UNIVERSITY TRANSPORTATION CENTER FOR RAILWAY SAFETY
Award Number: 69A3552348340

Center for Multidisciplinary Research Excellence in Cyber-Physical Infrastructure Systems (MECIS)
Award Number: 2112650

# High School Curriculum

# Week Overview

1. Intro to Python
2. Sensor Programming
3. Driving the Rover
4. Hardware Applications
5. Final Challenge

# Day 1

- Ice Breaker (Logic Labyrinth)
- Engineering Design Process
- Algorithm Design Process
- Intro to Variables
- Understanding Logic
- Active Sign Creation

# Ice Breaker: Logic Labyrinth (Thinking like a computer)

- Make teams of 4-5 students, each team will try to complete a simple task using the functions provided.
- Computers today can perform very complex tasks such as driving cars autonomously, or give you the answer to your homework

# Ice Breaker (Continue)

Allowed commands for your “robot”:

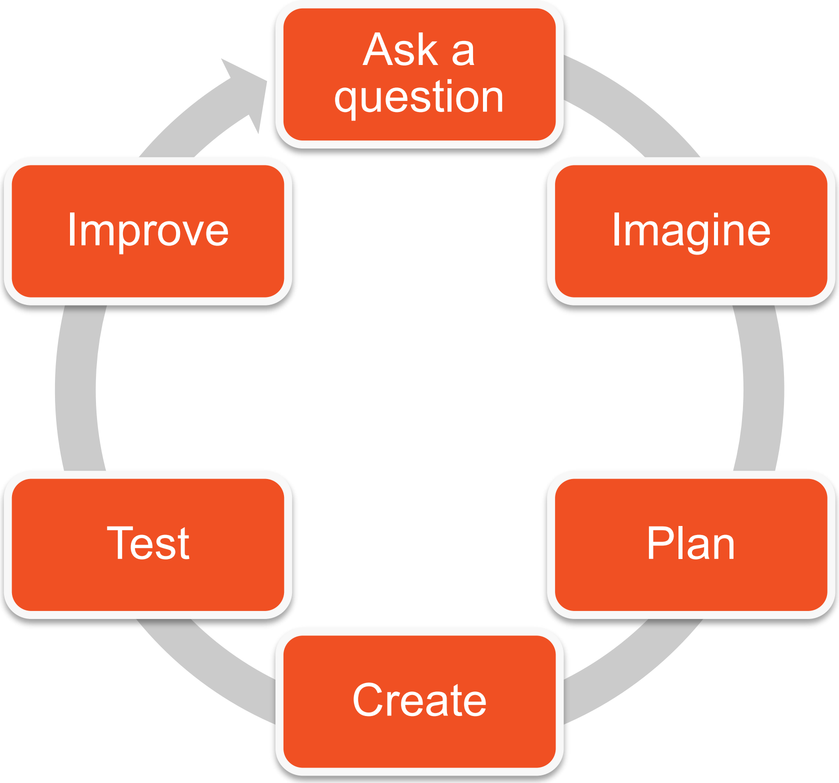
- Move forward (continuously)
- Move Back (continuously)
- Take # steps forward
- Take # steps back
- Turn right # degrees
- Turn left # degrees
- Stop

# Ice Breaker (Continue)

- In reality computers are very dumb, they can only do simple tasks like adding, subtracting, reading and writing data in memory. That is it! For the computer to complete the complex tasks AI can do we need a bunch of operations that all use those 4 actions.
- How do we create complex things like moving a robot?
    - Answer: Abstractions/Functions

# Engineering Design Cycle (Process)

# Engineering Design Cycle (Process)



Ask a question

Imagine

Plan

Create

Test

Improve

# What is Engineering?

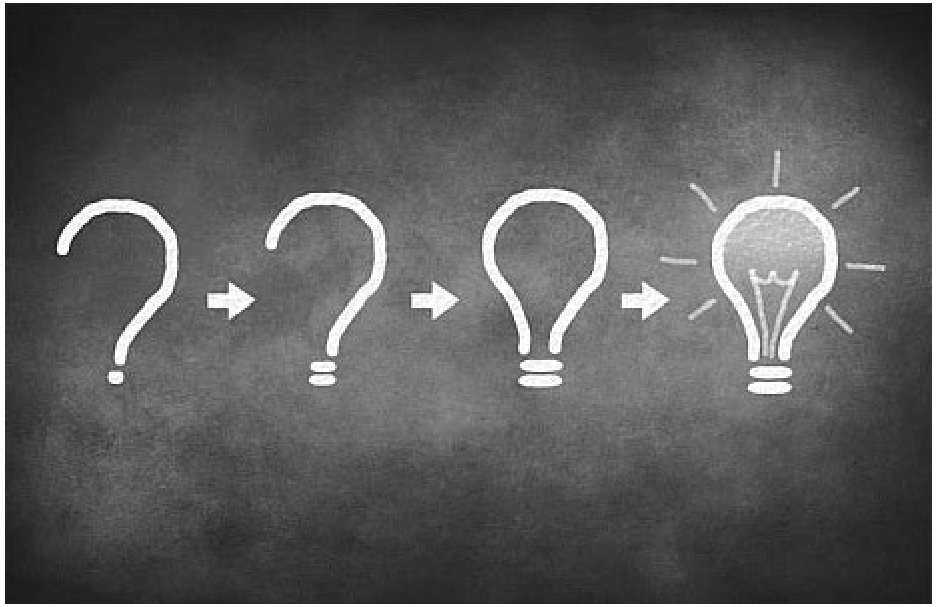
- Engineering is the process of creating technology
  - Can you name some examples of technology we use every day?

# ASK

- Students identify the problem, requirements that must be met, and constraints that must be considered.

# Imagine

- Students brainstorm solutions and research ideas.  They also identify what others have done.



# Plan

- Students choose two to three of the best ideas from their brainstormed list and sketch possible designs, ultimately choosing a single design to prototype.

# Create

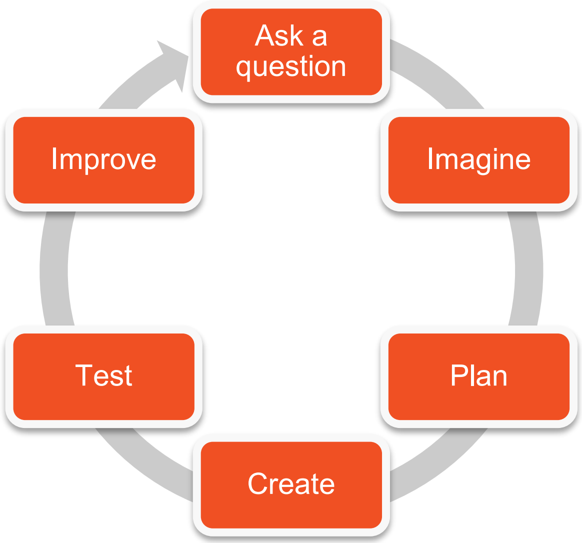
- Students build a working model, or prototype, that aligns with design requirements and that is within design constraints.

# Experiment

- Students evaluate the solution through testing; they collect and analyze data; they summarize strengths and weaknesses of their design that were revealed during testing.

# Improve

- Based on the results of their tests, students make improvements on their design. They also identify changes they will make and justify their revisions.



Ask a question

Imagine

Plan

Create

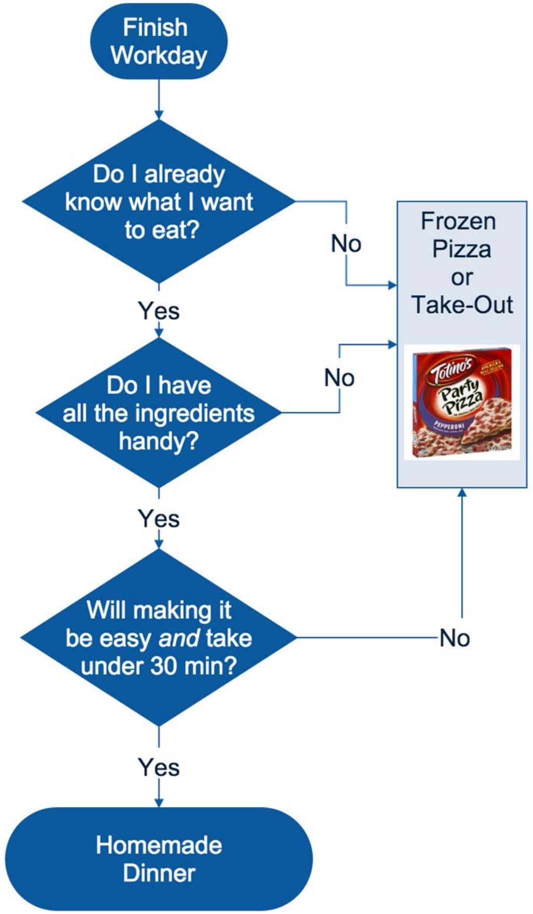
Test

Improve

# Algorithms and Pseudocode

# Algorithms

- An algorithm is a set of instructions for solving a problem or accomplishing a task. One common example of an algorithm is a recipe, which consists of specific instructions for preparing a dish or meal.


Finish Workday
Do I already know what I want to eat?
No
Yes
Frozen Pizza or Take-Out
Do I have all the ingredients handy?
No
Yes
Will making it be easy *and* take under 30 min?
No
Yes
Homemade Dinner

# Algorithm Example

# Pseudocode

- Pseudocode looks almost like real code, but it does not follow any specific syntax for any language, instead it is meant to be written almost in plain English to break down steps in a set of instructions. E.g.

```
Pseudocode for deciding what to eat:

when it is the end of the day
        Then Do I know what to eat?
        if yes
                then Do I have all ingredients?
                If yes
                        then Is it going to be complete in < 30 min?
                        If yes
                                Then Start cooking script
                        else Order take out
        else Order take out
Else continue working
```

# What do you know about programming?

What is Python?

