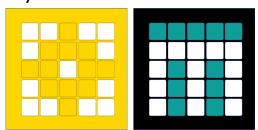# FUNCTIONS

BY SANJAY AND ARVIND SESHAN

LESSON OBJECTIVES

- Learn to create and use functions

- Learn why a function is useful

FUNCTIONS

- Python functions are similar to algebraic functions

    - $f(x)=3x^2$

    - $f(3)=27 \rightarrow f(3)$ returns 27

- Functions are defined as a set of code that takes one or more input values and returns one or more results

- Functions are very versatile. You can put as much code as you want, as many inputs as you want, and return any data you want

- Indentation is needed to make sure only the code you want in the function runs when it is called

```python
def f(x):
    y = 3*x**2 # y=3x^2
    return y
```

```python
def g():
    for i in range(7):
        print(i)
    return "Hello"
```

CONSTRUCTING A FUNCTION

- A function definition starts with:

  def YOUR_NAME_HERE(PARAMETERS):

  - The code that is indented below it runs when the function is called

- You can name the function whatever you want. However, the name must start with a letter (generally lowercase name)

  - A good naming convention for functions and variables is camelCase (the first word is all lowercase and the rest start capital). All words are conjoined. E.g. myFunction()

- The parameters are listed comma separated inside the parentheses following the function name.

  - IMPORTANT: These parameters are local variables and can only be used inside the function.

```
def g():
    for i in range(7):
        print(i)
    return "Hello"
```

```
def h(a,b):
    for i in range(7):
        print(i, a, b)
    return "Hello"
```

CALLING/RUNNING A FUNCTION

- To call a function, anywhere in you code place the function name, followed by parentheses with the desired parameter values

  - The yellow highlighted line at right calls the function g(a,b)

- The line that calls the function proceeds to run the function code, where a and b are replaced temporarily with the parameter values

- After the function is run, the code proceeds as usual

```
def g(a, b):
    print("Hello", a, b)


print("Running....")
g(2, 3)
print("Done!")


Output:
Running....
Hello 2 3
Done!
```

FUNCTIONS WITH RETURNS

- Place "return DATA" within a function to output DATA as an result of the function

- The function g() returns the value 10, which can be used in the program

```python
def g(a, b):
    print("Hello", a, b)
    return 10

print("Running....")
print(g(2, 3))
print("Done!")

Output:
Running....
Hello 2 3
10
Done!
```

BUILT-IN FUNCTIONS

■ There are many important functions built in

■ See a list of important ones below

```python
print("Type conversion functions:")
print(bool(0))   # convert to boolean (True or False)
print(float(42)) # convert to a floating point number
print(int(2.8))  # convert to an integer (int)

print("Basic math functions:")
print(abs(-5))   # absolute value
print(max(2,3))  # return the max value
print(min(2,3))  # return the min value
print(pow(2,3))  # raise to the given power (pow(x,y) == x**y)
print(round(2.354, 1)) # round with the given number of digits

print("Pause/sleep")
import time
time.sleep(4) # sleep for n seconds
```

WHEN DO YOU USE FUNCTIONS?

■ Great for repetitive tasks

■ Moving distance, turning, etc.

■ Great for organizing and simplifying code

WHAT MAKES A USEFUL FUNCTION

- Note: Making functions with inputs/outputs are very useful. However, you need to be careful not to make the function too complicated.

- Question: Look at the list of three functions below. Which ones do you think are useful to use?

  - Turn90degrees (Turns the robot 90 degrees)

  - TurnDegrees with an angle and power input

  - TurnDegrees with angle, power, coast/brake, etc. inputs

- Answer:

  - Turn90degrees may be used often, but you will be forced to make other MyBlocks for other angles. This will not be fixable later.

  - TurnDegrees with angle and power as inputs is probably the best choice.

  - TurnDegrees with angle, power, coast/brake, etc. might be most customizable, but some of the inputs might never be used.

VARIABLE SCOPE IN FUNCTIONS

■ What do you think this code will do?

```python
y = 7
def f(x):
    print(x)
    print(y)
f(4)
print(y)
print(x)
```

VARIABLE SCOPE IN FUNCTIONS

■ What do you think this code will do?

```
y = 7
def f(x):
    print(x)
    print(y)
f(4)
print(y)
print(x)
```

Output:
4
7
7
NameError: name 'x' is not defined

Hmmm....there seems to be an error

11

VARIABLE SCOPE IN FUNCTIONS CONT.

- We mentioned that function parameters are local variables…..that means that those "variables" can only be accessed within the function

- The print(x) on the last line is outside the function and therefore the variable x cannot be read

- The variables defined outside the function are considered global, meaning they can be used anywhere

- Note that if a local and global variable share the same name, the local one will be called, unless specified

```
y = 7
def f(x):
    print(x)
    print(y)
f(4)
print(y)
print(x)
```

Red is the function scope. Yellow is the global scope. Red can also access global variables

Advanced: use "global x" in a function to forcibly use a global variable over a local one

VARIABLE SCOPE EXAMPLE

- Same variable names in different scopes

```python
y = 7
x = 2
def f(x):
    print(x)
    print(y)
f(4)
print(y)
print(x)
```

Output:
4
7
7
2

In this case the x from the global scope is used on the last line, while the local one is used in the function (not overwriting the global one)

OBJECTS AND METHODS

- Objects are somewhat like a set of functions but are initialized and "saved" to a variable.

  - In Python, everything is technically an object (even ints, strings, etc.)

- Objects are created using a call to a constructor function

  - E.g., var = object()

- Methods are a special type of function associated with an object

- To call a method, you must have a variable or value of that type to call

  - The variable/value you use is an implicit input to the method

- The special variable types associated with SPIKE Prime/MINDSTORMS expose a range of different methods to control your robot. We will go over these types and their methods in later lessons.

- For example, strings have a variety of methods for various purposes

  - Some examples are shown to the right

  - Full list of string methods are listed at https://www.w3schools.com/python/python_ref_string.asp

```
s = str("Test")
s.upper() # TEST
s.lower() # test
s.find("T") # 0
```

14

CHALLENGE

■ Create a function with parameter n that adds up all numbers from 0 to n, where n is an integer

■ It should return the answer

■ Hints:

　■ You will use a loop and return statement

SOLUTION

```python
def sumToN(n):
    total = 0
    counter = 1
    while (counter <= n):
        total += counter
        counter += 1
    return total

print(sumToN(5), 1+2+3+4+5)
```

CREDITS

- This lesson was created by Sanjay and Arvind Seshan for Prime Lessons
- More lessons are available at www.primelessons.org