está utilizando una versión 1 de Edison?

   Ed. EdisonVersion = Ed.V2 _____

2. ¿Cuántos parámetros de entrada tiene cada uno de los siguientes comandos?

   Ed.PlayBeep() _____

   Ed.TiempoEspera() _____

   Ed.LeftLed() _____

Nombre_____

Ed.DriveRightMotor() _____

3. Puede cambiar el código de configuración para usar pulgadas en lugar de centímetros como su unidad de medida a lo largo del programa. Si hizo esto, ¿cómo debería escribirse esa línea del código de configuración?

_____

_____

4. Si hay errores en su código después de haber hecho clic en el botón 'Comprobar código', ¿en qué ventana aparecerán los errores?

_____

_____

Nombre_____

Lección 1: Hoja de trabajo 1.4 – Descargar un programa de prueba Abra el programa llamado 'Test_Program' desde la ventana 'Ejemplos' en EdPy.

Así es como se ve el programa de prueba:

Test_Program

```
1
2  #-------------Setup----------------
3
4  import Ed
5
6  Ed.EdisonVersion = Ed.V2
7
8  Ed.DistanceUnits = Ed.TIME
9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #--------Your code below-----------
12
13
14
15 while True:
16     Ed.PlayBeep()
17     Ed.LeftLed(Ed.OFF)
18     Ed.RightLed(Ed.ON)
19     Ed.Drive(Ed.SPIN_RIGHT, 5, 350)
20     Ed.TimeWait(20, Ed.TIME_MILLISECONDS)
21     Ed.PlayBeep()
22     Ed.LeftLed(Ed.ON)
23     Ed.RightLed(Ed.OFF)
24     Ed.Drive(Ed.SPIN_LEFT, 5, 350)
25     Ed.TimeWait(20, Ed.TIME_MILLISECONDS)
26
```

Edison mira cada línea del programa una a la vez, luego hace lo que dice la línea.

Sin embargo, hay algunas líneas que Edison se saltará.

Mire la línea 2, que comienza con un carácter '#' (hash). Cuando una línea comienza con este carácter, se denomina 'línea de comentario'. Edison ignorará los caracteres que vienen después del '#' en una línea y pasará a la siguiente línea. En programación, usamos líneas de comentarios para documentar nuestro código para ayudar a realizar un seguimiento de las cosas y para que otras personas puedan entender el programa.

Descargar el programa a Edison Para descargar un programa a Edison, conecte el cable EdComm a la toma de auriculares de la computadora y suba el volumen al máximo. Enchufe el otro extremo del cable EdComm en Edison como se muestra.

Nombre_____

Para descargar el programa de prueba, siga estos pasos:

1. Encienda Edison, luego presione el botón de grabación (redondo) de Edison una vez 2. Conecte Edison a la computadora usando el cable EdComm y confirme el el volumen está al máximo
3. Presione el botón 'Programar Edison' en la esquina superior derecha de la aplicación EdPy 4. Siga los pasos en la ventana emergente, luego presione 'Programar Edison'

Una vez que el programa termine de descargarse, desconecte el cable EdComm. Presione el botón de reproducción (triángulo) una vez para ejecutar el programa.

Tu turno:

1. ¿Qué hizo el robot cuando presionaste el botón de reproducción?

_____

_____

_____

_____

2. Mira los comandos de Python en el programa y piensa en lo que hizo Edison cuando jugaste el programa. Describe cómo se relacionan entre sí.

_____

_____

_____

_____

3. Explique cómo llega el programa de la computadora al robot.

_____

_____

_____

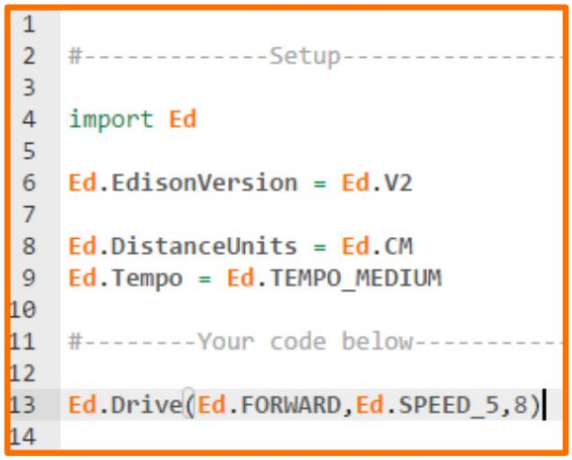
_____

Nombre_____

Lección 2: Hoja de trabajo 2.1 – Conducir el robot hacia adelante

En esta actividad, debe escribir un programa para hacer avanzar su robot Edison.

Mira el siguiente programa:



```
1
2  #------------Setup---------------
3
4  import Ed
5
6  Ed.EdisonVersion = Ed.V2
7
8  Ed.DistanceUnits = Ed.CM
9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #--------Your code below----------
12
13 Ed.Drive(Ed.FORWARD,Ed.SPEED_5,8)
14
```

Paso 1: comience a escribir el programa anterior en la aplicación EdPy escribiendo 'Ed' en la línea 13.

Cuando comience a escribir 'Ed' en la línea 13, verá que aparece un cuadro emergente. El cuadro de aviso muestra una lista de posibles comandos que puede seleccionar. Esta es una característica de la aplicación EdPy llamada 'finalización de línea de comando', y hace que programar sea más rápido.

Paso 2: Escriba 'Ed.Drive(' en la línea 13 y seleccione la función 'Ed.Drive()'.

Ed.Drive() es una función en Python que el código de configuración está importando desde el módulo de la biblioteca Edison 'Ed'.

Una función es una pieza de código que realiza un rol o trabajo en particular, según los parámetros que se ingresen. Todas las funciones que se importan de la biblioteca 'Ed' deben comenzar con 'Ed.' Esto le dice al programa a qué biblioteca debe ir para encontrar esa función.

Paso 3: Complete los parámetros de entrada.

Cuando una función tiene parámetros de entrada, debe ingresar un valor para cada uno.

La función Drive() tiene tres parámetros de entrada:

- dirección : la dirección en la que conducirá Edison
- velocidad : la velocidad a la que conducirá Edison •
  distancia : la cantidad de unidades de distancia que recorrerá Edison

Diferentes parámetros de entrada toman diferentes valores. Por ejemplo, 'velocidad' toma Ed.SPEED_ un número del 1 al 10 (10 es el máximo).

Nombre_____

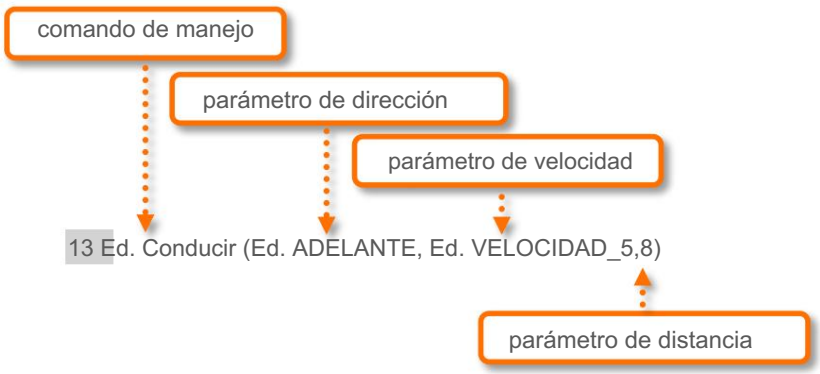
El tipo de unidades de distancia está controlado por la constante en la que se establece 'Ed.DistanceUnits' en el código de configuración. Hay tres unidades de distancia que puede utilizar:

- Centímetros, escrito como Ed.CM •
- Pulgadas, escrito como Ed.INCH •
- Tiempo, escrito como Ed.TIME

Nota: si tiene un robot Edison V1, debe usar el tiempo, así que asegúrese de que Ed.DistanceUnits = Ed.TIME. La unidad de distancia de tiempo es milisegundos, lo que significa que para conducir durante 2 segundos, debe configurar el parámetro de entrada de distancia en 2000 (ya que 2000 milisegundos = 2 segundos).

Paso 4: Revisa tu programa.

Echemos un vistazo más de cerca a la función de accionamiento y los parámetros en este programa.



Una vez que haya escrito el programa completo, haga clic en el botón 'Comprobar código' y mire la ventana 'Resultado del compilador' para asegurarse de que no ha cometido ningún error al escribir.

Los errores tipográficos se denominan "errores de sintaxis". Si escribe una palabra que no forma parte del lenguaje del programa EdPy, también llamado sintaxis, entonces el compilador EdPy no puede entender lo que debe hacer. Esto crea un error de sintaxis.

Si hay algún error en su código, arréglelo para que su código coincida con el ejemplo.

Paso 5: Descarga y prueba tu programa.

Una vez que compruebe que su código no tiene errores, descargue su programa a su Edison.

1. Encienda Edison, luego presione el botón de grabación (redondo) de Edison una vez
2. Conecte Edison a la computadora usando el cable EdComm y confirme el el volumen está al máximo
3. Presione el botón 'Programar Edison' en la esquina superior derecha de la aplicación EdPy 4. Siga los pasos en la ventana emergente, luego presione 'Programar Edison'

Una vez que se descarga el programa, desconecte el cable EdComm. Use la hoja de actividad 2.1 o cinta de color para marcar las líneas de 'inicio' y 'finalización' en un escritorio o en el piso como área de prueba para su

Nombre_____

programa. Presione el botón de reproducción (triángulo) una vez para ejecutar el programa y ver qué sucede.

Paso 6: Experimente con su programa.

Mide la distancia entre tu línea de salida y tu línea de meta. Intenta modificar tu programa para que tu robot Edison termine de conducir justo antes de la línea de meta. Experimente para ver qué funciona.

Tu turno:

1. ¿A qué constante configuró 'Ed.DistanceUnits' en el código de configuración?

   _____

2. ¿Qué número necesitaba ingresar como parámetro de entrada de distancia para hacer Edison conducir desde el principio hasta la línea de meta?

   _____

3. Experimente conduciendo el robot Edison a diferentes velocidades. Lo que hace el robot cuando conduce a Edison a velocidad 10? ¿Notas algún cambio en la precisión de Edison cuando conduce a velocidad 10?

_____

_____

_____

_____

Nombre_____

Lección 2: Hoja de trabajo 2.2 – Conducir el robot hacia atrás

En esta actividad, debe escribir un programa para hacer retroceder su robot Edison.

Cada vez que escribe un programa para Edison en EdPy, siempre sigue los mismos pasos básicos:

- Paso 1: Verifique que el código de configuración esté usando las constantes que desea.
- Paso 2: Escriba el programa, seleccione las funciones que desea utilizar y complete el parámetros de entrada con los valores que desee.
- Paso 3: Verifique que no haya errores en su programa usando el botón 'Verificar código'.
- Paso 4: Descargue y pruebe su programa usando su robot Edison.

Recuerde, si tiene un robot Edison V1, asegúrese de que Ed.DistanceUnits = Ed.TIME. El parámetro de distancia para Ed.TIME está en milisegundos.

Tu turno:

Tarea 1: conducir hacia atrás

Escribe el siguiente programa:

```
1
2  #-------------Setup----------------
3
4  import Ed
5
6  Ed.EdisonVersion = Ed.V2
7
8  Ed.DistanceUnits = Ed.CM
9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #--------Your code below-----------
12
13 Ed.Drive(Ed.BACKWARD,Ed.SPEED_5,8)
14
```

Use la hoja de actividad 2.1 o cinta de color para marcar las líneas de 'inicio' y 'finalización' en un escritorio o en el piso como área de prueba para su programa. Experimente para ver si puede hacer que su Edison conduzca hacia atrás desde el principio hasta la meta.

Tarea 2: Usa la constante 'Ed.DISTANCE_UNLIMITED'

Hay varias formas de programar su Edison para que avance y retroceda.
Otra forma es usar 'Ed.DISTANCE_UNLIMITED' para el parámetro de distancia. Esta constante enciende los dos motores impulsores de Edison.

A diferencia de cuando usa un valor numérico para el parámetro de distancia, Ed.DISTANCE_UNLIMITED no especifica un valor exacto después del cual los motores se detendrán. En cambio, enciende los motores y luego pasa a la siguiente línea de código. Se necesitará una parada de los motores más adelante en el código.

Nombre_____

El uso de Ed.DISTANCE_UNLIMITED puede ser útil cuando desea escribir un programa en el que los motores solo se detengan una vez que ocurra algún otro evento, por ejemplo, cuando se detecte un obstáculo.

Mira el siguiente programa:

```
1
2  #-------------Setup----------------
3
4  import Ed
5
6  Ed.EdisonVersion = Ed.V2
7
8  Ed.DistanceUnits = Ed.CM
9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #--------Your code below-----------
12
13 Ed.Drive(Ed.BACKWARD, Ed.Ed.SPEED_6, Ed.DISTANCE_UNLIMITED)
14 Ed.TimeWait(200, Ed.TIME_MILLISECONDS)
15 Ed.Drive(Ed.STOP, Ed.SPEED_10, 0)
16
```

Este programa enciende los motores Edison para conducir hacia atrás, luego espera 200 milisegundos antes de apagar los motores.

Escriba un programa utilizando el parámetro Ed.DISTANCE_UNLIMITED para conducir hacia atrás. Use la hoja de actividad 2.1 o cinta de color para marcar las líneas de 'inicio' y 'finalización' en un escritorio o en el piso como área de prueba para su programa. Experimente con su programa para ver si puede hacer que su Edison conduzca hacia atrás desde la meta hasta la línea de salida.

1. Establezca la velocidad en su programa en Ed.SPEED_6. ¿Cuántos milisegundos ¿Necesita ingresar a la función TimeWait() para hacer que Edison conduzca hacia atrás desde el final hasta la línea de inicio?

   _____

2. Experimente con diferentes parámetros de entrada de velocidad y TimeWait(). ¿Cuáles son los los tiempos más rápidos y más lentos que puede hacer que Edison conduzca hacia atrás desde la meta hasta la línea de salida?

   Lo más rápido: _____ El más lento: _____

Nombre_____

Lección 2: Hoja de trabajo 2.3: Adelante, luego atrás En esta actividad, debe escribir un programa para impulsar su robot Edison hacia adelante y luego hacia atrás.

Recuerde seguir los cuatro pasos básicos de programación para Edison usando EdPy:

- Paso 1: Verifique que el código de configuración esté usando las constantes que desea. • Paso 2: Escriba el programa, seleccione las funciones que desea usar y complete los parámetros de entrada con los valores que desea.
- Paso 3: Verifique que no haya errores en su programa usando el botón 'Verificar código'. • Paso 4: Descargue y pruebe su programa usando su robot Edison.

Recuerde, si tiene un robot Edison V1, asegúrese de que Ed.DistanceUnits = Ed.TIME. El parámetro de distancia para Ed.TIME está en milisegundos.

Tu turno:

Tarea 1: Escriba el siguiente programa:

```
1
2  #-------------Setup----------------
3
4  import Ed
5
6  Ed.EdisonVersion = Ed.V2
7
8  Ed.DistanceUnits = Ed.CM
9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #--------Your code below-----------
12
13 Ed.Drive(Ed.FORWARD,Ed.SPEED_5,8)
14 Ed.Drive(Ed.BACKWARD,Ed.SPEED_5,8)
15
```

Use la hoja de actividad 2.1 o cinta de color para marcar las líneas de 'inicio' y 'finalización' en un escritorio o en el piso como área de prueba para su programa.

1. ¿Cuáles son los valores correctos de los parámetros de entrada de distancia para que Edison conduzca hacia adelante, luego hacia atrás entre la salida y la meta?

   Adelante _____ Hacia atrás _____

Tarea 2: escribir un nuevo programa que haga que Edison conduzca hacia adelante y luego hacia atrás entre las líneas de salida y llegada. Esta vez, use el parámetro Ed.DISTANCE_UNLIMITED y las funciones Ed.TimeWait(). Use la Hoja de actividad 2.1 o cinta de color para marcar las líneas de 'inicio' y 'finalización' en un escritorio o en el piso como área de prueba para su programa.

Nombre_____

2. ¿Cómo es su nuevo programa? Escribe tu código abajo.

Nombre_____

Lección 2: Hoja de trabajo 2.4 – Expresiones en Python En esta actividad, aprenderá sobre un elemento importante del código que usamos cuando programamos en Python: las expresiones.

¿Qué son las expresiones?

Una expresión es una pregunta que se puede resolver como 'verdadera' o 'falsa'. Por ejemplo: '¿Es A menor que B?' o '¿Es A lo mismo que B?'

En código, las expresiones se escriben usando notaciones matemáticas en lugar de palabras.

En el código de configuración, ha visto que se usa la notación =. Por ejemplo, Ed.DistanceUnits = Ed.CM. Usar la notación 'A = B' significa 'establecer A en el mismo valor que B'.

Las expresiones también usan notaciones. Estas son algunas de las notaciones básicas en expresiones que podemos usar en Python:

Expresión A	Significado
== EL !=	¿A es lo mismo que B?
EL >	¿A no es igual a B?
EL >=	¿Es A mayor que B?
EL < EL	¿Es A mayor o igual que B?
<= B	¿A es menor que B?
	¿A es menor o igual que B?

Las expresiones comparan el lado izquierdo con el lado derecho de la notación en la expresión.

Puede reemplazar la 'A' y la 'B' en la lista de expresiones anterior con cualquier valor o función que devuelva un valor. También puede hacer operaciones matemáticas con esos valores. Por ejemplo, '(A + 2) > B' significa '¿Es A más 2 mayor que B?'

En código, las expresiones funcionan en un orden específico. Cuando su expresión incluye matemáticas a un valor o llama a una función, la expresión resolverá las matemáticas o la función primero. Luego comparará el lado izquierdo de la expresión con el lado derecho y resolverá como 'verdadero' o 'falso'.

¿Para qué se usan las expresiones en Python?

Las expresiones se utilizan junto con otros elementos del código, como los bucles 'while' y las declaraciones 'if', para cambiar el flujo del código. Estos elementos permiten que el código se mueva de manera diferente al flujo secuencial de línea 1 → línea 2 → línea 3.

Nombre_____

Tu turno:

Practica resolver expresiones. Primero, escriba lo que significa cada expresión, luego resuélvalo en verdadero o falso.

Si A = 2 y B = 4, ¿qué significa cada una de las siguientes expresiones y en qué se resuelve cada una (verdadero o falso)?

1. (A*2) == segundo

   Significado: _____

   Resuelve a: _____

2. A >= B

   Significado: _____

   Resuelve a: _____

3. (A+A) != B

   Significado: _____

   Resuelve a: _____

4. (A-1) < (B-3)

   Significado: _____

   Resuelve a: _____

Nombre_____

Lección 2: Hoja de trabajo 2.5 – Conducción activada por teclado En esta actividad, debe escribir un programa para conducir su robot Edison hacia adelante solo cuando se presiona el botón redondo o el botón triangular. Para ello, utilizaremos expresiones y el bucle 'while'.

Mira el siguiente programa:

```
1
2  #-------------Setup----------------
3
4  import Ed
5
6  Ed.EdisonVersion = Ed.V2
7
8  Ed.DistanceUnits = Ed.CM
9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #--------Your code below-----------
12 Ed.ReadKeypad()
13 while Ed.ReadKeypad() == Ed.KEYPAD_NONE:
14     pass
15 Ed.Drive(Ed.FORWARD, Ed.SPEED_6,8)
16
```

Este programa usa un bucle 'while' con una expresión.

En Python, un ciclo 'while' repite una declaración o grupo de declaraciones mientras una condición dada es VERDADERA. Comprueba la condición, que se escribe como una expresión, antes de ejecutar el cuerpo del bucle.

Mientras que la expresión se evalúa como VERDADERO, el programa repite los comandos en el bucle. Cuando la expresión se evalúa como FALSO, el programa pasa a la siguiente línea de código fuera del bucle.

Usar sangría Python
usa sangría para agrupar instrucciones o comandos.

En Python, todas las declaraciones sangradas por el mismo número de espacios de caracteres se consideran un solo bloque de código.

Mira la línea 14 del programa. Debido a que 'pass' está sangrado, está dentro del bucle. Sin embargo, la línea 15 del programa no está sangrada, por lo que la línea 15 no está dentro del bucle.

