Lesson 6

Comparison Operators

CS IN SCHOOLS

Learning objectives

By the end of this lesson, you should be able to:

- understand a variety of comparison operators
- write conditional statements using comparison operators
- understand how to apply the `or` operator to check multiple conditional statements

Revision: while

Can you spot the 5 errors in this code?

```
# Connect to the micro:bit
m1 = Micro_bit()

// Enter a loop
keep_going = true
while keep_going == True:
    # Get state of button A
    buttonA = m1.getButtonA()
    buttonB = m1.getTheButtonB()

    Primt("A: " + str(btnA) + ", B: "+str(btnB))

    # So the loop doesn't spam
    sleep(0.5)
```

Revision: while

Can you spot the 5 errors in this code?

```
# Connect to the micro:bit
m1 = Micro_bit()

// Enter a loop
keep_going = true
while keep_going == True:
    # Get state of button A
    buttonA = m1.getButtonA()
    buttonB = m1.getTheButtonB()

    Primt("A: " + str(btnA) + ", B: "+str(btnB))

    # So the loop doesn't spam
    sleep(0.5)
```

Micro_bit() → Microbit()

// → #

true → True

.getTheButtonB() → getButtonB()

Primt → print

Revision: while

Why do we now use `while` instead of `label` and `goto`?

```
# Connect to the micro:bit
m1 = Microbit()

# Enter a loop
keep_going = True
while keep_going == True:
    # Get state of button A
    buttonA = m1.getButtonA()
    buttonB = m1.getButtonB()

    print("A: " + str(btnA) + ", B: "+str(btnB))

    # So the loop doesn't spam
    sleep(0.5)
```

Revision: while

Why do we now use `while` instead of `label` and `goto`?

- To prevent spaghetti code!

```
# Connect to the micro:bit
m1 = Microbit()

# Enter a loop
keep_going = True
while keep_going == True:
    # Get state of button A
    buttonA = m1.getButtonA()
    buttonB = m1.getButtonB()

    print("A: " + str(btnA) + ", B: "+str(btnB))

    # So the loop doesn't spam
    sleep(0.5)
```

Today's Inputs and Outputs

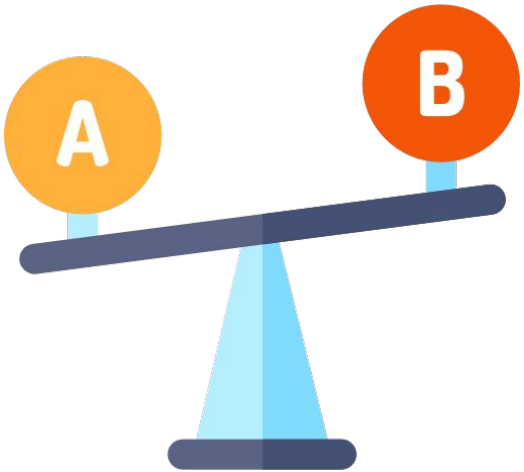
Not used yet
Used
New

- Today we will be using the following **inputs** and **outputs**:

	Input	Output	Processing	Communication
Your computer	Keyboard Mouse Touch screen Microphone	Monitor/Screen Speakers	CPU Graphics cards	Wifi Bluetooth Ethernet
Your Microbit	Buttons Thermometer Accelerometer Magnetometer Touch sensor Light sensor	25 x LED lights Speakers		Bluetooth Radio

Comparison Operators

- **Comparison operators** are usually used in code to compare numbers
- You may have used these in maths!
- On the next slide, we have some commonly used **comparison operators**, and we will go through how they work



Comparison Operators: equal to (==)

- This is the symbol for equal to: `==`
- It checks if the value on the left is the **same as** the value on the right

```
14 == 14
```

- Is 14 the **same as** 14? Yes - `True`!

Comparison Operators: not equal to (!=)

- This is the symbol for not equal to: `!=`
- It checks if the value on the left is **not equal to** the value on the right

```
18 != 14
```

- Is 18 **not equal to** 14? Yes - `True`!

