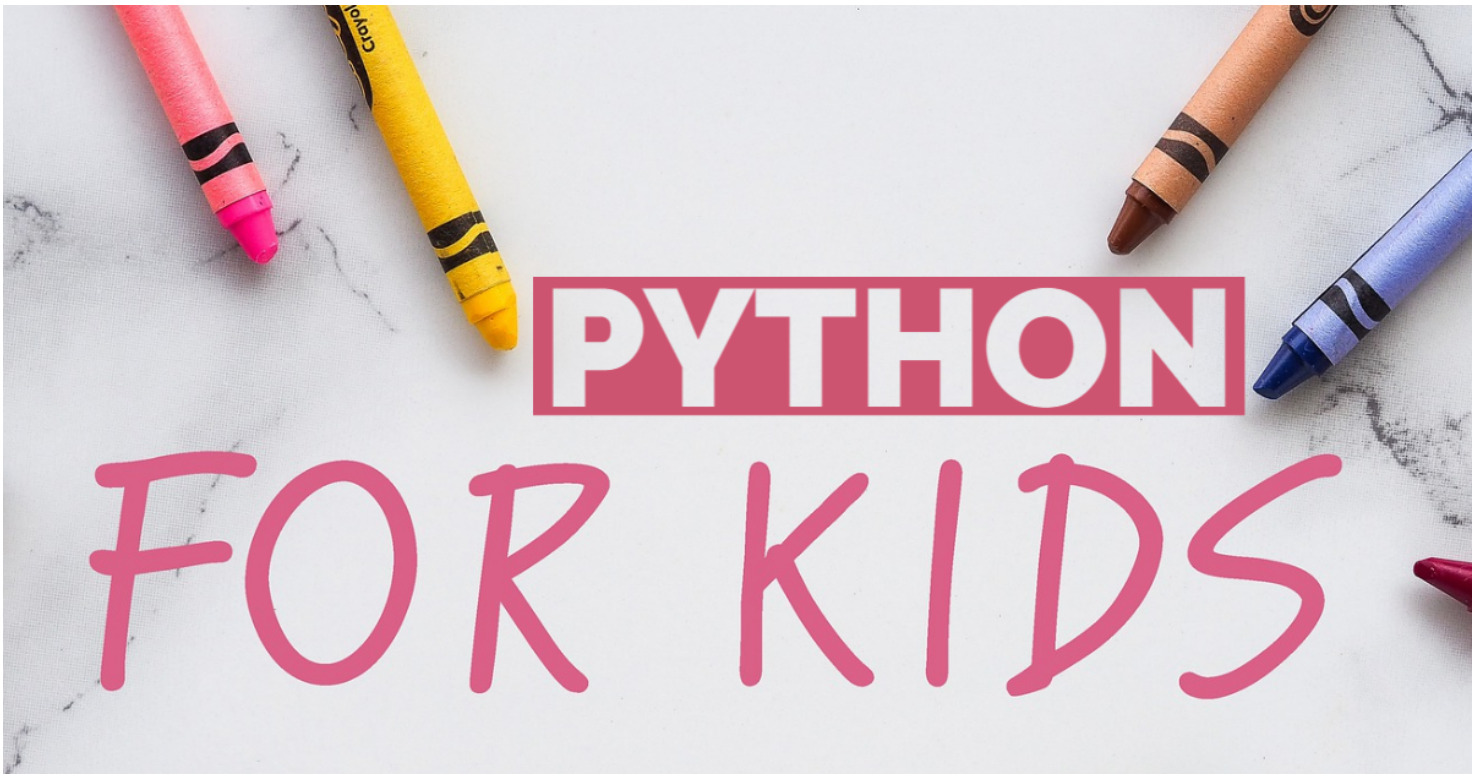


FIRST EDITION – CHAPTER 9 REV 1

Kevin Thomas & Katie Henry

Forward

Over the next decade we will personally witness the full Cyber and Automation revolution in a way even twenty years ago would never have been thought possible.

The future careers will be focused primarily on Computer Engineering. It is fundamental that YOU as a parent have a full appreciation and understanding of this reality and the fact that YOUR child MUST start early in their Engineering development if they are going to thrive in the coming years.

Python For Kids is a bite-sized, step-by-step FREE book and video series designed for EVERYONE. As a parent YOU can on-demand work through these projects together and give them a gift that you would otherwise never be able to provide which is providing them the foundational knowledge to get started in the world of microcontrollers.

I want to thank Katie Henry who is the Head Of Partnerships For North America at the Microbit Educational Foundation for her brilliant contribution and video series for this book.

Katie is also a Former Classroom Teacher and has the experience and vision necessary to help teach and inspire the next generation of Engineers!

If you have questions you can also contact us on our subreddit **r/micropython** as well.

Table Of Contents

Chapter 1: Goals

Welcome to a brand-new series for kids who want to learn the FUN way to program in Python.

This tutorial will be a step-by-step instruction series where parents can learn to code with their kids and non-technical educators with their students and enrich their lives and prepare them for the next generation of work when they mature into adults.

Our goal is to teach kids to have FUN building simple projects and getting familiar with Python in a way that opens their mind toward technical creativity in ways they would never have experienced before! If you are parent this is a great opportunity for you to help set your child into the drivers seat as over the next decade we will continue to transition to a world which is significantly more automated to which future generations (YOUR KIDS) will need to have an upper hand when understanding and interacting with technology in ways you or I never had to imagine if they are to sustain in the coming years.