# What is Python?

# Coding:

**go to further.pi-top.com**

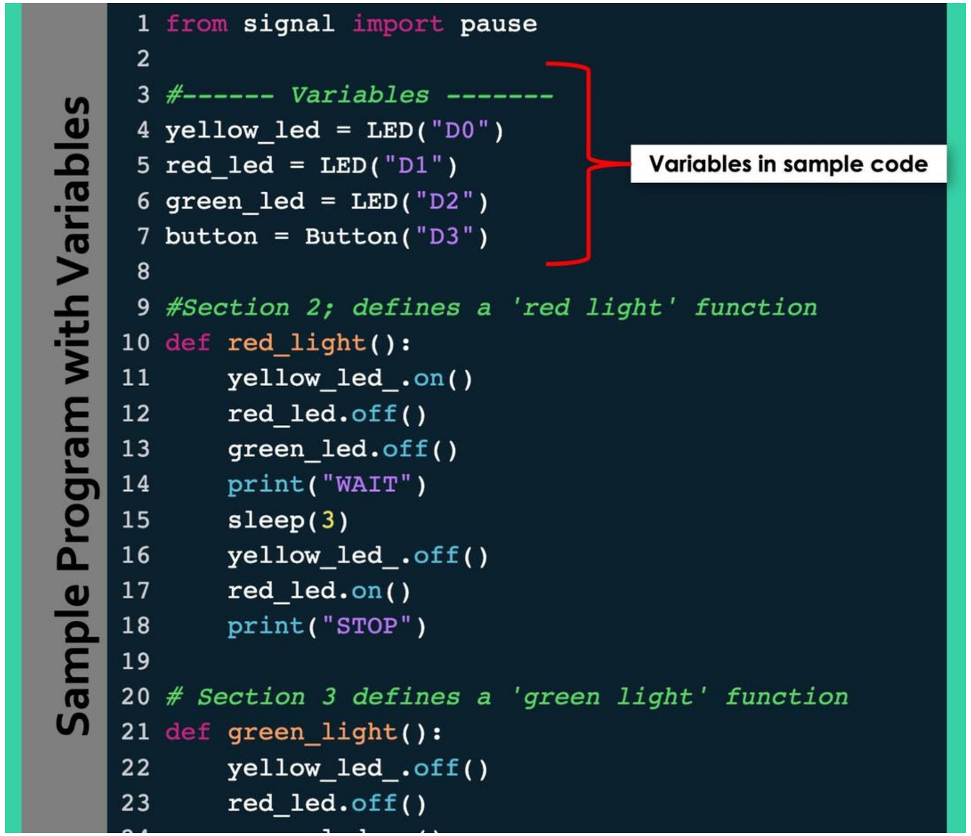
**Join: 2024 UTCRS Summer Camps**
**Code: 4FVZV8K73F**

# Further Course: Intro to Variables

1. What is a Variable?
2. Naming Variables
3. Rules for Variables

# What is a Variable?

- Variables are declarations or names written close to the top of a script. They are used to store and manipulate pieces of information. Variables allow us to assign values, declare what ports or pins are used, use calculations, etc.

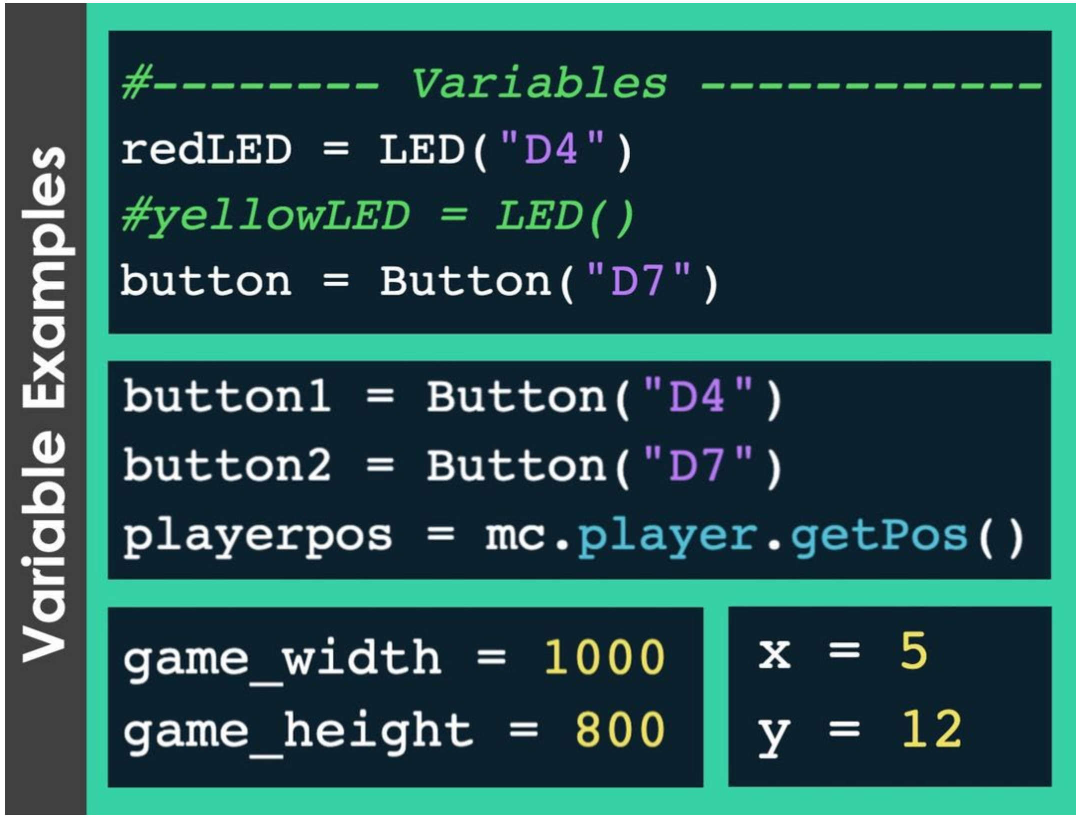


```
from signal import pause

#------ Variables -------
yellow_led = LED("D0")
red_led = LED("D1")
green_led = LED("D2")
button = Button("D3")

#Section 2; defines a 'red light' function
def red_light():
    yellow_led_.on()
    red_led.off()
    green_led.off()
    print("WAIT")
    sleep(3)
    yellow_led_.off()
    red_led.on()
    print("STOP")

# Section 3 defines a 'green light' function
def green_light():
    yellow_led_.off()
    red_led.off()
```

# Naming Variables

- To create a variable, it must be given a name and a value (or some type of “defining” information).

  For instance, speed = 120.
- The variable is named speed, with an integer value of 120.
- It is important to use a good name that is easy to remember and one that isn't closely related to other variable names. If we use something like walk1 and walk2 it may be hard to remember the difference between the type of walking. But if we used walk_slow and walk_fast it quickly tells us the difference and is easy to remember later in the code.
- See the image on the right for other variable name examples.



```
#-------- Variables ------------
redLED = LED("D4")
#yellowLED = LED()
button = Button("D7")
```

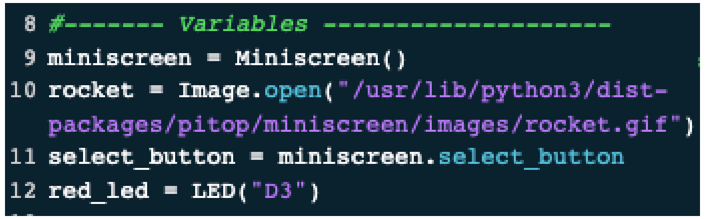
```
button1 = Button("D4")
button2 = Button("D7")
playerpos = mc.player.getPos()
```

```
game_width = 1000
game_height = 800
```

```
x = 5
y = 12
```

# Rules for Variables

- Variables can contain words, letters, underscores, and numbers but it is best to use a word or letter to start it off.
- Python code is case sensitive so if capital letters are used, they must always be written the same way throughout the script. For instance, Walk and walk are totally different variable names. Symbols: - / # @ are not to be used when naming variables. Avoid using python commands, like import or print.



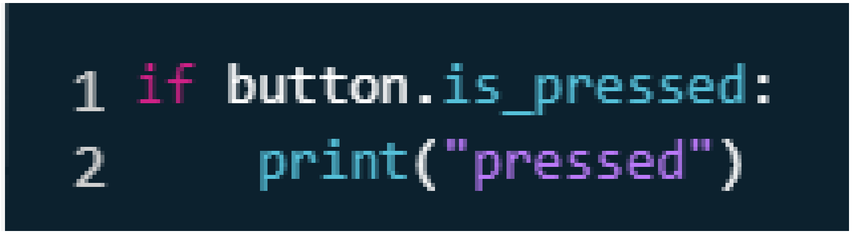
```
#------- Variables --------------------
miniscreen = Miniscreen()
rocket = Image.open("/usr/lib/python3/dist-packages/pitop/miniscreen/images/rocket.gif")
select_button = miniscreen.select_button
red_led = LED("D3")
```

# Further Course: Understanding Logic

- Python Principles: if Statements
- Python Principles: Loops

# Python Principles: if Statements

- What are if statements?
  - Conditionals, such as if and else, are used for decision making operations and in branching.



```python
if button.is_pressed:
    print("pressed")
```

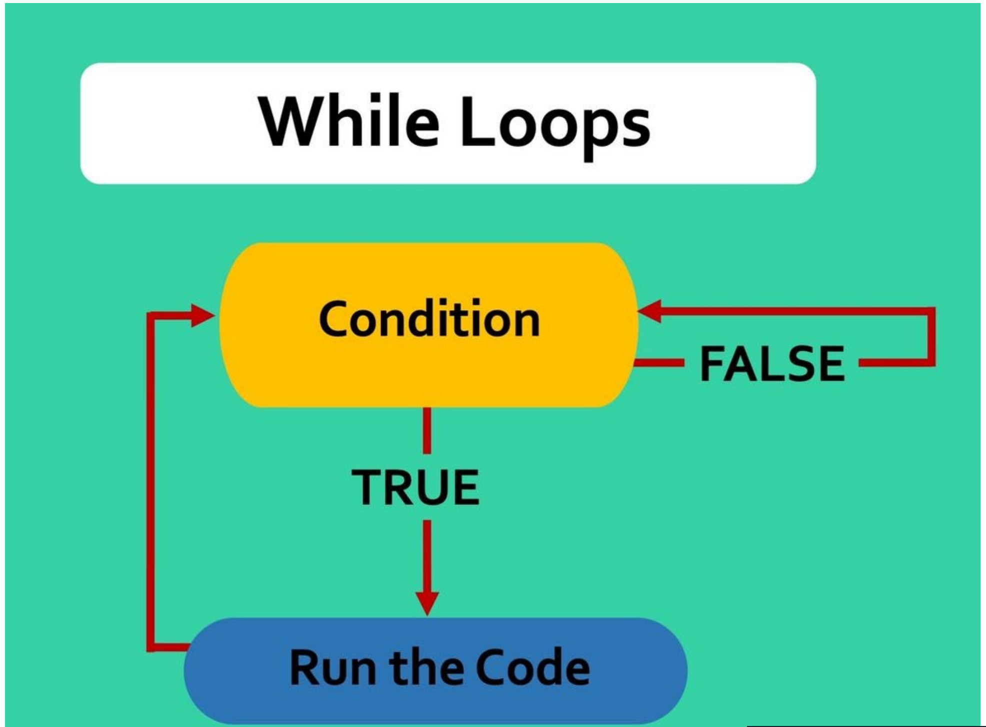
  - If statements are used when programmers want to give the computer a condition to evaluate before activating and to activate any code.

