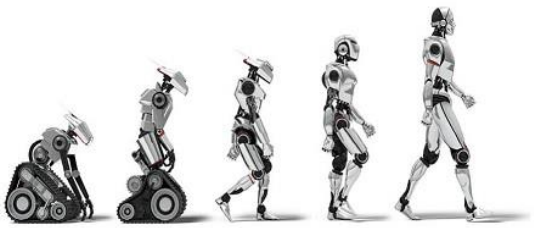
# Robotics – Edison

# Reinforcement Handout

## What is Robotics?

Robotics is a branch of engineering that involves the conception, design, manufacture, and operation of robots. This field overlaps with electronics, computer science, artificial intelligence, mechatronics, nanotechnology and bioengineering.

Whenever the word ROBOTICS is used we think of a machine which would have metal arms, legs, bionic eyes and the robotic voice. This machine is called humanoid or android. Robotics is all about automation of any process or task. A robot can contain numerous electronic or non-electronic parts like motors, battery, chassis, wirings, sensors, computer boards for programming etc.



Science-fiction author Isaac Asimov is often given credit for being the first person to use the term robotics in a short story composed in the 1940s. In the story, Asimov suggested three principles to guide the behavior of robots and smart machines. Asimov's Three Laws of Robotics, as they are called, have survived to the present:

1. Robots must never harm human beings.
2. Robots must follow instructions from humans without violating rule 1.
3. Robots must protect themselves without violating the other rules

## Why Robotics?

Robots make our life easier and much safer, robots can do tasks which are dangerous for humans like bomb diffusion, security guard, and mars rover etc. Besides the dangerous tasks robots can perform tasks which humans are not good at such as:


**Safety:** Safety is the most obvious advantage of utilizing robotics. Heavy machinery, machinery that runs at hot temperature, and sharp objects can easily injure a human being. By delegating dangerous tasks to a robot.

**Speed, consistency & Production:** Robots don't get distracted or need to take breaks. They don't request vacation time or ask to leave an hour early. A robot will never feel stressed out and start running slower. They also don't need to be invited to employee meetings or training session. Robots can work all the time, and this speeds up production. Robots never need to divide their attention between a multitudes of things. Their work is never contingent on the work of other people. They won't have unexpected emergencies, and they won't need to be relocated to complete a different time sensitive task. They're always there, and they're doing what they're supposed to do. Automation is typically far more reliable than human labor.

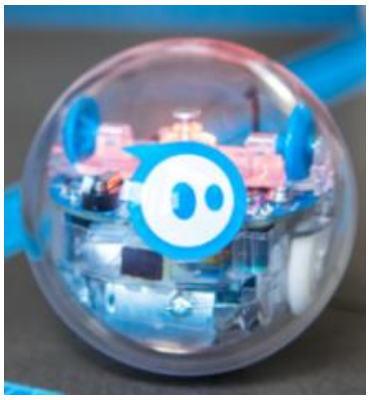
**Perfection & Accuracy:** Robots will always deliver quality. Since they're programmed for precise, repetitive motion, they're less likely to make mistakes. In some ways, robots are simultaneously an employee and a quality control system. A lack of quirks and preferences, combined with the eliminated possibility of human error and the outcomes are quite accurate than the humans.

**Job Creation:** Robots don't take jobs away. They merely change the jobs that exist. Robots need people for monitoring and supervision. The more robots we need, the more people we'll need to build those robots. By training your employees to work with robots, you're giving them a reason to stay motivated in their position with your company. They'll be there for the advancements and they'll have the unique opportunity to develop a new set of tech or engineering related skills.

Robots are taking over the world. OK, not really. Not yet. But they are becoming increasingly prevalent in almost every industry, from healthcare and manufacturing to defense and education.

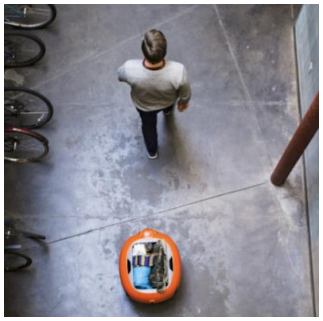
Here is the list of the organizations working on robotics as life essentials:

SPHERO



Sphero invented a now world-famous app-enabled robotic ball, which is used in classrooms all over the world to teach through play. In addition to the original ball, other products include the Sphero 2.0 and the Sphero Mini as well as app-enabled racing robots named Ollie and Darkside. The company's Sphero Edu app is a hub for programming its robots and more.

GITA



From the Piaggio Group that brought you the Vespa scooter comes Piaggio Fast Forward; a robotics company dedicated to creating lightweight mobility solutions for people and goods. The company's flagship robot, gita, is a mobile carrier that follows people around and carries up to 45 pounds. Gita can be used to carry everything from heavy books between classes to groceries.