There will be three things you need to partake in this series. The first is an coding tool we will use to write our simple programs which is FREE provided by the micro:bit Educational Foundation to which we will access via the web.

STEP 1 - Purchase micro:bit V2

Please make sure you select the micro:bit v2 microcontroller from the link below and make sure to check if the USB cable is included in the purchase as that may be sold separately. You can select from any of the official micro:bit retailers.

https://microbit.org/buy

These are the only tools you will need to embark on this incredible journey to get your child prepared for the next IoT generation and solidify their role as a leader in technology!

In our next lesson we will work step-by-step on working with the FREE online micro:bit MicroPython Web Editor to begin programming!

Chapter 2: “Hello World”

Today we will set up our development environment so we can write simple and FUN programs with our new micro:bit so let's get started!

We will then create our first program!

STEP 1: Navigate To Web Editor

https://python.microbit.org/v/beta

```
# Add your Python code here. E.g.
from microbit import *


while True:
    display.scroll('Hello, World!')
    display.show(Image.HEART)
    sleep(2000)
```

STEP 2: Connect Your micro:bit V2 Into Your Computer

STEP 3: Press The Connect Button In The micro:bit MicroPython Web Editor

STEP 4: Click "BBC micro:bit CMSIS-DAP" & CONNECT

STEP 5: Rename Script Name To helloworld_program

STEP 6: Click Flash

STEP 7: WATCH THE micro:bit V2 SCROLL 'Hello, World!' & SHOW A HEART!

Congrats! You did GREAT! You made your first program on the micro:bit V2 and you did a FANTASTIC job!

In our next lesson we will have FUN With Images!

Chapter 3: FUN With Images

Today we are going to make some FUN little images for our little friend to make and interact with us.

If your micro:bit is not connected to the micro:bit MicroPython Web Editor, please follow the steps in Lesson 2, specifically **STEP 2** and **STEP 3**:

STEP 2: Connect Your micro:bit V2 Into Your Computer

STEP 3: Press The Connect Button In The micro:bit MicroPython Web Editor

If you are connected you are all ready to start!

STEP 1: Open micro:bit MicroPython Web Editor

https://python.microbit.org/v/beta

STEP 2: Type Code & Name It images_program

```
from microbit import *


display.show(Image.HAPPY)
```

STEP 3: Click Save

STEP 4: Click Download Python Script

STEP 5: Click Flash

STEP 6: SEE THE FUN AS OUR NEW FRIEND MAKES A SMILE AT US!

(Look at the front of the badge to see the image.)

STEP 7: Let's Create Our OWN Image!

Let's create our very OWN shape! Each LED pixel on the physical display can be set to one of ten values. If a pixel is set to 0 (zero) then it’s off. It literally has zero brightness. However, if it is set to 9 then it is at its brightest level. The values 1 to 8 represent the brightness levels between off (0) and full on (9).

Let's create a little house. We can start by overwriting our image_program file and naming it **my_image_program** and type the following into the editor.

```
from microbit import *


house = Image("00900:"
              "09090:"
              "90009:"
              "05550:"
              "05950")


display.show(house)
```

STEP 8: Click Save

STEP 9: Click Download Python Script

STEP 10: Click Flash

STEP 11: SEE THE FUN AS WE JUST MADE A LITTLE HOUSE!

(Look at the front of the micro:bit to see the image.)

It's your turn! Ask our little friend to draw some of his other favorite images for you! You can try something like this on line 3 by replacing line 3 in our code with one of the below statements to which you will SAVE, DOWNLOAD and FLASH as described above.

You can name each new program anything you like! For example you could do the following for the heart image and after you SAVE, DOWNLOAD and FLASH you can start the process over and try the next one till you get through all of the examples below.

Here’s a list of the built-in images:

```
display.show(Image.HEART)
display.show(Image.HEART_SMALL)
display.show(Image.SMILE)
display.show(Image.SAD)
display.show(Image.CONFUSED)
display.show(Image.ANGRY)
display.show(Image.ASLEEP)
display.show(Image.SURPRISED)
display.show(Image.SILLY)
display.show(Image.FABULOUS)
```

```
display.show(Image.MEH)
display.show(Image.YES)
display.show(Image.NO)
display.show(Image.TRIANGLE)
display.show(Image.TRIANGLE_LEFT)
display.show(Image.CHESSBOARD)
display.show(Image.DIAMOND)
display.show(Image.DIAMOND_SMALL)
display.show(Image.SQUARE)
display.show(Image.SQUARE_SMALL)
display.show(Image.RABBIT)
display.show(Image.COW)
display.show(Image.MUSIC_CROTCHET)
display.show(Image.MUSIC_QUAVER)
display.show(Image.MUSIC_QUAVERS)
display.show(Image.PITCHFORK)
display.show(Image.XMAS)
display.show(Image.PACMAN)
display.show(Image.TARGET)
display.show(Image.TSHIRT)
display.show(Image.ROLLERSKATE)
display.show(Image.DUCK)
display.show(Image.HOUSE)
display.show(Image.TORTOISE)
display.show(Image.BUTTERFLY)
display.show(Image.STICKFIGURE)
display.show(Image.GHOST)
display.show(Image.SWORD)
display.show(Image.GIRAFFE)
display.show(Image.SKULL)
display.show(Image.UMBRELLA)
display.show(Image.SNAKE)
```

Feel free to share your code in the comments below! I would LOVE to see what YOU create!

In our next lesson we will have some FUN with numbers!

Chapter 3: FUN With Numbers

Today we are going to have FUN with our little friend and work with numbers and make a simple adding and subtracting calculator!

