Functions

Remember when you are in math class and you are told `y = 2x + 4` was the equation for y? You can say that it is a function for `y` in that `y` is given the value of the summation of twice `x` and `4`. Let's write it in a different way: `f(x) = 2x + 4`. There is a function, called `f` that takes in the value `x` and it returns the summation of twice `x` and `4`. Instead of `f`, we could call it `g` or `f1`. Maybe `cat`. It could be anything you want.

An example

```
def add(x,y):
  answer = x + y
  return answer
```

- `def`
- def is short for *define* and it is used to define a function. You use it whenever you want to make a function.
- `add`
- This is the name of the function. Try to use names that are meaningful so when you look at them you get an idea of what it does.
- `(x,y)`
- Every function has twice parenthesis, an open one and a close one, at the end of the function. Anything that goes inside the parenthesis are things that the function uses as input. To go along with our function earlier, `f(x)`, the function `f` has the input `x`.
- `:`
- This is Python's grammar (also known as syntax) to say that the definition of a function will begin on the next line.
- `answer = x + y`
- Whenever you have `=` in Python, it does not mean that both sides are equal. They technically are equal but that is not what's going on here. It means that you give the value from the right side of the `=` to the left side of the `=`
- `answer` gets the value of the summation of `x+y`. This is also known as `answer` is *assigned* the value of the summation of `x+y`.
- `return answer`
- `return` is another *keyword* (a word that has a special purpose in a language) like `:`. It means that the output of the function `add` is what follows it. In this case, the output is `answer`.

With what we know now, we know the function takes in two values, `x` and `y`; adds `x` and `y` and assigns the value to `answer`; and returns `answer` as the output.

Here is the code again:

```
def add(x,y):
  answer = x + y
  return answer
print add(1,2)
```

On the first three lines we have *defined* our function and on line 4 we are *calling* the function. Calling a function means that we are using the function to do a task. `print add(1,2)` has Python `print` the `return` value of `add(1,2)`, which is `3`.

Let's make another one:

```
def mult(x,y):
  return x*y
```

This one is slightly different. It's different because I did not create an `answer` variable to save what I have done. Instead of that, I can just return the result of `x*y`. Also, I can reuse `x` and `y` in this function. This will be fully explained later, but the point is that these are variables that are being used inside `add` or `mult` not both. Let's continue:

```
def f(x):
  return add(mult(2,x),4)
print f(4)
```

To come full circle, we have made a function `f(x)` that returns the summation of twice x and 4. To add to what we can do, we can not only call a function inside another function like we are doing with `add`, we are calling the function `mult` as an argument inside the function `add`.

The purpose of showing these different ways of building functions is to show you the tools to mix and match the code to your purpose. I could have defined `f` as the following:

```
def f(x):
  product = mult(2,x)
  summation = add(product,4)
  return summation
```

This definition is does the exact same thing as the first one; it does not matter. Functions can have variables or no variables. They could call functions inside of them or not call them.

Some common mistakes

The following code are consider errors and will not work well.

```
print foo(5)
```

```
def foo(x):
  return -1*x
```

This does not work because Python reads a file from top to bottom. If you call a function before defining it, Python will throw an syntax error. You need to switch the order of them.

```
def foo(x):

  def bar(y):
    z = x + y
    return z
  print bar(2)

foo(5)
```

This defines a function called `foo` which defines another function called `bar` inside of it. This does not work and will cause Python to throw a syntax error. If you want to call a function inside another function, you must define the one used inside the order first. In this case, you need to define bar first, and then define foo. A second syntax error is that the function `bar` uses the variable `x`, but `x` is not defined inside function `bar`. This might not be easily understandable because the `x` variable is typed twice within four lines and you'd think that you could use it. To make sure it's understood, I will rewrite the correct code.

```
def bar(x,y):
  z = x + y
  return z
def foo(x):
  print bar(2,x)
foo(5)
```

This does what the code before possibly means but there are no errors.