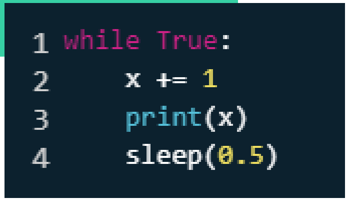
# Python Principles: Loops

- What are Loops?
  - Loops are used when programmers want to execute repetitive code when a program runs. A loop will run the same set of code over and over again when written in a program.
  - Think of this as a place where you would put the ongoing actions that a game character would need to have. If they should take more than one step when a button is pressed, then it is most convenient when in a loop.

# Python Principles: Loops

- While Loops (Infinite Loops)
  - What happens when a programmer isn't sure how many times to run the code? Or if they want to make it run repeatedly until they press stop? In this type of situation, a while loop is used. While loops are loops that will run if and when a condition is met. It is a type of Boolean expression - meaning the condition can only be true or false.

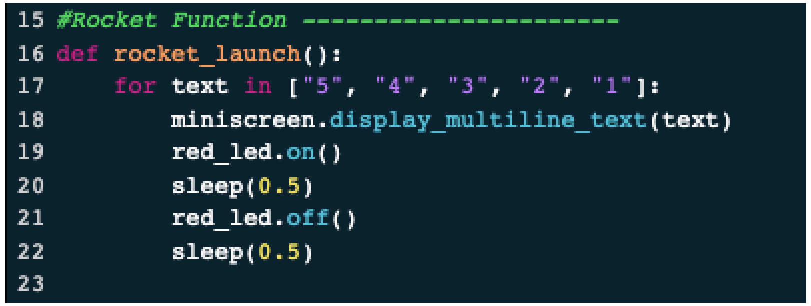




```python
while True:
    x += 1
    print(x)
    sleep(0.5)
```

# Python Principles: Loops

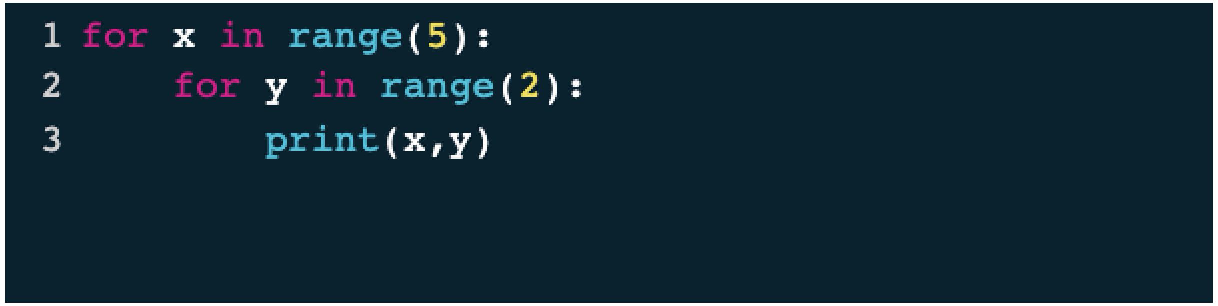
- for loops
  - When the programmer wants a section of code to happen a set number of times, the for loop is used. The amount of times is known as an iteration.
  - For loops iterate over a given sequence, or run the same block of code over and over until the sequence ends. They basically will run a body of code for a specified amount of time or only as long as the code that is written needs to run. Complete exercise 6 below to practice this concept.



```
#Rocket Function ----------------------
def rocket_launch():
    for text in ["5", "4", "3", "2", "1"]:
        miniscreen.display_multiline_text(text)
        red_led.on()
        sleep(0.5)
        red_led.off()
        sleep(0.5)

```

# Python Principles: Loops

- Nested Loops
  - Python allows us to use a loop inside of another loop. Yes, there may be times that for loops need to be written inside of other loops. While loops or infinite loops can also contain for loops, depending on what the programmer wants to happen. This is easy to do, but always make sure that the lines are indented properly.



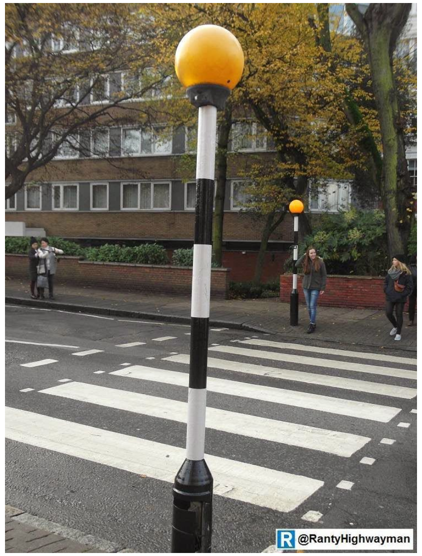
```python
for x in range(5):
    for y in range(2):
        print(x,y)
```

# Further Course: Programming Hardware

- Traffic Lights
- Python Principles: Functions (Optional)

# Traffic Lights

- Belisha Beacons
  - marks pedestrian crossings, such as zebra crossings, in the United Kingdom and other countries



# Traffic Lights

- Traffic lights normally consist of three signals, transmitting meaningful information to road users through colors and symbols, including arrows and bicycles.

# Traffic Lights

- What other traffic lights do you know?
- Can you recreate it?