We will use what we call a variable to hold a number for as long as you might want. Think of a variable as a little tiny box that you can put a number in and do something with and then take it out and put another number in to replace it.

In our case we are simply going to create a variable and call it counter to which we will set to 0 and use the A button to add 1 to and the B button to subtract 1 from.

Let's teach our little friend to add and subtract numbers by clicking its buttons. When you press the A button it will add a number and scroll it on his little screen and when you press the B button it will subtract a number and scroll it on his little screen.

Let's look at a picture of our little friend and see where the A and B buttons are. This should look familiar as this is a picture from our "Hello World" lesson.

Let's have some FUN and code up our little friend to be a little calculator!

If your micro:bit is not connected to the micro:bit MicroPython Web Editor, please follow the steps in Lesson 2, specifically **STEP 2** and **STEP 3**:

STEP 2: Connect Your micro:bit V2 Into Your Computer

STEP 3: Press The Connect Button In The micro:bit MicroPython Web Editor

If you are connected you are all ready to start!

STEP 1: Open micro:bit MicroPython Web Editor

https://python.microbit.org/v/beta

STEP 2: Type Code & Name It numbers_program

```
from microbit import *


counter = 0


while True:
    if button_a.was_pressed():
        counter = counter + 1
        display.scroll(str(counter))
    if button_b.was_pressed():
        counter = counter - 1
        display.scroll(str(counter))
```

STEP 3: Click Save

STEP 4: Click Download Python Script

STEP 5: Click Flash

STEP 6: SEE THE FUN AS OUR NEW FRIEND WILL ADD AND SUBTRACT NUMBERS WHEN WE PRESS ITS BUTTONS!

(Look at the front of the badge to see the image.)

Now that you have created a little calculator how about you try some FUN new things on your own! Go back to the code and on line 7 try changing it to something like the below statement to which you will SAVE, DOWNLOAD and FLASH as described above.

```
counter = counter + 10
```

You can also change the number or variable on line 10 by changing it to something like the below statement to which you will SAVE, DOWNLOAD and FLASH as described above.

```
counter = counter - 10
```

Feel free to share your code in the comments below! I would LOVE to see what YOU create!

In our next lesson we will have some FUN with words!

Chapter 5: FUN With Words

Today we are going to have FUN with our little friend and work with words and have him type our name and say hi to us!

We will use a word variable which we call a string to customize our little friends ability to say hello to us.

Instead of typing my name in **STEP 2** on line 3, as you can see below, type in your own name instead.

If your micro:bit is not connected to the micro:bit MicroPython Web Editor, please follow the steps in Lesson 2, specifically **STEP 2** and **STEP 3**:

STEP 2: Connect Your micro:bit V2 Into Your Computer

STEP 3: Press The Connect Button In The micro:bit MicroPython Web Editor

If you are connected you are all ready to start!

STEP 1: Open micro:bit MicroPython Web Editor

https://python.microbit.org/v/beta

STEP 2: Type Code & Name It words_program

```
from microbit import *


name = 'Kevin'


while True:
    display.scroll('Hi ' + name + '!')
```

STEP 3: Click Save

STEP 4: Click Download Python Script

STEP 5: Click Flash

STEP 6: SEE THE FUN AS OUR NEW FRIEND SAYS HI TO US!

(Look at the front of the badge to see the image.)

Now that you taught our little friend to say hi to us how about you try some other words and have him say different things to us. You can try changing the below statement to something like this on line 6 to which you will SAVE, DOWNLOAD and FLASH as described above.

```
display.scroll(name + ', thank you for learning Python with me!  I cannot wait to see all the fun things we make!')
```

Feel free to share your code in the comments below! I would LOVE to see what YOU create!

In our next lesson we will have some FUN with word lists!

Chapter 6: FUN With Word Lists

Today we are going to have FUN with our little friend and work with word lists and have him tell us his favorite foods!

A word list is a collection of items that are all within one variable. This is very handy when you want to do more advanced programming in the future when dealing with Python lists which are nothing more than arrays.

If your micro:bit is not connected to the micro:bit MicroPython Web Editor, please follow the steps in Lesson 2, specifically **STEP 2** and **STEP 3**:

STEP 2: Connect Your micro:bit V2 Into Your Computer

STEP 3: Press The Connect Button In The micro:bit MicroPython Web Editor

If you are connected you are all ready to start!

STEP 1: Open micro:bit MicroPython Web Editor

https://python.microbit.org/v/beta

STEP 2: Type Code & Name It word_lists_program

```
from microbit import *


favorite_foods = ['pizza', 'ice cream', 'cookies']


while True:
    display.scroll(
                   'I love to eat ' +
                   favorite_foods[0] + ', ' +
                   favorite_foods[1] + ', ' +
                   'and ' +
                   favorite_foods[2] + '!'
                  )
```

STEP 3: Click Save

STEP 4: Click Download Python Script

STEP 5: Click Flash

STEP 6: SEE THE FUN AS OUR NEW FRIEND TELLS US ALL HIS FAVORITE FOODS!

(Look at the front of the badge to see the image.)

Now that you taught our little friend to tell us all his favorite foods how about you type in your three favorite foods. You can try something like this on line 3 however type in your three favorite foods instead to which you will SAVE, DOWNLOAD and FLASH as described above.

```
favorite_foods = ['candy', 'chips', 'cake']
```

Feel free to share your code in the comments below! I would LOVE to see what YOU create!

In our next lesson we will have some FUN with sounds!