Comparison Operators: == and !=

- Look at the statements below, and decide whether they are `True` or `False`

`12 == 11`	True / False	`11 != 11`	True / False
`90 == 92`	True / False	`19 != 52`	True / False
`30 == 30`	True / False	`36 != 36`	True / False
`42 == 39`	True / False	`22 != 89`	True / False

Comparison Operators: == and !=

- Look at the statements below, and decide whether they are `True` or `False`

`12 == 11`	True / **False**	`11 != 11`	True / **False**
`90 == 92`	True / **False**	`19 != 52`	**True** / False
`30 == 30`	**True** / False	`36 != 36`	True / **False**
`42 == 39`	True / **False**	`22 != 89`	**True** / False

Comparison Operators: greater than (>)

- This is the symbol for greater than: `>`
- It checks if the value on the left is **greater than** the value on the right

```
18 > 12
```

- Is 18 **greater than** 12? Yes - `True`!

Comparison Operators: less than (<)

- This is the symbol for less than: `<`
- It checks if the value on the left is **less than** the value on the right

$$14 < 22$$

- Is 14 **less than** 22? Yes - `True`!

Comparison Operators: > and <

- Look at the statements below, and decide whether they are `True` or `False`

`12 > 11`	True / False	`10 < 11`	True / False
`90 > 92`	True / False	`99 < 52`	True / False
`12 > 30`	True / False	`14 < 36`	True / False
`42 > 39`	True / False	`22 < 89`	True / False

Comparison Operators: > and <

- Look at the statements below, and decide whether they are `True` or `False`

`12 > 11`	**True** / False	`10 < 11`	**True** / False
`90 > 92`	True / **False**	`99 < 52`	True / **False**
`12 > 30`	True / **False**	`14 < 36`	**True** / False
`42 > 39`	**True** / False	`22 < 89`	**True** / False

Comparison Operators: greater than or equal to (>=)

- This is the symbol for greater than or equal to: `>=`
- It checks if the value on the left is **greater than** or **equal to** the value on the right

```
18 >= 12
```

- Is 18 **greater than or equal to** 12? Yes, greater than - `True`!

Comparison Operators: less than or equal to (<=)

- This is the symbol for less than or equal to: `<=`
- It checks if the value on the left is **less than** or **equal to** the value on the right

14 <= 22

- Is 14 **less than or equal to** 22? Yes, less than - `True`!

Comparison Operators: >= and <=

- Look at the statements below, and decide whether they are `True` or `False`

`10 >= 11`	True / False	`19 <= 11`	True / False
`92 >= 92`	True / False	`19 <= 52`	True / False
`12 >= 30`	True / False	`36 <= 36`	True / False
`42 >= 39`	True / False	`22 <= 89`	True / False

Comparison Operators: >= and <=

- Look at the statements below, and decide whether they are `True` or `False`

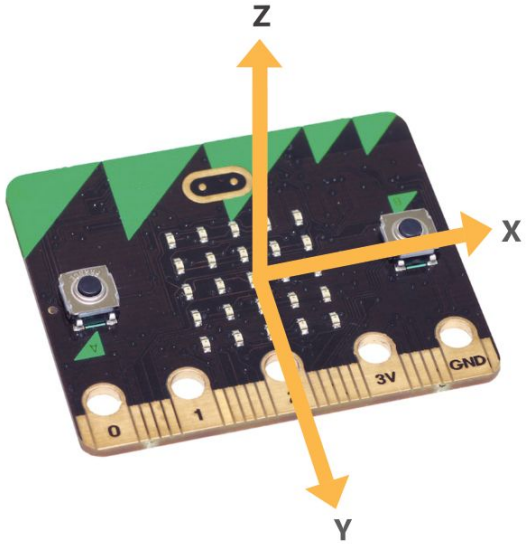
`10 >= 11`	True / **False**	`19 <= 11`	True / **False**
`92 >= 92`	**True** / False	`19 <= 52`	**True** / False
`12 >= 30`	True / **False**	`36 <= 36`	**True** / False
`42 >= 39`	**True** / False	`22 <= 89`	**True** / False

Comparison Operators

- Python uses **comparison operators** as part of `if` statements and `while` loops
- We can use them in our code to run certain sections of code when certain conditions are met
- An example of this is a fall detector

Comparison Operators

- Remember the **accelerometer**?
- This gave us the acceleration on our micro:bit in the X, Y and Z directions
- When the device is resting on the table, LED screen up, the acceleration in the Z direction is negative (less than 0)
- We can use this fact to write a program to detect when the micro:bit is upside down



Demo: Fall Detector

Fall Detector

Comparison in Code

```
# Connect to the micro:bit
r2d2 = Microbit()

# Enter a loop
keep_going = True
while keep_going == True:
    # Get acceleration in Z direction
    accZ = r2d2.getAccelerometerZ()

    # Show current acceleration
    print("Z: " + str(accZ))

    # Check if the acceleration is greater than 0
    # This means the device is upside down!
    if accZ > 0:
        keep_going = False
        say("Help!")

    # Sleep so the program doesn't spam
    sleep(0.25)
```