MOXI

Diligent's AI-enabled robots are designed to work with people in everyday environments. The company's autonomous "Moxi" robot can be left alone to perform time-consuming logistical tasks in hospitals like setting up patient rooms and restocking supply rooms. Capable of navigating hospital hallways and other tight spaces, Moxi is even imbued with "social intelligence" that's conveyed through its head movements and LED eyes.

ANYBOTS


Equipped with a speaker, camera and video screen, Anybots robots serve as remote avatars that are controlled through a browser-based interface and connect to the Web over Wi-Fi. Say you're in Chicago and you want to also be in Taiwan. Your robot — which has a built-in guidance system, live video streaming capabilities and is steered with the arrow keys on your computer's keyboard — can act as a stand-in.

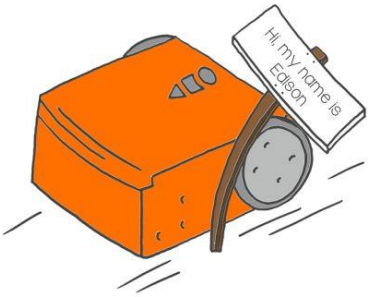
ATLAS


Boston Dynamics makes a host of different robots that have human- and animal-like dexterity. A few examples: There's SpotMini, "a nimble robot that handles objects, climbs stairs, and will operate in offices, homes and outdoors"; Atlas, a "dynamic humanoid" that "uses balance and whole-body skills to achieve two-handed mobile manipulation"; and WildCat, a speedy quadruped that "uses a galloping gait much like a dog or horse and leans into turns in order to maintain traction and balance."


HV-100

According to Harvest, its HV-100 model was the world's "first fully autonomous robot that works alongside people in unmodified industrial environments." Today, more than 30 of them serve major agricultural players across the U.S. to help increase productivity, efficiency and plant quality. Harvest's robots lesson the load when it comes to manual labor so their human counterparts can focus on other facets of the growing process.

## Meet Edison:




Edison robots are a complete STEAM teaching resource designed to bring coding to life. Edison is expandable robotics system, which works with any LEGO brick compatible building system, the robots can be used as the programmable base for an incredible assortment of engineering and STEM projects.

Edison can be programmed through 4 platforms:

- Barcodes and Remote Control
- EdBlocks – Graphical Language
- EdScratch – Scratch Language interface
- EdPy – Python Programming Interface

Edison is equipped with all the sensors, outputs and motors needed to introduce you to the amazing world of robotics.

Edison's creator, Brenton O'Brien says that "*A robot is a machine that can behave autonomously*" which is also a simple definition of robotics.

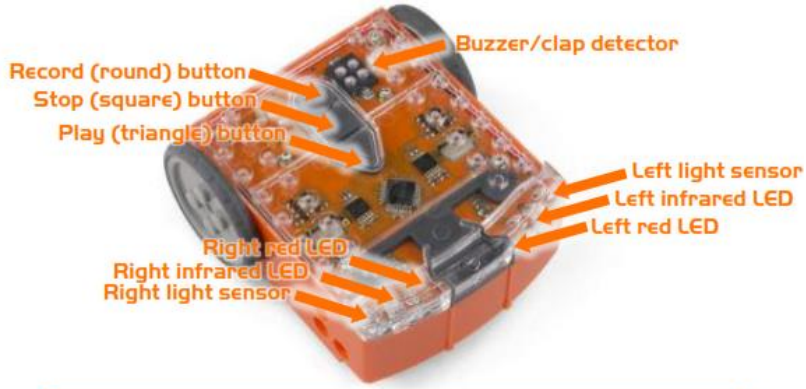
Robotics wouldn't be possible without electronics. Your Edison robot has its own electronics which you can see through the transparent top. There are resistors, capacitors, transistors, motors and more. The most important electronic part is Edison's microcontroller.

The microcontroller is like Edison's brain. It's where all the robot's thinking' happens. Edison's microcontroller is very similar to the processor chip inside a computer, only much smaller. Just like a processor chip in a computer, Edison's microcontroller contains programs. These programs are what allow Edison to 'think' and make decisions.

Edison carries three hardware buttons. With the record button, you download a program to the robot, the play button, starts the execution of the program and by pressing the stop button, the program is stopped. It is also equipped with different sensors in order to understand its environment and interact with it.

The following YouTube channel is very beneficial to develop the coding skills for programming Edison robot through EdScratch and EdPy: https://www.youtube.com/user/microbric/videos

**Edison's sensors, buttons and switches:**
To use Edison, you're going to need to know where all of Edison's sensors are located and become familiar with the robot's three buttons.