Chapter 7: FUN With Music

IS EVERYONE EXCITED? IS EVERYONE READY TO GET THEIR JAM ON?

Today we are going to have FUN with our little friend and have him jam out some tunes for us as the party starts right now!

If your micro:bit is not connected to the micro:bit MicroPython Web Editor, please follow the steps in Lesson 2, specifically **STEP 2** and **STEP 3**:

STEP 2: Connect Your micro:bit V2 Into Your Computer

STEP 3: Press The Connect Button In The micro:bit MicroPython Web Editor

If you are connected you are all ready to start!

STEP 1: Open micro:bit MicroPython Web Editor

https://python.microbit.org/v/beta

STEP 2: Type Code & Name It music_program

```
from music import *


play(NYAN)
```

STEP 3: Click Save

STEP 4: Click Download Python Script

STEP 5: Click Flash

STEP 6: SEE THE FUN AS OUR NEW FRIEND IS JAMMIN OUT HIS FAVORITE TUNE!

It's your turn! Ask our little friend to play some of his other favorite songs for you! You can try something like this on line 3 by replacing line 3 in our code with one of the below statements to which you will SAVE, DOWNLOAD and FLASH as described above.

Here’s a list of the built-in songs:

play(DADADADUM)
play(ENTERTAINER)

```
play(PRELUDE)
play(ODE)
play(RINGTONE)
play(FUNK)
play(BLUES)
play(BIRTHDAY)
play(WEDDING)
play(FUNERAL)
play(PUNCHLINE)
play(PYTHON)
play(BADDY)
play(CHASE)
play(BA_DING)
play(WAWAWAWAA)
play(JUMP_UP)
play(JUMP_DOWN)
play(POWER_UP)
play(POWER_DOWN)
```

Feel free to share your code in the comments below! I would LOVE to see what YOU create!

In our next lesson we will have some FUN with talking robots!

Chapter 8: FUN With Talking Robots

Today...

We...

Build...

An...

Interactive...

Robot...

Today we are going to have the MOST FUN YET by chatting to Mr. George who is a robot who lives inside our microcontroller and he CAN'T WAIT to talk to you!

If your micro:bit is not connected to the micro:bit MicroPython Web Editor, please follow the steps in Lesson 2, specifically **STEP 2** and **STEP 3**:

STEP 2: Connect Your micro:bit V2 Into Your Computer

STEP 3: Press The Connect Button In The micro:bit MicroPython Web Editor

If you are connected you are all ready to start!

STEP 1: Open micro:bit MicroPython Web Editor

https://python.microbit.org/v/beta

STEP 2: Type Code & Name It talking_robots_program

```
import gc

import speech


def talk(words):
    """Talk to your friend Mr. George

    Parameters
    ----------
    words : str
        The words to say to your friend Mr. George
    Returns
```

```
    -------
    None
    """
    gc.collect()
    words = words.lower()

    if words.find('how are you') != -1:
        speech.say('I am doing great!')
    elif words.find('what\'s up') != -1:
        speech.say('The sky.')
    elif words.find('morning') != -1:
        speech.say('I love to watch the sun rise in the morning!')
    elif words.find('afternoon') != -1:
        speech.say('I get hungry around lunch time.')
    elif words.find('evening') != -1:
        speech.say('I get sleepy in the evening.')
    elif words.find('night') != -1:
        speech.say('I get sleepy when it is night time.')
    elif words.find('tell me something') != -1:
        speech.say('I am a robot who loves to teach Piethon.')
    elif words.find('hello') != -1:
        speech.say('Hello to you!')
    elif words.find('hi') != -1:
        speech.say('Hi to you!')
    elif words.find('thank you') != -1:
        speech.say('It is my pleasure!')
    elif words.find('bye') != -1:
        speech.say('It was nice talking to you!')
    elif words.find('help') != -1:
        speech.say('I am always here to help!')
    elif words.find('what can you do') != -1:
        speech.say('I can teach Piethon programming.')
    elif words.find('name') != -1:
        speech.say('My name is Mr. George it is nice to meet you!')
    elif words.find('how old are you') != -1:
        speech.say('I was born in September of the year twenty twenty.')
    elif words.find('question') != -1:
        speech.say('I always try to answer questions.')
    elif words.find('joke') != -1:
        speech.say('What did the chicken cross the road?')
        speech.say('To get to the other side.')
    elif words.find('love') != -1:
        speech.say('I love pizza!')
    elif words.find('love you') != -1:
        speech.say('Thank you so kind of you!')
    elif words.find('love people') != -1:
        speech.say('I want to help people by teaching them Piethon!')
    elif words.find('hobby') != -1:
        speech.say('I like to teachin Piethon to people!')
    elif words.find('you live') != -1:
        speech.say('I live in side the little microcontroller here.')
    elif words.find('made you') != -1:
        speech.say('Kevin Thomas created me inspired by the great people at
MicroPiethon.')
    elif words.find('your job') != -1:
```

```
        speech.say('I teach Piethon.')
    elif words.find('you do') != -1:
        speech.say('I like to teach Piethon.')
    # ADD MORE CODE HERE
    else:
       pass
```

STEP 3: Click Save

STEP 4: Click Download Python Script

STEP 5: Click Flash

STEP 6: Click Open Serial

STEP 7: Type Into REPL The Following & Press Enter (Ask Mr. George A Question)

```
>>> import main
>>> main.talk('What is your name?')
```

STEP 8: HEAR MR. GEORGE TELL YOU HIS NAME!