Comparison in Code

- Connecting to the micro:bit

```
# Connect to the micro:bit
r2d2 = Microbit()

# Enter a loop
keep_going = True
while keep_going == True:
    # Get acceleration in Z direction
    accZ = r2d2.getAccelerometerZ()

    # Show current acceleration
    print("Z: " + str(accZ))

    # Check if the acceleration is greater than 0
    # This means the device is upside down!
    if accZ > 0:
        keep_going = False
        say("Help!")

    # Sleep so the program doesn't spam
    sleep(0.25)
```

Comparison in Code

- Set `keep_going` to be `True`

```
# Connect to the micro:bit
r2d2 = Microbit()

# Enter a loop
keep_going = True
while keep_going == True:
    # Get acceleration in Z direction
    accZ = r2d2.getAccelerometerZ()

    # Show current acceleration
    print("Z: " + str(accZ))

    # Check if the acceleration is greater than 0
    # This means the device is upside down!
    if accZ > 0:
        keep_going = False
        say("Help!")

    # Sleep so the program doesn't spam
    sleep(0.25)
```

Comparison in Code

- Here we enter the `while` loop, because `keep_going` is `True`

- This means the code indented and underneath will run

```
# Connect to the micro:bit
r2d2 = Microbit()

# Enter a loop
keep_going = True
while keep_going == True:
    # Get acceleration in Z direction
    accZ = r2d2.getAccelerometerZ()

    # Show current acceleration
    print("Z: " + str(accZ))

    # Check if the acceleration is greater than 0
    # This means the device is upside down!
    if accZ > 0:
        keep_going = False
        say("Help!")

    # Sleep so the program doesn't spam
    sleep(0.25)
```

Comparison in Code

- What is happening in these sections of code?

```
# Connect to the micro:bit
r2d2 = Microbit()

# Enter a loop
keep_going = True
while keep_going == True:
    # Get acceleration in Z direction
    accZ = r2d2.getAccelerometerZ()

    # Show current acceleration
    print("Z: " + str(accZ))

    # Check if the acceleration is greater than 0
    # This means the device is upside down!
    if accZ > 0:
        keep_going = False
        say("Help!")

    # Sleep so the program doesn't spam
    sleep(0.25)
```

Comparison in Code

- What is happening in these sections of code?
  - Getting the Z axis acceleration and storing it inside accZ
  - Print to the console what is stored inside accZ

```
# Connect to the micro:bit
r2d2 = Microbit()

# Enter a loop
keep_going = True
while keep_going == True:
    # Get acceleration in Z direction
    accZ = r2d2.getAccelerometerZ()

    # Show current acceleration
    print("Z: " + str(accZ))

    # Check if the acceleration is greater than 0
    # This means the device is upside down!
    if accZ > 0:
        keep_going = False
        say("Help!")

    # Sleep so the program doesn't spam
    sleep(0.25)
```

Comparison in Code

- In this line of code, Python is checking to see if the acceleration is greater than 0

- When the device is sitting on the table, LED screen up, the value of accZ is approximately -9.6 m/s^2

- Is this great than 0?

```
# Connect to the micro:bit
r2d2 = Microbit()

# Enter a loop
keep_going = True
while keep_going == True:
    # Get acceleration in Z direction
    accZ = r2d2.getAccelerometerZ()

    # Show current acceleration
    print("Z: " + str(accZ))

    # Check if the acceleration is greater than 0
    # This means the device is upside down!
    if accZ > 0:
        keep_going = False
        say("Help!")

    # Sleep so the program doesn't spam
    sleep(0.25)
```

Comparison in Code

- In this line of code, Python is checking to see if the acceleration is greater than 0
- When the device is sitting on the table, LED screen up, the value of accZ is approximately -9.6 m/s^2
- Is this great than 0? `False`!

```
# Connect to the micro:bit
r2d2 = Microbit()

# Enter a loop
keep_going = True
while keep_going == True:
    # Get acceleration in Z direction
    accZ = r2d2.getAccelerometerZ()

    # Show current acceleration
    print("Z: " + str(accZ))

    # Check if the acceleration is greater than 0
    # This means the device is upside down!
    if accZ > 0:
        keep_going = False
        say("Help!")

    # Sleep so the program doesn't spam
    sleep(0.25)
```