Funciones y constantes en este programa
Ed.ReadKeyPad() : esta función lee el estado del teclado de Edison. En otras palabras, determina si uno de los botones de Edison ha sido presionado o no. Ed.ReadKeyPad() devuelve un valor que indica qué botón se ha presionado: Ed.KEYPAD_NONE, Ed.KEYPAD_TRIANGLE o Ed.KEYPAD_ROUND.

Nombre_____

Esta función no funciona para el botón cuadrado. Esto se debe a que el botón cuadrado está diseñado solo para detener un programa. El botón cuadrado siempre detendrá la ejecución de cualquier programa cuando se presione.

Nota especial: uso de funciones de 'lectura' dentro de un bucle

Algunos tipos de datos se almacenan temporalmente en la memoria de Edison. Así es como la función Ed.ReadKeyPad() puede leer una pulsación de botón antes de llamar a la función de lectura en su código.

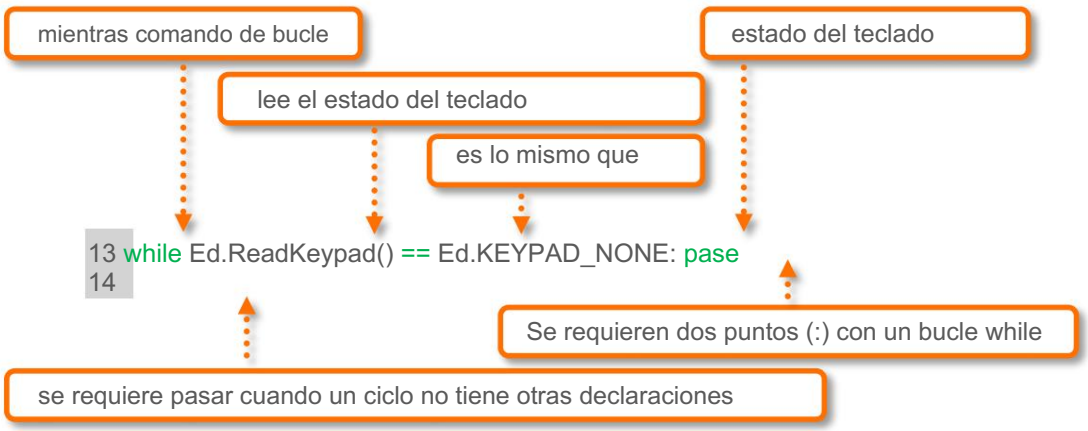
En este programa, queremos asegurarnos de que Ed.ReadKeyPad() en el bucle while esperará hasta que se presione un botón y no considerará ninguna pulsación de botón que suceda antes del bucle while. Es por eso que colocamos Ed.ReadKeyPad() en la línea arriba del bucle while (línea 12). Esto borrará cualquier pulsación de tecla previa antes del ciclo.

Siempre debe seguir este proceso cuando use una función de 'lectura' dentro de un bucle.

Tu turno:

Escribe el programa.

Asegúrate de que cuando escribas el ciclo while, incluyas los dos puntos y que sangres correctamente:



Descargue el programa y presione el botón triangular para iniciar el programa. Espere un poco, luego intente presionar el botón triangular o redondo.

1. ¿Qué hizo el robot cuando presionó el botón triangular o redondo unos segundos después de iniciar el programa?

_____

_____

Nombre_____

2. Vuelva a ejecutar el programa e intente presionar el botón cuadrado en lugar del redondo o botón de triángulo ¿Qué pasó? ¿Por qué pasó esto? Explicar.

_____

_____

_____

_____

3. Ahora intente agregar más código al final del programa. El nuevo código que escriba debe hacer que Edison conduzca hacia atrás después de presionar el botón redondo o triangular nuevamente. En otras palabras, su programa debe decirle a Edison que avance la primera vez que se presiona un botón, y luego retroceda cuando se presiona un botón nuevamente. Recuerde incluir los dos puntos y sangrar su código dentro del ciclo while. ¿Cómo es su nuevo programa? Escribe tu código abajo.

_____

_____

_____

_____

_____

_____

_____

_____

Nombre_____

Lección 2: Hoja de actividades 2.1

LÍNEA DE META

LÍNEA DE SALIDA

Nombre_____

Lección 3: Hoja de trabajo 3.1 – Girar a la derecha En esta actividad,
debe escribir un programa que haga que su robot Edison gire a la derecha.

Mira el siguiente programa:

```
1
2  #-------------Setup----------------
3
4  import Ed
5
6  Ed.EdisonVersion = Ed.V2
7
8  Ed.DistanceUnits = Ed.CM
9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #--------Your code below-----------
12 degreesToTurn = 90
13 Ed.Drive(Ed.SPIN_RIGHT,Ed.SPEED_6,degreesToTurn)
14
```

Mira la línea 13 del programa. Recuerde que la función Ed.Drive tiene tres parámetros de entrada: dirección, velocidad y distancia. En este programa, el parámetro de distancia no es un número sino 'grados para girar'.

Esta es una variable.

En Python, las variables son ubicaciones de memoria reservadas para almacenar valores. Esto significa que cuando crea una variable, reserva algo de espacio en la memoria del programa.

Una variable representa un valor que se establece en algún lugar de su programa.

Mira la línea 12 del programa. Aquí es donde se establece el valor de 'degreesToTurn'.
A esto se le llama asignar un valor a una variable. En Python, el signo igual (=) se usa para asignar valores a las variables.

Puede usar variables para almacenar valores que se usan en varios lugares a lo largo de un programa. Esto puede ser muy útil, especialmente si cambia el valor de una variable. Al usar variables, solo necesita hacer el cambio en una línea del código para ajustar el valor en todas partes donde se usa esa variable en su programa.

Tu turno:

Escriba el programa, luego descárguelo y ejecútelo con su Edison. Use la hoja de actividad 3.1 o cinta de color para marcar las marcas de los ángulos de 'inicio' y 'fin' en un escritorio o en el piso como área de prueba para su programa.

1. Describa lo que hace el robot y por qué lo hace cuando ejecuta el programa.

_____

_____

_____

Nombre_____

Agregue una nueva línea al final de su programa (después de la línea 13) y agregue el siguiente código a tu programa:

```
Ed.Drive(Ed.SPIN_LEFT,Ed.SPEED_6,degreesToTurn)
```

Descargue y ejecute el programa con su Edison en su área de prueba.

2. Describa lo que hace el robot y por qué lo hace cuando ejecuta la actualización. programa.

_____

_____

_____

Ahora edite su programa para que su Edison primero gire 180 grados a la derecha, luego 180 grados a la izquierda.

Sugerencia: recuerde que puede cambiar el valor de la variable gradosParaGirar.

Descargue y ejecute su programa actualizado con su Edison en su área de prueba para ver si su cambio fue exitoso.

3. ¿Qué línea o líneas de su programa cambió? Anote la línea actualizada o líneas.

_____

_____

_____

Los nombres de variables deben seguir ciertas reglas en Python. Por ejemplo, no se permiten caracteres especiales como # o $. Intente cambiar el nombre de la variable 'degreesToTurn'.
Experimente con diferentes nombres posibles y use el botón 'Comprobar código' para encontrar qué nombres están permitidos y cuáles no.

4. Da dos ejemplos de nombres de variables ilegales que hayas descubierto.

_____

_____

Nombre_____

Lección 3: Hoja de trabajo 3.2 – Girar a la izquierda 180°

En esta actividad, debe escribir dos programas diferentes para girar su robot Edison a la izquierda exactamente 180°.

Tu turno:

Tarea 1: girar a la izquierda exactamente 180°

Escriba un programa que haga que su robot Edison gire a la izquierda exactamente 180°.

Sugerencia: Trate de usar el programa que usó en la hoja de trabajo 3.1 como punto de partida.

Descargue su programa y pruébelo usando la hoja de actividades 3.1 o cinta de color para marcar las marcas de ángulo de 'inicio' y 'final' en un escritorio o en el piso como área de prueba para su programa. Recuerde experimentar con su programa. Si su Edison no gira exactamente 180°, intente ajustar sus parámetros de entrada y pruebe de nuevo.

1. ¿Cuáles son los parámetros de entrada que usó para hacer que el robot gire exactamente 180°? Si usted usó una variable, incluya el valor que le asignó a esa variable.

_____

_____

_____

Tarea 2: Gire exactamente 180° usando el comando Ed.DriveRightMotor()

EdPy tiene un comando llamado 'Ed.DriveRightMotor()' que hace que solo se mueva el motor derecho de Edison. Si solo se mueve el motor derecho, ¿hacia dónde girará Edison? Sostenga a Edison en sus manos e imite lo que sucederá si solo se mueve el motor correcto.

Busque el comando Ed.DriveRightMotor() en la ventana de documentación de la aplicación EdPy para ver cómo funciona esta función.

Luego escribe un programa que haga que tu robot gire 180° a la izquierda usando la función Ed.DriveRightMotor().

2. ¿Cuáles son los parámetros de entrada que usó para hacer que el robot gire a la izquierda exactamente 180°? usando el comando Ed.DriveRightMotor()?

_____

_____

_____

Nombre_____

Lección 3: Hoja de trabajo 3.3 – Giro a la derecha, luego giro a la izquierda En esta actividad, debe escribir un programa para que su robot Edison gire cuando se presiona el botón triangular.

Escribe el siguiente programa:

```
1
2  #-------------Setup----------------
3
4  import Ed
5
6  Ed.EdisonVersion = Ed.V2
7
8  Ed.DistanceUnits = Ed.CM
9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #--------Your code below-----------
12 degreesToTurn = 90
13 Ed.ReadKeypad()
14 while Ed.ReadKeypad() != Ed.KEYPAD_TRIANGLE:
15     pass
16 Ed.Drive(Ed.SPIN_RIGHT,Ed.SPEED_6,degreesToTurn)
17
```

Descargue su programa y pruébelo usando la hoja de actividades 3.1 o cinta de color para marcar las marcas de ángulo de 'inicio' y 'final' en un escritorio o en el piso como área de prueba para su programa.

Tu turno:

Escriba un programa para hacer que el robot gire a la derecha exactamente 90° cuando se presiona una vez el botón triangular, luego gire a la izquierda exactamente 270° cuando se presiona el botón triangular por segunda vez.

Recuerde poner Ed.ReadKeyPad() en la línea sobre cada bucle 'while' para borrar cualquier pulsación de tecla previa antes del bucle.

1. ¿Cómo es su programa? Escribe tu código abajo.

_____

_____

_____

_____

_____

_____

Nombre_____

Lección 3: Hoja de trabajo 3.4 – Mini laberinto

En esta actividad, debe escribir un programa que le permita a su robot Edison navegar con éxito a través de un laberinto.

Tu turno:

Escriba un programa para que su robot Edison conduzca a través del mini laberinto en la hoja de actividad 3.2 cuando presione el botón de reproducción (triángulo).

Para completar con éxito el laberinto, debes:

- hacer que Edison comience desde detrás de la línea de 'salida', • hacer que Edison se detenga después de cruzar la línea de 'meta', y • mantener a Edison dentro de las líneas del borde del laberinto.

Use el conocimiento de programación de robots que ha adquirido hasta ahora para escribir un programa que use múltiples funciones para permitir que Edison pase por las vueltas del laberinto.

Sugerencias:

Ed.Drive() Ed.	SPIN_RIGHT Ed.FORWARD Ed.SPIN_LEFT		

1. Describe la secuencia de movimientos que hizo tu robot para completar el laberinto.

_____

_____

_____

2. ¿Qué le resultó difícil al escribir este programa?

_____

_____

_____

_____

_____

Desafío 1: ¡Carrera!

¿Quién puede atravesar el laberinto más rápido, sin hacer trampa?

Recuerde: su robot debe comenzar detrás de la línea de salida, detenerse después de la línea de meta y no debe pasar por encima de las líneas fronterizas para ganar.

Nombre_____

3. ¿Con quién corriste? ¿Quién ganó la carrera?

Competidor: _____

Ganador: _____

4. ¿Cuál fue el tiempo del robot ganador a través del laberinto?

_____

Reto 2: Diseña tu propio laberinto

Diseña tu propio laberinto más desafiante con algunos giros más para que Edison navegue.
Escribe un programa para que Edison complete el laberinto con éxito. O intercambie laberintos con un compañero y escriba un programa para completar su laberinto con éxito.

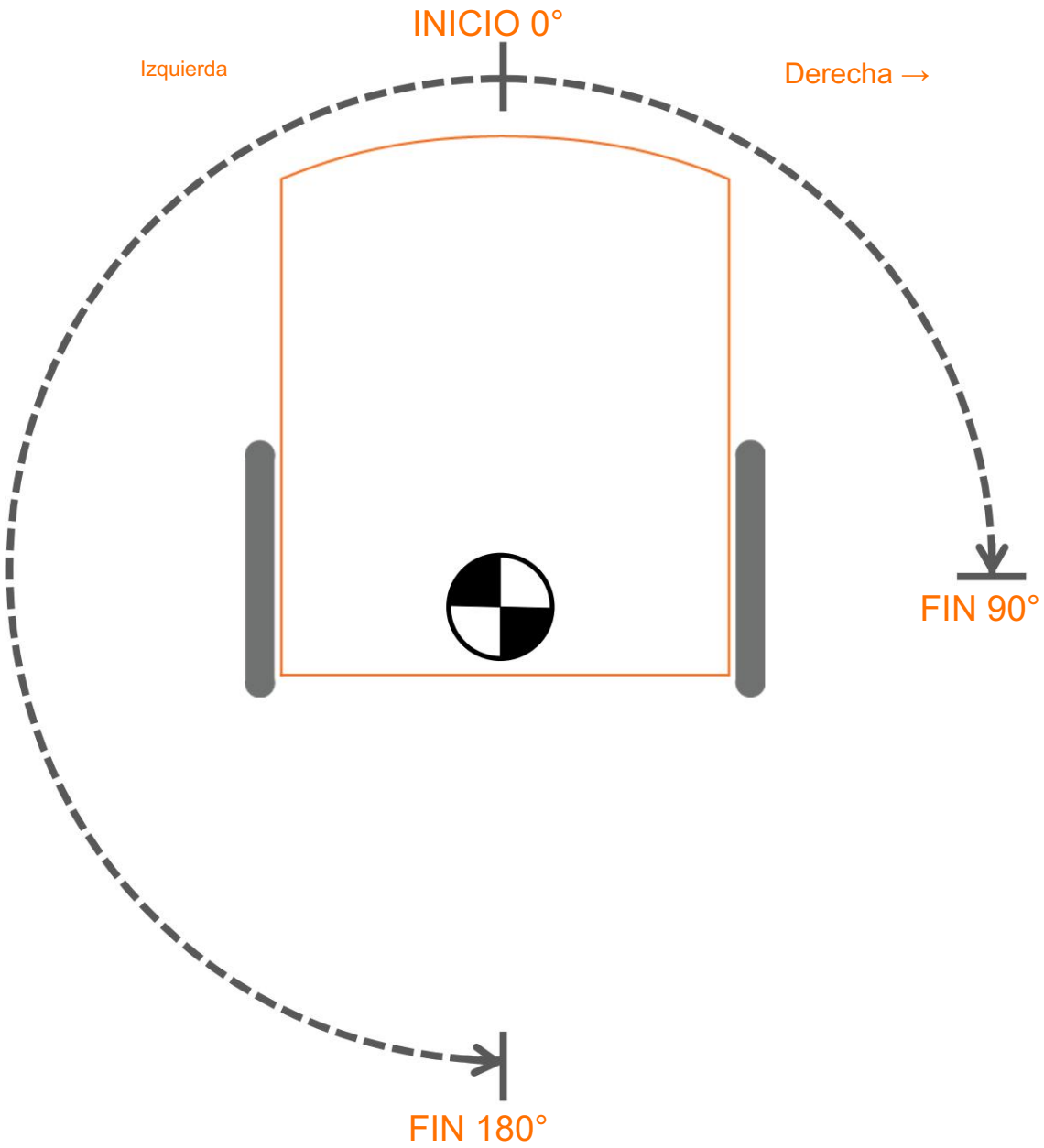
Recuerde: su robot debe comenzar detrás de la línea de salida, detenerse después de la línea de meta y no debe pasar por encima de las líneas fronterizas para ganar.

5. Dibuja una versión pequeña del laberinto que completaste en el cuadro.

Nombre_____

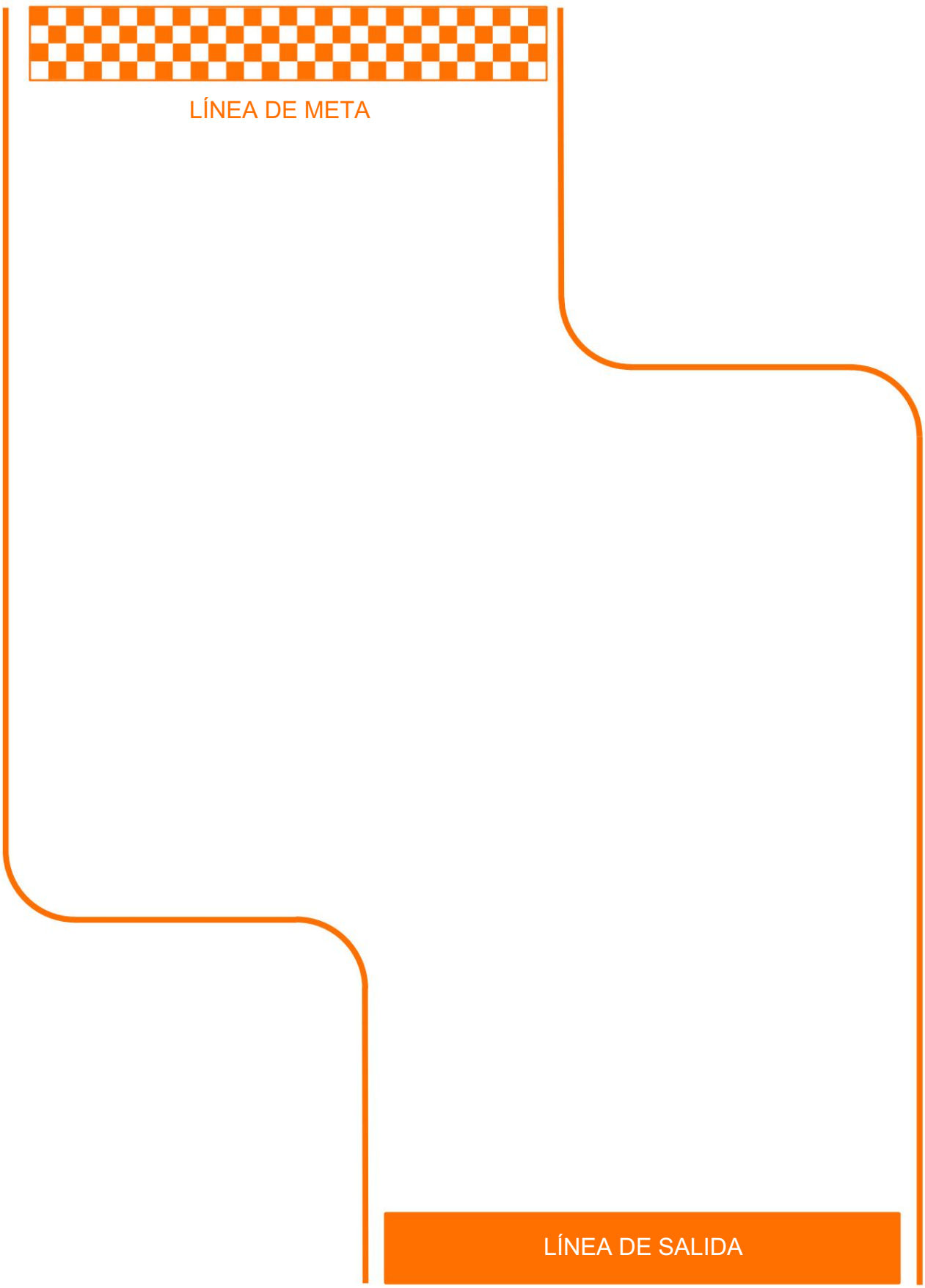
Lección 3: Hoja de actividad 3.1 – Girar Coloque su robot

Edison en el contorno como se muestra. Inicie siempre su robot desde el marcador de inicio (0°).