CONGRATULATIONS! You have brought to LIFE your FIRST TALKING ROBOT WHICH YOU CAN TALK TO DIRECTLY!

It's your turn! Ask Mr. George some more questions! You can try something like this by re-typing into the REPL with one of the below statements and press enter.

Here’s a list of the built-in questions and things to say to Mr. George:

```
main.talk('How are you?')
main.talk('What's up?')
main.talk('Good morning!')
main.talk('Good afternoon!')
main.talk('Good evening!')
main.talk('Good night!')
main.talk('Hello Mr. George!')
main.talk('Hi Mr. George!')
main.talk('Thank you Mr. George!')
main.talk('Bye Mr. George!')
main.talk('Help me Mr. George!')
main.talk('What can you do?')
main.talk('How old are you?')
main.talk('Do you like questions?')
main.talk('Tell me a joke.')
```

```
main.talk('What do you love?')
main.talk('Do you have a hobby?')
main.talk('Where do you live?')
main.talk('Who made you?')
main.talk('What is your job?')
main.talk('What do you do?')
```

Feel free to share your questions in the comments below! I would LOVE to see what YOU create!

In our next lesson we will have some FUN learning about dozens of FREE additional STEP-BY-STEP lessons online and discuss how YOU can teach one of your friends what you just learned except YOU will be the teacher!

Chapter 9: Basic I/O

Now that we have the fundamentals in place let’s take it to the next level!

MicroPython is a complete re-implementation of CPython which is the Python you use on your typical computer. Both CPython and MicroPython are written in the C programming language.

MicroPython is designed for microcontrollers of which the micro:bit is. MicroPython is crafted to work directly with the device's architecture to create just about any application you can dream of.

We will also work with that we call the REPL which is integrated in the MicroPython Web Editor which gives us very special access to the microcontroller in a very special way. REPL stands for repeat, evaluate, print and loop such that we can do a great deal of debugging or code inspection there.

By the end of the lesson we will have completed the following.

```
* Written a 0001_hello_world_repl.py app which will output "Hello World!"
to the REPL or web terminal or console.

* Written a 0002_hello_world.py app which will display "Hello World!"
on our micro:bit display LED matrix.

* Written a 0003_hello_world_talk.py app to which our micro:bit will
display a talking image in addition to hearing the device speak the
words "Hello World!"

* Written a 0004_basic_io_repl.py app which will demonstrate the
ability for us to obtain keyboard input and dynamically populate
logic in the REPL based on the user submission.
```

STEP 1: Open The MicroPython Web Editor

https://python.microbit.org/v/beta

STEP 2: Plug-In micro:bit To Computer USB

STEP 3: Click Connect Button

STEP 4: Click BBC micro:bit CMSIS-DAP & Click Connect

STEP 5: Select All Code From Demo Program

STEP 6: Delete Demo Code

STEP 7: Rename File To 0001_hello_world_repl

STEP 8: Type Code

```
print('Hello World!')
```

STEP 9: Click Load/Save

STEP 10: Click Download Python Script [NOTE: Might Prompt Warning - IGNORE]

STEP 11: Click Flash

STEP 12: Click Open Serial

STEP 13: Review Output

```
Hello World!
MicroPython v1.13 on 2020-12-03; micro:bit v2.0.0-beta.2 with nRF52833
Type "help()" for more information.
>>>
```

We begin our understanding of MicroPython with the *print* function. The *print* function in MicroPython is a *built-in* function which literally prints strings or words into the REPL. In order for the *print* function to be executed we need to add a pair of parenthesis () after the function name.

The words or string that goes between the parenthesis are what is called a *function argument*. In our case, we are going to pass a string surrounded by a set of single or double quotes. In this course we will be using the single quote convention primarily as it is simply a design choice.

The contents of the print function is nothing more than *print('Hello World!')* which in this situation will print out simply the words Hello World! to the REPL. Whatever word or words we put inside the parenthesis will determine what gets printed to the REPL.

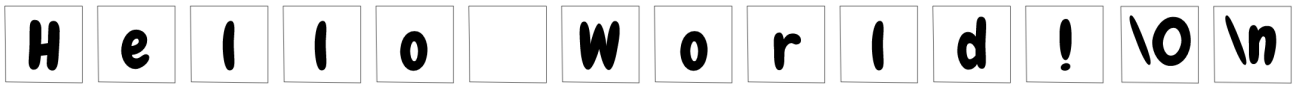
Let's dive into what a string is. A string is a string of characters or letters. Imagine if we had a bunch of little boxes on a table.

So we have our string, *Hello World!* which takes up 12 boxes.

Strangely if we count the boxes we see 14. Let's examine why.

Each box contains a letter or character which we refer to as an element in MicroPython. There is also what we call a null terminating character '\0' and a new line character '\n' that are two additional characters inside the print function. The good news is when the MicroPython team built MicroPython from C, they built-in what we refer to as *default arguments* inside the print function so you, the developer, does not have to type them every time you want to print something to the REPL.

Now let's look at the boxes with all of the letters, spaces, null terminating character and new line character.



That is exactly what is going on inside your computer's memory under the hood.

When programming we all make typos or mistakes. If you leave out a quote you will get what is referred to a *SyntaxError* as you will notice the color scheme on that line will be slightly off. This is an indicator of a syntax error which is nothing more than a syntactical error when the MicroPython interpreter parses your line of code.