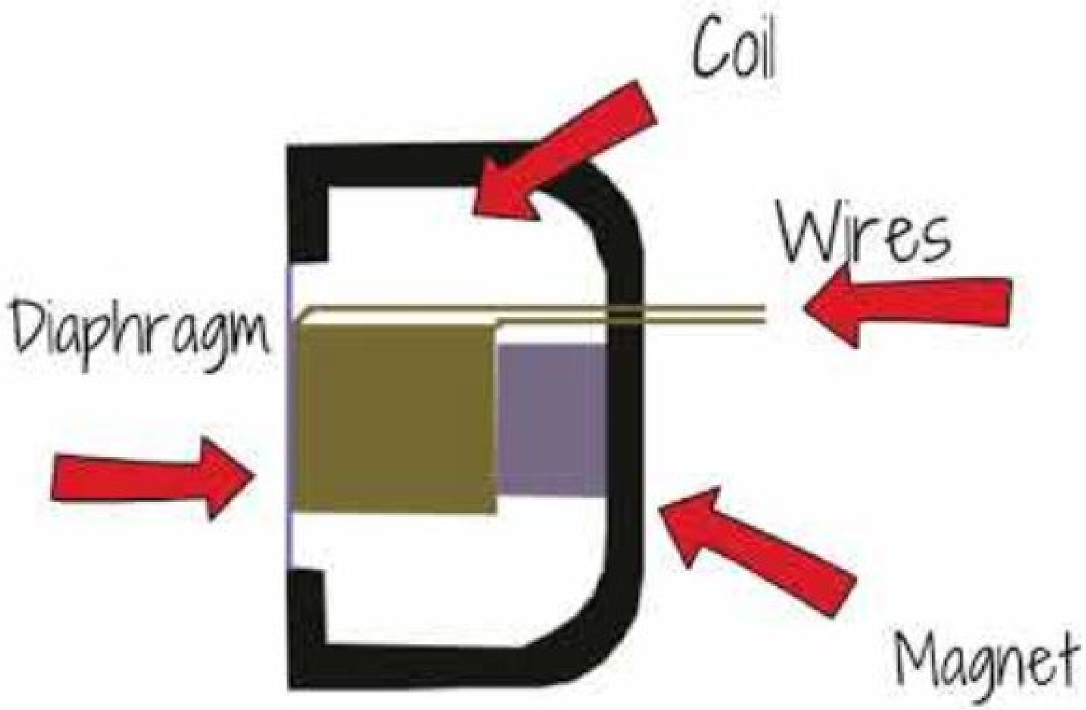
# Day 2

1. Understanding the Sensors
2. Active Sign Creation
3. Rover Build and Navigation

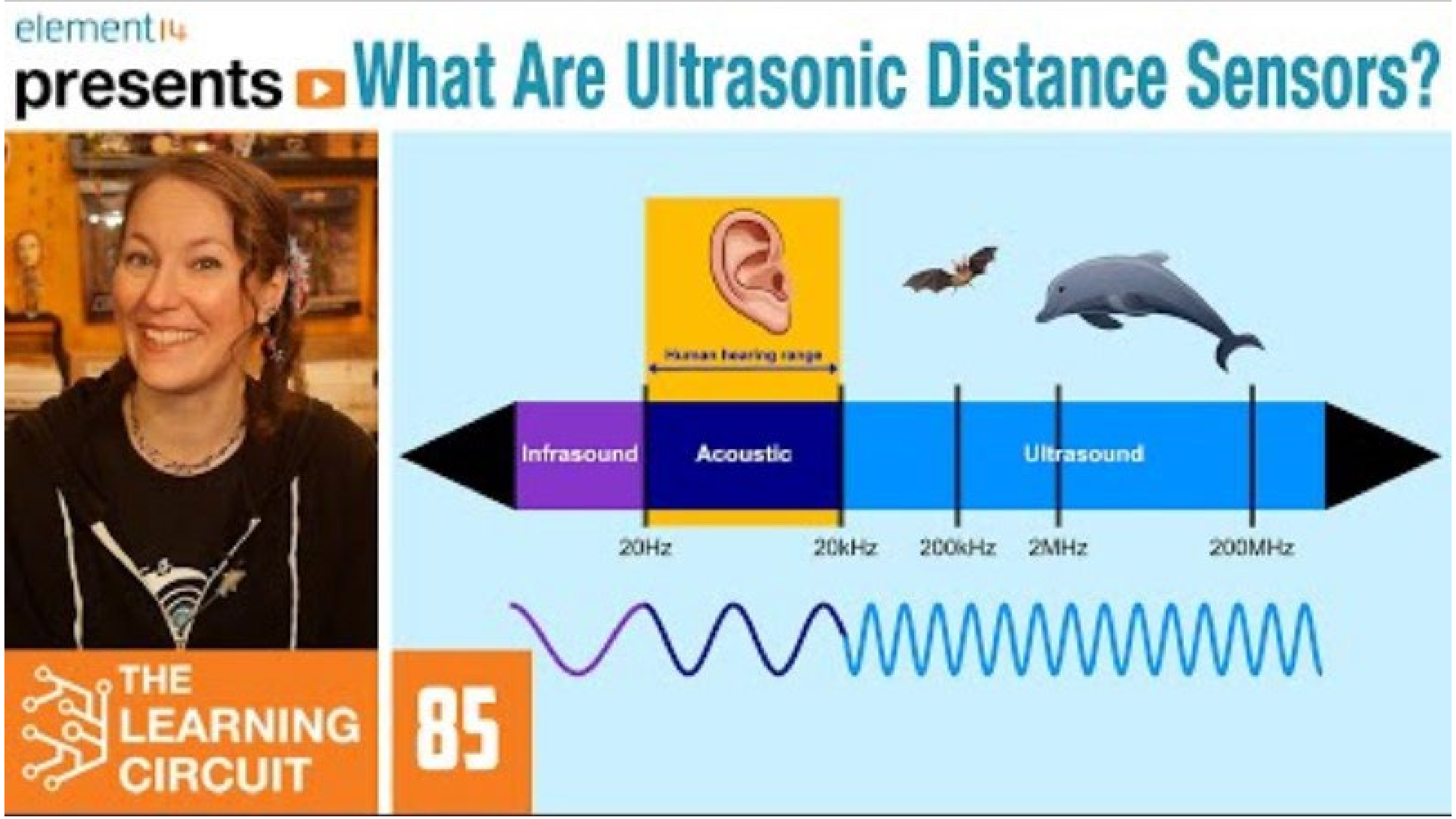
# Further Course: Understanding the Sensors

- Tutorial: Sound Sensor
- Tutorial: Ultrasonic Sensor
- Tutorial: Light Sensor

# Tutorial: Sound Sensor

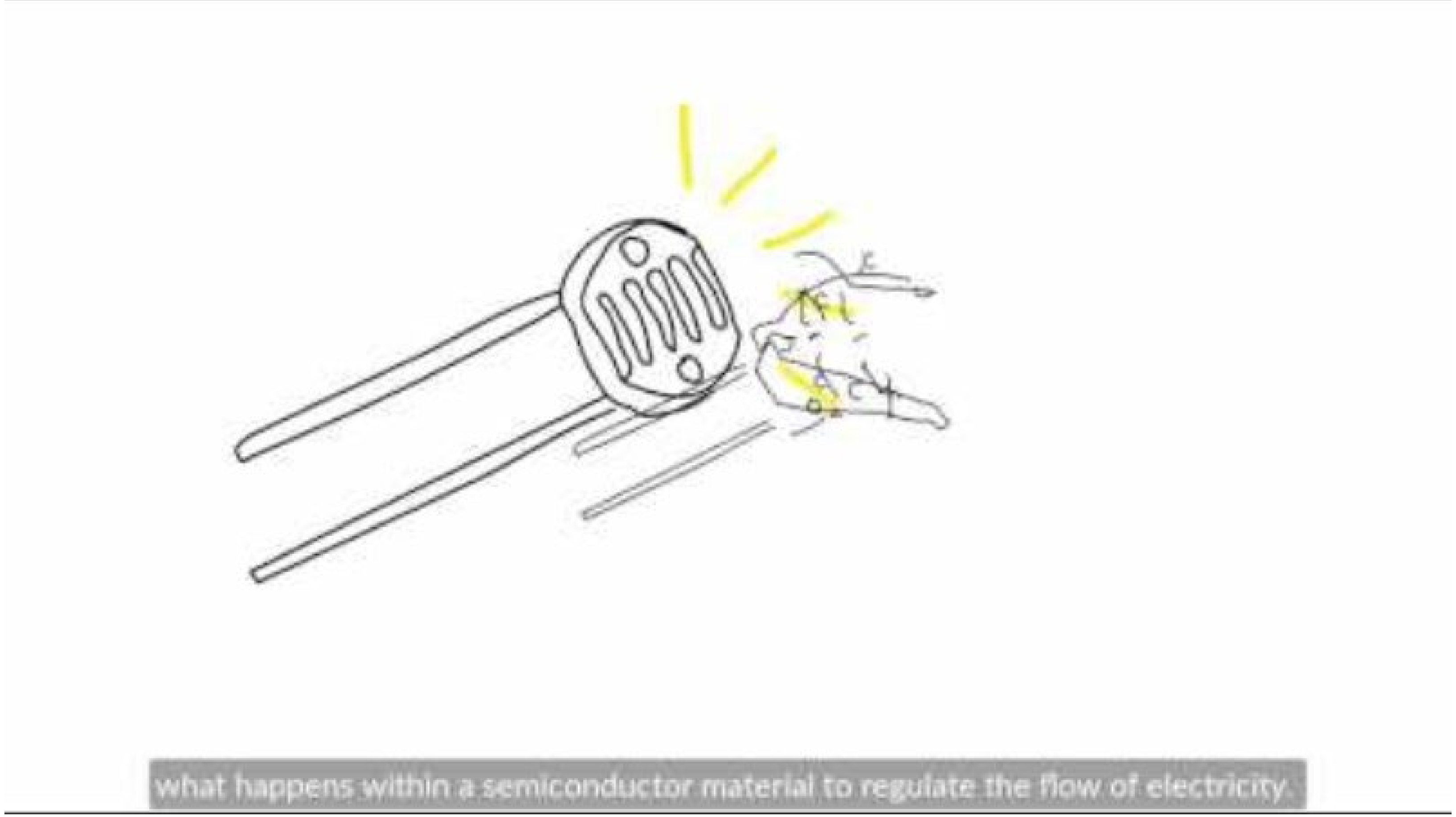


# Tutorial: Ultrasonic Sensor



element14 presents What Are Ultrasonic Distance Sensors?

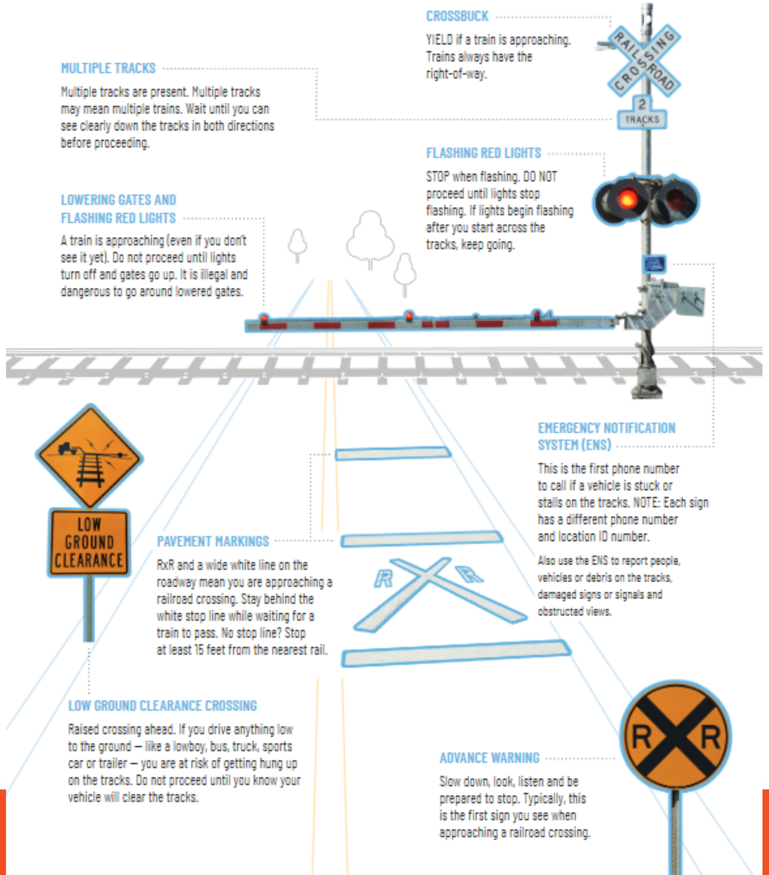
# Tutorial: Light Sensor



what happens within a semiconductor material to regulate the flow of electricity.

# Further Challenge: Active Sign Creation

## Why are railway crossing signs important?



**CROSSBUCK**

YIELD if a train is approaching. Trains always have the right-of-way.

**MULTIPLE TRACKS**

Multiple tracks are present. Multiple tracks may mean multiple trains. Wait until you can see clearly down the tracks in both directions before proceeding.

**FLASHING RED LIGHTS**

STOP when flashing. DO NOT proceed until lights stop flashing. If lights begin flashing after you start across the tracks, keep going.

**LOWERING GATES AND FLASHING RED LIGHTS**

A train is approaching (even if you don't see it yet). Do not proceed until lights turn off and gates go up. It is illegal and dangerous to go around lowered gates.

**EMERGENCY NOTIFICATION SYSTEM (ENS)**

This is the first phone number to call if a vehicle is stuck or stalls on the tracks. NOTE: Each sign has a different phone number and location ID number.

Also use the ENS to report people, vehicles or debris on the tracks, damaged signs or signals and obstructed views.

**PAVEMENT MARKINGS**

RxR and a wide white line on the roadway mean you are approaching a railroad crossing. Stay behind the white stop line while waiting for a train to pass. No stop line? Stop at least 15 feet from the nearest rail.

**LOW GROUND CLEARANCE CROSSING**

Raised crossing ahead. If you drive anything low to the ground — like a lowboy, bus, truck, sports car or trailer — you are at risk of getting hung up on the tracks. Do not proceed until you know your vehicle will clear the tracks.