Nombre_____

Lección 3: Hoja de actividades 3.2 – Mini laberinto



Nombre_____

Lección 4: Hoja de trabajo 4.1 – Conducir en un cuadrado En esta actividad, debe escribir un programa que le permita a su robot Edison conducir en forma de cuadrado.

Tu turno:

Escriba un programa para que cuando su robot Edison conduzca, forme un cuadrado. Utilice los comandos que ya ha aprendido, incluidos Ed.Drive() y una variable. Asegúrese de que su programa termine con su Edison en el mismo lugar donde comenzó.

Descarga tu programa y pruébalo usando la hoja de actividades 4.1, colocando tu Edison en el punto de 'inicio' y siguiendo las líneas. También puede hacer un cuadrado más grande con cinta de color para marcar las líneas y un punto de "inicio" en un escritorio o en el piso.

1. ¿Cómo es su programa? Escribe tu código abajo.

_____

_____

_____

_____

_____

_____

_____

2. ¿Cuántas llamadas de función usó en este programa?

_____

3. ¿Tiene líneas de código duplicadas en su programa? Si es así, ¿cuáles son y cuántas veces usaste cada uno?

_____

_____

_____

_____

Nombre_____

Lección 4: Hoja de trabajo 4.2: use un bucle para conducir en un cuadrado

En esta actividad, debe escribir un programa diferente que le permita a su robot Edison conducir en forma de cuadrado.

Usando la hoja de trabajo 4.1, escribió un programa que usaba los mismos comandos varias veces. Necesitabas usar Ed.Drive() con un parámetro de dirección de Ed.FORWARD cuatro veces, una para cada lado del cuadrado. También necesitaba usar Ed.Drive() con un parámetro de dirección de Ed.SPIN_LEFT cuatro veces para girar cada esquina.

¿Encontraste un poco aburrido escribir los mismos comandos muchas veces?

Repetir los mismos comandos una y otra vez no es un problema para una computadora, pero escribir un programa de esta manera no es muy eficiente. En su lugar, es mejor utilizar una estructura de bucle.

Mira a Mark Zuckerberg, creador de Facebook, explicar el concepto de bucles al programar: https://www.youtube.com/watch?v=hYvcoRkAkOU

¿Quién diría que ser un gran programador podría convertirte en uno de los multimillonarios más jóvenes del mundo? ¡Se estima que el patrimonio neto de Mark Zuckerberg supera los 70.500 millones de dólares estadounidenses!

Podemos escribir un programa para hacer que Edison conduzca en un cuadrado con menos código usando un bucle 'for'. Esto hará que escribir el programa sea más eficiente. Dado que necesitaremos usar menos líneas de código, usar el bucle 'for' también ayudará a reducir la probabilidad de escribir mal y tener un error de sintaxis en el programa.

El bucle 'for' y la función 'range()' en Python En Python, un bucle 'for' es una estructura de control que se puede utilizar para repetir conjuntos de comandos o declaraciones cualquier número de veces.

El uso de un ciclo 'for' le permite repetir (también llamado 'iterar sobre') un bloque de declaraciones tantas veces como desee.

El ciclo 'for' a menudo va junto con la función 'range()' en Python.

La función range() devuelve un conjunto de valores dentro de un cierto rango.

En EdPy, range() solo tiene un parámetro de entrada. Ese parámetro de entrada determina el límite superior del conjunto y el límite inferior siempre es 0.

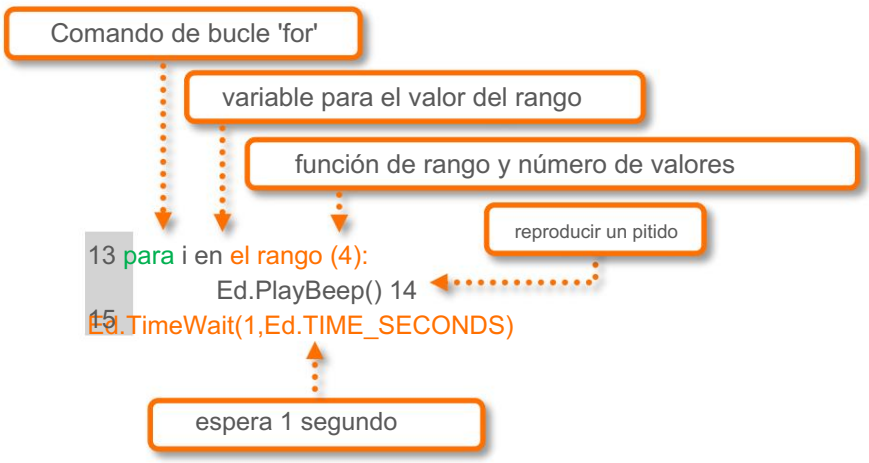
La función range() devuelve valores de 0 a (parámetro de entrada – 1).

Ejemplo:

range(4) → hay 4 valores en el conjunto: 0, 1, 2, 3.

Nombre_____

Veamos un ejemplo:



En este ejemplo, el bucle 'for' itera cuatro veces, lo que hace que la variable 'i' tenga los valores 0, 1, 2 y 3. Cada vez que itera, el bucle ejecuta el bloque de instrucciones que consta de Ed.PlayBeep() y Ed.TimeWait().

¿El resultado? El pitido se reproduce cuatro veces con un segundo de retraso entre cada pitido.

Tu turno:

Escriba un programa usando el ciclo 'for' y la función 'range()' para que cuando su robot Edison conduzca, forme un cuadrado. Debería poder completar el programa usando solo dos funciones Ed.Drive(), una para avanzar y otra para girar.

No olvide incluir dos puntos y una sangría adecuada dentro de su bucle.

Descarga tu programa y pruébalo usando la hoja de actividades 4.1, colocando tu Edison en el punto de 'inicio' y siguiendo las líneas. Asegúrese de que su programa termine con su Edison en el mismo lugar donde comenzó. También puede hacer un cuadrado más grande con cinta de color para marcar las líneas y un punto de "inicio" en un escritorio o en el piso.

1. ¿Cómo es su programa? Escribe tu código abajo.

_____

_____

_____

_____

_____

_____

Nombre_____

Lección 4: Hoja de trabajo 4.3: Conduzca en forma de triángulo y hexágono En esta actividad, debe escribir dos programas diferentes para que su robot Edison conduzca en forma de triángulo y luego en forma de hexágono.

Tu turno:

Tarea 1: conducir en un triángulo

Escriba un programa para que cuando su robot Edison conduzca, forme un triángulo.

Descarga tu programa y pruébalo usando la hoja de actividades 4.2, colocando tu Edison en el punto de 'inicio' y siguiendo las líneas. También puede hacer un triángulo más grande con cinta de color para marcar las líneas y un punto de "inicio" en un escritorio o en el piso.

1. ¿Cuántas veces se ejecutó el ciclo 'for' para la forma de su triángulo?

   _____

Tarea 2: conducir en un hexágono

Escriba un programa para que cuando su robot Edison conduzca, haga un hexágono.

Descarga tu programa y pruébalo usando la hoja de actividades 4.3, colocando tu Edison en el punto de 'inicio' y siguiendo las líneas. También puede hacer un hexágono más grande con cinta de color para marcar las líneas y un punto de "inicio" en un escritorio o en el piso.

2. ¿Cuántas veces se ejecutó el bucle 'for' para la forma de hexágono?

   _____

3. Deberías ver un patrón emergiendo entre el número de lados de la forma y el número de veces que se ejecuta el bucle 'for'. Describa ese patrón.

_____

_____

4. ¿Cuántas veces necesitaría ejecutar el ciclo 'for' para dibujar un patrón regular? (lo que significa que todos los lados son iguales) ¿Forma de 12 lados?

   _____

Nombre_____

Lección 4: Hoja de trabajo 4.4 – ¡Desafío! Conducir en círculos En esta actividad, debe escribir un programa para que su robot Edison conduzca en círculos.

Tu turno:

Escriba un programa en el que su robot Edison conduzca en círculos. Tu Edison necesita conducir en forma de círculo, no solo girar en un lugar.

Descarga tu programa y pruébalo usando la hoja de actividades 4.4, colocando tu Edison en el punto de 'inicio' y siguiendo la línea. También puede hacer que su robot conduzca alrededor de cualquier objeto circular, como un cubo de basura redondo o una mesa redonda.

Sugerencia: una forma con muchos cientos de lados muy pequeños puede aproximarse mucho a un círculo.

1. ¿Cuántas veces se ejecuta tu bucle para hacer que tu forma sea un círculo?

_____

_____

_____

2. ¿Qué distancia recorre su robot cada vez que ejecuta su bucle?

_____

_____

3. ¿Tu robot conduce en un círculo perfecto? Si no, ¿puedes pensar en una razón por la que no?

_____

_____

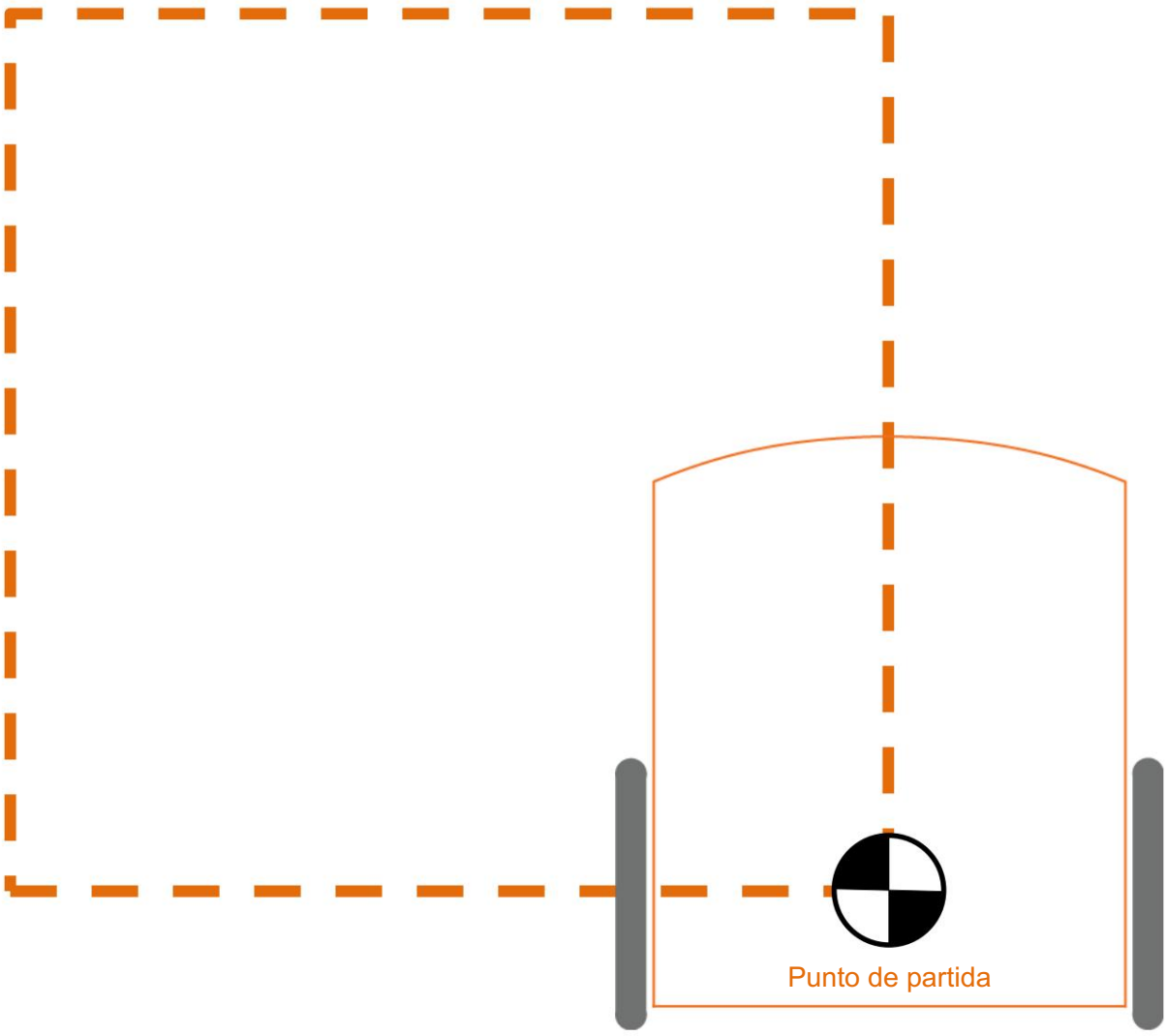
_____

Desafío opcional: dibuja tu forma

Adjunte un crayón o marcador de color a su robot usando alguna combinación de piezas de bloques LEGO o cinta adhesiva. Coloque su Edison en una hoja de papel y ejecute su programa circular. Mira como el marcador de color dibuja tu forma mientras el robot se mueve. Vea si su Edison dibuja un círculo verdadero o no.

