# Day 2

# Functions

A function lets you execute another piece of code.

Functions are **invoked** by using the name of the function, parentheses, and passing **arguments**.

# Anatomy of a Function Invocation

```
max(10, 15, 6)
```

# Function Invocation

Arguments can be variables.

```python
alice_score = 10
bob_score = 22
carol_score = 2
max(alice_score, bob_score, carol_score)
```

# Function Invocation

Function invocations can be arguments to other functions.

```
max(564, max(100, 689), 12)
```

# Modules

Many useful functions come from modules.

You get access to these functions by **import** ing them.

This is used a lot to interact with the Sense HAT.

# Module Import

```python
from random import randint
random(0, 10)
```

# Multiple Imports

Good modules come with **documentation** that explains what they do.

Here is the documentation for the built-in `string` module: https://docs.python.org/3/library/string.html

```
import string
print(string.digits)
print(string.punctuation)
```

# `for..in` Loop

It's common to want to run a piece of code a certain number of times.

For that, you can use `for..in`.

# for..in with range

range(a, b) returns numbers starting with a and going to
b - 1.

```python
for i in range(0, 5):
    print(i)
```

# `if` Condition

Often you only want to execute a piece of code **conditionally**.

For this you can use `if..else`.

# **if Condition**

Like with `while`, it takes in a condition to evaluate.

```python
if True:
    print("True!") # This is executed
else:
    print("False!") # This is not
```

# **if Condition**

Usually variable will be involved in the condition.

```python
is_even = False
if is_even:
    print("Even")
else:
    print("Odd")
```

# Loops and Conditionals Together

Loops and conditionals together can express a wide range of **algorithms**.

An **algorithm** is a sequence of steps for solving some problem.

# Find smallest $n$ such that $n^2 > 1000$

```python
still_searching = True
current_number = 0
while still_searching:
    square = current_number * current_number
    if square > 1000:
        print(current_number)
        print(square)
        still_searching = False
    else:
        current_number = current_number + 1
```

# Break

# Lab 2: Countdown Timer