Play (triangle) button – Press to start a program
Stop (square) button – Press to stop a program
Record (round) button – 1 press = download a program from a programming app
3 presses = read a barcode

EdPy is Edison's text-based programming application. The robot is controlled by a program, which contains the instructions and rules governing the robot's behavior. Any program you create in EdPy must be downloaded to Edison before the robot can perform the actions. The program can be changed only in the EdPy platform and in such cases, you will need to download the new program to the robot again.

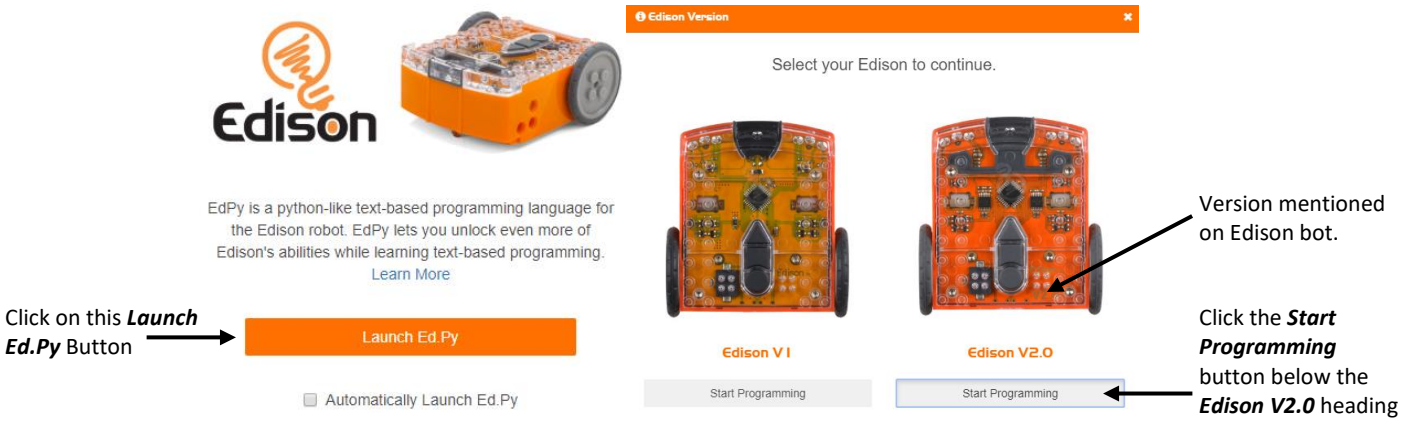
To connect Edison with the computer in order to download the program you create with EdPy, you need the EdComm cable which is Edison's special cable. The EdComm cable is used to download programs to Edison. It connects into the headphone/audio Socket/jack on your computer or tablet or phone or any other computing device which contains an audio jack and a working web browser to access EdPy app.

**Important Note:** Before you begin programming with EdPy you **MUST** make sure that your computer volume is **MAX** and it is important that **ALL sound enhancements are disabled**. This includes Altec Lansing enhancements, Realtec audio enhancements, Beats audio, Spacial sound, and all equalizers. Edison will fail to program if this is not done.

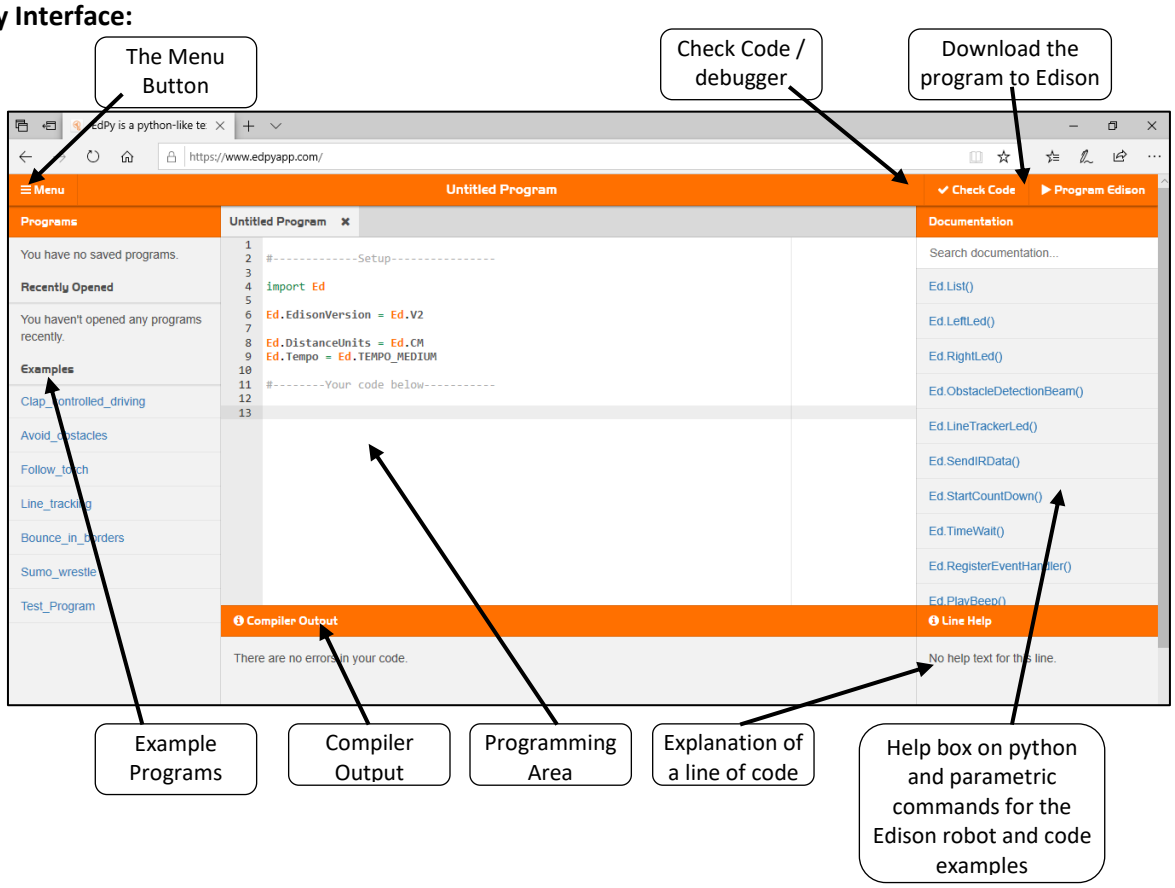
## EdPy App:

EdPy app is one the web based programming interface to program Edison. EdPy is a text-based programming language based on Python. Python, gives more flexibility to your programs and offers more precise control of your robot. EdPy application can be accessed online and without any installation on your computer or tablet.

In order to program Edison through EdPy App access this web: https://www.edpyapp.com/



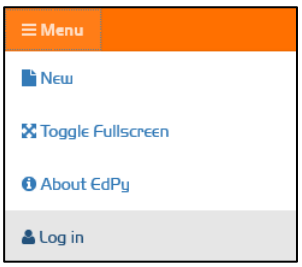

**EdPy Interface:**



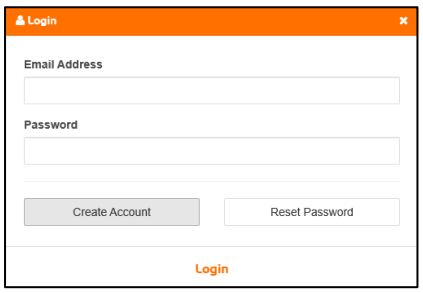
## EdPy Online Account:

Using the online EdPy programming environment you can save your work online and have access to your projects from any browser. All you need to do is to create an account and Login. The online EdPy programming environment can be used with or without an account. Without an account, you don't have the option to save a project so that you can work on it later.

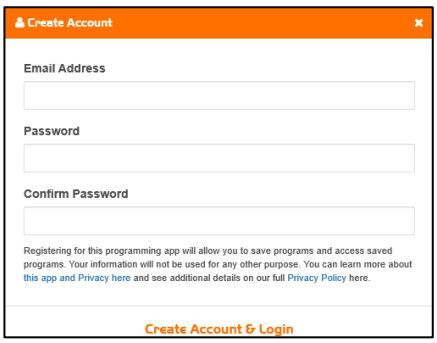
Step1: Click on ***Menu***




Step2: Click on ***Create Account***




Step3: Enter details and click on ***Create Account & Login*** to proceed




## Working with EdPy:

It is highly recommended to work on EdPy while logged into your account so all the work will be available/save for your convenience.

The Programming area in EdPy has numbered lines. All Edison programs must contain the setup code which is included in lines 1 to 11. When you start a new program, you should start typing in line 13. When you finish writing you should check the program for errors with the Check Code button. If the program has any errors they appear in the Compiler Output area.

The following program, will turn on the left LED of the robot for 10 seconds.

```
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.LeftLed(Ed.ON)
14 Ed.TimeWait(10, Ed.TIME_SECONDS)
```

There is no need to enter the mentioned below code and it will be available always whenever you start a new program, however you can customize the parameters as per your requirement.

- Import Ed is a predefined code to import the library of Edison functions in python.
- Ed.EdisonVersion = Ed.V2 is the code to define the version of Edison we are using which is Version 2.
- Ed.DistanceUnits = Ed.CM is the code to define the unit of measurements to drive the Edison.
- Ed.Tempo = Ed.TEMPO_MEDIUM is the code to define the music/sound pace.

After this line **#-------Your code below ---------** enter the code to work on Edison.

- Ed.LeftLed(Ed.ON) is to turn on the left LED of Edison
- Ed.TimeWait(10, Ed.TIME_SECONDS) is the code where we have define the interval between the LED to turn on.