Comparison in Code

- Is `accZ` greater than 0? `False`!
- So now the code will sleep and then return to the top of our while `loop`
- Why doesn't the code indented and underneath the `if` statement run?

```
# Connect to the micro:bit
r2d2 = Microbit()

# Enter a loop
keep_going = True
while keep_going == True:
    # Get acceleration in Z direction
    accZ = r2d2.getAccelerometerZ()

    # Show current acceleration
    print("Z: " + str(accZ))

    # Check if the acceleration is greater than 0
    # This means the device is upside down!
    if accZ > 0:
        keep_going = False
        say("Help!")

    # Sleep so the program doesn't spam
    sleep(0.25)
```

Comparison in Code

- Is this great than 0? `False`!
- So now the code will sleep and then return to the top of our while `loop`
- Why doesn't the code indented and underneath the `if` statement run? - Because the answer to the condition was `False`!

```
# Connect to the micro:bit
r2d2 = Microbit()

# Enter a loop
keep_going = True
while keep_going == True:
    # Get acceleration in Z direction
    accZ = r2d2.getAccelerometerZ()

    # Show current acceleration
    print("Z: " + str(accZ))

    # Check if the acceleration is greater than 0
    # This means the device is upside down!
    if accZ > 0:
        keep_going = False
        say("Help!")

    # Sleep so the program doesn't spam
    sleep(0.25)
```

Comparison in Code

- Will the loop run again?

```
# Connect to the micro:bit
r2d2 = Microbit()

# Enter a loop
keep_going = True
while keep_going == True:
    # Get acceleration in Z direction
    accZ = r2d2.getAccelerometerZ()

    # Show current acceleration
    print("Z: " + str(accZ))

    # Check if the acceleration is greater than 0
    # This means the device is upside down!
    if accZ > 0:
        keep_going = False
        say("Help!")

    # Sleep so the program doesn't spam
    sleep(0.25)
```

Comparison in Code

- Will the loop run again?
- Yes - because `keep_going` is still `True`!

```
# Connect to the micro:bit
r2d2 = Microbit()

# Enter a loop
keep_going = True
while keep_going == True:
    # Get acceleration in Z direction
    accZ = r2d2.getAccelerometerZ()

    # Show current acceleration
    print("Z: " + str(accZ))

    # Check if the acceleration is greater than 0
    # This means the device is upside down!
    if accZ > 0:
        keep_going = False
        say("Help!")

    # Sleep so the program doesn't spam
    sleep(0.25)
```

Comparison in Code

- So we update accZ
- and print it to the user
- Now this time, suppose the micro:bit is upside down, so:

accZ = +9.64

```
# Connect to the micro:bit
r2d2 = Microbit()

# Enter a loop
keep_going = True
while keep_going == True:
    # Get acceleration in Z direction
    accZ = r2d2.getAccelerometerZ()

    # Show current acceleration
    print("Z: " + str(accZ))

    # Check if the acceleration is greater than 0
    # This means the device is upside down!
    if accZ > 0:
        keep_going = False
        say("Help!")

    # Sleep so the program doesn't spam
    sleep(0.25)
```

Comparison in Code

- So we update `accZ`
- and print it to the user
- Now this time, suppose the micro:bit is upside down, so:

  `accZ = +9.64`

- Now, is this `True`?

```
# Connect to the micro:bit
r2d2 = Microbit()

# Enter a loop
keep_going = True
while keep_going == True:
    # Get acceleration in Z direction
    accZ = r2d2.getAccelerometerZ()

    # Show current acceleration
    print("Z: " + str(accZ))

    # Check if the acceleration is greater than 0
    # This means the device is upside down!
    if accZ > 0:
        keep_going = False
        say("Help!")

    # Sleep so the program doesn't spam
    sleep(0.25)
```

Comparison in Code

- So we update `accZ`
- and print it to the user
- Now this time, suppose the micro:bit is upside down, so:

`accZ = +9.64`

- Now, is this `True`? Yes - the code indented and underneath will run

```
# Connect to the micro:bit
r2d2 = Microbit()

# Enter a loop
keep_going = True
while keep_going == True:
    # Get acceleration in Z direction
    accZ = r2d2.getAccelerometerZ()

    # Show current acceleration
    print("Z: " + str(accZ))

    # Check if the acceleration is greater than 0
    # This means the device is upside down!
    if accZ > 0:
        keep_going = False
        say("Help!")

    # Sleep so the program doesn't spam
    sleep(0.25)
```

