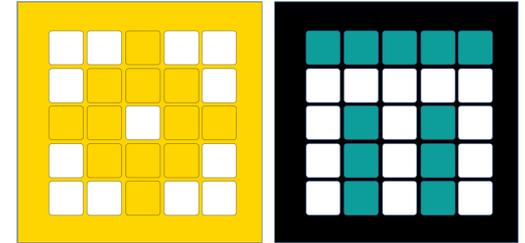


PRIME LESSONS

By the Makers of EV3Lessons



DATA TYPES, OPERATIONS, AND VARIABLES

BY SANJAY AND ARVIND SESHAN

This lesson uses SPIKE 3 software

LESSON OBJECTIVES

Learn the very basics of Python syntax (code)

Learn basic data types

Learn how to use basic operations

Learn basic variables

BASIC TYPES

Integers

Stores whole numbers

Floats

Stores decimals

Bool

Stores **True** or **False**

Strings

Stores text

These types are built-in to the python programming language

```
>> type(42)
int
>> type(42.1)
float
>> type(True)
bool
>> type(False)
bool
>> type("Hello World")
str
```

SPIKE PRIME/MINDSTORMS SPECIFIC TYPES

SPIKE Prime/Mindstorms provide libraries that define additional classes

These types are assigned/initialized to variables to access data or control sensors or motors

You can get load these with commands such as:

Spike Prime 3:

```
from hub import button, light, light_matrix, motion_sensor, port, sound
import runloop, color, color_matrix, color_sensor, device, distance_sensor,
force_sensor, motor, motor_pair, orientation
```

Mindstorms:

```
from mindstorms import MSHub, Motor, MotorPair, ColorSensor, DistanceSensor, A
pp
```

These types are slightly different than integers, strings, etc. but have similar properties

These SPIKE/MINDSTORMS specific types will be covered in later lessons

USING THE PRINT FUNCTION

We will cover functions in general in a later lesson. Here we just describe how to use the print function to display information in the console.

Print data to the “console”/output screen

```
>> print("Hello World")
Hello World
>> print(253.5)
253.5
```

Helpful note: Placing a # in front of text creates a comment. That code will not run.

```
>> # Comment
>> print("Hello World")
Hello World
```

USING THE HUB LIGHT MATRIX

We will cover how to use the hardware specific methods in general in a later lesson. Here we just describe how to use the hub display to show values.

This displays the word “Hi” on the SPIKE hub

```
from hub import light_matrix
import runloop

async def main():
    # write your code here
    await light_matrix.write("Hi!")

runloop.run(main())
```

This does the same for the MINDSTORMS hub

```
from mindstorms import MSHub
hub = MSHub()
hub.light_matrix.write(5.3)
hub.light_matrix.write("hello")
```

Note, the “from”, “import” and “hub =” lines need to only be included once at the beginning of your code. To use the light matrix, just use the “write” method calls later in your program.

VARIABLES

Variables store data

These are like variables in algebra

Data is of a given type

The content stored in a variable can be changed to a different value or even type

You can name the variable anything you like (in this case it is “x”). However, the variable name must start with a letter (generally lowercase)

```
>> x = 7
>> print(x)
7
>> x = "hi"
>> print(x)
"hi"
>> x = "bye"
>> print(x)
"bye"
```

OPERATIONS

You can write mathematical expressions using common operators:

add (+), subtract (-), divide (/), multiply (*), modulo (%) (remainder), exponent (**)

The “//” operator to integer divide. It will remove all decimals.

You can add numbers, floats, strings, and many more

You cannot interchange different types in operations (with the exception of floats, integers, and booleans)

Advanced: place “import math” at the beginning of your program to get access to more functions; e.g. “math.sqrt(n)” (square root)

```
>> print(5+10)
15
>> print(10/3)
3.3333333333333335
>> print(10//3)
3
>> print("ab"+"cd")
abcd
>> print(7+"ab")
TypeError: unsupported operand
type(s) for +: 'int' and 'str'
>> print(7, "ab")
7 ab
```



For those who are curious, the 10/3 output ends in a 5 because of something called “floating point approximation”. Basically, computers have to estimate when decimals are involved, so there is some inaccuracy

OPERATIONS ON VARIABLES

Operations on variables are not quite like algebra

Expressions are right-hand evaluated

The expression on the right of the = is evaluated first, then re-casted to the variable on the left side

In the example on the right, the $x+10$ is evaluated to 20 first, then x is set to 20, deleting the previous value

```
>> x = 10
>> print(x)
10
>> x = x+10
20
>> # Shorthand:
>> x+=10
30
```

CHALLENGE

Create a variable x and assign it a value

Create a variable y and make it equal the square root of x

Display y on the hub

CHALLENGE SOLUTION (SPIKE PRIME)

```
from hub import light_matrix
import asyncio

# Import standard Python math library to help with math functions
import math

async def main():
    # This creates the variable x and set it to 2
    x = 2
    # This creates y and sets it to square root of x (square root is the
    # same as the exponent power of 0.5)
    y = math.pow(x, 0.5)
    # This displays y. It must first convert it to a string
    # using the str() function
    await light_matrix.write(str(y))

asyncio.run(main())
```

CREDITS

This lesson was created by Sanjay and Arvind Seshan for Prime Lessons

Additional contributions by FLL Share & Learn community members.

More lessons are available at www.primelessons.org



This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-nc-sa/4.0/).