**ADVANCE WARNING**

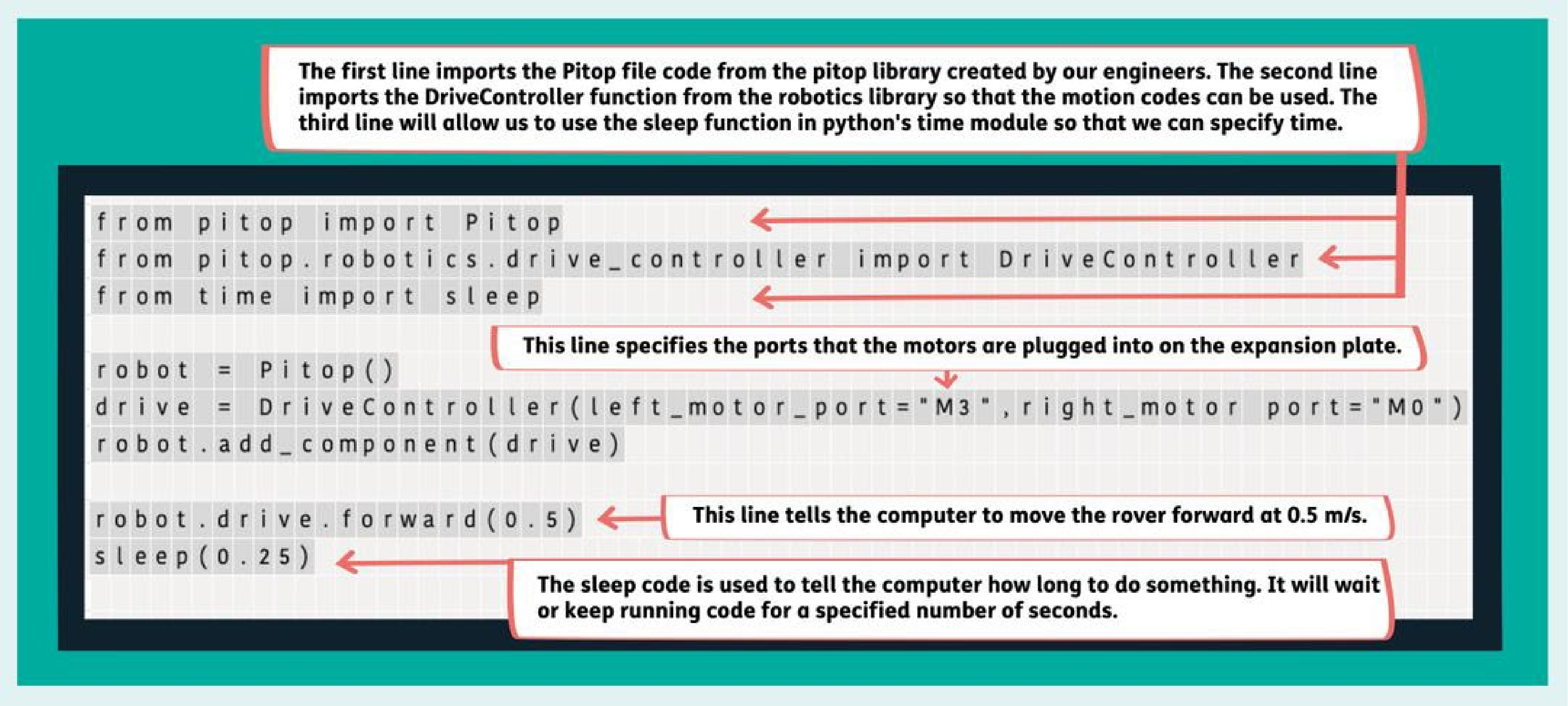
Slow down, look, listen and be prepared to stop. Typically, this is the first sign you see when approaching a railroad crossing.

# Further Course: Rover Build and Navigation

- Robotics 101 [1]: The Build
- Robotics 101 [2]: Navigation
- Reverse Parking Sensor

# Robotics 101 [1]: The Build

# Robotics 101 [2]: Navigation



The first line imports the Pitop file code from the pitop library created by our engineers. The second line imports the DriveController function from the robotics library so that the motion codes can be used. The third line will allow us to use the sleep function in python's time module so that we can specify time.

```
from pitop import Pitop
from pitop.robotics.drive_controller import DriveController
from time import sleep

robot = Pitop()
drive = DriveController(left_motor_port="M3",right_motor port="M0")
robot.add_component(drive)

robot.drive.forward(0.5)
sleep(0.25)
```

This line specifies the ports that the motors are plugged into on the expansion plate.

This line tells the computer to move the rover forward at 0.5 m/s.

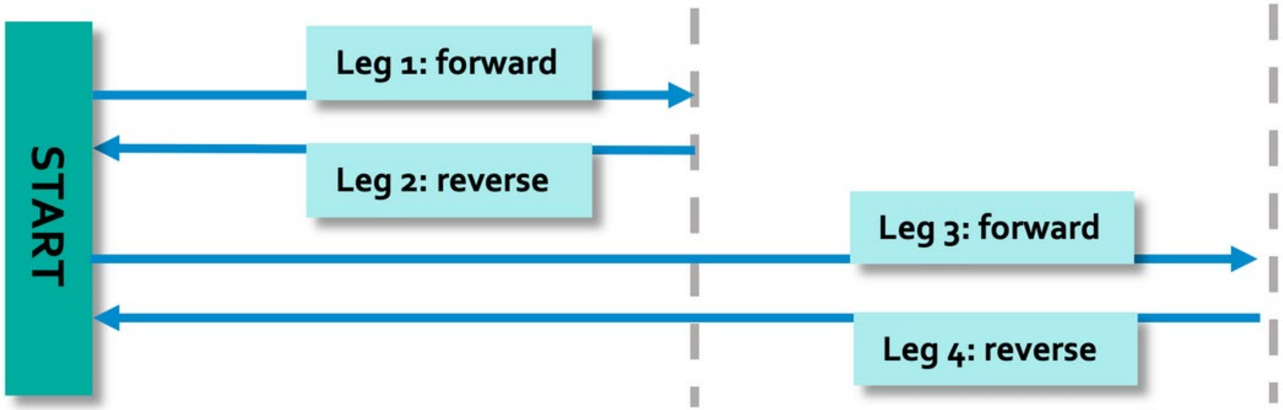
The sleep code is used to tell the computer how long to do something. It will wait or keep running code for a specified number of seconds.

# Navigation Challenges

- Change 1forward.py so the robot travels double the distance
- Change 1forward.py again to testing different distances, how do you move exactly 1 feet?
- Change 2reverse.py so the robot moves forward first then reverses to its original position.

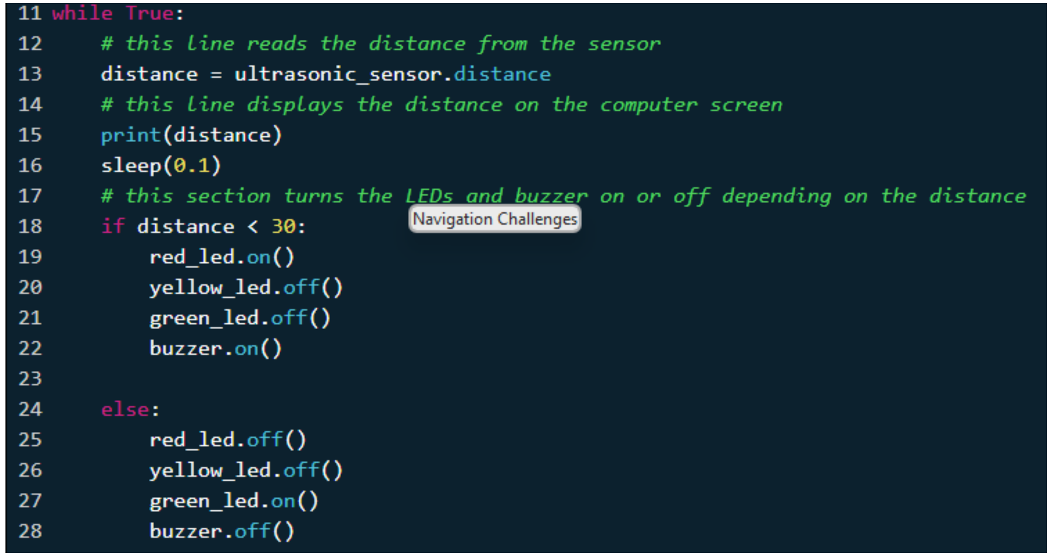
# Navigation Challenges

- Change 2reverse.py again so the robot moves forward 2 feet then reverses to its original position, then again but this time 4 feet. All in 1 program execution.



# Reverse Parking Sensor

- Create your own reverse parking sensor using sensor_control.py.
  - This code incorporates a while True loop. The beginning of the loop is the same as it was in a previous section, but now there is a conditional after the statements to read the distance from ultrasonic sensor and print the reading.
  - This conditional does the following: if something is within 30 cm of the sensor, it turns on the red LEDs and the buzzer.
  - If neither of these conditions are true (else), it turns off all the components except for the green LED.



```
while True:
    # this line reads the distance from the sensor
    distance = ultrasonic_sensor.distance
    # this line displays the distance on the computer screen
    print(distance)
    sleep(0.1)
    # this section turns the LEDs and buzzer on or off depending on the distance
    if distance < 30:
        red_led.on()
        yellow_led.off()
        green_led.off()
        buzzer.on()

    else:
        red_led.off()
        yellow_led.off()
        green_led.on()
        buzzer.off()
```

# Day 3

1. Rover Speed and Rotation
2. Rover Race
3. Rover Vision

# Further Challenge: Rover Speed and Rotation

1. Use speed.py to run your robot for several speed_factors and sleep_times and record the distance for each. Run the program, place your robot at a start location and press enter to run the first experiment, record the distance in the worksheet provided.

$$speed = \frac{distance}{time}$$

# Further Challenge: Rover Speed and Rotation

1. Use angular.py to run your robot for several rotation_angles and sleep_times and record the angular velocity for each. Run the program, place your robot at a start location and press enter to run the first experiment, use your knowledge so far to "guess" the time the robot needs to reach a desired angle. Calculate the angular velocity in radians/seconds in the worksheet provided.