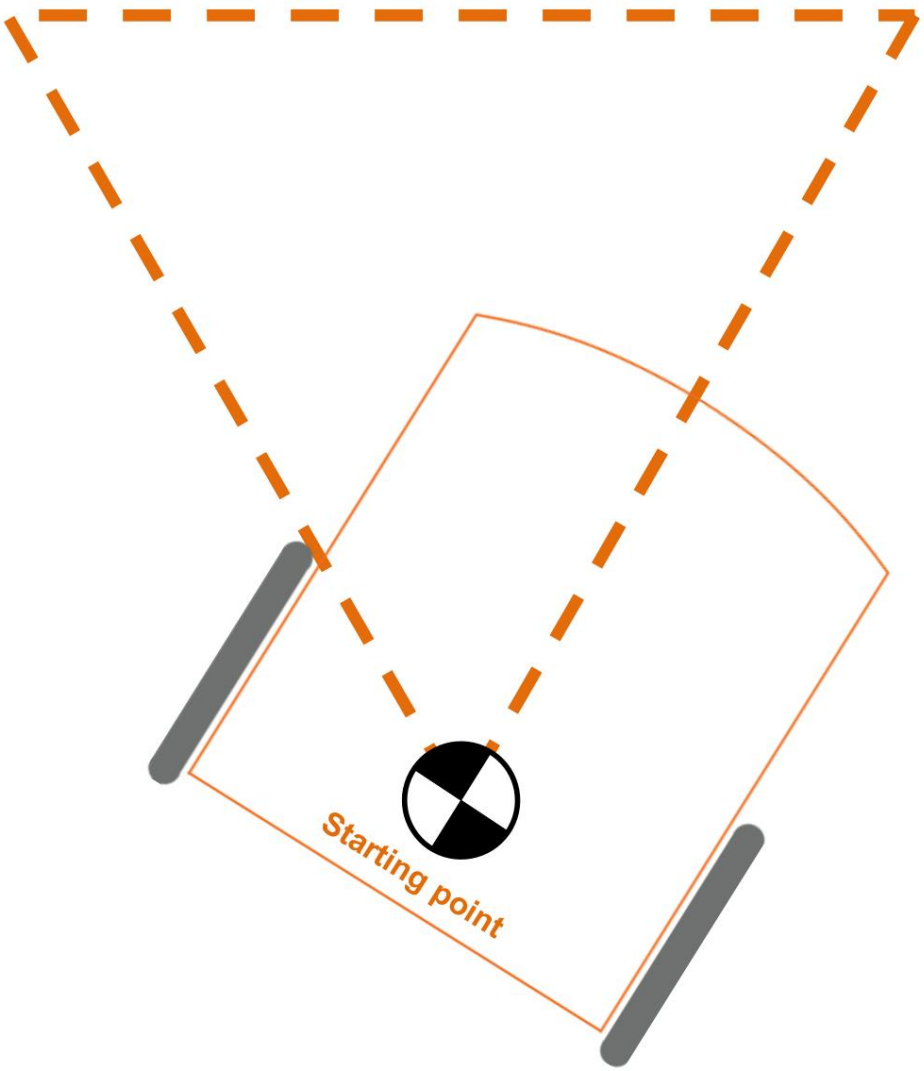
Nombre_____

Lección 4: Hoja de actividades 4.1



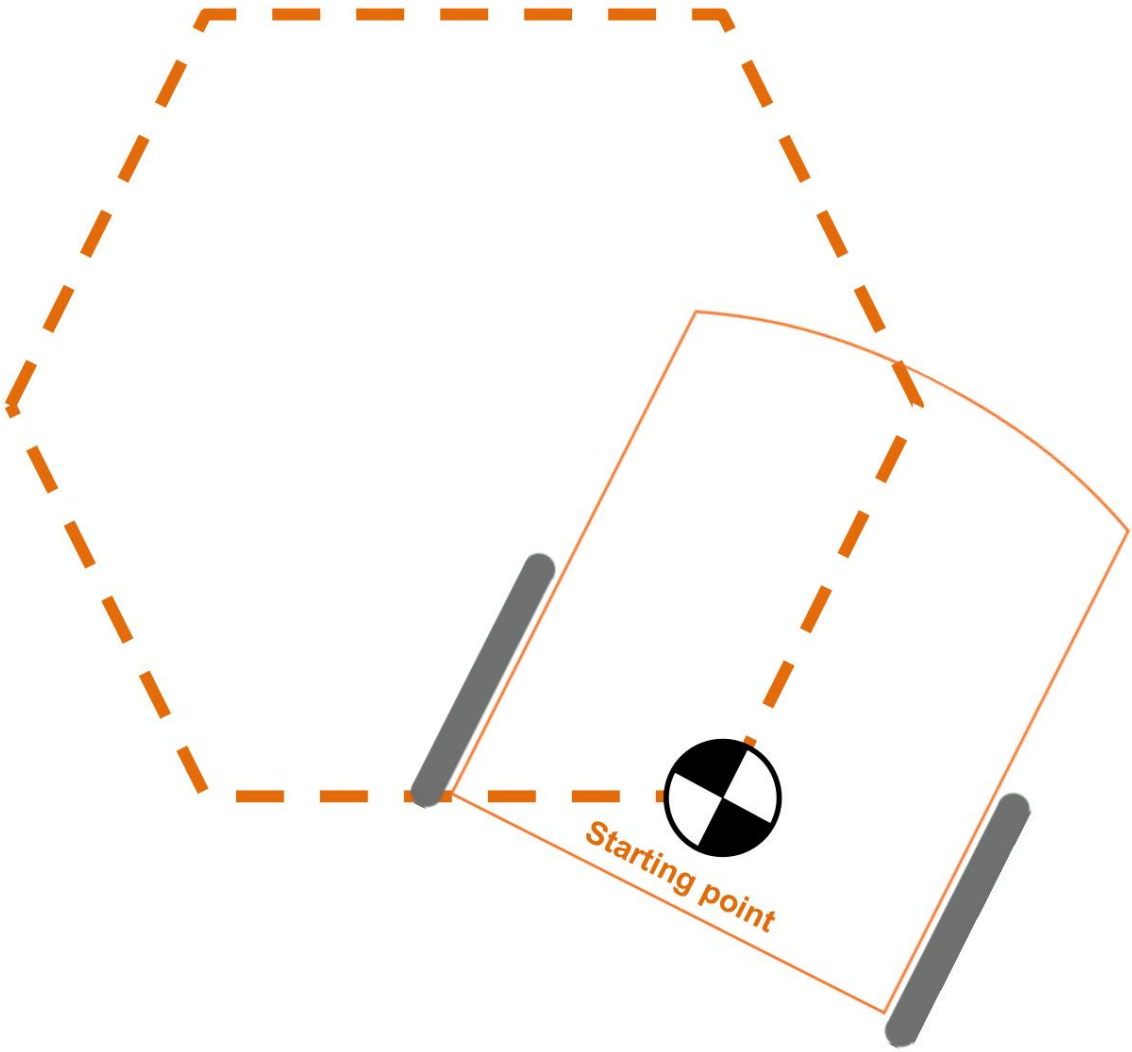
Nombre_____

Lección 4: Hoja de actividades 4.2



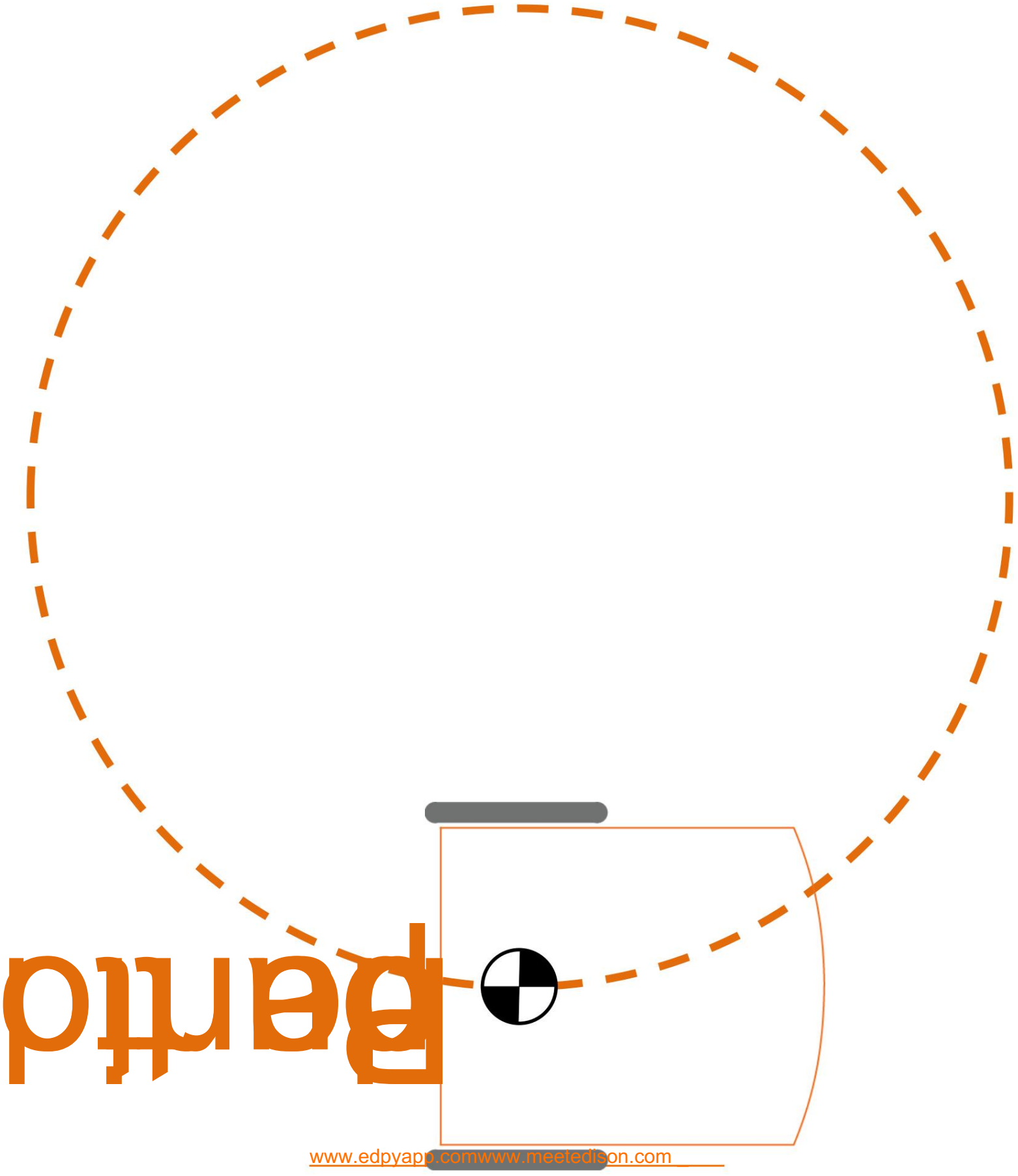
Nombre_____

Lección 4: Hoja de actividades 4.3



Nombre_____

Lección 4: Hoja de actividades 4.4



Nombre_____

Lección 5: Hoja de trabajo 5.1 – Tocar tonos En esta actividad, debe escribir un programa para hacer que Edison toque una nota musical y aprender cómo toca Edison los sonidos en un programa.

Puede reproducir notas musicales individuales a través del pequeño altavoz de Edison usando la función Ed.PlayTone() en EdPy.

La función Ed.PlayTone() toma dos parámetros de entrada: la nota y la duración. La nota determina qué nota tocar y la duración determina el período de tiempo dado que se debe tocar la nota.

Esta lista incluye los posibles valores de los parámetros:

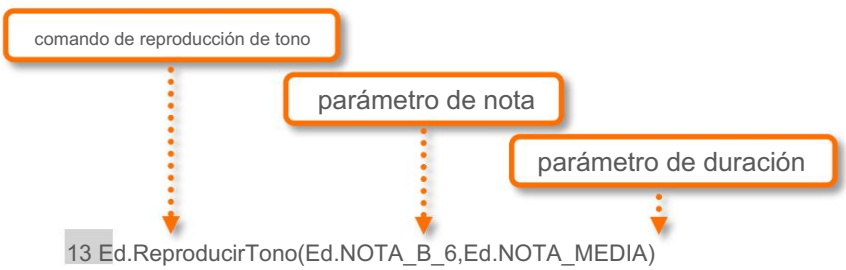
nota

Opciones de entrada de parámetros	Reproduce la nota musical
Ed.NOTE_A_6	la baja
Ed.NOTE_A_SHARP_6	la baja sostenida
Ed.NOTE_B_6	baja B
Ed.NOTE_C_7	C
Ed.NOTE_C_SHARP_7	C sostenida
Ed.NOTE_D_7	D
Ed.NOTE_D_SHARP_7	D sostenida
Ed.NOTE_E_7	Y
Ed.NOTE_F_7	F
Ed.NOTE_F_SHARP_7	F sostenida
Ed.NOTE_G_7	GRAMO
Ed.NOTE_G_SHARP_7	Sol sostenido
Ed.NOTE_A_7	A
Ed.NOTE_A_SHARP_7	La sostenido
Ed.NOTE_B_7	B
Ed.NOTE_C_8	alto C
Ed.NOTE_REST	descanso

duración

Opciones de entrada de parámetros	Reproduce la nota para
Ed.NOTE_SIXTEENTH	125 milisegundos
Ed.NOTE_EIGHTH	250 milisegundos
Ed.NOTE_QUARTER	500 milisegundos
Ed.NOTE_HALF	1000 milisegundos
Ed.NOTE_WHOLE	2000 milisegundos

Echemos un vistazo más de cerca a la función de tono de reproducción en un programa:



Utilizando las tablas de valores de los parámetros como referencia, ¿puede averiguar qué hará el programa?

Este programa reproducirá una nota B grave durante 1 segundo.

Nombre_____

Tu turno:

Tarea 1: tocar una nota

Escribe el siguiente programa:

```
#-------------Setup----------------

import Ed

Ed.EdisonVersion = Ed.V2

Ed.DistanceUnits = Ed.CM
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------

Ed.PlayTone(Ed.NOTE_A_SHARP_7,Ed.NOTE_HALF)

```

Descarga y prueba el programa para ver cómo suena.

Tarea 2: ¿ Tocar una nota y luego conducir? ¿O tocar una nota mientras conduce?

Cuando Edison reproduce sonidos, lo hace en segundo plano. Esto significa que tan pronto como Edison comience a reproducir el sonido, el programa pasará a la siguiente línea de código. El sonido seguirá sonando 'de fondo' mientras Edison continúa con el programa.

Si desea que Edison espere a que termine el sonido, debe usar la función Ed.ReadMusicEnd() en un bucle 'while'.

Escribe el siguiente programa:

```
#-------------Setup----------------

import Ed

Ed.EdisonVersion = Ed.V2

Ed.DistanceUnits = Ed.CM
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------
Ed.PlayTone(Ed.NOTE_C_8, Ed.NOTE_WHOLE)
while Ed.ReadMusicEnd()==Ed.MUSIC_NOT_FINISHED:
    pass
Ed.Drive(Ed.FORWARD, Ed.SPEED_6, 5)

```

Descarga y prueba el programa.

Nombre_____

1. Describa lo que sucedió cuando ejecutó este programa.

_____

_____

_____

2. Mira la línea 13 y 14 del programa. Recuerda que las expresiones comparan el lado izquierdo con el lado derecho de la notación en la expresión. ¿Qué está haciendo este bucle?

_____

_____

_____

Escribe el siguiente programa:

```
1
2  #-------------Setup----------------
3
4  import Ed
5
6  Ed.EdisonVersion = Ed.V2
7
8  Ed.DistanceUnits = Ed.CM
9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #--------Your code below-----------
12 Ed.PlayTone(Ed.NOTE_C_8, Ed.NOTE_WHOLE)
13 Ed.Drive(Ed.FORWARD, Ed.SPEED_6, 5)
14
```

Descarga y prueba el programa.

3. Describa lo que sucedió cuando ejecutó este programa.

_____

_____

_____

4. ¿Por qué este programa se comportó de manera diferente al último programa?

_____

_____

_____

Nombre_____

Lección 5: Hoja de trabajo 5.2 – Hacer una alarma

En esta actividad, debe escribir un programa para hacer que Edison active una alarma con la frecuencia que especifique.

Usando la función Ed.PlayTone(), puede personalizar la frecuencia exacta del sonido que produce el parlante de Edison usando números y variables.

Frecuencia en acústica

Como sabrás, el sonido viaja en ondas llamadas ondas sonoras. La acústica, la rama de la física que se ocupa del sonido y las ondas sonoras, analiza todo lo que tiene que ver con el sonido, incluida la forma de medirlo.

Una forma de medir el sonido es midiendo la frecuencia. La frecuencia es el número de ondas que pasan por un punto en un cierto período de tiempo.

La frecuencia se mide con mayor frecuencia en ciclos por segundo (ciclo/seg). La unidad básica de frecuencia es el hercio, abreviado Hz.

Un hercio es igual a una onda completa por segundo.

¿Sabías? El rango de audición humana es de 20 Hz ~ 20000 Hz.

Frecuencia y periodo

Además de las notas musicales que vienen preestablecidas en EdPy, también podemos programar a Edison para que reproduzca sonidos con diferentes frecuencias.

Para hacer esto, convertimos frecuencias en períodos, que Edison puede entender.

Un período es el tiempo que tarda una onda acústica en completar un ciclo completo. Como estamos usando hercios, medimos la frecuencia en ciclos por segundo.

En acústica, cuando aumenta el período, disminuye la frecuencia.

Veamos algunos ejemplos de cómo se relacionan la frecuencia y el período:

- Si una onda tiene un período de 0,5 segundos, tiene una frecuencia de 2 Hz porque puede completar 2 ciclos en 1 segundo.
- Si una onda tiene un período de 2 segundos, tiene una frecuencia de 0,5 Hz porque solo puede completar la mitad de un ciclo en 1 segundo.

Conversión de frecuencia a período para su programa

Para lograr que Edison reproduzca una frecuencia personalizada, necesitamos calcular el valor del período. Este es el número que ingresamos en el parámetro 'nota' en Ed.PlayTone().

Nombre_____

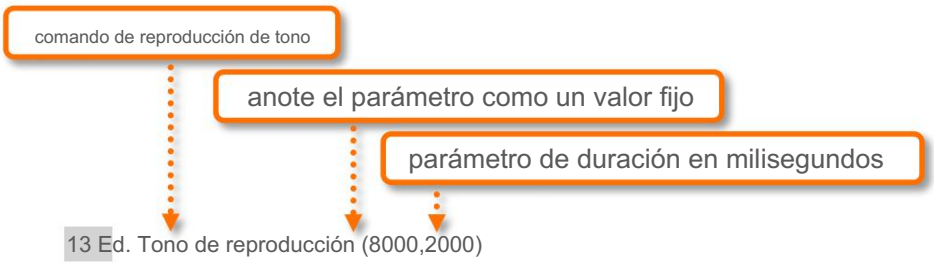
Para convertir una frecuencia en un período, divida el número 8,000,000 por la frecuencia deseada. Por ejemplo, para reproducir un sonido de 1 kHz (1000 ciclos por segundo):

$$\frac{8000000}{1000} = 8000$$

Tu turno:

Tarea 1: reproducir un tono personalizado

Escribe el siguiente programa:



13 Ed. Tono de reproducción (8000,2000)

Descárgalo y pruébalo para escuchar cómo suena este programa.

Tarea 2: reproducir una alarma

En este programa, queremos que Edison toque notas de período creciente.

Para hacer el programa de alarma, necesitará usar un bucle 'for', variables y la función range(). También necesita anidar un bucle 'while' en el programa.

Escribe el siguiente programa:

```
1
2  #-------------Setup----------------
3
4  import Ed
5
6  Ed.EdisonVersion = Ed.V2
7
8  Ed.DistanceUnits = Ed.CM
9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #--------Your code below-----------
12
13 for i in range(33):
14     Ed.PlayTone(100+(i*100), 1000)
15     while Ed.ReadMusicEnd()==Ed.MUSIC_NOT_FINISHED:
16         pass
17
```

Descárgalo y prueba cómo suena este programa.

Nombre_____

1. ¿Qué escuchas del robot? ¿Por qué está pasando esto?

_____

_____

_____

Una habilidad importante en la programación es poder 'rastrear' a través de un programa para comprender lo que está sucediendo. Los programadores realizan un seguimiento de código como método para simular manualmente la ejecución de su código para verificar que funciona correctamente antes de compilarlo manualmente.

El rastreo implica recorrer el programa línea por línea, registrando valores importantes. A menudo se hace para ayudar a encontrar errores o "errores" en el código, pero también es útil cuando solo necesita comprender lo que sucede en un programa.

Trate de "rastrear" lo que está sucediendo en el programa y responda las siguientes preguntas sobre el programa.

2. Complete la siguiente tabla calculando el parámetro de período para cada valor dado de 'i' en el código anterior. El primer valor se rellena por usted.

valor de yo	Parámetro de período [el primer parámetro de entrada a PlayTone()]
0	100
1 2	

3. ¿Cuál es el valor máximo de i?

_____

4. ¿Cuál es el valor máximo de la entrada del parámetro de período a PlayTone()? ¿función?

_____

5. ¿Cuántos tonos se reproducen?

_____

Nombre_____

¡Intentalo!

La aplicación de la acústica en la tecnología se denomina ingeniería acústica.

Prueba algo de ingeniería acústica por tu cuenta. Experimente modificando los parámetros de PlayTone() para hacer que el programa reproduzca una combinación diferente de sonidos.

Nombre_____

Lección 5: Hoja de trabajo 5.3 – Tocar una melodía En esta actividad, debe escribir un programa para hacer que Edison toque una melodía musical.

Puede hacer que Edison toque una melodía usando la función Ed.PlayTune() y un tipo especial de entrada llamado 'cadena'.

Usar una cadena para reproducir una melodía En Python, una 'cadena' es una lista de caracteres en orden. Un 'carácter' es cualquier cosa que pueda escribir en el teclado, como una letra, un número o un carácter especial como $ o #. Por ejemplo, 'Meet Edison' es una cadena de 11 caracteres (10 letras y 1 espacio).

En la aplicación EdPy, necesitamos usar una cuerda para tocar una melodía musical. A esto lo llamamos una 'cadena de sintonía'.

Las cadenas de melodías son una cadena especial de caracteres que representan melodías particulares. Las cadenas de sintonía se componen de entradas de notas y duración, que se representan mediante caracteres individuales.

Una cadena de melodía se ve así: “ndndndndnd…ndz” donde n es una nota de la tabla de notas y d es la duración de la tabla de duración:

Tabla de notas

Carácter de cuerda Toca una nota musical baja A
metro baja A
METRO aguda baja B
norte
C C
Cd Do sostenido
D
D re sostenido
Es Y
F F
F fa sostenido
gramo GRAMO
GRAMO sol sostenido
a A
un Un afilado
b B
O alto C
R resto
Con final de melodía

Tabla de duración

Carácter de cuerda Toca 1 nota
entera 2 media nota 4 negra 8 corchea
6 semicorchea

Todas las cadenas de melodías deben terminar con el carácter 'z' para terminar correctamente.

Para crear una cadena de melodía, debe llamar a la función Ed.TuneString(), que tiene dos parámetros de entrada. El tamaño de la cadena (en otras palabras, la cantidad de caracteres en la cadena) es el primer parámetro, y la cadena real que desea reproducir es el segundo parámetro.

Puede cambiar la velocidad de reproducción de su melodía cambiando la variable Ed.Tempo en el código de configuración.

Nombre_____

Tu turno:

Escribe el siguiente código para reproducir la melodía 'Mary Had a Little Lamb':

```
1
2  #-------------Setup----------------
3
4  import Ed
5
6  Ed.EdisonVersion = Ed.V2
7
8  Ed.DistanceUnits = Ed.CM
9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #--------Your code below-----------
12 maryLamb = Ed.TuneString(53,"e4d4c4d4e4e4e2d4d4d2e4g4g4e4d4c4d4e4e4e4e4d4d4e4d4c1z")
13
14 Ed.PlayTune(maryLamb)
15 while Ed.ReadMusicEnd() == Ed.MUSIC_NOT_FINISHED:
16     pass
17
```

Esta es la cadena de sintonía en el programa:

"e4d4c4d4e4e4e2d4d4d2e4g4g4e4d4c4d4e4e4e4e4d4d4e4d4c1z"

Experimente cambiando el valor Ed.Tempo en el código de configuración.

1. ¿Cuáles son los diferentes valores que puede tomar Ed.Tempo?

Sugerencia: recuerde que puede usar la función de autocompletar en EdPy. Intente escribir 'Ed.TEMPO' y vea todos los valores posibles para Ed.TEMPO que aparece en la función de autocompletar.

_____

_____

2. ¿Qué valor de Ed.TEMPO hará que la melodía suene más rápido?

_____

3. Modifique su programa para reproducir solo una parte de la melodía. Describa los cambios que Ud. tuvo que hacer a su programa para reproducir solo una parte de la melodía.

_____

_____

_____

_____

_____

Nombre_____

Lección 5: Hoja de trabajo 5.4 – Haz bailar a tu robot En esta actividad, escribirás un programa para hacer bailar a tu robot.

En la mayoría de los buenos espectáculos de danza, hay algunos movimientos o acciones que se repiten. Puedes hacer que tu Edison repita acciones en una rutina de baile usando el bucle 'for'.

El 'shimmy' es un movimiento de baile en el que mantienes el cuerpo quieto y mueves rápidamente los hombros hacia adelante y hacia atrás.

Mire el siguiente programa que hará que su robot Edison haga una versión de un shimmy:

```
1
2  #-------------Setup----------------
3
4  import Ed
5
6  Ed.EdisonVersion = Ed.V2
7
8  Ed.DistanceUnits = Ed.CM
9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #--------Your code below-----------
12
13 #Set up some variables
14 turnSpeed = Ed.SPEED_9
15 degreesToTurn = 20
16 numberOfTwists = 3
17
18 #Now shimmy!
19 Ed.Drive(Ed.SPIN_RIGHT,turnSpeed,degreesToTurn/2)
20 for i in range(numberOfTwists):
21     Ed.Drive(Ed.SPIN_LEFT,turnSpeed,degreesToTurn)
22     Ed.Drive(Ed.SPIN_RIGHT,turnSpeed,degreesToTurn)
23 Ed.Drive(Ed.SPIN_LEFT,turnSpeed,degreesToTurn/2)
24
```

Este programa utiliza variables para que sea fácil cambiar la velocidad de giro, el número de giros en el baile y los grados que girará Edison.

Tanto la línea 13 como la 18 comienzan con '#', lo que significa que estas líneas son líneas de código de comentario agregadas para facilitarnos la lectura del programa. Recuerde, Edison omitirá cualquier línea que comience con '#'.

Mire las líneas 19 y 23. En estas líneas, estamos haciendo un cálculo matemático en nuestro código para hacer que Edison gire solo la mitad de la cantidad de grados.

Tu turno:

Escribe el programa.

Descargue el programa a su Edison y ejecútelo para ver el baile en acción.

Nombre_____

1. ¿Cuántas veces gira el robot hacia la izquierda?

   _____

2. ¿Cuántas veces gira el robot hacia la derecha?

   _____

3. El primer giro a la derecha es solo la mitad de la distancia de todos los giros dentro del bucle 'for' porque esta línea tiene el parámetro de entrada 'degreesToTurn/2'. ¿Por qué quieres esta línea en el programa? Intente eliminar las matemáticas (el /2) y vuelva a ejecutar el programa. ¿Que notaste? (Sugerencia: mire cuánto se mueve Edison hacia la izquierda en comparación con el punto de inicio).