EdPy works on python and will work with the same rules use for python like, case sensitivity and indentation, therefore you need to be very careful entering the code according to python style.
**Create the above mentioned program and save it on the online account.**

## Edison Drive Functions with EdPy:

The Edison robot has two motors that allow it to perform a handful of movements. With its two motors, the robot can move forward and backward and also spin both left and right.

To move your Edison robot you use the drive function, which has three parameters: direction, speed and distance.

The constant "Ed.DistanceUnits" that is set in the Setup code controls the measurement of distance. There are three distance units: centimeters which is written as Ed.CM, inches which is written as Ed.INCH and time which is written as Ed.TIME.

```
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(direction, speed, distance)
```

**Ed.Drive Direction Parameters:**

**Ed.FORWARD -** Edison drives forwards.
**Ed.BACKWARD -** Edison drives backwards.
**Ed.FORWARD_RIGHT -** Edison uses one wheel to turn forwards right (clockwise).
**Ed.BACKWARD_RIGHT -** Edison uses one wheel to turn backwards right (counterclockwise).
**Ed.FORWARD_LEFT -** Edison uses one wheel to turn forwards left (counterclockwise).
**Ed.BACKWARD_LEFT -** Edison uses one wheel to turn backwards left (clockwise).
**Ed.SPIN_RIGHT -** Edison spins on the spot to the right (clockwise).
**Ed.SPIN_LEFT -** Edison spins on the spot to the left (counterclockwise).
**Ed.STOP -** Stops Edison immediately.

To make Edison drive forward/backward enter this code:

- While entering the code you will observe that a code hint menu will appear, from which we can select the entire code by pressing tab and there would be no need to type the whole code. Using the code hint menu will help you program the code in less time and above all, there will be less chance for syntax errors.
- Check the code from **Check Code** button from the right top corner.
- Click on **Program Edison** button right top corner.
- Make sure the Edison is connected to your computer's headphone jack via the EdComm Cable.
- And also make sure your computer's volume at maximum and all sound enhancements are disabled.

```
11 #--------Your code below-----------
12
13 Ed.Drive(Ed.FORWARD, Ed.SPEED_4, 9)
```

```
11 #--------Your code below-----------
12
13 Ed.Drive(Ed.BACKWARD, Ed.SPEED_3, 7)
```

```
Ed.Drive(Ed.FORWARD, ed.sp, distance)
    Ed.SPEED_FULL   Edison
    Ed.SPEED_10     Edison
    Ed.SPEED_9      Edison
    Ed.SPEED_8      Edison
    Ed.SPEED_7      Edison
    Ed.SPEED_6      Edison
    Ed.SPEED_5      Edison
    Ed.SPEED_4      Edison
```

Always remember that while using the Ed.FORWARD and Ed.BACKWARD command, the "Distance" parameter works as **number of steps(cm/inch) Edison has to move.**

```
11 #--------Your code below-----------
12
13 Ed.Drive(Ed.FORWARD, Ed.SPEED_4, 9)
```

Whereas, while using Ed.FORWARD_RIGHT, Ed.FORWARD_LEFT, Ed.SPIN_RIGHT, Ed.SPIN_LEFT, the "Distance" parameter works as **the angle at which Edison has to turn.** This means that a code like "Ed.Drive(Ed.SPIN_LEFT, Ed.SPEED_5, 90)" will make the Edison robot turn 90 degrees counterclockwise.

```
11  #--------Your code below-----------
12
13  Ed.Drive(Ed.FORWARD_RIGHT,Ed.SPEED_5,120)
14  Ed.Drive(Ed.FORWARD,Ed.SPEED_5,10)
```

**Creating a square with Edison:**

After successfully programming the Edison to move backward and forward, program the Edison to move in square form.

- Ed.Drive(Ed.FORWARD, Ed.SPEED_6, 20) - This code will drive Edison forward to 20cm at the speed of 6
- Ed.Drive(Ed.SPIN_LEFT, Ed.SPEED_6, 90) - This code will rotate the Edison to 90 degree at the speed of 6

```
13  Ed.Drive(Ed.FORWARD, Ed.SPEED_6, 20)
14  Ed.Drive(Ed.SPIN_LEFT, Ed.SPEED_6, 90)
15  Ed.Drive(Ed.FORWARD, Ed.SPEED_6, 20)
16  Ed.Drive(Ed.SPIN_LEFT, Ed.SPEED_6, 90)
17  Ed.Drive(Ed.FORWARD, Ed.SPEED_6, 20)
18  Ed.Drive(Ed.SPIN_LEFT, Ed.SPEED_6, 90)
19  Ed.Drive(Ed.FORWARD, Ed.SPEED_6, 20)
```

## Using Variables for Edison

A Python variable is a reserved memory location to store values. In other words, a variable in a python program gives data to the computer for processing.

```
11  #--------Your code below-----------
12  degreesToTurn = 90
13  Ed.Drive(Ed.SPIN_RIGHT,Ed.SPEED_6,degreesToTurn)
```

This program will make Edison robot turn right using a value from the variable. ***degreesToTurn = 90*** is the variable with the parameters of 90. This variable is going to use as a reference in the program to turn the Edison robot to 90 degree.