$$angular\ velocity\ (\omega) = \frac{radians\ (\Delta\theta)}{time\ (t)}$$

# Further Challenge: Rover Speed and Rotation

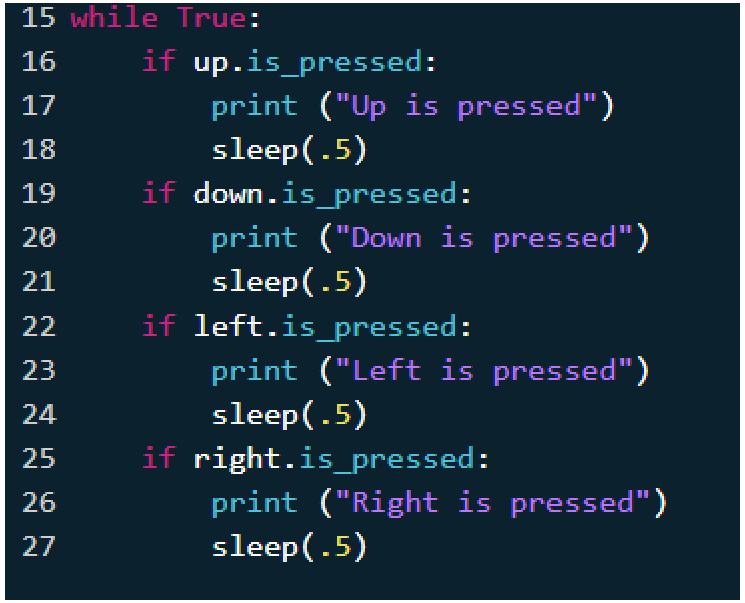
1. What is the top speed? How long does it take to reach that speed?
2. What happens when you keep the speed factor the same but change the sleep time?
3. Describe how to modify the code so that the robot reverses direction at similar speeds to a forward motion.
4. The rover weighs 3.18 kilograms and Perseverance, the Mars rover, weighs about 45 kilograms. How do you think mass affects speed?
5. NASA's Mars Rover drives with a top speed of 0.16 kilometers per hour (0.1 miles per hour). Calculate the speed of your rover and compare it to that of the Perseverance rover.

# Further Challenge: Rover Race (Tutorial USB Keyboard)

- There are 2 ways we can interact with the Pitop[4] using a keyboard
  - Polling (direction_arrows.py)
  - Event Handling (control_LED_buzzer.py)

# Polling

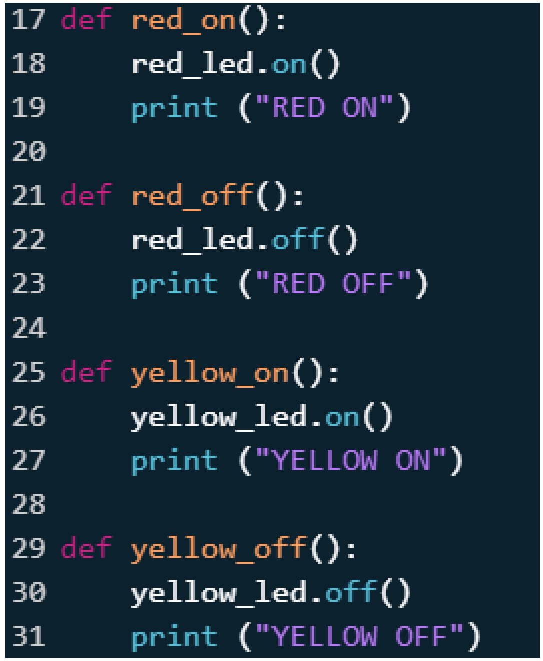
- Polling means we wait for a loop to check things one by one, add too many things and the robot will appear to become unresponsive, but in reality, the robot was probably checking  if up is pressed when you pressed a down in your keyboard and the input will look like it was dropped. Try pressing the key longer, now it responds!



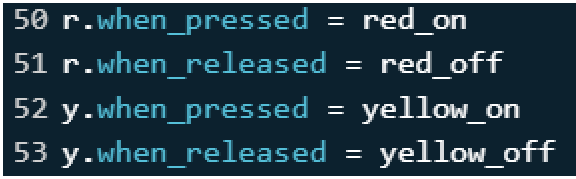
```
while True:
    if up.is_pressed:
        print ("Up is pressed")
        sleep(.5)
    if down.is_pressed:
        print ("Down is pressed")
        sleep(.5)
    if left.is_pressed:
        print ("Left is pressed")
        sleep(.5)
    if right.is_pressed:
        print ("Right is pressed")
        sleep(.5)
```

# Event Handling

- Unlike polling, events are handled on threads, what are those!? Think of them as little processes that will just listen to keys being press. With events you can immediately react to a key being pressed instead of relying on a loop being in the right position at the right time.



```python
def red_on():
    red_led.on()
    print ("RED ON")

def red_off():
    red_led.off()
    print ("RED OFF")

def yellow_on():
    yellow_led.on()
    print ("YELLOW ON")

def yellow_off():
    yellow_led.off()
    print ("YELLOW OFF")
```



```python
r.when_pressed = red_on
r.when_released = red_off
y.when_pressed = yellow_on
y.when_released = yellow_off
```

# Rover Race

- Can you create your own controller to drive on a race?
- Will you use polling or event handling?
- Will you want to go full speed (1) or less than that?

# Further Challenge: Rover Race (Tutorial USB Keyboard)

- How responsive is the robot using a loop? Did it get slower the more keys you added?
- How did your robot design affect your performance in the race?

# Further Course: Rover Vision

- Robotics 101 [7]: Rover Vision
- (Optional) Robotics 101 [8]: Path Finder

# Robotics 101 [7]: Rover Vision

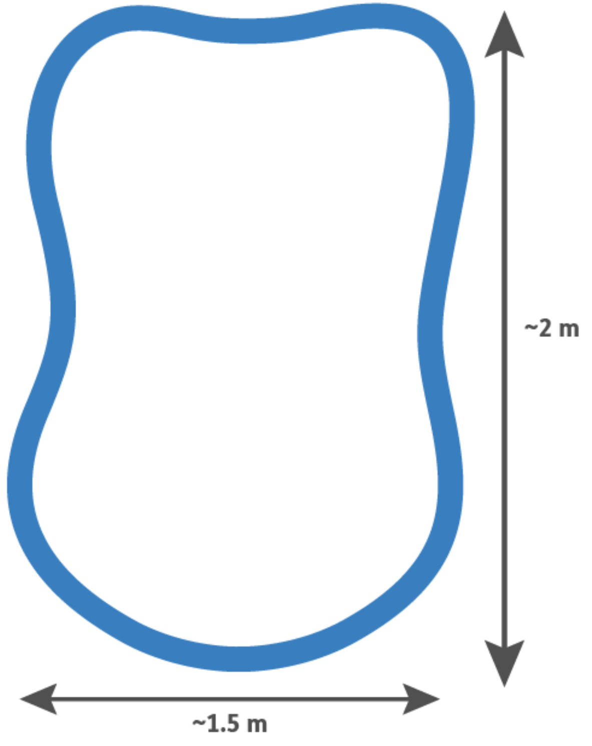
- Ensure you have the standard Alex Build and a USB camera for this lesson
- (Optional) Run the zero.py and follow the instructions on Further to calibrate the servos
- Explore the servos!

# Robotics 101 [7]: Rover Vision

- What does the robot see?
  - Run camera5.py
  - Verify the camera has the correct orientation
- What can we do with the camera feed?
  - The camera component has a cool event handler called “on_frame” that runs on every frame it receives.

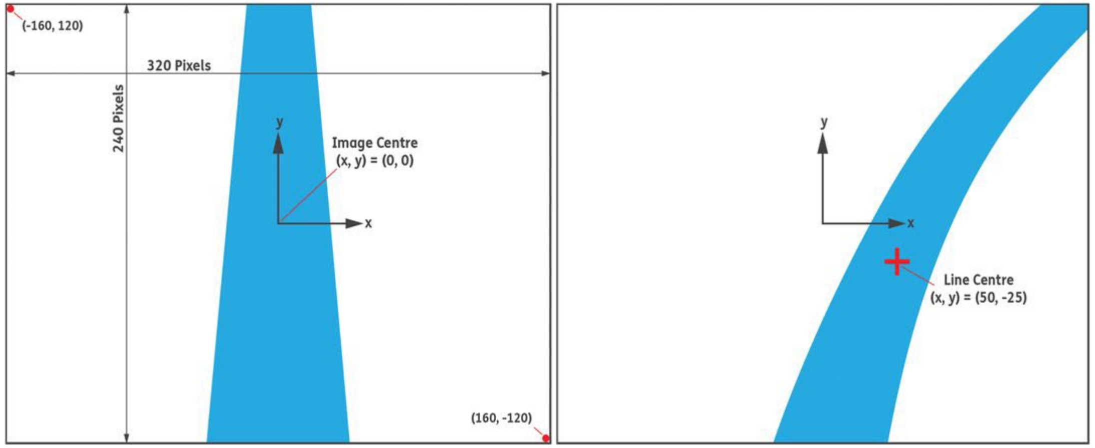
# Robotics 101 [7]: Rover Vision

- If we want to build a line follower rover, we'll need a line to follow - grab the blue tape that was included in your Robotics Kit. Set up a line track on your floor based on the following:
- Keep the track simple with gentle curves, we will make a more advanced one later!
- You can make it whatever size you want, but approximately 1.5m x 2m should be big enough.



# Robotics 101 [7]: Rover Vision

- How does the robot follow a line with only a camera?
  - The Pitop has a built-in color detector and can find the blue tape edges, with this we can find the middle section of the blue edges with some math.
  - The objective is to keep the center of the camera in the center of the tape.