Comparison in Code

- Now `keep_going` will be set to `False`, and the micro:bit will call for help!

```
# Connect to the micro:bit
r2d2 = Microbit()

# Enter a loop
keep_going = True
while keep_going == True:
    # Get acceleration in Z direction
    accZ = r2d2.getAccelerometerZ()

    # Show current acceleration
    print("Z: " + str(accZ))

    # Check if the acceleration is greater than 0
    # This means the device is upside down!
    if accZ > 0:
        keep_going = False
        say("Help!")

    # Sleep so the program doesn't spam
    sleep(0.25)
```

Comparison in Code

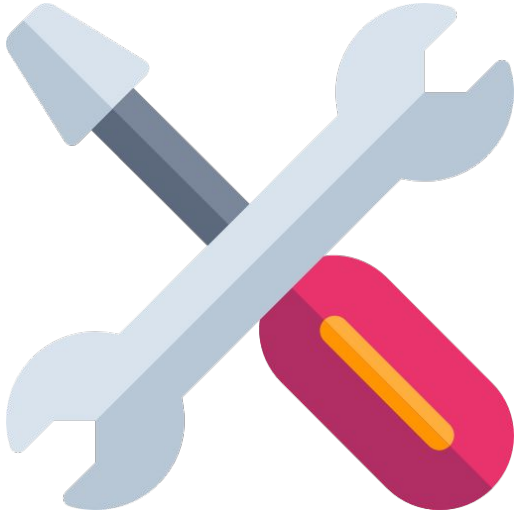
- Now `keep_going` will be set to `False`, and the micro:bit will call for help!

- It will then sleep, and the loop will end because `keep_going` is now `False`

```
# Connect to the micro:bit
r2d2 = Microbit()

# Enter a loop
keep_going = True
while keep_going == True:
    # Get acceleration in Z direction
    accZ = r2d2.getAccelerometerZ()

    # Show current acceleration
    print("Z: " + str(accZ))

    # Check if the acceleration is greater than 0
    # This means the device is upside down!
    if accZ > 0:
        keep_going = False
        say("Help!")

    # Sleep so the program doesn't spam
    sleep(0.25)
```

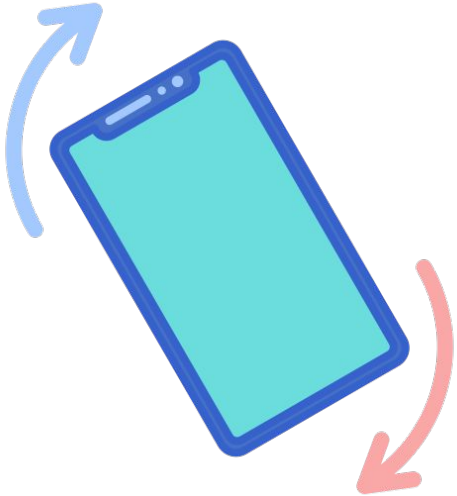
Activity Time!

Exercise 1: Fixing A Fall Detector



Activity 06.01

Exercise 2: X Tilt Checker



Activity 06.02

Exercise 3: Y Tilt Checker

Activity 06.03

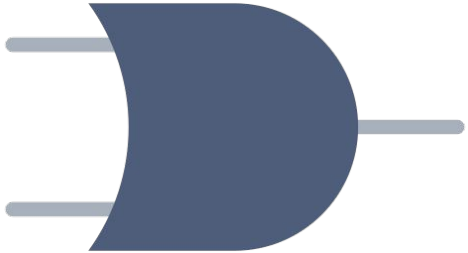
Demo: A or B

A or B

or

- `or` is used in Python to check for multiple conditions

- `or` works the same way in Python as making a choice in real life
  - Imagine you are hungry
  - You can eat a sandwich `or` some fruit, (`or` both) but the outcome is the same either way
  - Your hunger is satisfied



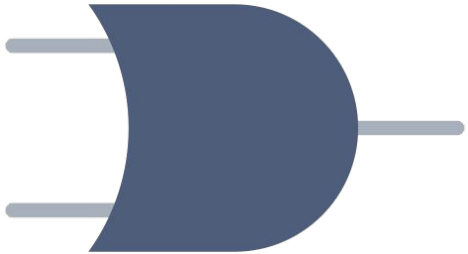
or

- `or` can be used in `while` loops or `if` statements to produce the same outcome for more than one condition