_____

_____

_____

_____

¡Intentalo!

Experimente con el programa. Intenta cambiar las variables para cambiar la forma en que baila Edison. ¡Cambia el número de grados que girará Edison, la velocidad a la que girará Edison, el número de giros en el baile o los tres!

Nombre_____

Lección 5: Hoja de trabajo 5.5 – ¡Desafío! Baila con música ¡Bailar es más divertido con música! En esta actividad, escribirás un programa que combina pasos de baile con algunos tonos o una melodía.

Tu turno:

Escribe y ejecuta el siguiente programa que combina un baile 'shimmy' con algunos tonos:

```
#-------------Setup----------------

import Ed

Ed.EdisonVersion = Ed.V2

Ed.DistanceUnits = Ed.CM
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------

#Set up my variables
turnSpeed = Ed.SPEED_9
degreesToTurn = 60
numberOfTwists = 3

#Now dance to the music!
Ed.Drive(Ed.SPIN_RIGHT, turnSpeed, degreesToTurn/2)
Ed.PlayTone(Ed.NOTE_C_7, Ed.NOTE_SIXTEENTH)
for i in range(numberOfTwists):
    Ed.Drive(Ed.SPIN_LEFT, turnSpeed, degreesToTurn)
    Ed.PlayTone(Ed.NOTE_A_7, Ed.NOTE_SIXTEENTH)
    Ed.Drive(Ed.SPIN_RIGHT, turnSpeed, degreesToTurn)
    Ed.PlayTone(Ed.NOTE_C_7, Ed.NOTE_SIXTEENTH)
Ed.Drive(Ed.SPIN_LEFT, turnSpeed, degreesToTurn/2)
Ed.PlayTone(Ed.NOTE_A_7, Ed.NOTE_SIXTEENTH)
```

Ahora diseñe su propio baile para su Edison, agregando algunos tonos o usando una cuerda de melodía. ¿Puedes sincronizarlo para que Edison baile al compás de la música?

1. Describe los movimientos de baile de tu robot. ¿Hay algo en su programa que realmente le gustó? Si es así, descríbalo.

_____

_____

_____

_____

_____

Nombre_____

2. ¿Qué combinación de tonos o notas tocó junto con su baile?

_____

_____

_____

_____

Nombre_____

Lección 6: Hoja de trabajo 6.1: el LED parpadea en respuesta a un aplauso En esta actividad, debe escribir un programa
utilizando el sensor de detección de aplausos de Edison para que el robot haga parpadear una luz LED cada vez que detecta un aplauso.

Lo primero que debe hacer es planificar el programa.

Diagramas de flujo en la programación

Los programadores profesionales generalmente planifican su programa antes de comenzar a escribir su código. El uso de diagramas de flujo es una forma en que los programadores pueden organizar y planificar sus programas.

La idea de un diagrama de flujo es resumir gráficamente lo que sucede en el código sin necesidad de entrar en todos los detalles. Los diagramas de flujo permiten a un programador visualizar y comunicar cómo funcionará el 'flujo' del programa.

En un diagrama de flujo, un programa se representa usando diferentes formas y flechas. Cada forma representa un elemento diferente en el programa, y las flechas muestran cómo funcionan los elementos juntos.

Hay cinco símbolos principales utilizados en los diagramas de flujo:

Símbolo	Nombre	Función
	Terminator (inicio/fin)	Un óvalo representa un punto inicial o final.
	Flecha	Una línea que actúa como conector, mostrando las relaciones entre formas representativas.
	De entrada y salida	Un paralelogramo representa una entrada o una salida.
	Proceso	Un rectángulo representa un proceso o acción.
	Decisión	Un diamante representa una decisión.

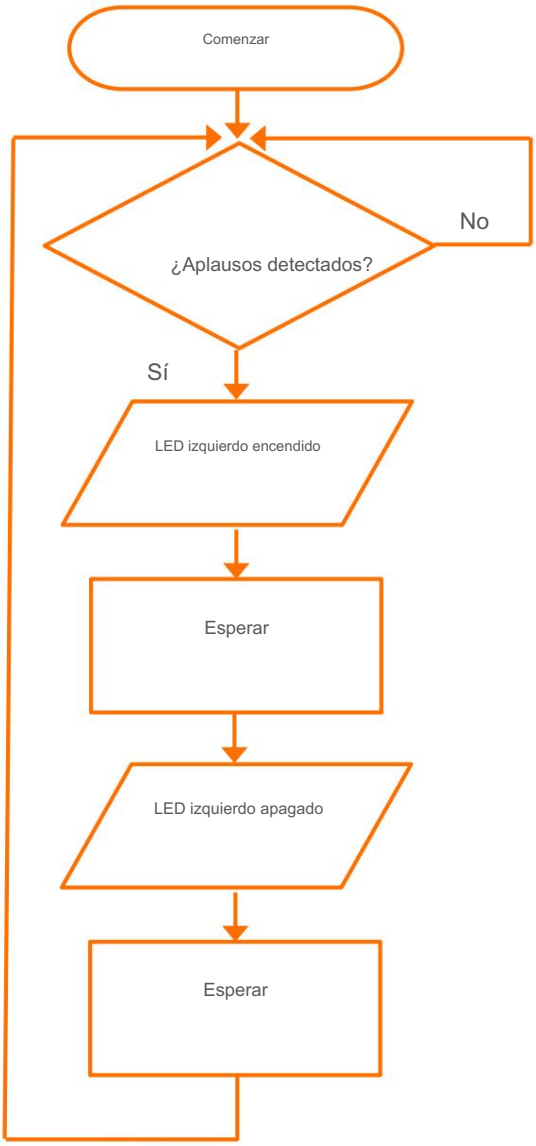
Los diagramas de flujo más complicados también pueden usar formas adicionales con diferentes significados.

Al hacer un diagrama de flujo para planificar un programa, a menudo se agregan palabras dentro de las formas o al lado de las flechas. Estas palabras son breves resúmenes del proceso o decisión.

Veamos un diagrama de flujo de ejemplo que resume el programa que queremos hacer.

Nombre_____

Aquí hay un diagrama de flujo para un programa que le dirá a su robot Edison que espere un aplauso, luego haga parpadear el LED izquierdo:



Este programa utilizará el sensor de detección de sonido de Edison para determinar si se ha producido o no un aplauso. El resultado determina cómo fluye el código a continuación.

Cuando mira este diagrama de flujo, puede notar que no tiene un terminador de 'fin'.
Esto se debe a que este programa usa un bucle 'while' configurado de manera que el programa continúe indefinidamente.

Hacer un bucle infinito A veces
puede querer escribir un programa que no tenga un final, pero que se repita para siempre.
En programación, esto a menudo se denomina bucle infinito.

Puede usar un ciclo infinito para hacer que el programa esté planeado en el diagrama de flujo.

Nombre_____

Mira el siguiente programa:

```
1
2  #-------------Setup----------------
3
4  import Ed
5
6  Ed.EdisonVersion = Ed.V2
7
8  Ed.DistanceUnits = Ed.CM
9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #--------Your code below-----------
12 while True:
13     Ed.ReadClapSensor()
14     while Ed.ReadClapSensor() != Ed.CLAP_DETECTED:
15         pass
16     Ed.LeftLed(Ed.ON)
17     Ed.TimeWait(50, Ed.TIME_MILLISECONDS)
18     Ed.LeftLed(Ed.OFF)
19     Ed.TimeWait(50, Ed.TIME_MILLISECONDS)
20
```

Este programa está representado por nuestro diagrama de flujo e incluye un bucle infinito.

Mire la línea 12 del programa, que usa un bucle 'while'.

Los bucles 'mientras' siempre necesitan una condición. El ciclo repetirá cualquier código sangrado mientras esa condición se resuelve como 'verdadera'.

Si queremos que el ciclo 'while' se repita infinitamente, en lugar de dar una condición que el programa debe evaluar, reemplazamos la condición con 'Verdadero'.

'Verdadero' siempre se resuelve en 'verdadero'. Al establecer la condición en 'Verdadero', codificamos la condición de nuestro bucle 'mientras' para que sea 'verdadero'.

En programación, la codificación rígida es donde obligas a que algo sea de una manera específica escribiéndolo explícitamente.

Al usar 'Verdadero' como condición para el bucle 'while' en nuestro programa, la condición del bucle nunca puede ser falsa y se repetirá infinitamente.

Tu turno:

Escriba el código anterior para programar su robot Edison para que el LED izquierdo parpadee cuando aplaude. Descárgalo y prueba para ver cómo funciona.

1. ¿Cuál es la distancia más lejana de su Edison que puede estar y aún tener
   ¿El robot siente cuando aplaudes?

   _____

Nombre_____

2. ¿Cuál es el propósito de tener llamadas a la función TimeWait() en el código? Qué pasaría si no los tuviéramos? Sugerencia: intente ejecutar el programa sin las llamadas a la función TimeWait().

_____

_____

_____

3. Observe el programa y el diagrama de flujo para comparar cómo se relacionan entre sí el código del programa y el diagrama de flujo. Explique qué sucede en el código cuando el resultado de la decisión representada en el diagrama de flujo es 'no'.

_____

_____

_____

¡Intentalo!

¿Puedes cambiar el programa para que ambos LED se enciendan cuando aplaudes?

Nombre_____

Lección 6: Hoja de trabajo 6.2 – Conducir en respuesta a un aplauso En esta actividad, debe escribir un programa para hacer que su robot Edison avance en respuesta a un aplauso.

Tu turno:

Tarea 1: conducir hacia adelante cuando se detecta un aplauso

Escriba y ejecute el siguiente programa:

```
1
2  #-------------Setup----------------
3
4  import Ed
5
6  Ed.EdisonVersion = Ed.V2
7
8  Ed.DistanceUnits = Ed.CM
9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #--------Your code below-----------
12 Ed.ReadClapSensor()
13 while Ed.ReadClapSensor() == Ed.CLAP_NOT_DETECTED:
14     pass
15 Ed.Drive(Ed.FORWARD,Ed.SPEED_8,10)
16
```

1. ¿Por qué es necesario realizar una lectura inicial del sensor de palmadas en la línea 12? Qué es esto haciendo? Sugerencia: Vuelva a consultar la hoja de trabajo 2.5.

_____

_____

_____

Tarea 2: conducir hacia adelante y luego hacia atrás cuando se detecta un aplauso

El sensor de sonido del robot Edison no solo es sensible a los aplausos. Los sensores pueden responder a cualquier sonido fuerte detectado, por lo que puede tocar cerca del altavoz del robot para activar el sensor de sonido.

Los motores, engranajes y ruedas de Edison emiten sonidos al girar, lo que puede activar el sensor de sonido. Para evitar que el sonido de la conducción del robot active el sensor de sonido, debe modificar el programa.

Deberá agregar una llamada a la función TimeWait() con un parámetro de entrada de aproximadamente 350 milisegundos para dar tiempo a que los motores del robot se detengan.

También necesita usar un ReadClapSensor() para borrar el sensor de aplausos.

Nombre_____

Escribe el siguiente programa:

```
#-------------Setup----------------

import Ed

Ed.EdisonVersion = Ed.V2

Ed.DistanceUnits = Ed.CM
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------
Ed.Drive(Ed.FORWARD,Ed.SPEED_8,10)

Ed.TimeWait(350,Ed.TIME_MILLISECONDS)
Ed.ReadClapSensor()
while Ed.ReadClapSensor() == Ed.CLAP_NOT_DETECTED:
    pass
Ed.Drive(Ed.BACKWARD,Ed.SPEED_8,10)

```

Descarga y prueba el programa.

2. Por lo general, escribimos un diagrama de flujo para ayudarnos a planificar nuestro código. Para practicar, escribe un diagrama de flujo para que coincida con el código que acaba de escribir. Dibuje su diagrama de flujo a continuación o intente usar un programa como Google Slides para hacer su diagrama de flujo.

Nombre_____

Lección 6: Hoja de trabajo 6.3: Diseñe su propia función En esta actividad, diseñará su propia función y la usará para escribir un programa para Edison.

¿Qué son exactamente las funciones?

Ahora que ha estado programando durante un tiempo con Edison y EdPy, ha utilizado una variedad de funciones diferentes de la biblioteca EdPy.

Como sabe, una función es una pieza de código que realiza una función o trabajo particular en el programa según los parámetros de entrada que se utilicen.

Pero es posible que no se haya dado cuenta de que todas las funciones que ha utilizado hasta ahora han ejecutado varias líneas de código cuando las ejecuta el programa.

Esto se debe a que una función es un bloque de código organizado y reutilizable que se utiliza para realizar una sola acción relacionada.

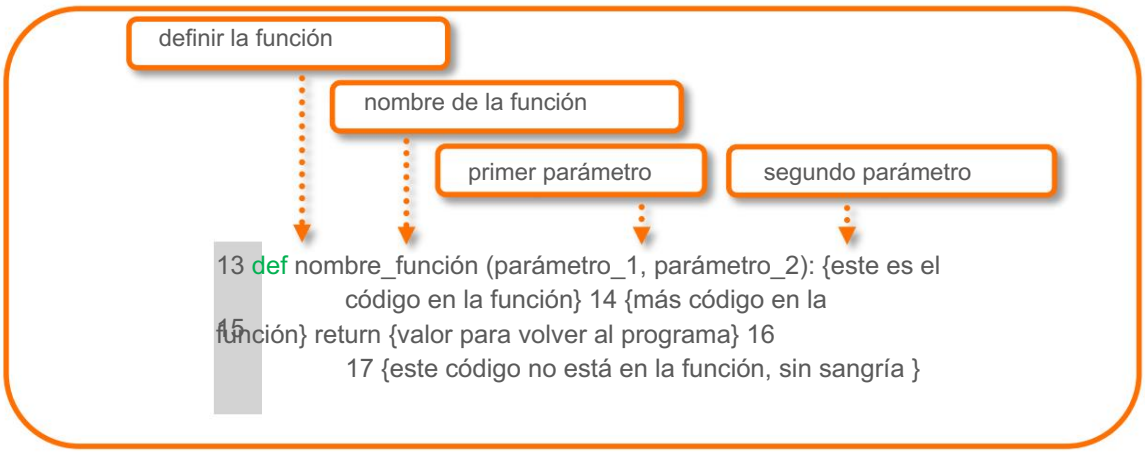
Las funciones son muy útiles porque nos permiten programar de una manera más modular, usando el mismo bloque de código en varios puntos a lo largo del programa. Para que su programa ejecute todas esas líneas de código, solo necesita escribir una línea: llamar a esa función.

Las funciones hacen que la reutilización del código sea más fácil y eficiente cuando programamos.

Mire un breve video de code.org que explica por qué las funciones son tan útiles en la programación: https://youtu.be/8T5acEwfJbw

Creando tu propia función Hasta ahora, cada vez que has usado una función en EdPy, la has llamado desde la biblioteca Ed. También puedes hacer tus propias funciones.

En Python, las funciones se ven así:



Cualquier cosa con sangría es una parte de la función. Cualquier cosa que no tenga sangría no es parte de la función, sino la siguiente línea de código en el programa.

Es importante tener en cuenta que las funciones que crea no son diferentes de las funciones que ha usado de la biblioteca Ed hasta ahora.

Nombre_____

Cuando llama a su función (con o sin parámetros), el programa salta de la llamada al código de la función (el código sangrado). Luego, el programa ejecuta este código antes de regresar a la línea donde realizó la llamada.

Si configura su función para que devuelva un valor, cuando el programa vuelve a la línea donde realizó la llamada, la llamada a la función se reemplaza por el valor que ha devuelto la función.

Esto es importante porque el programa siempre resolverá funciones, luego órdenes matemáticas y luego expresiones en este orden.

Organizando tu código

La convención en EdPy es escribir la definición de tus funciones al final de tu código después de que ya hayas usado tu función en tu programa.

Esto es para que su código sea agradable y ordenado. Al organizar su código de esta manera, todas sus funciones, ya sea que esté utilizando las suyas propias o llamando funciones de una biblioteca, se pueden escribir en la parte principal de su programa de una manera organizada y visualmente limpia. Sus definiciones de funciones pueden ubicarse fuera de esto, más abajo en la parte inferior del programa.

Tu turno:

Tarea 1: Practicar la definición de una función

Mira el siguiente programa:

```
1
2  #-------------Setup----------------
3
4  import Ed
5
6  Ed.EdisonVersion = Ed.V2
7
8  Ed.DistanceUnits = Ed.CM
9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #--------Your code below-----------
12 directionToMove=Ed.SPIN_LEFT
13 speedToMoveAt=Ed.SPEED_8
14 distanceToMove=360
15
16 #call the function
17 moveOnClap(directionToMove, speedToMoveAt, distanceToMove)
18
19
20 #user defined functions
21 def moveOnClap(direction, speed, distance):
22     Ed.ReadClapSensor()
23     while Ed.ReadClapSensor()==Ed.CLAP_NOT_DETECTED:
24         pass
25     Ed.Drive(direction, speed, distance)
26
27
```

Nombre_____

La línea 17 del programa es la llamada de función. Las líneas 21 a 25 están definiendo la función. Recuerde, la convención es definir su función en la parte inferior de su programa para mantener todo limpio y organizado.

Escriba el programa y descárguelo a su robot Edison. Ejecute el programa para ver cómo funciona.

1. Podríamos escribir una función para hacer que el robot Edison maneje en forma cuadrada. Completa las palabras que faltan en la siguiente función para terminar de escribir esta función.

```
def driveInaSquare():

    para i en el rango (___):

        Ed.Drive(_____,_____,_____)

        Ed.Drive(_____,_____,_____)
```

2. Escriba la sintaxis del código para llamar a esta función. En otras palabras, ¿qué harías escriba en su programa para llamar a esta función?

_____

Escribe un programa que impulse a Edison en múltiples casillas. Deberá definir la función 'driveInaSquare' y su programa deberá llamar a esta función más de una vez.
Descarga y ejecuta el programa para ver cómo funciona.

3. ¿El programa hace lo que esperaba que hiciera? Si no, describe lo que esperado y lo que el programa realmente hizo. Describa cualquier problema que haya tenido para que su programa funcione.

_____

_____

_____

_____

_____

Tarea 2: Diseña tu propia función

Escribe tu propia función que hará que Edison haga algo cuando aplaudas. ¡Puedes hacer bailar a Edison, encender un LED, girar en círculos o cualquier otra cosa que quieras!

Nombre_____

Paso 1: Diseño

Primero, escriba un diagrama de flujo que resuma su programa gráficamente. Haga su diagrama de flujo dibujándolo en papel o use un programa como Google Slides. Asegúrese de usar las formas correctas en su diagrama de flujo para representar el inicio, los procesos, los puntos de decisión y el flujo del programa.

Las formas que necesitará dependerán de su diagrama de flujo. Como mínimo, necesitará la forma de inicio ovalada, la forma de proceso de rectángulo y la forma de decisión de diamante en su diagrama de flujo.

Paso 2: Código

Traduzca su idea a código en la aplicación EdPy. Use su diagrama de flujo como guía y codifique su función para incluir cada paso que presentó en su diagrama de flujo.

Paso 3: Prueba

Escriba un programa de prueba que incluya su función y código adicional para 'ejercer' su función. (Ejercer una función significa llamarla en su código). Vea si su función funciona como lo planeó y esperaba que funcionara. De lo contrario, revise su diagrama de flujo y código, ajustándolo según sea necesario. ¡Experimenta para ver qué funciona!

4. Describe qué hizo tu función.

_____

_____

_____

_____

_____

_____

5. Describa cualquier problema que haya tenido.

_____

_____

_____

_____

_____

Nombre_____

Lección 7: Hoja de actividad 7.1 – Calibrar obstáculo detección

Puedes regular la sensibilidad del sistema de detección de obstáculos de Edison. Al hacer que el sistema de detección de obstáculos sea más sensible, Edison puede detectar obstáculos más lejanos. Al hacer que el sistema sea menos sensible, Edison solo detectará obstáculos muy cercanos. Use esta hoja de actividades para ajustar el sistema de detección de obstáculos de su Edison.

Paso 1: leer el código de barras

1. Coloque a Edison en el lado derecho, mirando hacia el código de barras 2. Presione el botón de grabación (redondo) tres veces 3. Edison avanzará y escaneará el código de barras

Código de barras: calibre la detección de obstáculos

Paso 2: establece la sensibilidad máxima

Después de escanear el código de barras, coloque a Edison sobre una mesa o escritorio y elimine cualquier obstáculo frente a Edison. Luego presione el botón de reproducción (triángulo). Edison está ahora en modo de calibración.