Most important function to use the variable is to have flexibility in program, rather than entering data directly into a program, we use variables to represent the data. Then, when the program is executed, the variables are replaced with real data.

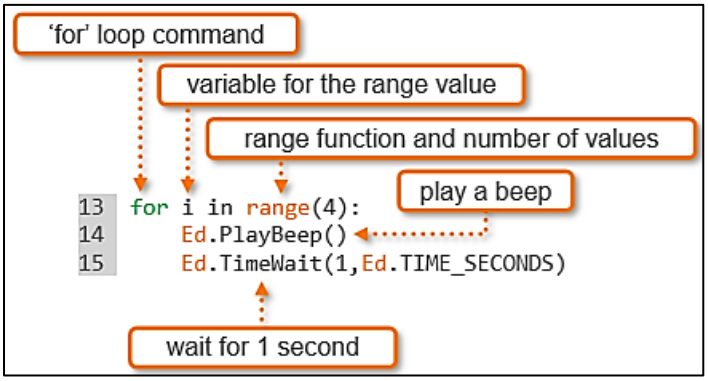
## Using Loops to program Edison

### For Loop

In Python, a 'for' loop is a control structure which can be used to repeat sets of commands or statements any number of times.

Using a 'for' loop allows you to repeat (also called 'iterate over') a block of statements as many times as you like.

The 'for' loop often goes together with the 'range()' function in Python.




In EdPy, range() only has one input parameter. That input parameter determines the upper limit of the set and the lower limit is always 0, or in a simpler definition it can be used to define the times of loop should repeat itself.

**Creating a square with Edison:**

Using loop to create a program which has several repetitive code is a smart decision. Following program will create a square using a loop.

```
12  for x in range(4):
13      Ed.Drive(Ed.FORWARD, Ed.SPEED_5, 1)
14      Ed.Drive(Ed.SPIN_LEFT, Ed.SPEED_5, 90)
```

**Creating a Circle with Edison:**

It may not be possible to drive in a perfect circle, but a shape with thousands of very small sides can closely approximate a circle. **degreesToTurn** is a variable which is having a value of 1.

```
degreesToTurn = 1
for x in range(360):
    Ed.Drive(Ed.FORWARD, Ed.SPEED_5, 1)
    Ed.Drive(Ed.SPIN_LEFT, Ed.SPEED_5, degreesToTurn)
```

**Perform the following practical task:**

Create a program to make your robot drive forward for 3 seconds then flash both LED lights. After that, using the For loop, Edison is to drive in the form of a square and stop.

**While Loop**

While loop repeats a statement or group of statements while a given condition is TRUE. It tests the condition, which is written as an expression, before executing the loop body.

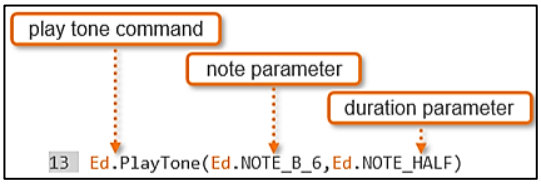
While the expression evaluates to TRUE, the program repeats the commands in the loop. When the expression evaluates to FALSE, the program moves on to the next line of code outside the loop.

In this program, we want the Edison to follow a torch and we need this program to run infinitely therefore we will use the while loop.

```
#loop forever
while True:
    if Ed.ReadLeftLightLevel()>Ed.ReadRightLightLevel():
        #If the left light level is higher, drive to the left
        Ed.Drive(Ed.FORWARD_LEFT, Ed.SPEED_4, Ed.DISTANCE_UNLIMITED)
    else:
        #otherwise, the light is on the right so drive to the right
        Ed.Drive(Ed.FORWARD_RIGHT, Ed.SPEED_4, Ed.DISTANCE_UNLIMITED)
```

## Play tunes on Edison:

Edison can play individual musical notes through its small speaker using the Ed.PlayTone() function in EdPy.




The Ed.PlayTone() function takes two input parameters: the note and the duration. The note determines what note to play and the duration determines the given length of time the note should be played.