- Imagine we have the line of code:

```
if num > 6 or num < 0:
```

- This would run if `num` was greater than 6, `or` less than 0



or in Code

- This is the code from the demo you just looked at
- What did the demo program do?

```
# Connect to the micro:bit
linda = Microbit()

# Enter a loop
keep_going = True
while keep_going == True:
    # Get button presses
    btnA = linda.getButtonA()
    btnB = linda.getButtonB()

    # Tell user they have pressed a button
    if btnA > 0 or btnB > 0:
        print("Button Pressed")

    # Sleep so the program doesn't spam
    sleep(0.25)
```

or in Code

- This is the code from the demo you just looked at
- What did the demo program do?
- It should have printed “Button Pressed” whenever you pressed button A or button B

```
# Connect to the micro:bit
linda = Microbit()

# Enter a loop
keep_going = True
while keep_going == True:
    # Get button presses
    btnA = linda.getButtonA()
    btnB = linda.getButtonB()

    # Tell user they have pressed a button
    if btnA > 0 or btnB > 0:
        print("Button Pressed")

    # Sleep so the program doesn't spam
    sleep(0.25)
```

or in Code

- This functionality came from this line of code

```
# Connect to the micro:bit
linda = Microbit()

# Enter a loop
keep_going = True
while keep_going == True:
    # Get button presses
    btnA = linda.getButtonA()
    btnB = linda.getButtonB()

    # Tell user they have pressed a button
    if btnA > 0 or btnB > 0:
        print("Button Pressed")

    # Sleep so the program doesn't spam
    sleep(0.25)
```

or in Code

- This functionality came from this line of code
- We know that the `btnA` and `btnB` variables will be `1` when the button is pressed, so by checking if they are greater than `0`, we know if they were pressed

```
# Connect to the micro:bit
linda = Microbit()

# Enter a loop
keep_going = True
while keep_going == True:
    # Get button presses
    btnA = linda.getButtonA()
    btnB = linda.getButtonB()

    # Tell user they have pressed a button
    if btnA > 0 or btnB > 0:
        print("Button Pressed")

    # Sleep so the program doesn't spam
    sleep(0.25)
```

or in Code

- Notice we have two **full conditional statements**:

```
btnA > 0 or btnB > 0
```

- We <u>did not</u> write:

```
btnA or btnB > 0
```

- The above line of code will not work!

```
# Connect to the micro:bit
linda = Microbit()

# Enter a loop
keep_going = True
while keep_going == True:
    # Get button presses
    btnA = linda.getButtonA()
    btnB = linda.getButtonB()

    # Tell user they have pressed a button
    if btnA > 0 or btnB > 0:
        print("Button Pressed")

    # Sleep so the program doesn't spam
    sleep(0.25)
```

or in Code

- We can use `or` as many times as we like in one `if` statement or `while` loop
- Now it's time for you to try it!

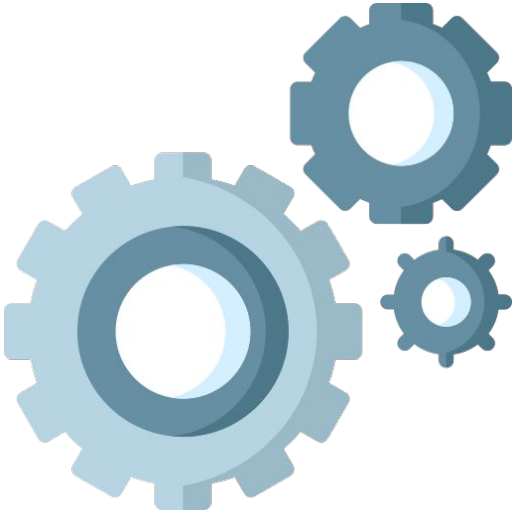
```
# Connect to the micro:bit
linda = Microbit()

# Enter a loop
keep_going = True
while keep_going == True:
    # Get button presses
    btnA = linda.getButtonA()
    btnB = linda.getButtonB()

    # Tell user they have pressed a button
    if btnA > 0 or btnB > 0:
        print("Button Pressed")

    # Sleep so the program doesn't spam
    sleep(0.25)
```

Activity Time!

Exercise 4: Don’t Hurt Micro

Activity 06.04

Worksheet: Input → Process → Output



See lesson 4 for a reminder on IPOs

Worksheet

Summary

- `>` checks if the number on the left is greater than the right
- `<` checks if the number on the left is less than the right
- `>=` checks if the number on the left is greater than or equal to the right
- `<=` checks if the number on the left is less than or equal to the right
- `or` can be used to check multiple conditional statements in one line of code

License Information