Let's look at an example:

```
print('Hello World!)
```

Notice we are missing the other quote mark after the exclamation point. Let's click Flash and Open Serial. Notice our little micro:bit is making a sad face and telling us something useful as well as what we see in the Serial REPL.

```
Traceback (most recent call last):
  File "main.py"
SyntaxError: invalid syntax
MicroPython v1.13 on 2020-12-03; micro:bit v2.0.0-beta.2 with nRF52833
Type "help()" for more information.
>>>
```

In addition, MicroPython will give you an error if you start a line of code that is not at the very beginning of the line as it will be

an indentation error as it will say it sees an unexpected indent error.

Let's look at an example:

```
print('Hello World!')
```

If you notice, the word print starts three spaces after it should.

Click Flash and Open Serial.

```
Traceback (most recent call last):
  File "main.py"
IndentationError: unexpected indent
```

We see our little micro:bit making another frown as well which is very helpful! We can see in the serial REPL an *IndentationError* as well.

Now that you have a handle on all of the steps to create, save, flash and REPL your code we will try some additional examples as well to help solidify these concepts.

Let's clear out our code and rename our new file to **0002_hello_world.py** and type the following code below then click Load/Save, Download Python Script and Flash.

```
from microbit import display


display.scroll('Hello World!')
```

Here we see the string *Hello World!* scroll across our micro:bit display. This is how we can work with output without having to use the REPL.

We see it only scroll once which is what we want. We will look at loops later in this course.

First we see *from microbit import display* which means there is a what we refer to as a *MicroPython module* which is nothing more than a *.py* MicroPython file or library of functions that help us make fun programs easily without having to do all of the low-level implementation to animate the screen in addition to all of the other pieces to make any functionality work.

We will get into more of what MicroPython modules are in later lessons.

We see *display* which is what we refer to as a *MicroPython* function which for our purposes represents a real-world object. In our case it is the display lights on the micro:bit. We see a dot . which means we are going to then extend our display object and make it do something. After the dot we see the world *scroll* which adds additional functionality to the display function to scroll the text.

Do not worry about fully understanding these concepts. I just wanted to bring them to your attention so if you had very basic questions about what each of these pieces mean you can at least get a very high-level understanding of what they are referred to. You do not at this stage have to understand how it all connects yet. This will take time and patience and will be an amazing journey for you which we will take together!

Let's clear out our code and rename our new file to **0003_hello_world_talk.py** and type the following code below then click Load/Save, Download Python Script and Flash.

```
from microbit import display, Image
from speech import say


SPEED = 95


display.show(Image.SURPRISED)
say('Hello World!', speed=SPEED)
display.show(Image.HAPPY)
```

Here we see our little micro:bit display a talking image then literally speak the string *Hello World!* and then display a smiling image.

This is now a third way to work with output with our micro:bit as we started off with text output in the REPL then we saw how our micro:bit could use the display to scroll output and finally here we can see how we can combine the display with speech functionality to allow it to talk to us.

Think of the amazing things you can do now just with this very basic toolset. The possibilities are unlimited and with each lesson we will keep learning and making this journey the most amazing ever!

Let's, on a very high level, review what is going on in the code so far. It is important to remember that we are at the beginning of our journey and like any journey the literal amount of new things we encounter all at once can be intimidating however we will develop these out so that when you continue to come across them in future lectures they will become more solidified in your understanding.

We see *from microbit import display, Image* which means let's take a walk into the microbit module (library) and go and grab the display book (function) and also before we leave the library we need to grab the Image book (class).

We then see *from speech import say* so let's walk across the street and go into the speech module (library) and checkout the say book (function).

Now we have all three books in hand, display, Image and say so we have the tools necessary to write our little app.

We then see a word all in caps called *SPEED* and it is assigned to the number *95*. This is what we call a constant or something that will remain the same and not change for the duration of our app. We make it in all CAPS to remind us that this will not change during the app however we can change the value in one place if we want to adjust rather than changing it all over the app.

We then see *display.show(Image.SURPRISED)* which simply displays the little talking image.

We then see *say('Hello World!')* which allows our little micro:bit to say the words Hello World! to us through it's little speaker.

Finally we see *display.show(Image.HAPPY)* which displays a happy face to us and will stay in that position until new code is flashed.

I know this is a good deal of new info for you but just spend a few minutes each day typing this into your MicroPython Web Editor and trying new strings to the REPL, display and to our micro:bit to speak back to you.

Learning is like exercise, the more you do the better you become at it.

Let's clear out our code and rename our new file to **0004_basic_io_repl.py** and type the following code below then click Load/Save, Download Python Script, Flash and Open Serial.

```
# We introduce the concept of a variable to which
# we reserve a little box in our computer's memory
# to hold the string which we are going to type
# when prompted to provide our favorite food and
# favorite drink
favorite_food = input('What is your favorite food? ')
favorite_drink = input('What is your favorite drink? ')

# Here we use MicroPython's built-in format method
# which is part of the string module's Formatter
# class as the following line of code will provide
# a response back based to the console based on
# our two variables which we inputted above
print('I love {0} and {1} as well!'.format(favorite_food, favorite_drink))
```

I want to introduce the concept of adding comments. We see a *#* and then everything after the *#* on a line is what we call a *comment*. These are helpful to remind us what is going on in our code.

When we start out we can use as many comments as we want. As we get more comfortable we will tend to use fewer comments as we will get a better handle of what is going on by looking at the Python code.

Earlier we saw the concept of a constant which holds a value that does not change when the app runs however now we are going to extend that concept to the variable.

A variable holds a value in those little boxes like we saw earlier and we can use this to store any information we want during our app's run. The difference here is that variables can change during our app and not stay constant.

