Week 6

Intro Python

http://cadtx.pw/intropy6

Objectives

Chapter 6

- Functions
- Codingbat

What are functions?

- Functions are blocks of code designed to do a specific task that you can re-use
- Instead of typing the same code over and over again you can just write a function and call it
- Some built in python functions are `str()`, `abs()`, `len()`, etc.
- Functions are called with their name followed by any *arguments* in parentheses
- len(my_list)

Writing our own functions

- You define functions using the `def` keyword, followed by the name of your function, parentheses and a colon
- It's good practice to write a comment describing what the function does, this is called a *docstring*
- The code within the body of the function is indented
- You can call the function using its name and parentheses

```
1 def greet_user():
2     """ Display a simple greeting. """
3     print("Hello!")
4
5 greet_user()
```

```
Hello!
>>>
```

Passing information to a function

- Modified slightly our function greet_user() can greet people by name
- By adding the *parameter* `username` you allow the function to accept any value you specify
- The function will now expect us to provide a value for `username` each time you call it

```
1 def greet_user(username):
2     """ Display a simple greeting. """
3     print("Hello, " + username.title() + "!")
4
5 greet_user('bevo')
6 greet_user('sean')
7 greet_user()
```

```
Hello, Bevo!
Hello, Sean!
>>>
```

```
    greet_user()
TypeError: greet_user() missing 1 required positional argument: 'username'
>>>
```

Multiple arguments

```
6  def describe_pet(animal_type, pet_name):
7      """ Display information about a pet. """
8      print("I have a " + animal_type + ".")
9      print("My " + animal_type + "'s name is " + pet_name.title() + ".")
10
11 describe_pet("dog", "august")
12 describe_pet("august", "dog")
```

- There are multiple ways to pass your function multiple arguments
- The default is with *positional arguments* which need to be in the same order the parameters were written
- Order matters! Sometime it will cause errors

```
I have a dog.
My dog's name is August.
I have a august.
My august's name is Dog.
>>>
```

Keyword Arguments

- A keyword argument is a name-value pair that you pass to a function
- You don't have to write the function any differently, you're just more explicit when you call the function

```
I have a dog.
My dog's name is August.
I have a dog.
My dog's name is August.
>>>
```

```
6  def describe_pet(animal_type, pet_name):
7      """ Display information about a pet. """
8      print("I have a " + animal_type + ".")
9      print("My " + animal_type + "'s name is " + pet_name.title() + ".")
10
11 describe_pet(animal_type = "dog", pet_name = "august")
12 describe_pet(pet_name = "august", animal_type = "dog")
```

Default values

- When writing a function you can specify a default value for each parameter which will be used if no arguments are specified in the function call
- Now if no `animal_type` is specified `'dog'` will be used
- Note that any parameter with a default value must be listed after the others so positional arguments can function correctly

```
6  def describe_pet(pet_name, animal_type = "dog"):
7      """ Display information about a pet. """
8      print("I have a " + animal_type + ".")
9      print("My " + animal_type + "'s name is " + pet_name.title() + ".")
10
11 describe_pet('August')
12 describe_pet('bevo', 'cow')
```

```
I have a dog.
My dog's name is August.
I have a cow.
My cow's name is Bevo.
>>>
```

Return values

- Functions don't have to display their output directly
- Instead they can process some data and return a value or set of values
- The value a function returns is aptly called the *return value*
- The return statement takes a value from inside the function and sends it back to the line that called it

```
11 def get_formatted_name(first_name, last_name):
12     """ Return a full name, neatly formatted """
13     full_name = first_name + ' ' + last_name
14     return full_name.title()
15
16 instructor = get_formatted_name('sean', 'yu')
17 print(instructor)
```

```
Sean Yu
>>>
```

```
16 print("My name is " + get_formatted_name('sean', 'yu'))
```

```
My name is Sean Yu
>>>
```

Making arguments optional

- You can make arguments optional through the use of default values and control structures

```
11 def get_formatted_name(first_name, last_name, middle_name = ''):
12     """ Return a full name, neatly formatted """
13     if middle_name:
14         full_name = first_name + ' ' + middle_name + ' ' + last_name
15     else:
16         full_name = first_name + ' ' + last_name
17     return full_name.title()
18
19 musician1 = get_formatted_name('carly', 'jepsen', 'rae')
20 musician2 = get_formatted_name('taylor', 'swift')
21 print(musician1)
22 print(musician2)
23
```

```
Running: cad lole.py
Carly Rae Jepsen
Taylor Swift
>>>
```