Name:_____________________

Micropython For EV3 Reference Sheet

Importing Modules

- These are the **libraries** that are used to control your robot
- Libraries: a collection of related **modules** that are used to make your code simpler and easier to use
- Modules: other people's code that we are stealing (with permission) in order for us to do certain tasks python does not have commands for so that we can do those tasks quickly and easily
- Here are the modules you need to import before any program using EV3 devices (the robots):

```
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: a name we assign to another command or an item that has information that changes regularly like the **input** in a sensor
- In the code for your robot, you need to define a variable for your brick, your drivebase, and every single sensor and motor you have on your robot
- You can define a variable anywhere in your code before using it but it is most helpful to do this at the top of your code together so that you can find them easier
- This is how you define a variable for your brick, it is useful for things like beeps

```
ev3 = EV3Brick() # This is your brick
```

- This is how you define variables for your motors

```
left_motor = Motor(Port.A) # This is the left motor and port
right_motor = Motor(Port.D) # This is the right motor and port
```

- This is how you define variables for your sensors

```
US = UltrasonicSensor(Port.S1)
CS = ColorSensor(Port.S2)
TS = TouchSensor(Port.S3)
GS = GyroSensor(Port.S4)
```

The Drive Base

- The Drive base is what controls your robot's movements, it is a very important variable that uses your left and right motor to make your robot turn and move.
  ```
  drivebase = DriveBase(left_motor, right_motor, wheel_diameter = ,
  axle_track = ) # This is your drivebase command
  ```
- It is necessary to tell the Drive Base command the motors on your robot, without it the robot will not be able to move in any direction.
- The **wheel diameter** is the distance between two opposite sides on the robot's wheels in millimeters. The robot uses this measurement to make sure it is moving at the correct distance.
- The **axle track** distance is what allows your robot to make accurate turns, without it your robot will not be able to properly turn. It is measured by finding the distance between the center point of your two wheels in millimeters.

Movement Controls

- Movement controls are the commands you give to the robot to make it move.
- They let you do things such as going forward and backward, stopping, and turning
- Most movement commands are formatted by typing the variable name of the drivebase followed by a dot, and then the **parameters** in parenthesis
- If you want to change the direction the robot moves in, you can make the distance or speed negative instead of positive so that the wheels move in the opposite direction
  - This also applies to turning in the opposite direction
- The most common movement commands you will use are straight and turn commands but there are several others:
  - `drivebase.straight(distance) # This is the command to make your robot move in a straight for a given distance. The distance must be written in millimeters.`
  - `drivebase.turn(angle) # This is the command to make your robot turn at a certain angle. The angle must be written in degrees.`
  - `drivebase.run(speed) # This command will run the robot at a certain speed. This speed is in degrees per second`

- `drivebase.run_time(speed, time) # This is the command to make your robot move in a straight line at a given velocity for a certain amount of time. The velocity must be in degrees per second and the time is in milliseconds`
- `drivebase.run_angle(speed, rotation angle) # This is the command to make the robot move in a straight line for a given speed and rotation angle. The speed must be written in degrees per second, and the rotation angle must be written in degrees`
- `drivebase.drive(drive_speed, turn_rate) # This is the command to make your robot move straight at a given velocity and at a given turn rate for the wheels. The velocity must be written in millimeters per second. The turn rate must be written in degrees per second (how quickly the robot will turn)`
- `drivebase.stop() # Stops the robot from moving by letting the motors turn freely.`
- `drivebase.brake() # Stops the robot from moving by slowing down with brakes`
- `drivebase.hold() # Stops the robot from moving by forcibly holding the wheels in place`

Sensors:

- Ultrasonic Sensor
    - The ultrasonic sensor measures two things- it measures the distance from the robot to a solid object (like a wall) and it listens for other ultrasonic sensors (this is useful for trying to avoid other robots)
    - It does this by sending out sound waves so high pitched that we can not hear them and listening for them to return similarly to how a bat uses echolocation
    - In order to use distance use the following command: `US.distance(silent=True/False)# The silent option is optional but allows you to choose whether or not other ultrasonic sensors will be able to hear yours`
    - In order to listen for other ultrasonic sensors use the following command: `US.presence()# This will return as True or False`
- Color Sensor
    - The color sensor can be used to measure a wide variety of things. It can measure the intensity of light, the color of light, and the reflection of a surface.
    - The colors it can measure are black, blue, green, yellow, red, white, brown, and none

    - To measure the color of the light that is detected by the color sensor the following command must be used:
      ```
      CS.color()   # Returns the color or none
      ```
    - The command needed to measure the intensity of the light (how bright it is):
      ```
      CS.ambient()   # returns the light intensity as a percentage
      ```
    - The command to find out how reflective the surface the color sensor is facing is given by:
      ```
      CS.reflection()   # returns reflection as a percentage
      ```
- Touch Sensor
    - The touch sensor is used as a button that can be clicked, either by the robot running into a wall or by the students pressing it to trigger the robot to do a certain action.
    - The touch sensor has one command attributed to it. The **pressed()** command is used by the robot to check whether the touch sensor has been pressed.
    - In order to check whether the touch button has been pressed the following command must be used:
      ```
      TS.pressed()
      ```

Flow Controls

- A loop allows you to repeat a section of your code a certain amount of times or as long as a specific situation is met
- **For loops**
    - For loops are used to make your robot do a command or a set of commands a repeated amount of times.
    - The variable after the for in the for loop counts how many times the loop has repeated beginning at the number 0. This means that the first time the code runs the number will be 0, the second time it runs the number will be 1, the third time it runs it will be 2, etc.
    - The parameter for range is the number of times you want the code to repeat.
    - Remember to indent all the code you want to include in your loop after the loop command so python knows where to start and end the loop
    - This is an example of a for loop for a code that will make a square by repeating the straight and turn commands 4 times:
      ```
      for x in range(4):
            drivebase.straight(100)
            drivebase.turn(90)
      ```

- **While loops**
    - While loops are used to check whether a given statement or comparison is True, or False
    - It will run the code indented underneath it, repeating for as long as the condition you establish is met
    - A while loop will continue to loop until a certain action is no longer true, this can be achieved by having conditions be met that you want to affect your while loop.
        - For example, if you want the robot to move in a straight line as long as there are no obstacles. A while loop can be used to make the robot move straight as long as there is nothing in front of the robot. However, as soon as the robot encounters (senses) something in front of it the robot can be told to move in a different direction.
    - You can also use a while loop to make the code repeat forever this is done by putting all of your code underneath the following code:

      ```
      while True:
            drivebase.straight(100)
      ```
    - If you want to end a loop, you use a break statement

      ```
      while True:
            drivebase.straight(100)
            break()
      ```
    - If you want to repeat the code for a specific condition (for example, repeating a code for as long as the touch sensor is pressed) this is the **syntax** you will use:

      ```
            while TS.pressed()= True:
                  drivebase.straight(30)
      ```
- **If/Else statements**
    - If statements create a condition that needs to be met for a section of your code to run. You can set up different conditions with different code and also create a section where if none of the conditions are met a completely different code will run (called an else statement)
        - For example, if it is rainy I will bring an umbrella but if it is not rainy I will not.
        - In this example, the condition is rainy and the code would be what I am brining

- This may sound similar to while loops- this is because oftentimes you can achieve the same result using either in the code for your robot but one will be more efficient than the other
- There are 3 main things you can use in an if statement, if (the condition), elif (any additional conditions, you can only use if once in an if statement but can use elif as many times as you want), and else (when none of the if or elif conditions are met)
- If statements also use comparisons, like greater than, less than, greater than or equal, etc.
  - For example,

```
if US.distance() < 200: # this is the
                         condition set
      drivebase.straight(-90) # this is the code
      that will run when the wall is closer than
      200 mm
elif US.distance() = 200:
      drivebase.turn(90)
else: # the code under the else will only run
            if none of the conditions above are met
      drivebase.straight(90)
```

Useful Vocabulary:

➔ Input: information given to the computer (or robot) from the outside (like the letters you type into the computer the information a sensor gives the robot) → think of this as putting information into the computer

➔ Output: information given from the computer to something on the outside (like the images shown on the computer screen and the movement information the brick gives the motors) → think of this as putting information out of the robot

➔ Parameters: what is given to the code when you want it to run for a certain distance, time, or speed. It is seen as the numbers that are input into the drivebase.straight() command, so that the robot can move a certain distance. The distance given in this case is a parameter

➔ Syntax: the way we type things. The syntax is the spelling, commands, capitalization, and specific text you need to put for python to know what it is you are asking it to do.