We are also introducing the concept of basic input in MicroPython which we refer to as *input*. This is a built-in function like the *print* function that allows us to display a message in the REPL and then whatever we type will be then stored into the variable.

We see *favorite_food = input('What is your favorite food? ')* and all this does is display the words, What is your favorite food? and then allow us to type a string response and then it will be stored in *favorite_food*. For example if we typed *pizza* then the string *pizza* would be stored in the *favorite_food* variable.

We repeat the process for *favorite_drink* in the exact same way.

Finally we use the print function again and we use what we refer to as a *format method*. Notice we see *{0}* and *{1}* which are placeholders for our variables so what will happen is that if we used the word

pizza for *favorite_food* it would replace the *{0}* with *pizza* and if we used *Pepsi* for *favorite_drink* the *{1}* would be replaced with *Pepsi*.

Let's try it in the REPL! Let's click Open Serial.

```
What is your favorite food? pizza
What is your favorite drink? Pepsi
I love pizza and Pepsi as well!
MicroPython v1.13 on 2020-12-03; micro:bit v2.0.0-beta.2 with nRF52833
Type "help()" for more information.
>>>
```

It is now time for our first project!

Project 1 - Create a Candy Name Generator app - You are hired as a contract MicroPython Software Developer to help Mr. Willy Wonka rattle off whatever candy title he comes up with in addition to a flavor of that candy. When the program is complete, Mr. Willy Wonka will be able to type into the MicroPython REPL a candy title he dreams up in addition to the flavor of that candy. For example, Scrumpdiddlyumptious Strawberry.

STEP 1: Prepare Our Coding Environment

Let's clear out our code and rename our new file to **p_0001_candy_name_generator.py** and follow all of the steps we learned so far.

Give it your best shot and really spend some time on this so these concepts become stronger with you which will help you become a better MicroPython Developer in the future.

The real learning takes place here in the projects. This is where you can look back at what you learned and try to build something brand-new all on your own.

This is the hardest and more rewarding part of programming so do not be intimidated and give it your best!

If you have spent a few hours and are stuck you can find the solution here to help you review a solution. Look for the **Part_1_Basic_IO** folder and click on **p_0001_candy_name_generator.py** in GitHub.

https://github.com/mytechnotalent/Python-For-Kids

EXTRA CREDIT - Create a Talking Candy Name Generator app - If you are feeling adventurous, let's clear out our code and rename our new file

to **p_0001_candy_name_generator_ec.py** and follow all of the steps we learned so far.

Our task is to take the file we just created and add talking functionality to it.

Earlier in our lesson we learned how to program the micro:bit to talk so let's try to see if we an integrate that functionality into this app!

Please give it a few hours and if you do get stuck here is a solution to review. Look for the **Part_1_Basic_IO** folder and click on **p_0001_candy_name_generator_ec.py** in GitHub.

https://github.com/mytechnotalent/Python-For-Kids

I really admire you as you have stuck it out and made it to the end of your first step in the MicroPython Journey! Great job!

In our next lesson we will learn about MicroPython data types and numbers!

Chapter 10: DataTypes & Numbers

Today we are going to discuss datatypes and numbers as it relates to MicroPython on our micro:bit.

By the end of the lesson we will have accomplished the following.

```
* Written a 0005_calculator_repl.py app which will output add, subtract,
multiply and divide two numbers in the REPL or web terminal or console.

* Written a 0006_square_footage_repl app which will take a width and height in
feet from the repl and print and display the square footage in our micro:bit display LED
matrix.

* Written a 0007_final_score_talk_repl app which will calculate a final
score of a player and indicate the result of a hardcoded boolean.
```

We will focus on 4 primary primitive datatypes that are built-in to MicroPython.

```
* string
* integer
* float
* boolean
```

string

We are familiar with the concept of the string from last lesson. A string is nothing more than a string of characters.

Let's open up our Python Web Editor (full instructions were in Part 1 if you need to double-check) and type the following.

```
name = 'Kevin'
print(name)


Kevin
```

We see that our name variable properly prints the word *'Kevin'*.

Let's demonstrate that a *string* really is a string of characters. Each letter or character in a string is referred to as an *element*. Elements in MicroPython are what we refer to as *zero-indexed* meaning that the first *element* starts at *0* not *1*.

Let's try an example:

```
name = 'Kevin'
print(name[0])
print(name[1])
print(name[2])
print(name[3])
print(name[4])

print(name[-1])

print(name[1:4])

print(name[1:])


K
e
v
i
n

n

evi

evin
```

We see that we do have 5 characters starting at *0* and ending with *4* so the first *element* is *0* and the fifth *element* is *4*. We can also see that the *-1* allows us to get the last *element*. In addition we can see *1:4* prints the 2nd, 3rd and 4th *element* (1, 2, 3) but not the 5th *element*. We see that if we do *1:* that will print the 2nd *element* and the remaining elements.

If we try print an element that is out of bounds we will get an *IndexError*.

```
name = 'Kevin'
print(name[5])


IndexError: str index out of range
```

A *string* is what we refer to as *immutable* as you can't change individual letters or characters in a string however you can change the entire string to something else if it is a variable.

Let's look at an example to illustrate.

```
# CAN'T DO
name = 'Kevin'
name[0] = 'L'

TypeError: 'str' object doesn't support item assignment

# CAN DO
name = 'Kevin'
print(name)
name = 'Levin'
print(name)


Kevin
Levin
```

We can see that we in fact can't change an individual element in a *string* but we can change the *string* to be something else completely by reassigning the variable *name*.