La sensibilidad izquierda se calibra primero.

1. Presione repetidamente el botón de reproducción (triángulo), que aumenta la sensibilidad, hasta que el rojo El LED de la izquierda parpadea.
2. Presione repetidamente el botón de grabación (redondo), que disminuye la sensibilidad, hasta que el LED deje de parpadear por completo.
3. Presione el botón de parada (cuadrado) para cambiar y calibrar el lado derecho.
4. Presione repetidamente el botón de reproducción (triángulo) hasta que el LED rojo derecho parpadee. Ahora presione repetidamente el botón de grabación (redondo) hasta que el LED deje de parpadear por completo.
5. Pulse el botón de parada para completar la calibración.

Nota especial: sensibilidad personalizada

Es posible establecer la distancia a la que se detectan los obstáculos. Para hacer esto, escanee el código de barras 'calibrar detección de obstáculos', coloque un obstáculo frente a Edison a la distancia que desea que Edison detecte obstáculos, presione el botón de reproducción y luego repita los pasos 1 a 5.

Nombre_____

Lección 7: Hoja de trabajo 7.1 – Detección de obstáculos por infrarrojos

En esta actividad, aprenderá más sobre la luz infrarroja (IR) y cómo Edison puede usar IR para detectar obstáculos.

¿Qué es la luz infrarroja (IR)?

Existe una amplia gama de luces, algunas de las cuales son visibles para el ojo humano y otras no. El infrarrojo, también llamado IR, no es visible para los humanos.

> ¿Sabías? Aunque la gente no puede verlo, el infrarrojo es un tipo de luz.
> Por lo tanto, funcionará en la oscuridad. ¡Es por eso que puede encender un televisor con un control remoto incluso si no hay luces encendidas en la habitación!

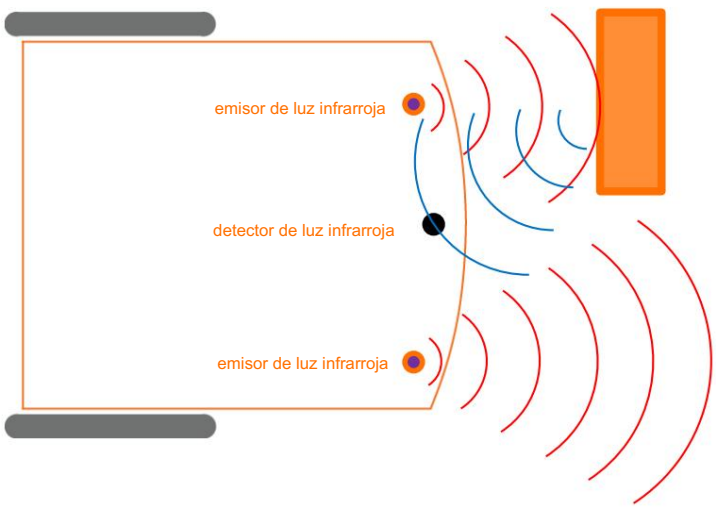
Edison y los infrarrojos

El robot Edison está equipado con un sistema de infrarrojos que le da una especie de "visión" al robot. Este sistema de infrarrojos permite a Edison detectar obstáculos alrededor del robot.

El sistema de infrarrojos de Edison se compone de dos diodos emisores de luz IR (o LED) en la parte delantera. Uno está a la izquierda y el otro a la derecha. Edison también tiene un sensor IR en el frente, directamente en el medio.

Para que Edison detecte obstáculos, la luz infrarroja se emite hacia adelante desde los LED IR izquierdo y derecho. Si la luz IR encuentra un obstáculo, como una pared, se refleja hacia Edison. El sensor IR de Edison detecta la luz reflejada.

Mira la siguiente ilustración:



Edison emite luz IR (que se muestra en rojo) desde los LED IR izquierdo y derecho del robot. La luz IR reflejada (que se muestra en azul) rebota en los obstáculos y es detectada por el sensor IR de Edison.

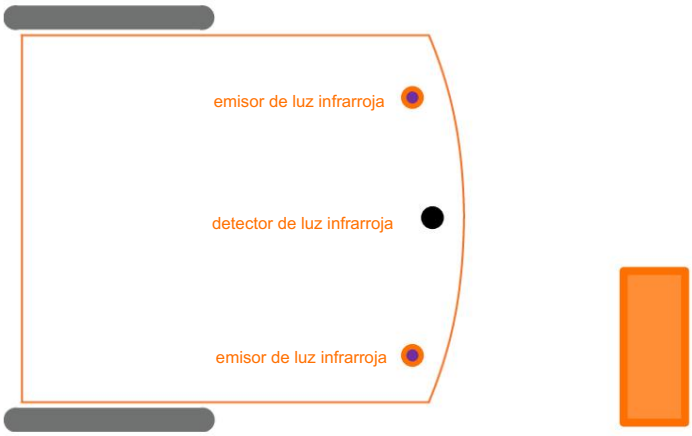
En esta imagen, hay un obstáculo enfrente del lado izquierdo de Edison, pero no del lado derecho. Es por eso que solo se refleja la luz IR del emisor izquierdo.

A partir de la señal recibida, Edison puede determinar que hay un obstáculo a la izquierda, pero ningún obstáculo a la derecha.

Nombre_____

Tu turno:

1. Dibuje la luz IR emitida y la luz IR reflejada para este obstáculo.



Nombre_____

Lección 7: Hoja de trabajo 7.2: Detectar un obstáculo y detenerse En esta actividad, deberá escribir un programa para hacer que su robot Edison conduzca hasta que encuentre un obstáculo y luego se detenga.

Mira el siguiente programa:

```
1
2  #-------------Setup----------------
3
4  import Ed
5
6  Ed.EdisonVersion = Ed.V2
7
8  Ed.DistanceUnits = Ed.CM
9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #--------Your code below-----------
12
13 Ed.ObstacleDetectionBeam(Ed.ON)
14
15 Ed.Drive(Ed.FORWARD,Ed.SPEED_5,Ed.DISTANCE_UNLIMITED)
16
17 while Ed.ReadObstacleDetection() != Ed.OBSTACLE_AHEAD:
18     pass
19 Ed.Drive(Ed.STOP,1,1)
20
```

Este programa le dice a Edison que conduzca hasta que encuentre un obstáculo.

Hay un par de cosas importantes a tener en cuenta sobre este programa.

Mira la línea 13 del programa. Esta línea enciende el haz de detección de obstáculos de Edison. Siempre que desee utilizar el haz de detección de obstáculos de Edison en un programa EdPy, siempre debe encender el haz antes de utilizarlo en el programa.

Ahora, mira la línea 15 del programa. Esta línea establece la velocidad en 5 en este programa. Al usar la detección de obstáculos, debe usar una velocidad ligeramente más baja para permitir que el robot detecte un obstáculo antes de chocar con él. Si la velocidad es demasiado rápida, el robot chocará contra los obstáculos antes de poder detectarlos.

Tu turno:

Tarea 1: selecciona tus obstáculos

Debe elegir buenos obstáculos para usar con Edison. Si un obstáculo es demasiado pequeño o no refleja suficiente luz infrarroja, Edison no puede detectarlo. Seleccione un objeto que sea opaco pero no demasiado oscuro (por ejemplo, no negro) y al menos tan alto como Edison.

Para este programa, la pared de la habitación sería un buen obstáculo.

Nombre_____

Tarea 2: Prepara tu Edison

Cuando desee ejecutar un programa utilizando la detección de obstáculos, es una buena idea verificar que la detección de obstáculos de su robot Edison esté calibrada a la distancia que desea. Use la hoja de actividad 7.1 para calibrar su Edison. Es posible que deba calibrar su robot con una sensibilidad personalizada para asegurarse de que su robot pueda detectar y detenerse en su obstáculo.

Tarea 3: escribir y ejecutar el programa

Escriba el programa usando la aplicación EdPy y descárguelo a su robot Edison. Luego ejecute el programa para ver cómo funciona.

Experimente usando diferentes obstáculos para ver lo que Edison puede y no puede detectar. También puede intentar configurar la calibración de detección de obstáculos de Edison a diferentes distancias para ver qué sucede.

1. Elimine la línea 'Ed.ObstacleDetectionBeam(Ed.ON)' de su programa. Intentar descargar y ejecutar el programa ajustado. ¿Qué pasó? ¿Por qué sucede eso?

_____

_____

_____

2. Piense en dónde ha visto antes este tipo de detección invisible en el mundo real. mundo. Describa un ejemplo.

_____

_____

_____

3. ¿Dónde más cree que podría usarse este tipo de tecnología de detección? Escribir escriba al menos una idea de cómo podría utilizar esta tecnología.

_____

_____

_____

_____

Nombre_____

Lección 7: Hoja de trabajo 7.3 – Evasión de obstáculos

En esta actividad, escribirá un programa para hacer que su robot Edison conduzca hasta que encuentre un obstáculo, gire y se aleje.

Mira el siguiente programa:

```
#-------------Setup----------------

import Ed

Ed.EdisonVersion = Ed.V2

Ed.DistanceUnits = Ed.CM
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------

Ed.ObstacleDetectionBeam(Ed.ON)

Ed.Drive(Ed.FORWARD,Ed.SPEED_5,Ed.DISTANCE_UNLIMITED)

while Ed.ReadObstacleDetection() != Ed.OBSTACLE_AHEAD:
    pass
Ed.Drive(Ed.SPIN_RIGHT,Ed.SPEED_5,180)
Ed.Drive(Ed.FORWARD,Ed.SPEED_5,10)
```

Este programa le dice a Edison que avance hasta que se detecte un obstáculo. Una vez que Edison detecta un obstáculo, el programa le dice a Edison que gire 180° y se aleje 10 cm.

Tu turno:

Tarea 1: escribir y ejecutar el programa

Escriba el programa usando la aplicación EdPy y descárguelo a su robot Edison. Luego ejecute el programa para ver cómo funciona.

Después de ejecutar el programa, mírelo de nuevo y piense en cómo podría cambiar el programa. Intenta modificar el código para que el robot se comporte de forma diferente cuando detecte un obstáculo por delante. (Sugerencia: intente usar sonido y luces).

1. Piense en cómo podría mejorar el programa original. ¿Qué podrías cambiar?
   para que el programa haga algo más que dar la vuelta y alejarse después de encontrar un obstáculo?

_____

_____

_____

Nombre_____

Tarea 2: errores de sintaxis y errores lógicos

Los programadores a menudo cometen errores tipo 'typo', llamados errores de sintaxis. Es importante ser bueno para detectar sus propios errores de sintaxis para que pueda corregir su código.

También puede obtener otro tipo de error, llamado error lógico, cuando programa. En codificación, cualquier error que no sea un error de sintaxis es un error lógico.

Cuando hay un error lógico en un programa, el código no se comporta de la manera que el programador espera. En EdPy, si tiene un error lógico en su programa, el programa aún se descargará a Edison. Sin embargo, cuando ejecute el programa, no se comportará de la manera que cree que debería.

Un error lógico puede ser tan simple como usar la función incorrecta o dejar de lado una función. Un ejemplo de un error lógico sería escribir un programa que utilice la detección de obstáculos pero que no encienda el haz de detección de obstáculos.

Mira el siguiente programa:

```
1
2  #-------------Setup----------------
3
4  import Ed
5
6  Ed.EdisonVersion = Ed.V2
7
8  Ed.DistanceUnits = Ed.CM
9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #--------Your code below-----------
12
13 Ed.ObstacleDetectionBeam(Ed.ON)
14
15 Ed.Drive(Ed.FORWARD,Ed.SPEED_5, Ed.DISTANCE_UNLIMITED)
16
17 While Ed.ReadObstacleDetection() != Ed.OBSTACLE_NONE
18 pass
19 Ed.Drive(Ed.SPIN_RIGHT,Ed.SPEED_5,145)
20 Ed.Drive(Ed.FORWARD, Ed.SPEED_5,20)
21
```

El programa está diseñado para hacer que el robot Edison conduzca hasta que encuentre un obstáculo, luego gire 135° y se aleje del obstáculo 20 cm.

Hay cinco errores en el programa. ¿Puede identificar los cinco errores en el programa y determinar si son errores de sintaxis o errores lógicos? Escribe tus respuestas en la tabla de la página siguiente.

Nombre_____

Sugerencia: puede escribir este programa en EdPy y usar el botón 'Verificar código' para ayudarlo a encontrar los errores de sintaxis.

Nº de error	Nº de línea	Tipo de error (sintaxis o lógico)	Error de descripción
1			
2			
3			
4			
5			

Nombre_____

Lección 7: Hoja de trabajo 7.4 – Detectar un obstáculo como evento

En esta actividad, escribirá un programa basado en eventos para hacer que su robot Edison avance continuamente mientras evita obstáculos.

Programación impulsada por eventos

En programación, un evento es algo que ocurre fuera del código del programa y que afecta la forma en que se ejecuta el programa. Un evento puede ser la presión de un botón o la transmisión de información desde un sensor.

Muchos lenguajes de programación, incluido Python, permiten a los programadores crear código que puede responder a algún conjunto de eventos. Este tipo de programación se llama 'programación dirigida por eventos'.

Cada vez que escriba un programa dirigido por eventos, también necesitará escribir código que maneje estos eventos. Hay dos formas principales de hacer esto, haciendo que el programa espere en un ciclo o usando interrupciones.

Interrumpimos esta lección para hablar de... interrupciones Como ya sabe, los programas generalmente se mueven secuencialmente a través del código, línea por línea. Sin embargo, hay formas de permitir que el código se mueva de manera diferente, como mediante el uso de bucles.

También puede afectar la forma en que se ejecuta un programa mediante el uso de una 'interrupción'.

Una interrupción es una sección de código que pausa el programa principal y se ejecuta solo. Una vez que se completa el código de interrupción, el programa vuelve a donde lo dejó en el programa principal.

Las interrupciones son siempre funciones definidas en alguna parte del programa. Por lo general, son secciones cortas de código diseñadas para realizar una tarea específica sin causar una interrupción importante en el flujo del programa principal.

Los programadores usan interrupciones porque las interrupciones permiten que un programa reaccione a un evento en cualquier momento mientras el programa se está ejecutando. En otras palabras, mediante el uso de interrupciones, un programador no tiene que predecir exactamente cuándo ocurrirá el evento durante un programa.

Controladores de eventos e interrupciones

Cuando use interrupciones en la programación dirigida por eventos, necesitará usar 'controladores de eventos'.

Un controlador de eventos es una forma de vincular una interrupción a un evento específico.

Para usar un controlador de eventos, primero debe configurar o "registrar" el controlador de eventos en su código. Una vez registrado, el controlador de eventos monitorea constantemente su evento dado.
Cada vez que ocurre ese evento, el controlador de eventos activa la interrupción, que llama y ejecuta la función, luego regresa al programa principal.

Nombre_____

Mira el siguiente programa:

```
1
2  #-------------Setup---------------
3
4  import Ed
5
6  Ed.EdisonVersion = Ed.V2
7
8  Ed.DistanceUnits = Ed.CM
9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #--------Your code below-----------
12 Ed.ObstacleDetectionBeam(Ed.ON)
13 Ed.RegisterEventHandler(Ed.EVENT_OBSTACLE_AHEAD, "avoidObstacle")
14
15 while True:
16     Ed.Drive(Ed.FORWARD,Ed.SPEED_5,Ed.DISTANCE_UNLIMITED)
17
18 def avoidObstacle():
19     Ed.Drive(Ed.SPIN_RIGHT,Ed.SPEED_5,180)
20     Ed.ReadObstacleDetection()
21
```

En EdPy, usamos la función 'Ed.RegisterEventHandler' para registrar un controlador de eventos.

La función 'Ed.RegisterEventHandler' tiene dos parámetros. El primer parámetro es el evento que va a ocurrir y el segundo parámetro es la función que se llamará cada vez que ocurra ese evento.

Mira la línea 13 del programa. Esta línea está registrando un controlador de eventos para que cada vez que ocurra 'EVENT_OBSTACLE_AHEAD', se llame a la función 'avoidObstacle'.

Tu turno:

Escriba el programa usando la aplicación EdPy y descárguelo a su robot Edison. Luego ejecute el programa para ver cómo funciona.

1. Describe lo que hace el robot.

_____

_____

_____

_____

Nombre_____

2. En este programa, cuando el robot detecta un obstáculo adelante, ¿qué líneas de código se ejecutarán? ¿Por qué el programa ejecuta estas líneas?

_____

_____

_____

_____

3. ¿Por qué se incluye Ed.ReadObstacleDetection en este programa? En otras palabras, ¿qué está haciendo la línea 20? Sugerencia: consulte la hoja de trabajo 2.5 para obtener una pista o intente eliminar la línea 20 y ejecute el programa.

_____

_____

_____

_____

¡Intentalo!

¿Qué más podría hacer Edison cuando detecta un obstáculo adelante? Intente modificar el código para que Edison realice un comportamiento diferente cuando detecte un obstáculo más adelante. Experimente con el programa para ver qué funciona y qué no.

Nombre_____

Lección 7: Hoja de trabajo 7.5 – Obstáculo derecho e izquierdo detección

En esta actividad, escribirá un programa para que Edison reaccione ante obstáculos a la izquierda o derecha del robot. Para hacer esto, usaremos 'sentencias if'.

Si declaraciones

Una parte importante de la codificación es tomar decisiones. La forma más común de hacer esto es usar una 'sentencia if'.

Una 'sentencia if' pregunta si una condición es verdadera o falsa. Si el resultado es verdadero, entonces el programa ejecuta el bloque de sentencias que sigue a la sentencia if. Si el resultado es falso, el programa ignora las declaraciones dentro de la declaración if y pasa a la siguiente línea de código fuera de la declaración if.

Mira el siguiente programa:

```
1
2  #-------------Setup----------------
3
4  import Ed
5
6  Ed.EdisonVersion = Ed.V2
7
8  Ed.DistanceUnits = Ed.CM
9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #--------Your code below-----------
12 Ed.ObstacleDetectionBeam(Ed.ON)
13
14 while True:
15     if Ed.ReadObstacleDetection() != Ed.OBSTACLE_NONE:
16         Ed.PlayBeep()
17         Ed.ReadObstacleDetection()
18
```

Este programa usa una declaración if para darle al robot la capacidad de tomar decisiones sin la guía humana. Cuando un robot puede tomar decisiones por sí mismo de esta manera, se le llama robot autónomo.

Tu turno:

Tarea 1: pitido si hay un obstáculo

Escriba el programa anterior usando la aplicación EdPy, descárguelo a su robot Edison y ejecute el programa. Luego intente mover un obstáculo, como su mano, dentro y fuera del haz de detección de obstáculos de Edison para ver qué sucede.

Nombre_____

1. Edison ahora puede reaccionar de manera diferente a diferentes estímulos, tomando una 'decisión' sobre qué hacer. ¿Significa esto que Edison tiene inteligencia? Sugerencia: es posible que desee investigar un poco sobre la inteligencia artificial para ayudarlo a decidir.

_____

_____

_____

_____

_____

Tarea 2: emite un pitido si hay un obstáculo, de lo contrario enciende la luz

Además de decirle a un programa qué hacer cuando una condición de declaración if es verdadera, también puede decirle a un programa qué hacer si esa condición es falsa.

If, else

El uso de sentencias if con 'else' nos permite escribir programas tomando decisiones más complicadas. Las declaraciones if/else son básicamente una forma de tomar una decisión entre dos cosas.

Vea a Bill Gates, fundador de Microsoft, explicar las declaraciones if/else y la toma de decisiones en la programación: https://youtu.be/fVUL-vzrIcM

En Python, la sintaxis de if/else es:

```
if expresión:
    sentencia(s) else:

sentencia(s)
```

El programa se mueve secuencialmente de arriba hacia abajo, comenzando con la condición if. Si la instrucción if es verdadera, el programa ejecuta el código sangrado para la expresión if y omite el else. Sin embargo, si la declaración if es falsa, el programa omite esa sección del código sangrado y ejecuta el código sangrado 'else' en su lugar.

