Nombre:_____

Micropython para la hoja de referencia de EV3

Módulos de importación

- Estas son las **bibliotecas** que se usan para controlar su robot
- Bibliotecas: una colección de **módulos** que se usan para hacer que su código sea más simple y más fácil de usar
- Módulos: el código de otras personas que estamos robando (con permiso)no tiene comandos para que podamos realizar esas tareas de forma
- Python

```
from pybricks.hubs import EV3Brick # you need this to access the brick
from pybricks.ev3devices import Motor, UltrasonicSensor # essentially, just list every motor and sensor you will need to use in the program needed. Don't use spaces and capitalize the first letter of each word
from pybricks.parameters import Port # this is where information will communicate with the sensors
from pybricks.tools import wait # if it's being used, you need to put it
from pybricks.robotics import DriveBase # you need this to move properly
```

Variables

- Variable: un nombre que asignamos a otro comando o un elemento que tiene información que cambia regularmente como la **entrada** en un sensor
- En el código de su robot, necesita definir una variable para su ladrillo, su drivebase, y cada sensor y motor que tiene en su robot
- Puedes definir una variable en cualquier parte de su código antes de usarlo, pero es más útil hacerlo en la parte superior de su código para que pueda encontrarlos más fácilmente.
- Así es como defines una variable para tu ladrillo, es útil para cosas como pitidos

```
ev3 = EV3Brick() # Este es tu ladrillo
```

- Así es como defines las variables para tus motores

```
left_motor = Motor(Port.A) # Este es el motor izquierdo y el puerto
right_motor= Motor(Port.D) # Este es el motor derecho y el puerto
```

- Así es como define las variables para sus sensores

```
US = UltrasonicSensor(Port.S1)
```

```
CS = ColorSensor(Port.S2)
TS = TouchSensor(Port.S3)
GS = GyroSensor(Port.S4)
```

La base de movimiento

- La base de movimiento es lo que controla los movimientos de su robot, es una variable muy importante que utiliza su motor izquierdo y derecho para hacer que tu robot gire y se mueva.
  `drivebase = DriveBase(left_motor, right_motor, wheel_diameter = , axle_track = )` *# Este es su comando drivebase.*
- Es necesario decirle al comando Drive Base los motores de su robot, sin él, el robot no podrá moverse en ninguna dirección.
- El **diámetro de la rueda** es la distancia entre dos lados opuestos de las ruedas del robot en milímetros. El robot utiliza esta medida para asegurarse de que se mueve a la distancia correcta.
- La **de la pista del eje** es lo que le permite a su robot hacer giros precisos, sin ella, su robot no podrá girar correctamente. Se mide encontrando la distancia entre el punto central de sus dos ruedas en milímetros.

Controles

- Los controles de movimiento son los comandos que le das al robot para que se mueva.
- Le permiten hacer cosas como avanzar y retroceder, detenerse y girar.
- La mayoría de los comandos de movimiento se formatean escribiendo el nombre de la variable de la base de la unidad seguido de un punto y luego los **parámetros** entre paréntesis
- Si desea cambiar la dirección en que se mueve el robot. puede hacer que la distancia o la velocidad sean negativas en lugar de positivas para que las ruedas se muevan en la dirección opuesta.
  - Esto también se aplica al girar en la dirección opuesta
- Los comandos de movimiento más comunes que usará son los comandos recto y de giro, pero hay varios otros:
  - `drivebase.straight(distancia)` *# Este es el comando para hacer que su robot se mueva en línea recta por una distancia determinada. La distancia debe escribirse en milímetros.*

- **drivebase.turn(ángulo)** *# Este es el comando para hacer que su robot gire en un cierto ángulo. El ángulo debe escribirse en grados.*
- **drivebase.run(velocidad)** *# Este comando ejecutará el robot a cierta velocidad. Esta velocidad está en grados por segundo*
- **drivebase.run_time(velocidad, tiempo)** *# Este es el comando para hacer que su robot se mueva en línea recta a una velocidad determinada durante un cierto período de tiempo. La velocidad debe estar en grados por segundo y el tiempo en milisegundos*
- **drivebase.run_angle(velocidad, ángulo de rotación)** # Este es el comando para hacer que el robot se mueva en línea recta para una velocidad y un ángulo de rotación dados. La velocidad debe escribirse en grados por segundo, y el ángulo de rotación debe escribirse
- **drivebase.drive(velocidad, velocidad de giro)** *enruedas La velocidad debe escribirse en milímetros por segundo. La velocidad de giro debe escribirse en grados por segundo (qué tan rápido girará el robot)*
- **drivebase.stop()** *# Evita que el robot se mueva dejando que los motores giren libremente.*
- **drivebase.brake()** *# Detiene el movimiento del robot reduciendo la velocidad con los frenos*
- **drivebase.hold()** *# Evita que el robot se mueva manteniendo las ruedas en su lugar a la fuerza*

Sensores:

- Sensor Ultrasonico
  - El sensor ultrasónico mide dos cosas: mide la distancia desde el robot hasta un objeto sólido (como una pared) y escucha otros sensores ultrasónicos (esto es útil para tratar de evitar a otros robots)
  - Lo hace enviando ondas de sonido tan altas que no podemos escucharlas y escuchando cómo regresan de manera similar a cómo un murciélago usa la ecolocalización
  - Para usar la distancia, use el siguiente comando:
    US.distance(silent=True/False) *# La opción silenciosa es opcional pero le permite elegir si otros sensores ultrasónicos podrán escuchar el suyo o no.*
  - Para escuchar otros sensores ultrasónicos use el siguiente comando:
    US.presence() *# Esto devolverá como Verdadero o Falso*