**Music Notes for Edison**

| Parameter input options | Plays musical note |
|---|---|
| Ed.NOTE_A_6 | low A |
| Ed.NOTE_A_SHARP_6 | low A sharp |
| Ed.NOTE_B_6 | low B |
| Ed.NOTE_C_7 | C |
| Ed.NOTE_C_SHARP_7 | C sharp |
| Ed.NOTE_D_7 | D |
| Ed.NOTE_D_SHARP_7 | D sharp |
| Ed.NOTE_E_7 | E |
| Ed.NOTE_F_7 | F |
| Ed.NOTE_F_SHARP_7 | F sharp |
| Ed.NOTE_G_7 | G |
| Ed.NOTE_G_SHARP_7 | G sharp |
| Ed.NOTE_A_7 | A |
| Ed.NOTE_A_SHARP_7 | A sharp |
| Ed.NOTE_B_7 | B |
| Ed.NOTE_C_8 | high C |

**Duration Parameters**

| Parameter input options | Plays note for |
|---|---|
| Ed.NOTE_SIXTEENTH | 125 milliseconds |
| Ed.NOTE_EIGHTH | 250 milliseconds |
| Ed.NOTE_QUARTER | 500 milliseconds |
| Ed.NOTE_HALF | 1,000 milliseconds |
| Ed.NOTE_WHOLE | 2,000 milliseconds |

## A Robot with Logic and Senses:

An important part of coding is making decisions. The most common way to do this is to use an 'if statement'. An '***if*** statement' asks whether a condition is true or false. If the result is true, then the program executes the block of statements following the '***if*** statement'. If the result is false, the program ignores the statements inside the '***if*** statement' and moves to the next line of code outside of the '***if*** statement'.

```
#--------Your code below-----------
Ed.ObstacleDetectionBeam(Ed.ON)

while True:
    if Ed.ReadObstacleDetection() != Ed.OBSTACLE_NONE:
        Ed.PlayBeep()
        Ed.ReadObstacleDetection()
```

For Edison we can create a program which makes a decision using more than two conditions. To do this, you use another Python syntax structure:

- ***if*** expression:
  statement(s)
- ***elif*** expression:
  statement(s)
- ***else***:
  statement(s)

***'Elif'*** is how you say 'else if' in Python. You can use ***elif*** to write a program with multiple ***if*** conditions.

A program using ***if/elif/else*** still moves sequentially from the top down. Once the program runs any indented code inside any part of the ***if*** statement structure, it will skip the rest of the structure and move on to the next line of code outside the structure.

This means that if the ***if*** statement at the top is true, the program runs the indented code for the ***if*** expression and skips any ***elif*** sections as well as the ***else*** section if there is one. If the ***if*** statement is false, however, the program skips that section of indented code and moves to the first ***elif*** section.

Again, if the first ***elif*** condition is true, the program runs its indented code and skips everything below it in the ***if*** statement structure (any other ***elifs*** and the ***else*** condition if there is one). If this ***elif*** condition is false, the program moves to the next part of the ***if*** statement structure and so on.

This program has three different paths that it can take when an obstacle is detected based on where the detected obstacle is in relation to Edison.

```
#--------Your code below-----------
Ed.ObstacleDetectionBeam(Ed.ON) #turn on obstacle detection

while True:
    Ed.Drive(Ed.FORWARD, Ed.SPEED_1, Ed.DISTANCE_UNLIMITED)

    obstacle=Ed.ReadObstacleDetection()

    if obstacle>Ed.OBSTACLE_NONE: #there is an obstacle
        Ed.Drive(Ed.BACKWARD, Ed.SPEED_5, 7)
        if obstacle==Ed.OBSTACLE_LEFT:
            Ed.Drive(Ed.SPIN_RIGHT, Ed.SPEED_5, 90)
        elif obstacle==Ed.OBSTACLE_RIGHT:
            Ed.Drive(Ed.SPIN_LEFT, Ed.SPEED_5, 90)
        elif obstacle==Ed.OBSTACLE_AHEAD:
            Ed.Drive(Ed.SPIN_RIGHT, Ed.SPEED_5, 180)
        Ed.ReadObstacleDetection() #clear any unwanted detections

```

**Perform the following practical task:**

Write the above program using the EdPy app and download it to your Edison robot. Then run the program to see how it works.

***From this point onwards we will make programs in EdPy using the variables, loops, conditions, motors and sensors of Edison.***

## Obstacle Detection:

Edison has two obstacle detection beams, one on the front left and other is on the front right.

This program tells Edison to drive until it encounters an obstacle.

There are a couple of important things to notice about this program.

```
#--------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.STOP,1,1)

```

Look at line 13 of the program. This line turns Edison's obstacle detection beam to 'on'. Whenever you want to use Edison's obstacle detection beam in an EdPy program, you always need to turn the beam to 'on' before the beam is used in the program.

Now, look at line 15 of the program. This line sets the speed to 5 in this program. When using obstacle detection, you need to use a slightly lower speed to allow the robot to detect an obstacle before colliding with it. If the speed is too fast, the robot will crash into obstacles before being able to detect them.

