Programming best practices

Variables:

- Don't hard code values into your program. This limits reusability!
- Use descriptive names. Your future self will have trouble remembering what `x, i, sbg`, or `krf_int` mean. Avoid abbreviations if possible.
- Come up with naming conventions with your team. Everyone should use the same format, such as `first_name` or `firstName` or `first_name_string`.

Functions:

- Functions are bits of reusable code that do a specific action.
- Smaller, more precise functions are better than bigger, more complex functions. `draw_plans(), build_walls(), build_roof(), build_electric(), build_sewer(), build_interior()` as a sequence of events is much easier to fix, improve, and understand than a giant function called `build_house()`.
- Like a variable, a function has a descriptive name. `calculate_area_triangle()` or `break_lines()` are good names.
- Just like variables, have everyone in your team use the same naming conventions. This makes your code more consistent throughout the program/project.
- Define your functions early, then do the work at the end.

```
def function_a(variable):
     do stuff

def function_b(other_variable):
     do other stuff

cats = 2
dogs = 5

answer = function_a(cats) / function_b(dogs)
print(answer)
```

Monitoring variable values:

- You want to make sure that everything is working correctly when building your program. Check that the outputs and variables are indeed what you expect them to be in both value and type. If you are expecting something to be a name, for example, make sure the variable returns `"Georgina"` rather than `5`.
- An easy way to monitor your variables is to print them to the console (terminal) when you are running the program. At some point in your code, write a print statement (like "`print("name : " + name)`") for your variable of interest. Even if the code is buggy after it, the print statement will

work and you will see the value(s) you are interested in before an error message.

Comments:

- Comments are extremely useful to help a reader (you from the future, a teammate, a person who found your code on GitHub) understand your code.
  ```
  // This section gets all the information from the file
  ```
- You can also use comments when you are working on a bit of code.
- Comment out old code before writing new code: this lets you use selected parts of your code while you are figuring out what works.
  ```
  //cats = 5
  cats = 3
  ```