# Robotics 101 [7]: Rover Vision

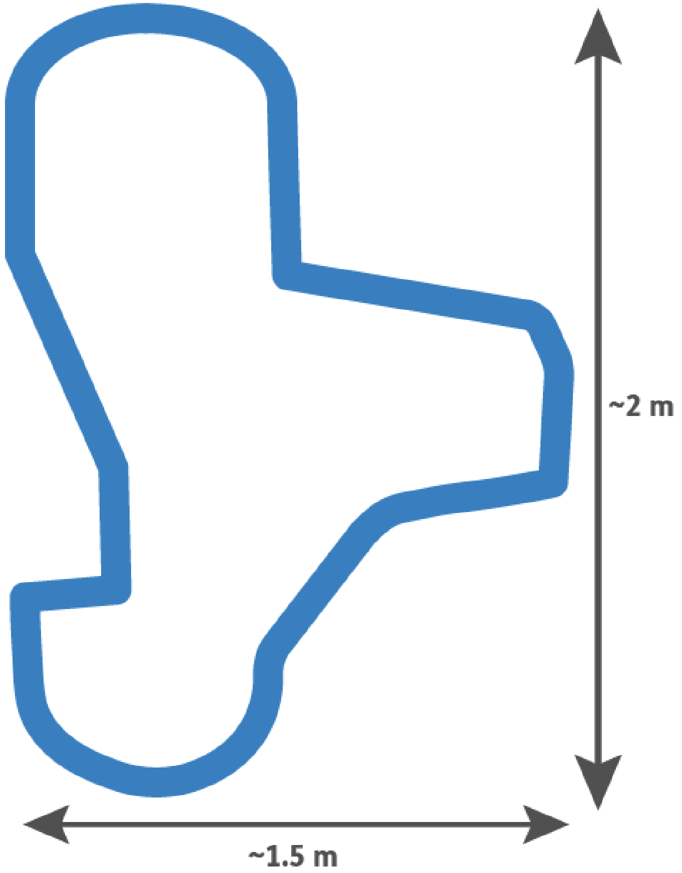
- Using 7line.py complete the code in 8LRM.py and build a line following robot!

# Robotics 101 [7]: Rover Vision

- What is the fastest forward speed value you achieved?
- List and describe two real life examples where a line detection process is used. If you plan to use a search site to help you answer the question, be sure to reference the source(s).
- If the processes from this lesson were to be applied in real-world applications, what potential issues could arise for the user?
- What is the benefit of using functions in your code?

# Robotics 101 [8]: Path Finder

- Build a better version of the line following robot from the previous version.
- Try to complete a track on the right



# Day 4

- Collision Simulation
- Forklift Assembly Challenge

# Further Challenge: Collision Simulation

- You are locomotive engineer driving at top speed when suddenly a truck tries to go around the railway crossing gates and gets stuck, you have no time to break and crash with the truck
- What happens? You've probably can guess the car is completely destroyed
- How far does the train drag the car along?

# Further Challenge: Collision Simulation

- Consider a robot and a box involved in a collision. We want to calculate the kinetic energy before and after the collision and the work done by the robot on the box.
- What is Kinetic Energy?
- What is Work?

# Kinetic Energy

- Energy of Motion
- Imagine the scenario of the robot rolling across the floor toward the cardboard box. This robot isn't just sitting still; it's moving—maybe slowly, maybe quickly. When it crashes into the box, something happens: the box might slide across the floor, tip over, or even get crushed, depending on how fast and heavy the robot is.
- The robot has something called kinetic energy. Kinetic energy is the energy an object possesses because of its motion. Whenever something is moving, it carries energy with it, and this energy can be transferred to other objects when they interact.

$$\text{Kinetic Energy (KE)} = \frac{1}{2} \times mass(m) \times velocity(v)^2$$

# Understand Kinetic Energy

- Kinetic Energy depends on the object’s mass (weight in kilograms(Kg)) and its current velocity (speed in a direction).
- Take a moment and reflect what would be the difference of a toy car crashing into a wall at 10 m/s (22.3 mph), versus a train crashing into a wall at 10 m/s. Which one would cause more damage? A toy car weight approx. 0.04 Kilograms (0.08 lb), and a train weights approx. 15,000 tons or 15000000 Kg

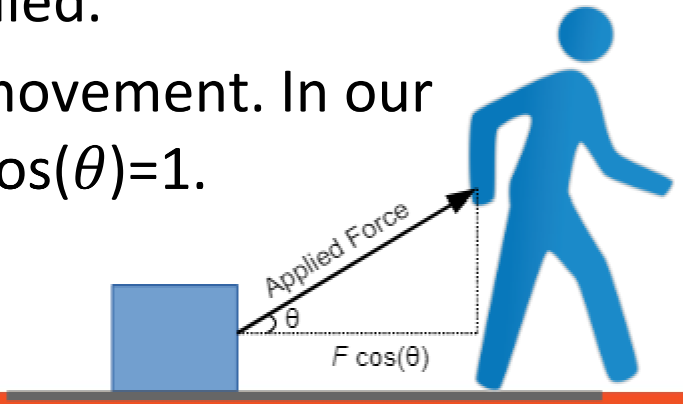
$$\text{Kinetic Energy (KE)}=\frac{1}{2}\times mass(m)\times velocity(v)^2$$

# Work

- In everyday language, "work" can mean many things, but in physics, it has a very specific definition:

$$\text{Work (W)} = \text{Force (F)} \times \text{Distance (d)} \times \cos(\theta)$$

- Force (F): The push or pull acting on an object. Equal to the product of mass and acceleration of the object.
- Distance (d): How far the object moves while the force is applied.
- Theta (θ): The angle between the force and the direction of movement. In our case, the force and movement are in the same direction, so cos($\theta$)=1.



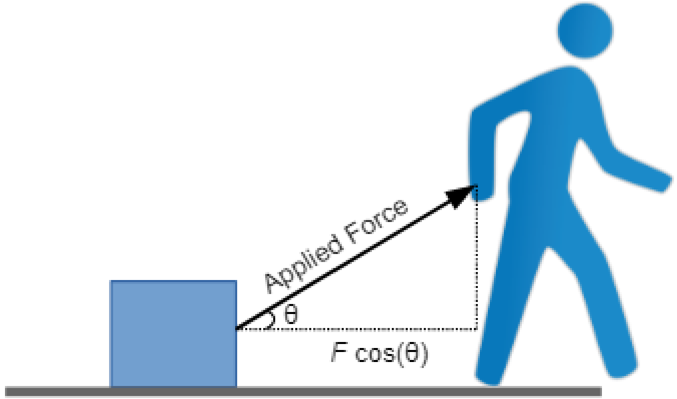
# Work (Simplified)

- Since the force and movement are in the same direction:

$$\text{Work (W)=Force (F)}\times\text{Distance (d)}$$

- Work can also be calculated with the change in Kinetic Energy

$$\text{Work (W)=}KE_{final} - KE_{inital}$$



# Run your Experiment

- Run the experiment with your own Alex robot.
- The approximate weight of the standard Alex build is 3.18 kilograms.
- Repeat the crash several times to account for random factors such as friction, motor malfunction, etc.

# Further Challenge: Forklift Assembly

- In this challenge you will use your knowledge on Encoder and Servo motors to design and build a forklift to move a train wheel-axle assembly to predestined locations.

# What is a Wheel-Axle Assembly?

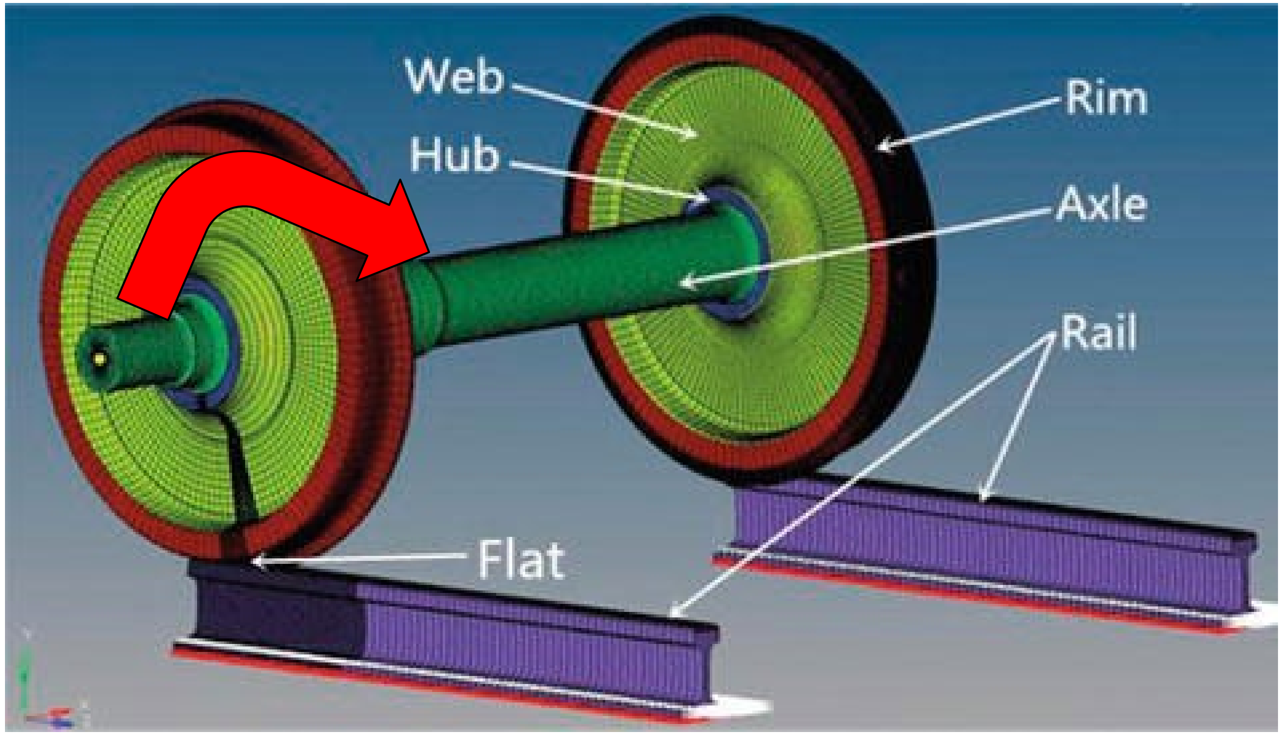
The axle connects both wheels where they move together while traveling at the same speed.

Approximately weighs 400 pounds (depending on size & material)

# Wheel-Axle Assembly

The axle connects both wheels where they move together while traveling at the same speed.

Approximately weighs 400 pounds (depending on size & material)



# How Are They Damaged?

Most of the time is due to **sudden braking(stopping)**.

The **more weight & load the freight train carries, the more dangerous** it can become.

# Train Derailments

What caused the train derailment in Ohio? Could it have been prevented?