**Right and left obstacle detection:**
In this program Edison is going react to obstacles on the left or right. To do this, we will use conditional statements like ***if*** and ***elif***.

This program has three different paths that it can take when an obstacle is detected based on where the detected obstacle is in relation to Edison.

```
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
```

**Perform the following practical task:**
Write the above program using the EdPy app and download it to your Edison robot. Then run the program to see how it works.

## Clap Control Drive:

The Edison robot's sound sensor is not just sensitive to claps. The sensors can respond to any loud sound detected or vibrations similar to that sound, which is why you can tap near the speaker on the robot to trigger the sound sensor.
Edison's motors, gears and wheels all make sounds as they turn, which can trigger the sound sensor. To prevent the sound of the robot driving from triggering the sound sensor, you need to alter the program.

```
11  #--------Your code below-----------
12  Ed.Drive(Ed.FORWARD,Ed.SPEED_8,10)
13
14  Ed.TimeWait(350,Ed.TIME_MILLISECONDS)
15  Ed.ReadClapSensor()
16  while Ed.ReadClapSensor() == Ed.CLAP_NOT_DETECTED:
17      pass
18  Ed.Drive(Ed.BACKWARD,Ed.SPEED_8,10)
```

You will need to add a TimeWait() function call with an input parameter of about 350 milliseconds to give the robot's motors time to stop.

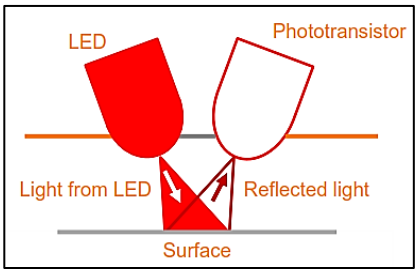
You also need to use a ReadClapSensor() to clear the clap sensor.

**Perform the following practical task:** Write the above program using the EdPy app and download it to your Edison robot. Then run the program to see how it works.

## Line Tracking Sensor:

In this program, you will learn about the Edison's line tracking sensor and how Edison can use this sensor to determine if it is on a reflective or non-reflective surface. How does Edison's line tracking sensor work?

Your Edison robot is equipped with a line tracking sensor, located near the power switch on the bottom of the robot. This sensor is made up of two main electronic components:
A red light emitting diode (LED), and A phototransistor (light sensor).
This image represents a cross-section of Edison's line tracking sensor.
The line tracking sensor's LED shines light onto the surface that the Edison robot is driving on.
The phototransistor component is a light sensor. The phototransistor measures the amount of light that is reflected back up from the surface beneath Edison.



In this activity, you will write a program so that your Edison robot will drive forward on a white (reflective) surface until a black (non-reflective) line is crossed.

```
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)
```

Look at line 13. This line calls the function Ed.LineTrackerLed() and turns the state to 'on'.

Just like with Edison's obstacle detection beam, to use the line tracking sensor in a program, you must first turn the sensor on. Turning the line tracking sensor on will also activate the line tracker's red LED.

Now, look at line 19. This line calls the Ed.PlayBeep() function. This line doesn't affect the way the line tracking program works. Instead, this line's purpose is for debugging.

**Practical task:**

Write a program that will keep Edison inside a black border using the robot's line tracking sensor.
By this program Edison robot will bounce in border.

```
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
```

**Light Sensors:**

In this activity, we will use a program to have your Edison robot turn the two LED lights on when it gets dark.

```
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)
```

In this program, we are using the 'less than' (<) symbol to determine the path that the program will take.

If the value returned from the Ed.ReadLeftLightLevel() function is less than 100, then the activateBothLights() function is called with the input parameter of Ed.ON. Otherwise, the program moves to the 'else' part of the if statement, which also calls the activateBothLights() function, but with the input parameter of Ed.OFF.

**Light following**:

In this activity, we will write a program so that your Edison robot will follow the light from a torch (flashlight).

```
11  #--------Your code below-----------
12
13  while True:
14      if Ed.ReadRightLightLevel() - Ed.ReadLeftLightLevel() < 0:
15          Ed.Drive(Ed.FORWARD_LEFT, Ed.SPEED_5, Ed.DISTANCE_UNLIMITED)
16      else:
17          Ed.Drive(Ed.FORWARD_RIGHT, Ed.SPEED_5, Ed.DISTANCE_UNLIMITED)
```

This program compares the light level between the right light sensor and the left light sensor to determine the flow of the program.

The presence of the torch on either the left or right of the robot will cause the robot to read a higher light level on that side of the robot. The logic of this program says that when the right light level minus the left light level is less than zero, the robot drives left towards the higher source of light, else the robot drives to the right.

**Perform the following practical task:** Write the above program of light sensor and light following using the EdPy app and download it to your Edison robot. Then run the program to see how it works.