When you add strings together you concatenate them rather than add them. Let's see what happens when we add two strings that are numbers.

```
print('1' + '2')


12
```

We can see we get the *string '12'* which are not numbers as they are strings concatenated together.

integer

An *integer* or *int* is a whole number without decimal places.

```
print(1 + 2)


3
```

Here we see something that we would naturally expect. When we add two integers together we get another integer as shown above.

float

A *float* is a number with fractions or decimals. One thing to remember about a *float* is that if you add, multiply, subtract or divide an *integer* with a *float* the result will ALWAYS be a *float*.

```
print(10.2 + 2)


12.2
```

boolean

A *boolean* has only two values which are either *True* or *False*. Make sure you keep note that you must use a capital letter at the beginning of each word.

A *True* value means anything that is not *0* or *None* and a *False* value is *None* or *0*.

The *None* keyword is used to define a null value, or no value at all. *None* is not the same as *0*, *False*, or an *empty string*.

None is a datatype of its own (*NoneType*) and only *None* can be *None*.

```
is_happy = True
print(is_happy)

is_angry = False
print(is_angry)

score = None
print(score)


True
False
None
```

A *boolean* can be used in so many powerful way such as setting an initial condition in an app or changing conditions based on other conditions, etc.

Type Checking & Type Conversion

We can check the datatype very easily by doing the following.

```
my_string = 'Kevin'
print(type(my_string))

my_int = 42
print(type(my_int))

my_float = 77.7
print(type(my_float))
```

```
my_boolean = True
print(type(my_boolean))


<class 'str'>
<class 'int'>
<class 'float'>
<class 'bool'>
```

You can also convert datatypes by what we refer to as casting or changing one datatype to another.

```
my_int = 42
print(type(my_int))


<class 'int'>
<class 'str'>
```

Math Operations In MicroPython

In MicroPython we have an order of operations that are as follows.

```
Parentheses ()
Exponents **
Multiplication *
Division /
Addition +
Subtraction -
```

If you look at them together you can see we have *PEMDAS*. This is a way we can remember.

In addition *multiplication* and *division* are of equal weight and the calculation which is left-most will be prioritized. This is the same for *addition* and *subtraction*.

```
print(5 * (9 + 5) / 3 - 3)

# First: (9 + 5) = 14
# Second: 5 * 14 = 70
# Third: 70 / 3 = 23.33334
# Fourth: 23.33334 - 3 = 20.33334


20.33334
```

APP 1

Let's create our first app for the day and call it **0005_calculator_repl.py**.

```
first_number = int(input('Enter First Number: '))
second_number = int(input('Enter Second Number: '))

my_addition = first_number + second_number
my_subtraction = first_number - second_number
my_multiplication = first_number * second_number
my_division = first_number / second_number

print('Addition = {0}'.format(my_addition))
print('Subtraction = {0}'.format(my_subtraction))
print('Multiplication = {0}'.format(my_multiplication))
print('Division = {0}'.format(my_division))
print(type(my_division))


Enter First Number: 3
Enter Second Number: 3
Addition = 6
Subtraction = 0
Multiplication = 9
Division = 1.0
<class 'float'>
```

Notice when we use division in MicroPython that the result is a float. This will always be the case.

APP 2

Let's create our second app for the day and call it **0006_square_footage_repl.py**:

```
from microbit import display

length = float(input('Enter length: '))
width = float(input('Enter width: '))

square_footage = length * width

print('Your room size is {0} square feet.'.format(square_footage))
display.scroll('Your room size is {0} square feet.'.format(square_footage))


Enter length: 4
Enter width: 5
Your room size is 20.0 square feet.
```

APP 3

Let's create our third app for the day and call it **0007_final_score_talk_repl.py**:

```
from microbit import display, Image
from speech import say

SPEED = 95

player_score = int(input('Enter Player Score: '))
player_score_bonus = int(input('Enter Player Score Bonus: '))
player_has_golden_ticket = True

player_final_score = player_score + player_score_bonus

display.show(Image.SURPRISED)
print('Player final score is {0} and has golden ticket is
{1}.'.format(player_final_score, player_has_golden_ticket))
say('Player final score is {0} and has golden ticket is {1}.'.format(player_final_score,
player_has_golden_ticket))
display.show(Image.HAPPY)


Enter Player Score: 4
Enter Player Score Bonus: 5
Player final score is 9 and has golden ticket is True.
```

Project 2 - Create a Talking Mad Libs app

Today we are going to create a talking mad libs app and call it **p_0002_talking_madlibs.py**:

Start out by thinking about the prior examples from today and spend an hour or two making a logical strategy based on what you have learned.

The app will first set the *SPEED* of the speech module. It will then get a noun from the user and then get a verb from the user and then finally create a madlib. Print this out and have the speech module talk out the result.

Give it your best shot and really spend some time on this so these concepts become stronger with you which will help you become a better MicroPython Developer in the future.

The real learning takes place here in the projects. This is where you can look back at what you learned and try to build something brand-new all on your own.

This is the hardest and more rewarding part of programming so do not be intimidated and give it your best!

If you have spent a few hours and are stuck you can find the solution here to help you review a solution. Look for the **Part_2_DataTypes_+_Numbers** folder and click on **p_0002_talking_madlibs.py** in GitHub.

https://github.com/mytechnotalent/Python-For-Kids

I really admire you as you have stuck it out and made it to the end of your second step in the MicroPython Journey! Great job!

In our next lesson we will learn about MicroPython conditional logic!