Nombre_____

Escribe el siguiente programa:

```
1
2  #-------------Setup----------------
3
4  import Ed
5
6  Ed.EdisonVersion = Ed.V2
7
8  Ed.DistanceUnits = Ed.CM
9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #--------Your code below-----------
12 Ed.ObstacleDetectionBeam(Ed.ON)
13
14 while True:
15     if Ed.ReadObstacleDetection() != Ed.OBSTACLE_NONE:
16         Ed.LeftLed(Ed.OFF)
17         Ed.PlayBeep()
18         Ed.TimeWait(100, Ed.TIME_MILLISECONDS)
19         Ed.ReadObstacleDetection()
20     else:
21         Ed.LeftLed(Ed.ON)
22
```

Descarga y ejecuta el programa. Intente mover un obstáculo dentro y fuera del haz de detección de obstáculos de Edison para ver qué sucede.

Este programa tiene dos vías: una para si se detecta un obstáculo y otra para si no se detecta ningún obstáculo.

If, elif, else

También puede crear un programa que tome una decisión usando más de dos condiciones. Para hacer esto, usa otra estructura de sintaxis de Python:

if expresión:
sentencia(s) elif
expresión: sentencia(s)
else:

declaraciones)

'Elif' es como dices 'else if' en Python. Puede usar elif para escribir un programa con múltiples condiciones if.

Un programa que usa if/elif/else todavía se mueve secuencialmente de arriba hacia abajo. Una vez que el programa ejecuta cualquier código sangrado dentro de cualquier parte de la estructura de la instrucción if, omitirá el resto de la estructura y pasará a la siguiente línea de código fuera de la estructura.

Esto significa que si la declaración if en la parte superior es verdadera, el programa ejecuta el código sangrado para la expresión if y omite cualquier sección elif, así como la sección else si la hay. Sin embargo, si la instrucción if es falsa, el programa omite esa sección de código sangrado y pasa a la primera sección elif.

Nombre_____

Nuevamente, si la primera condición elif es verdadera, el programa ejecuta su código sangrado y omite todo lo que está debajo de él en la estructura de la instrucción if (cualquier otro elif y la condición else, si la hay). Si esta condición elif es falsa, el programa pasa a la siguiente parte de la estructura de la instrucción if y así sucesivamente.

Tarea 3: Detectar un obstáculo a la izquierda o a la derecha

Mira el siguiente programa:

```
1
2  #-------------Setup----------------
3
4  import Ed
5
6  Ed.EdisonVersion = Ed.V2
7
8  Ed.DistanceUnits = Ed.CM
9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #--------Your code below-----------
12 Ed.ObstacleDetectionBeam(Ed.ON) #turn on obstacle detection
13
14 while True:
15     Ed.Drive(Ed.FORWARD, Ed.SPEED_1, Ed.DISTANCE_UNLIMITED)
16
17     obstacle=Ed.ReadObstacleDetection()
18
19     if obstacle>Ed.OBSTACLE_NONE: #there is an obstacle
20         Ed.Drive(Ed.BACKWARD, Ed.SPEED_5, 7)
21         if obstacle==Ed.OBSTACLE_LEFT:
22             Ed.Drive(Ed.SPIN_RIGHT, Ed.SPEED_5, 90)
23         elif obstacle==Ed.OBSTACLE_RIGHT:
24             Ed.Drive(Ed.SPIN_LEFT, Ed.SPEED_5, 90)
25         elif obstacle==Ed.OBSTACLE_AHEAD:
26             Ed.Drive(Ed.SPIN_RIGHT, Ed.SPEED_5, 180)
27         Ed.ReadObstacleDetection() #clear any unwanted detections
28
```

Este programa tiene tres caminos diferentes que puede tomar cuando se detecta un obstáculo en función de dónde se encuentra el obstáculo detectado en relación con Edison.

Escriba el programa usando la aplicación EdPy y descárguelo a su robot Edison. Luego ejecute el programa para ver cómo funciona.

2. Cuando ejecute este programa, explique con sus propias palabras qué hace el robot cuando:

Obstáculo detectado adelante: _____

Obstáculo detectado a la derecha: _____

Obstáculo detectado a la izquierda: _____

Nombre_____

Lección 8: Hoja de trabajo 8.1 – Sensor de seguimiento de línea

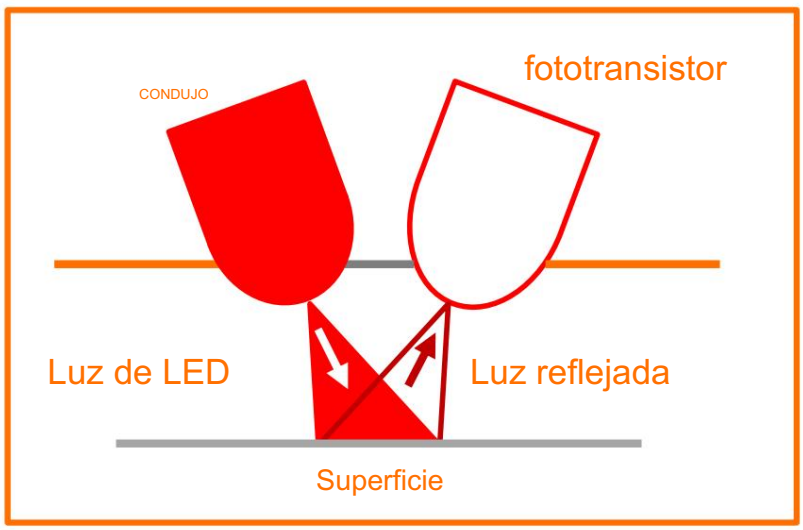
En esta actividad, aprenderá más sobre el sensor de seguimiento de línea del robot Edison y cómo Edison puede usar este sensor para determinar si está en una superficie reflectante o no reflectante.

¿Cómo funciona el sensor de seguimiento de línea de Edison?

Su robot Edison está equipado con un sensor de seguimiento de línea, ubicado cerca del interruptor de encendido en la parte inferior del robot. Este sensor se compone de dos componentes electrónicos principales:

1. un diodo emisor de luz roja (LED) y 2. un fototransistor (sensor de luz).

Esta imagen representa una sección transversal del sensor de seguimiento de línea de Edison:



El LED del sensor de seguimiento de línea ilumina la superficie que conduce el robot Edison en.

El componente fototransistor es un sensor de luz. El fototransistor mide la cantidad de luz que se refleja desde la superficie debajo de Edison.

Tu turno:

Tarea 1: negro contra blanco

Cuando se refleja más luz de regreso al fototransistor de Edison, da una lectura de luz más alta.

Experimente para ver si una superficie blanca o negra reflejará más luz.

Use la hoja de actividad 8.1 o una hoja de papel blanco y una hoja negra. Encienda Edison y presione el botón redondo dos veces para que se encienda el LED de seguimiento de línea. Levante ligeramente a Edison del papel y mire de cerca el punto redondo de luz que el LED brilla en la superficie. Compara qué tan brillante aparece el punto de luz cuando se coloca sobre una superficie negra y luego sobre una superficie blanca.

1. ¿El punto de luz LED parece más brillante cuando se coloca sobre una superficie negra o blanca?

_____

Nombre_____

Tarea 2: rojo, verde y azul

Como viste en la tarea 1, se refleja más luz en una superficie blanca que en una superficie negra. Es por eso que la luz parece más brillante en una superficie blanca.

Cuando el fototransistor está sobre una superficie blanca, da una lectura de luz más alta que cuando está sobre una superficie negra. Por lo tanto, una superficie negra se considera "no reflectante" y una superficie blanca se considera "reflectante".

La capacidad del fototransistor para determinar si Edison está sobre una superficie reflectante o no reflectante es lo que permite que el robot se programe para responder a la superficie sobre la que se conduce.

2. Piense en cómo respondería el rastreador de líneas a cada uno de los siguientes colores de superficie. ¿El rastreador de línea vería cada color como reflectante o no reflectante?

   Recuerde, el LED del rastreador de línea de Edison emite una luz roja. (Sugerencia: puede probar sus respuestas usando la hoja de actividades 8.1.)

   superficie roja _____

   superficie verde _____

   superficie azul _____

Video: los humanos no necesitan postularse

El seguimiento de línea es un comportamiento robótico muy básico que se utiliza en muchas fábricas y almacenes automatizados en la actualidad. ¿Será igual en el futuro? ¿Cómo será el lugar de trabajo del futuro con más robots?
¿Existirá siquiera?

Mire el video Humans Need Not Apply para obtener más información sobre lo que podría significar la adopción generalizada de robots para el futuro de los humanos en el trabajo: https://www.youtube.com/watch?v=7Pq-S557XQU (15 minutos)

Nombre_____

Lección 8: Hoja de trabajo 8.2 – Conducir hasta una línea negra

En esta actividad, escribirá un programa para que su robot Edison avance sobre una superficie blanca (reflectante) hasta que se cruce una línea negra (no reflectante).

Mira el siguiente programa:

```
1
2  #-------------Setup----------------
3
4  import Ed
5
6  Ed.EdisonVersion = Ed.V2
7
8  Ed.DistanceUnits = Ed.CM
9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #--------Your code below-----------
12
13 Ed.LineTrackerLed(Ed.ON)
14
15 Ed.Drive(Ed.FORWARD,Ed.SPEED_6,Ed.DISTANCE_UNLIMITED)
16
17 while True:
18     if Ed.ReadLineState() == Ed.LINE_ON_BLACK:
19         Ed.PlayBeep()
20         Ed.Drive(Ed.STOP,Ed.SPEED_6,0)
```

Mire la línea 13. Esta línea llama a la función Ed.LineTrackerLed() y cambia el estado a 'on'.

Al igual que con el haz de detección de obstáculos de Edison, para usar el sensor de seguimiento de línea en un programa, primero debe encender el sensor. Al encender el sensor de seguimiento de línea, también se activará el LED rojo del rastreador de línea.

Ahora, observe la línea 19. Esta línea llama a la función Ed.PlayBeep(). Esta línea no afecta la forma en que funciona el programa de seguimiento de línea. En cambio, el propósito de esta línea es para la depuración.

Depuración La

depuración es el proceso de encontrar 'bugs' o errores en su programa. A menudo, los programadores colocarán líneas como la línea 19 en este programa en su código para realizar un seguimiento del flujo del programa.

Digamos que ejecuta su programa, pero el robot no se detiene en la línea negra. Hay dos razones posibles: (1) es posible que el robot no esté detectando la línea negra o (2) podría haber un error en el comando final Ed.Drive().

Si escuchamos el pitido, sabríamos que se detectó la línea negra. Por lo tanto, sabemos que el error estuvo en el siguiente comando. Este código de depuración adicional nos ayuda a determinar el error más fácilmente.

También se pueden usar otras funciones para la depuración, como el comando Ed.LeftLed(). Por ejemplo, puede usar este comando para encender el LED izquierdo para indicar que se ha llegado a cierto punto en el programa.

Nombre_____

Tu turno:

Escriba el programa usando la aplicación EdPy y descárguelo a su robot Edison. Use la línea negra en la hoja de actividades 8.1 para probar el programa. También puede dibujar una línea negra en una hoja de papel blanco o usar cinta aislante negra en un escritorio blanco.

> Nota: Siempre que utilice el sensor de seguimiento de línea de Edison en un programa, siempre inicie el robot en la superficie blanca (reflectante), nunca en la superficie negra (no reflectante).

Coloque a Edison en la superficie blanca y conduzca el robot hacia la línea negra.

Luego intente ejecutar el programa nuevamente usando cada una de las tres líneas de colores en la hoja de actividades 8.1, una a la vez. Conduce a Edison hacia cada línea de color para comprobar si el robot detectará o no la línea y se detendrá.

1. ¿Hay colores que Edison no puede detectar muy bien? Si es así, ¿qué color(es)?

   _____

2. ¿Por qué crees que obtuviste la respuesta que obtuviste para la pregunta número 1? ¿Por qué no puedo? ¿Edison detectó ese(s) color(es)?

_____

_____

_____

3. Imagina que estás programando tu robot Edison para conducir una carrera de slalom con tres banderas de slalom. Describa cómo podría utilizar las funciones Ed.PlayBeep(), Ed.LeftLed() o Ed.RightLed() en un programa con fines de depuración y qué harían.

_____

_____

_____

_____

_____

_____

_____

Nombre_____

Lección 8: Hoja de trabajo 8.3 – Conducir dentro de un borde

En esta actividad, debe escribir un programa que mantendrá a Edison dentro de un borde negro utilizando el sensor de seguimiento de línea del robot.

Primero, necesitamos planificar el programa usando pseudocódigo.

Pseudocódigo

La planificación de su programa antes de comenzar a codificar es una habilidad importante y útil en la programación.

Una forma de hacerlo es mediante el uso de un diagrama de flujo para resumir el flujo de su programa, que aprendió en la lección 6. El pseudocódigo es otra forma de representar su programa antes de comenzar a codificar.

El pseudocódigo es una forma de escribir un programa de forma simplificada y fácil de leer.
El pseudocódigo se parece a un lenguaje de programación simplificado, pero no se basa en ningún lenguaje de programación específico, por lo que no tiene sintaxis. En cambio, el pseudocódigo usa el idioma inglés para describir lo que hará el programa. Es por eso que el pseudocódigo también se llama 'inglés estructurado'.

Cuando planifique su programa usando pseudocódigo, debe sangrar de la misma manera que lo haría en el lenguaje de programación para que el pseudocódigo sea fácil de leer y comprender.

Aquí hay un ejemplo de un pseudocódigo que describe un programa que hace que el robot Edison permanezca dentro de un borde no reflectante usando el sensor de seguimiento de línea:

```
Encienda el rastreador de línea
bucle para siempre
    Impulsar
    Si el robot detecta una línea negra, entonces
        Detener
        reproducir un pitido
        Girar a la derecha 135°
```

Aquí está el código correspondiente en EdPy:

```
11  #--------Your code below-----------
12
13  Ed.LineTrackerLed(Ed.ON)
14
15  while True:
16      Ed.Drive(Ed.FORWARD, Ed.SPEED_3, Ed.DISTANCE_UNLIMITED)
17      if Ed.ReadLineState() == Ed.LINE_ON_BLACK:
18          Ed.Drive(Ed.STOP, Ed.SPEED_3, Ed.DISTANCE_UNLIMITED)
19          Ed.PlayBeep()
20          Ed.Drive(Ed.SPIN_RIGHT, Ed.SPEED_5, 135)
21
```

Compara el pseudocódigo con el programa. ¿Ves cómo se tradujo el pseudocódigo en el programa Python?

Nombre_____

Tu turno:

Escribe el siguiente programa:

```
1
2  #-------------Setup----------------
3
4  import Ed
5
6  Ed.EdisonVersion = Ed.V2
7
8  Ed.DistanceUnits = Ed.CM
9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #--------Your code below-----------
12
13 Ed.LineTrackerLed(Ed.ON)
14
15 while True:
16     Ed.Drive(Ed.FORWARD, Ed.SPEED_3, Ed.DISTANCE_UNLIMITED)
17     if Ed.ReadLineState() == Ed.LINE_ON_BLACK:
18         Ed.Drive(Ed.STOP, Ed.SPEED_3, Ed.DISTANCE_UNLIMITED)
19         Ed.PlayBeep()
20         Ed.Drive(Ed.SPIN_RIGHT, Ed.SPEED_5, 135)
21
```

Descargue el programa a su robot Edison y ejecútelo para ver cómo funciona.

Use la hoja de actividad 8.2 como borde para probar el programa. También puede crear su propio borde usando una hoja grande de papel y un marcador negro grueso o usar cinta aislante negra en un escritorio blanco o en el piso para crear un borde grande.

Modifique el programa:
intente eliminar la parada eliminando la línea 18 del programa. Luego experimente cambiando el programa para usar diferentes velocidades. Pruebe sus programas modificados para ver qué sucede.

1. ¿Qué tan rápido puede ir el robot antes de que haya problemas?

   _____

2. ¿Qué sucede cuando el robot conduce demasiado rápido?

_____

_____

_____

Nombre_____

Lección 8: Hoja de trabajo 8.4 – Seguir una línea

En esta actividad, utilizará el rastreador de línea del robot Edison para escribir un programa en EdPy que haga que Edison siga cualquier línea negra.

Para hacer esto, primero necesitamos crear un algoritmo para seguir una línea negra.

¿Qué es un algoritmo?

Digamos que quieres enseñar a tus amigos cómo hacer pasteles de frutas. Si sabes que todos tus amigos tienen manzanas, puedes escribir una receta de pastel de manzana.

Sin embargo, es posible que no todos tus amigos tengan manzanas. ¿Qué pasa si uno de tus amigos tiene arándanos, otro tiene cerezas y un tercero tiene manzanas? No todos pueden seguir la receta de la tarta de manzana. Tendrías que escribir una receta separada para cada fruta diferente.

¿Qué pasa si no sabes qué fruta tiene cada uno de tus amigos? ¿Cómo podrías enseñarles a hacer pasteles de frutas?

No importa qué fruta tengan, todos tus amigos deben seguir las mismas instrucciones básicas: hacer la masa, llenar el pastel con la fruta y luego hornear el pastel.

Este nuevo conjunto de instrucciones es un ejemplo de un algoritmo.

Un algoritmo es un amplio conjunto de instrucciones para resolver un conjunto de problemas. Un algoritmo establece un proceso o conjunto de reglas a seguir para resolver cualquier problema en el conjunto.

Algoritmos en programación

En programación, a menudo tenemos conjuntos de problemas que queremos resolver.

En esta actividad, por ejemplo, queremos que Edison siga cualquier línea negra. Nuestro conjunto de problemas, por lo tanto, es 'siga cualquier línea negra'. Cualquier línea específica que hagas para que Edison la siga es un problema nuevo dentro de este conjunto.

Digamos que dibujas una línea negra para que la siga Edison. Para que Edison siga su línea, puede escribir un programa que haga que Edison conduzca por la ruta exacta de la línea. Sin embargo, si crea una nueva línea, deberá escribir un programa completamente nuevo para esa nueva línea.

En su lugar, puede crear un algoritmo.

El algoritmo produce un programa que funcionará para todos los problemas del conjunto. De esta manera, no se necesita un programa completamente nuevo para cada problema nuevo.

Para resolver nuestro conjunto de problemas, necesitamos que el algoritmo produzca un conjunto de instrucciones que funcionen para cualquier línea negra.

Puede planificar un algoritmo usando pseudocódigo o un diagrama de flujo, tal como lo hace cuando planifica un programa.

Nombre_____

Aquí hay un algoritmo en pseudocódigo que le permitirá a Edison seguir cualquier línea negra:

```
Encienda el rastreador de línea
bucle para siempre
    Si el robot detecta una línea negra, entonces
        Conducir adelante a la derecha
    Demás
        Conducir adelante a la izquierda
```

Este algoritmo dice que si el sensor de seguimiento de línea está en una superficie no reflectante (negra), entonces el robot Edison debe avanzar hacia la derecha, moviendo el sensor hacia la superficie blanca. Si el sensor de seguimiento de línea no está en una superficie negra, entonces el robot debe avanzar hacia la izquierda, moviendo el sensor hacia la superficie negra. De esta forma, el robot avanza continuamente y sigue el borde de la línea.

Observe que no se mencionan velocidades ni distancias en el pseudocódigo. Ese tipo de detalles generalmente se dejan en la etapa de codificación.

Tu turno:

Traduzca el algoritmo de pseudocódigo a un programa de Python para hacer que su robot Edison siga cualquier línea negra. Experimente con diferentes velocidades y distancias para lograr el seguimiento de línea más suave. Descargue su código en Edison y pruébelo usando la pista en la hoja de actividad 8.2.

1. ¿Cuál fue la mejor combinación de velocidad y distancia para lograr una línea suave?
   seguimiento que encontraste?

_____

2. ¿Cómo se veía su programa en Python? Escriba su código aquí.

_____

_____

_____

_____

_____

_____

_____

_____

¡Intentalo!

Nombre_____

Haz tu propia línea con un marcador negro sobre papel blanco o cinta adhesiva negra. ¿Puede Edison seguir tu línea?