- Sensor de Color
    - El sensor de color se puede usar para medir una amplia variedad de cosas. Puede medir la intensidad de la luz, el color de la luz y el reflejo de una superficie.
    - Los colores que puede medir son negro, azul, verde, amarillo, rojo, blanco, marrón y ninguno
    - Para medir el color de la luz que detecta el sensor de color se debe utilizar el siguiente comando:
      `CS.color()  # Devuelve el color o ninguno`
    - El comando necesario para medir la intensidad de la luz (cuán brillante es):
      `CS.ambient()  # devuelve la intensidad de la luz como un porcentaje`
    - El comando para averiguar qué tan reflectante es la superficie a la que se enfrenta el sensor de color está dado por:
      `CS.reflection()  # devuelve la reflexión como un porcentaje`
- Sensor Tactil
    - El sensor táctil se usa como un botón en el que se puede hacer clic, ya sea por el robot que choca contra una pared o por los estudiantes que lo presionan para activar el robot y realizar una determinada acción.
    - El sensor táctil tiene un comando atribuido a él. El **pressed()** para comprobar si se ha pulsado el sensor táctil.
    - Para verificar si se ha presionado el botón táctil, se debe usar el siguiente comando:
      `TS.pressed()`

Controles de flujo

- Un bucle le permite repetir una sección de su código una cierta cantidad de veces o siempre que se cumpla una situación específica
- **For Loops**
    - For se utilizan para hacer que su robot ejecute un comando o un conjunto de comandos una cantidad repetida de veces.
    - La variable después de for en el ciclo for cuenta cuántas veces se ha repetido el ciclo comenzando en el número 0. Esto significa que la primera vez que se ejecuta el código, el número será 0, la segunda vez que se ejecuta el número será 1, la la tercera vez que se ejecuta serán 2, etc.

- El parámetro para rango es el número de veces que desea que se repita el código.
- Recuerde sangrar todo el código que desea incluir en su ciclo después del comando de ciclo para que Python sepa dónde comenzar y finalizar el ciclo
- Esto es un ejemplo de un codigo que utiliza los bucles para que el robot haga un cuadrado 4 veces:

```
for x in range(4):
    drivebase.straight(100)
    drivebase.turn(90)
```

- **While Loops**
  - while se utilizan para verificar si una declaración dada o una comparación es verdadera o falsa
  - Ejecutará el código sangrado debajo. , repitiendo mientras se cumpla la condición que establezca.
  - Un bucle while continuará hasta que una determinada acción ya no sea cierta, esto se puede lograr si se cumplen las condiciones que desea que afecten a su bucle while.
    - Por ejemplo, si desea que el robot se mueva en línea recta siempre que no haya obstáculos. Se puede usar un bucle while para hacer que el robot se mueva en línea recta siempre que no haya nada delante del robot. Sin embargo, tan pronto como el robot encuentra (siente) algo frente a él, se le puede indicar al robot que se mueva en una dirección diferente.
  - También puede usar un ciclo while para hacer que el código se repita para siempre. Esto se hace colocando todo su código debajo del siguiente código:

```
while True:
    drivebase.straight(100)
```

  - Si desea finalizar un ciclo, use una instrucción break

```
while True:
    drivebase.straight(100)
    break()
```

  - Si desea repetir el código para una condición específica (por ejemplo, repetir un código mientras se presiona el sensor táctil), esta es la **sintaxis** que usará:

```
while TS. pressed()= True:
```

```
        drivebase.straight(30)
```

- **Sentencias If/Else Las sentencias**
    - If crean una condición que debe cumplirse para que se ejecute una sección de su código. Puede configurar diferentes condiciones con un código diferente y también crear una sección en la que, si no se cumple ninguna de las condiciones, se ejecutará un código completamente diferente (llamado sentencia else).
        - Por ejemplo, si llueve, traeré un paraguas, pero si llueve, no lluvioso no lo haré.
        - En este ejemplo, la condición es lluviosa y el código sería lo que estoy trayendo
    - Esto puede sonar similar a los bucles while, esto se debe a que a menudo puede lograr el mismo resultado utilizando el código de su robot, pero uno será más eficiente que el otro
    - Hay 3 cosas principales que puede usar en una declaración if, if (la condición), elif (cualquier condición adicional, solo puede usar if una vez en una declaración if pero puede usar elif tantas veces como desee), y else (cuando no se cumple ninguna de las condiciones if o elif) Las
    - declaraciones If también usan comparaciones, como mayor que, menor que, mayor que o igual, etc.
        - Por ejemplo,

```
if US.distance() < 200: # esta es la
                        condición establecida
      drivebase.straight(-90) # este es el código
      que se ejecutará cuando la pared esté a menos
      de 200 mm
      elif US.distance() = 200:
            drivebase.turn(90)
      else: # el código debajo el resto solo se
            ejecutara condiciones anteriores
            drivebase.straight(90)
```

Vocabulario Esencial

➔ Entrada: información dada a la computadora (o robot) desde el exterior (como las letras que escribe en la computadora, la información que un sensor le da al robot) → piense en esto como poner información en la computadora

- ➔ Salida: información que la computadora le da a algo en el exterior (como las imágenes que se muestran en la pantalla de la computadora y la información de movimiento que el ladrillo le da a los motores) → piense en esto como poner información del robot
- ➔ Parámetros: lo que se le da al código cuando desea que se ejecute durante una determinada distancia, tiempo o velocidad. Se ve como los números que se ingresan en el comando drivebase.straight(), para que el robot pueda moverse una cierta distancia. La distancia dada en este caso es un parámetro
- ➔ Sintaxis: la forma en que escribimos las cosas. La sintaxis es la ortografía, los comandos, las mayúsculas y el texto específico que debe colocar para que Python sepa qué es lo que le está pidiendo que haga.