Functions

Defining Functions

def

```
def [function_name]([## arguments]):
    [function content]
    ## return [return_variable]
```

Parameters

Type	Syntax
no parameter	`()`
mandatory parameter	`(mandatory_param_name)`
multiple parameters	`(param_1, param_2)`
optional parameters	`(mandatory_param, optional_param=default_val)`
keyword paramteres	`(**name_of_param_dict)`

return Statement

Type	Syntax
simple	`return x`
modifying	`return x + 1`
conditional	`return x if x==0 else y`

Learn more about functions in Python here here.

Recursive Functions

If a function is *recursive* it means that it calls on itself inside its body.

```
def [rec_function_name]([## arguments]):
    if [base_case]:
        return [variable]
    else:
        return [rec_function_name]([## arguments])
```

When to use Recursion

Examples of when to consider recursion include (but are by no means limited to):

1. When you find yourself writing similar loops inside loops.
2. When you need to process unknown "depth" (but you know that it ends eventually).
3. When the problem can be broken into smaller identical pieces.
4. When the data structure references itself (e.g. linked lists, graphs, etc.).
5. When the problem involves backtracking different paths.

Functions: General Notes

1. **The function body cannot be empty.** While a function needs neither arguments nor return statements, to make a function *do nothing*, you need to use a `pass` statement (just as in conditionals).
2. **Always make sure of the function return type.** When using a function's output for further computations, make sure fits (your code may break otherwise).
3. **Always make sure your recursion ends.** Much like infinite loops, infinite calls on a function are a good way of crashing your computer.