Nombre _____

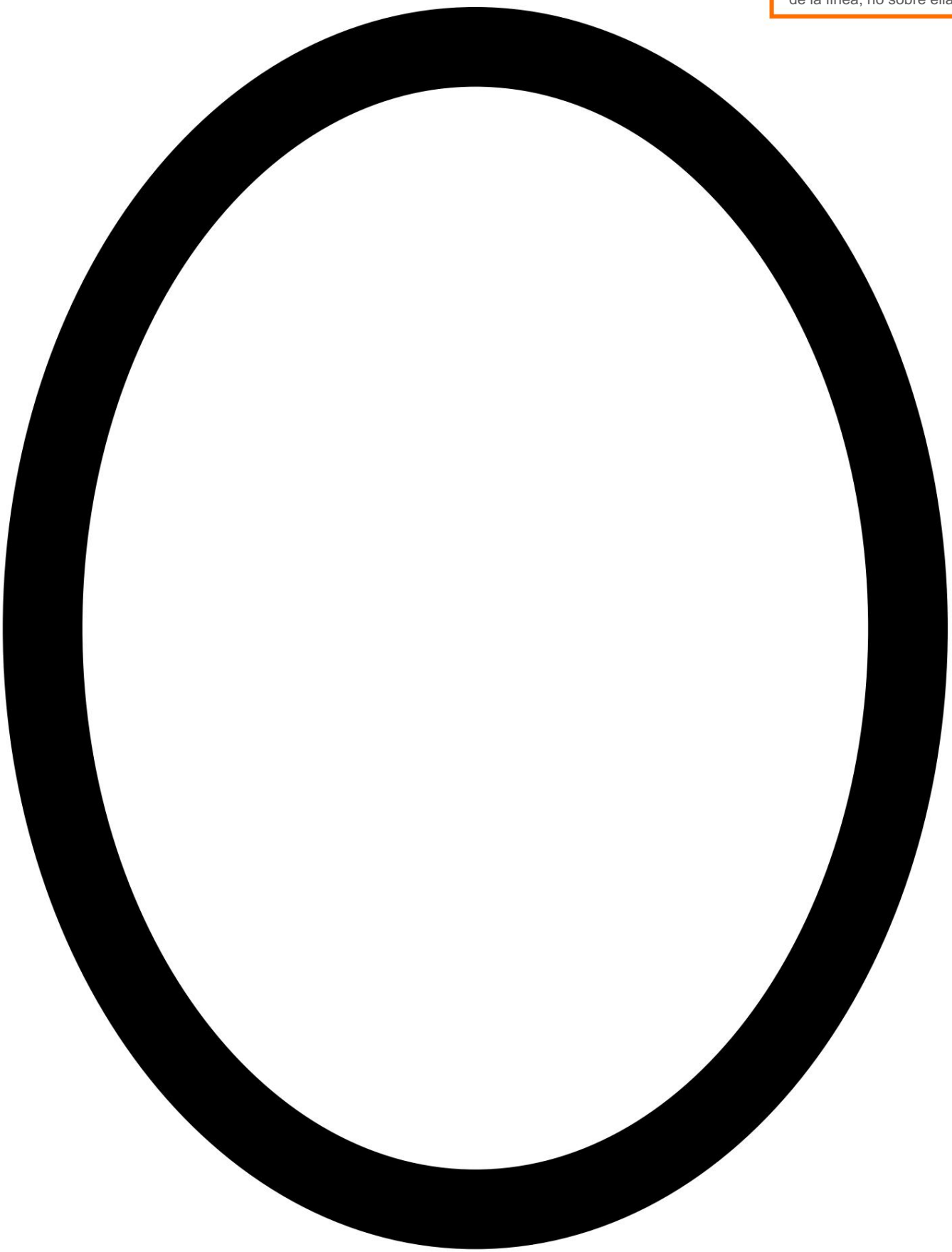
Lección 8: Hoja de actividades 8.1

Nombre_____

¡Recordar!

Siempre ponga en marcha el robot al lado de la línea, no sobre ella.

Lección 8: Hoja de actividades 8.2



Nombre_____

Lección 9: Hoja de trabajo 9.1 – Alarma de luz

En esta actividad, escribirá un programa para hacer que su robot Edison haga sonar una alarma cuando se enciendan las luces de la habitación.

Uso de los sensores de luz de Edison en los programas

Su robot Edison tiene dos sensores de luz, uno a la izquierda y otro a la derecha, que pueden detectar la luz visible. Podemos usar estos sensores para programar a Edison para que responda a la luz.

En un programa, podemos hacer que estos sensores lean la cantidad de luz detectada y la devuelvan como un valor digital.

Mira el siguiente programa:

```
1
2  #-------------Setup----------------
3
4  import Ed
5
6  Ed.EdisonVersion = Ed.V2
7
8  Ed.DistanceUnits = Ed.CM
9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #--------Your code below-----------
12
13 while Ed.ReadLeftLightLevel() < 100:
14     pass
15 while True:
16     Ed.PlayBeep()
17
```

Este programa usa la función Ed.ReadLeftLightLevel() en la línea 13. Esta función le dice al programa que lea el nivel de luz del sensor de luz izquierdo en Edison y devuelva un valor entre 0 y 1023.

Debido a que la función Ed.ReadLeftLightLevel() devuelve un valor, podemos hacer operaciones matemáticas con ese valor. El primer bucle de este programa utiliza las matemáticas para determinar qué hacer. Este primer bucle dice pasar (en otras palabras, no hacer nada) mientras que el valor devuelto por la función Ed.ReadLeftLightLevel() es 'menor que' (<) 100.

Cuando el valor devuelto es mayor que (>) 100, el programa sale del primer ciclo y pasa al siguiente ciclo, que suena una alarma continuamente.

Tu turno:

Escriba el programa y descárguelo a su robot Edison. Coloque el robot Edison en la oscuridad antes de presionar el botón de reproducción.

Una vez que haya apagado las luces o haya bloqueado el sensor de luz izquierdo de Edison, presione el botón de reproducción. Cuando se encienden las luces, o se retira lo que sea que esté bloqueando el sensor de luz izquierdo de Edison, el robot activará la alarma.

Nombre_____

1. Piense en una situación de la vida real en la que este tipo de alarma sería útil. Describe el situación y cómo se podría utilizar la alarma.

_____

_____

_____

_____

2. ¿Qué cambios deben hacerse en el programa para convertirlo en una alarma oscura?

_____

_____

_____

_____

Nombre_____

Lección 9: Hoja de trabajo 9.2 – Luces automáticas En esta actividad, escribirá un programa para que su robot Edison encienda las dos luces LED cuando oscurezca.

Mira el siguiente programa:

```
1
2  #-------------Setup----------------
3
4  import Ed
5
6  Ed.EdisonVersion = Ed.V2
7
8  Ed.DistanceUnits = Ed.CM
9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #--------Your code below-----------
12 Ed.Drive(Ed.FORWARD, Ed.SPEED_5, Ed.DISTANCE_UNLIMITED)
13
14 while True:
15     if Ed.ReadLeftLightLevel() < 100:
16         activateBothLights(Ed.ON)
17     else:
18         activateBothLights(Ed.OFF)
19
20
21 def activateBothLights(stateOfLed):
22     Ed.LeftLed(stateOfLed)
23     Ed.RightLed(stateOfLed)
24
```

En este programa, estamos usando el símbolo 'menor que' (<) para determinar la ruta que tomará el programa.

Si el valor devuelto por la función Ed.ReadLeftLightLevel() es inferior a 100, se llama a la función activateBothLights() con el parámetro de entrada de Ed.ON. De lo contrario, el programa pasa a la parte 'else' de la sentencia if, que también llama a la función activarBothLights(), pero con el parámetro de entrada Ed.OFF.

Tu turno:

Escriba el programa y descárguelo a su robot Edison. Deberá crear algunos "túneles" para que Edison los atraviese y bloquearán la luz para ver cómo se encienden las luces delanteras de Edison.

Ejecute el programa con Edison conduciendo dentro y fuera de los túneles para ver cómo funciona.

Luego intente experimentar con el valor (100) en la declaración if en la línea 15 para ver qué sucede cuando ejecuta el programa con un número mayor o menor.

Nombre_____

1. ¿Qué sucede cuando haces que el valor en la instrucción if sea más alto?

_____

2. ¿Qué sucede si reduce el valor de la instrucción if?

_____

¡Intentalo!

¿Hará alguna diferencia si usa la función ReadRightLightLevel() en lugar de ReadLeftLightLevel()? Modifique su programa con este cambio y pruébelo para ver.

Nombre_____

Lección 9: Hoja de trabajo 9.3 – Seguimiento de la luz En esta actividad, escribirá un programa para que su robot Edison siga la luz de una antorcha (linterna).

Mira el siguiente programa:

```
#-------------Setup----------------

import Ed

Ed.EdisonVersion = Ed.V2

Ed.DistanceUnits = Ed.CM
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------

while True:
    if Ed.ReadRightLightLevel() - Ed.ReadLeftLightLevel() < 0:
        Ed.Drive(Ed.FORWARD_LEFT, Ed.SPEED_5, Ed.DISTANCE_UNLIMITED)
    else:
        Ed.Drive(Ed.FORWARD_RIGHT, Ed.SPEED_5, Ed.DISTANCE_UNLIMITED)
```

Este programa compara el nivel de luz entre el sensor de luz derecho y el sensor de luz izquierdo para determinar el flujo del programa.

La presencia de la antorcha a la izquierda o a la derecha del robot hará que el robot lea un nivel de luz más alto en ese lado del robot. La lógica de este programa dice que cuando el nivel de luz derecho menos el nivel de luz izquierdo es menor que cero, el robot se desplaza hacia la izquierda hacia la fuente de luz más alta, de lo contrario, el robot se desplaza hacia la derecha.

Tu turno:

Tarea 1: rastrear el programa

En la hoja de trabajo 5.2, aprendió sobre 'rastrear' un programa. Cuando hay muchos cálculos y diferentes valores posibles en un programa, puede ser útil rastrear el programa. Esto le permite comprender los diferentes valores que pueden ocurrir y predecir el comportamiento asociado.

1. Complete la siguiente tabla de seguimiento para este programa. El comportamiento esperado debe puede ser 'conducir hacia adelante a la derecha' o 'conducir hacia adelante a la izquierda'.

	Luz_derecha Luz_izquierda 200		Comportamiento esperado
Antorcha a la derecha	100		
Antorcha a la izquierda	100	200	
sin antorcha	100	100	

Nombre_____

Tarea 2: escribir y ejecutar el programa

Escriba el programa y descárguelo a su robot Edison. Después de presionar el botón de reproducción, encienda una antorcha en Edison. El robot conducirá hacia la luz. Usa la linterna para controlar por dónde conduce Edison.

2. ¿Qué sucede si cambia el símbolo 'menor que' (<) por un símbolo mayor que (>)? en este programa?

_____

_____

¿Qué opinas?

El programa de seguimiento de la luz anterior demuestra un comportamiento robótico que es muy similar al de una polilla atraída por una farola en una noche cálida.

¿Es inteligente una polilla que se siente atraída por la luz? ¿Qué pasa con un robot con el mismo comportamiento?

¿Por qué un insecto que es atraído por la luz está vivo, pero no un robot?

Nombre_____

Lección 10: Hoja de trabajo 10.1 – Robot vampiro En esta actividad basada en desafíos, aplicará todo lo que ha aprendido hasta ahora para crear un programa con muchos comportamientos interesantes para que su robot Edison realice mientras se transforma en un robot vampiro.

Para convertir tu Edison en un robot vampiro, usaremos código orientado a objetos.

Objetos y clases en Python Python es un lenguaje de programación orientado a objetos. El código orientado a objetos es una forma poderosa de reducir la complejidad dentro de un programa. Es por eso que muchos programadores usan código orientado a objetos en programas grandes y complicados.

En un lenguaje de programación orientado a objetos, en lugar de solo usar variables y funciones independientes, los programas también pueden usar 'objetos'.

Un objeto es una colección de funciones y variables que están relacionadas entre sí. Los objetos están definidos por una 'clase'.

Una clase (también llamada definición de clase) es como un plano que describe cómo hacer un objeto.

En un programa, puede crear muchos objetos diferentes de una clase y luego manipular estos objetos individualmente en su programa para diferentes propósitos. Todos los objetos creados a partir de la clase tendrán el mismo 'plano' cuando se creen, pero luego puede hacer cosas diferentes a diferentes objetos a lo largo de su programa.

Ejemplo: clase 'Forma' El ejemplo más común de una clase es una Forma.

Todas las formas, como cuadrados, triángulos y hexágonos, tienen atributos comunes. Por ejemplo, todas las formas tienen algún número de lados y algún número de vértices. También hay cosas comunes que haces con las formas, como encontrar el área o el perímetro de la forma.

Podemos usar estos elementos comunes para crear una clase para Shape en Python. Al crear la clase, estamos configurando un plan con solo los elementos comunes conocidos. Entonces podremos crear muchos objetos diferentes de esa clase, como 'cuadrado' y 'triángulo'.

La sintaxis para definir una clase en Python es 'clase' y luego el nombre que desea darle a esa clase, seguido de dos puntos (:). Por ejemplo:

```
forma de clase :
```

Cualquier cosa que incluya como código sangrado en las líneas siguientes está dentro de la definición de la clase.

Nombre_____

Mire el siguiente ejemplo de una clase para Shape en Python:

```
clase Forma: def
        __init__(self,x,y): self.x = x self.y
                = y


        def area(self): return
                self.x * self.y

        perímetro def (self): return 2
                * self.x + 2 * self.y
```

Esta clase dice que el modelo para cualquier objeto creado a partir de la clase Forma tendrá dos parámetros de entrada (x e y). La clase también define tres funciones: 'init', 'area' y 'perimeter'.

La función __init__ y 'self' Cada clase
debe tener siempre una función __init__.

La función __init__ es un código de inicio. El término 'init' significa 'inicialización'.

Cuando el programa necesita crear un objeto, el programa entra en la definición de clase y ejecuta la función __init__. Esta función contiene código que configura el objeto, lo que normalmente significa establecer los valores iniciales de las variables del objeto.

Las funciones en una clase siempre deben tener 'self' como su primer parámetro. Esto le dice al objeto creado que use sus propias funciones y variables.

Ejemplo: objeto 'rectángulo' Una
vez que hemos creado una definición de clase, podemos usarla para crear objetos en nuestro programa.

Digamos que queremos crear un objeto 'rectángulo' en la clase Forma.

En Python, cuando desea crear un objeto a partir de una clase, utiliza la sintaxis
nombre_objeto = nombre_clase(parámetros).

Mira el ejemplo de creación de objeto 'rectángulo':

```
rectángulo = Forma (100,45)
```

En un programa, esto llama a la función __init__, que se ejecuta, creando un objeto 'rectángulo' usando el modelo de la definición de la clase Shape. El objeto 'rectángulo' está configurado para que 100 sea el valor de entrada 'x' y 45 como su valor de entrada 'y'. (No es necesario que ingrese 'self'; esa es solo una referencia para que el programa sepa dónde buscar).

Una vez que el programa crea el objeto, puede acceder a las funciones de ese objeto definidas por la clase.

Nombre_____

Para llamar a una de estas funciones en Python, utiliza la sintaxis nombre_objeto.nombre_función(parámetros).

Por ejemplo:

```
rectángulo.área()
```

En este ejemplo, la función rectángulo.area() devolverá el valor del parámetro de entrada 'x' del objeto rectángulo multiplicado por su parámetro de entrada 'y'.

Una forma de usar esta función particular sería almacenar este valor en una variable en su programa. Por ejemplo:

```
AreaOfRectangle = rectángulo.area()
```

Configurando una clase en EdPy

En EdPy, podemos usar una clase para crear diferentes objetos en un programa para Edison.

Aquí hay una clase simple para un robot vampiro con una función __init__ y otras dos funciones: una para el comportamiento diurno del robot y otra para el comportamiento nocturno. La clase Vampiro también contiene una variable para la edad del robot:

```
class Vampire:

    def __init__(self,age):
        self.age = age

    def doDayTimeBehaviour(self):
        Ed.LeftLed(Ed.OFF)
        Ed.RightLed(Ed.OFF)
        Ed.PlayTone(Ed.NOTE_A_7, self.age)
        while Ed.ReadMusicEnd() == Ed.MUSIC_NOT_FINISHED:
            pass
        Ed.TimeWait(1, Ed.TIME_SECONDS) # for debugging purposes and as a placeholder

    def doNightTimeBehaviour(self):
        Ed.RightLed(Ed.ON)
        Ed.LeftLed(Ed.ON) # for debugging purposes and as a placeholder
```

Esta definición de clase te permitirá crear diferentes objetos Vampiro.

Luego puede usar este objeto Vampiro en un programa para su robot Edison.

Nombre_____

Escriba el siguiente código y verifique que comprende lo que está sucediendo en el programa:

```
#-------------Setup----------------

import Ed

Ed.EdisonVersion = Ed.V2

Ed.DistanceUnits = Ed.CM
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------

myEdisonVampire = Vampire(21)

while True:
    # if it is the daytime
    if Ed.ReadLeftLightLevel() > 100:
        myEdisonVampire.doDayTimeBehaviour()
    #it is night time
    else:
        myEdisonVampire.doNightTimeBehaviour()

class Vampire:

    def __init__(self,age):
        self.age = age

    def doDayTimeBehaviour(self):
        Ed.LeftLed(Ed.OFF)
        Ed.RightLed(Ed.OFF)
        Ed.PlayTone(Ed.NOTE_A_7, self.age)
        while Ed.ReadMusicEnd() == Ed.MUSIC_NOT_FINISHED:
            pass
        Ed.TimeWait(1, Ed.TIME_SECONDS) # for debugging purposes and as a placeholder

    def doNightTimeBehaviour(self):
        Ed.RightLed(Ed.ON)
        Ed.LeftLed(Ed.ON) # for debugging purposes and as a placeholder

```

Tu turno:

Tarea 1: diseñar los comportamientos de los vampiros

Ahora es el momento de diseñar algunos comportamientos más interesantes para su robot vampiro. ¡Ser creativo! Puede hacer que su robot reaccione a las pulsaciones de botones, aplausos y obstáculos. Considere usar el manejo de eventos en su programa. Puedes hacer múltiples objetos de vampiros con las edades que quieras. También puede modificar el programa base para que sus objetos tengan múltiples parámetros de entrada.

Piense en lo que desea incluir en su programa, luego trabaje en el diseño de su programa creando primero diagramas de flujo y luego pseudocódigo basado en los diagramas de flujo.

Nombre_____

Paso 1: Diagramas de flujo

Dibuje diagramas de flujo para los comportamientos diurnos y nocturnos, ya sea a mano o usando un programa como Google Slides o un programa de dibujo similar. Incluya sus diagramas de flujo aquí. Use papel adicional si lo necesita.

Nombre_____

Paso 2: Pseudocódigo

Traduzca sus diagramas de flujo a pseudocódigo. Escriba su pseudocódigo aquí. Use papel adicional si lo necesita.

Nombre_____

Tarea 2: escribe tu programa

Traduce tu pseudocódigo a código Python usando la aplicación EdPy. Descárguelo a su robot Edison y ejecute su programa para probarlo.

Tarea 3: Resolución de problemas

En la programación, es muy común que no todo salga como uno pensaba que saldría la primera vez. ¡Sin sudar! Recuerde esto: Thomas Edison fracasó 10,000 veces antes de inventar con éxito la bombilla. Entonces, ¡ sigue intentándolo!

Aprendemos de los problemas que encontramos, por lo que es un buen hábito tomarse un tiempo para pensar en lo que salió mal. Describe los problemas que tuviste, las ideas que se te ocurrieron para solucionarlos y las soluciones que encontraste en intentos posteriores. Utilice papel adicional, si es necesario.

Nombre_____

Tarea 4: Describa algunas de las estructuras de programación en su programa

Elija tres estructuras de programación diferentes que utilizó en su programa. Describa lo que hace cada uno a continuación.

1. Nombre de la estructura de programación:

   _____

   ¿Qué hace?

_____

_____

_____

_____

2. Nombre de la estructura de programación:

   _____

   ¿Qué hace?

_____

_____

_____

_____

3. Nombre de la estructura de programación:

   _____

   ¿Qué hace?

_____

_____

_____

_____

Tarea 5: Presenta tu programa

Demuestre su robot vampiro a su pareja, grupo o clase. Habla sobre tus ideas, el programa, los problemas que encontraste y cómo los resolviste.