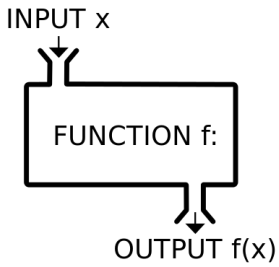
Functions



Functions are reusable blocks of code. We use functions to *name* blocks of code.
Then we can call a block of code by its name.
Functions are a *powerful* tool for creating abstractions to hide the implementation.
Variables are the nouns in our code and functions are our *verbs*. They *do* things.
The ability to name abstractions allows us to create solutions in the *language* of the problem.

Built-in Functions

`len`, `type`, `abs`, `str`, `sum`, `range`, etc... are built-in. All we do is "call" them to run them.

https://docs.python.org/3/library/functions.html

Defining Our Own Functions

Step 1: Define the function

```
def add_one(x):
  """This function adds 1 to any input. String inputs produce an error"""
  return x + 1
```

Step 2: Call the function

`add_one(5)` returns `6`. `add_one(add_one(1))` returns `3`.

Valuable Takeaways to Know

- `functions(run(from(inside(out))))`
- `` `methods.run().from().left().to().right()``
- Methods are functions defined and then called directly from objects, like lists or strings.
- Your ability to write and use functions defines your capability to customize your solutions.
- Recommend *always* returning values from functions. W/o `return`, functions return `None`
- Recommend using input variables and variables local to your function instead of globals.