# What can we do?

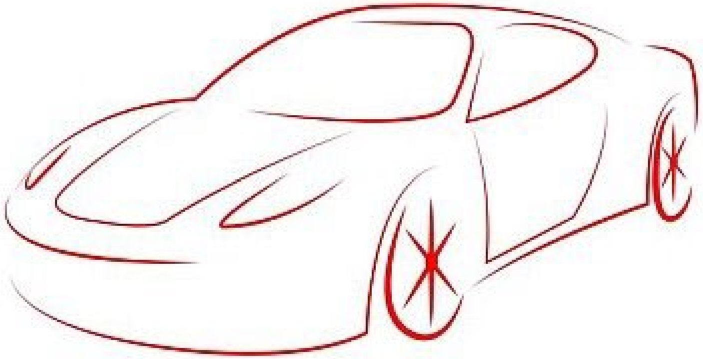
Severe braking over time can cause **fractures, scrapings & deterioration of the Wheel-Axle Assembly**...which can then cause **accidents**.

→

This needs **maintenance** very frequently. This is where Rail Flaw Detection can be used to **avoid** such accidents and failures.

# Also…

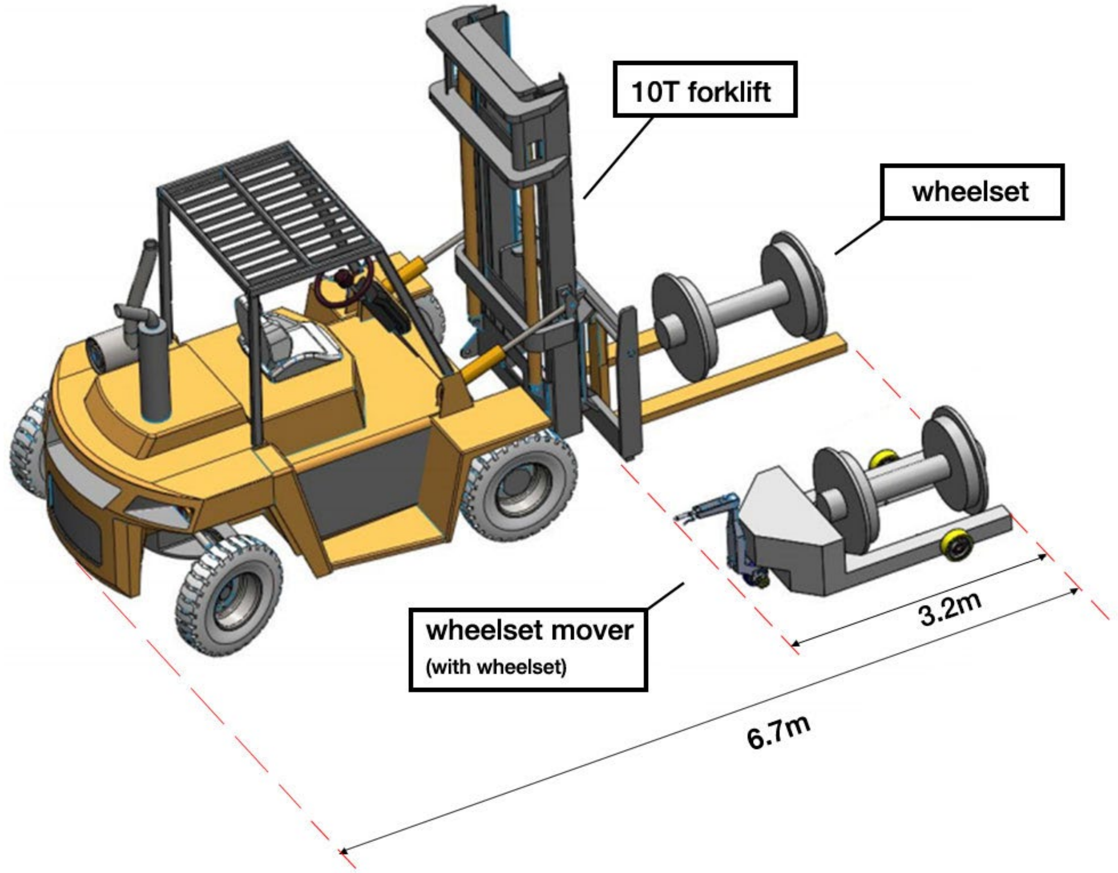
Just like we change the wheels of our cars very often (for every 50,000 miles traveled).....



....Freight trains need an exchange of Wheel-Axle Assemblies for every 500,000 miles as well.

# How Are They Moved?

Wheel-Axle Assemblies are moved by **forklifts** to make the process more **efficient(easy)**. Helps place Wheel-Axle Assemblies at a desirable spot as well.



# Build and Program a Forklift

# Day 5

Final Competition

# Final Competition

Instructions

- In this final competition the goal is to run a track as fast as possible without getting off track, or crashing into any obstacles, all while driving the robot through its camera feed only.

Rules before competition starts

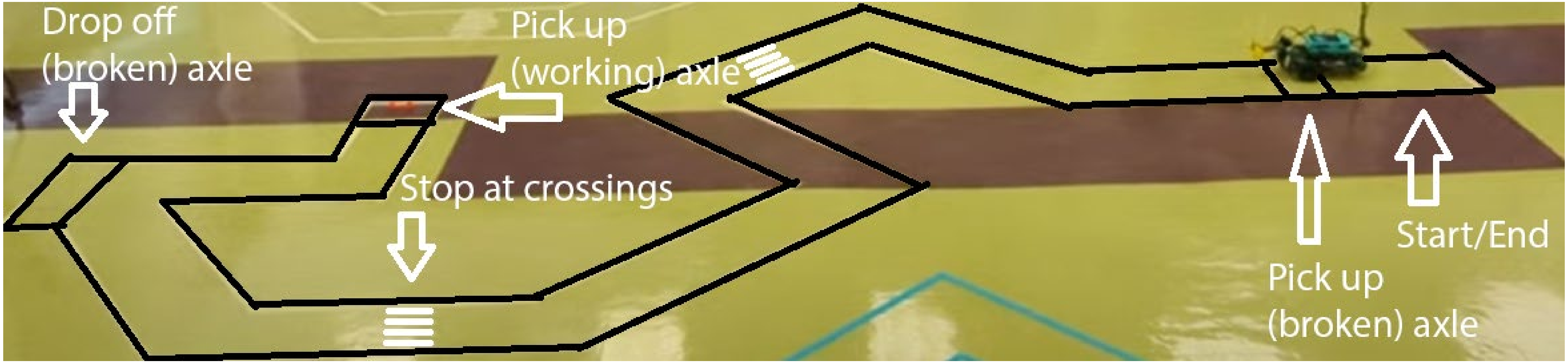
- You can test any section of the course as much as you want.
- Once the official competition begins you have 3 tries showing the final solution.

Rules during the competition

- You must run the entire track without looking at the robot, your only communication will be through the camera feed you will get from the robot. (Camera, Keyboard needed)
- You must pick up the broken wheel-axle assembly at the start of the track. (Forklift needed)
- Carry the wheel-axle assembly all the way to its drop off location.
- Pick-up the working wheel-axle assembly and bring it back to the beginning of the course.
- Stop at pedestrian crossings only if there are pedestrians crossing (Ultrasonic sensor needed)
- (Optional) Use the light sensor and LEDs to detect low lighting and turn on your headlights automatically.
- (Optional) Use the buzzer and LEDs to beep and blink your back lights when going in reverse.

**Failure to follow any of the rules above will result in a time penalty proportional to the rule broken.**

# Final Competition



# References

Josh Darnit (n.d.). *Exact Instructions Challenge - THIS is why my kids hate me. | Josh Darnit*. Exact Instructions Challenge - THIS is why my Kids Hate me. | Josh Darnit. Retrieved October 7, 2024, from https://www.youtube.com/watch?v=cDA3_5982h8 (Slide 18)

(2015, February 7). Why Did The Zebra Cross The Road? The Ranty Highwayman. Retrieved October 7, 2024, from https://therantyhighwayman.blogspot.com/2015/02/why-did-zebra-cross-road.html (Slide 34)

(n.d.). Sound Sensors. STEM Basics. Retrieved October 7, 2024, from https://www.youtube.com/watch?v=H3JacSVipVA&t=4s (Slide 39)

(n.d.). How Do Ultrasonic Distance Sensors Work? - The Learning Circuit. Element14 Presents. Retrieved October 7, 2024, from https://www.youtube.com/watch?v=2ojWO1QNprw&t=1s (Slide 40)

(n.d.). How does a photoresistor work. STEM Basics. Retrieved October 7, 2024, from https://www.youtube.com/watch?v=u9Riurh4y9U (Slide 41)

(2022). Terrifying Video Shows Train Crashing Into Truck. NDTV. Retrieved October 7, 2024, from https://www.youtube.com/watch?v=tMJK_PnZgzU&t=1s (Slide 42)

(2021). Pi-top [4] Robotics Kit: Build instructions - Alex. Pi-Top Team. Retrieved October 7, 2024, from https://www.youtube.com/watch?v=8c-T1KmL0lY (Slide 44)

(2009, February 6). Driver cited in Kent truck-train crash. Kent Reporter. Retrieved October 7, 2024, from https://www.kentreporter.com/news/driver-cited-in-kent-truck-train-crash/ (Slide 67 & 68)

(n.d.). What does energy and work mean? Khan Academy. Retrieved October 7, 2024, from https://www.khanacademy.org/science/physics/work-and-energy/work-and-energy-tutorial/a/what-is-work (Slide 71)

Jing, L. (2018). Wheel-Rail Impact by a Wheel Flat. InTech. doi: 10.5772/intechopen.70460 (Slide 76)

(n.d.). Moving rail bogie wheelsets with Electrodrive's Powered Wheelset Mover. Electrodrive. Retrieved October 7, 2024, from https://www.electrodrive.com.au/our-solutions/moving-train-bogies.aspx (Slide 77 & 81)

(2023, February 24). 3News Investigates: Railway experts weigh in on axle inspection after East Palestine train derailment. WKYC. Retrieved October 7, 2024, from https://www.wkyc.com/article/news/investigations/railway-experts-weigh-in-axel-inspection-east-palestine-train-derailment/95-8ca7d329-0d09-4603-bd4e-e7f70968b095 (Slide 78)

(2023). Pi-top. https://www.pi-top.com/

(2024). Welcome to Python. Python. https://www.python.org/