Functions, Part II

March 11,2020

Administrative Notes

Scope of variables

Scope means where a variable can be directly seen and used

Variables defined in the main program can be seen everywhere in the program. BUT - you should only directly use them in the main program. If you need their values in a function, pass them as arguments. If you use a main program variable directly in a function, without passing it as an argument, that's called a "global variable" and it will cost you major points!!!

Variables defined in functions can only be seen and used in the functions where they are defined. These are called "local variables."

Examples

A few examples of what scope means

The return statement

return is the statement used to pass values from the function back to the main program (or back to the calling function)

The syntax is

```
    return variable_name # you can optionally put the variable name in ()
```

For this class, a function will return one value.

Using returned values

If you want to use the value that’s returned, you have to do so in the calling program/function

If the function contains:

```
def factorial(num):
    Product = 1
    For i in range(num):
        Product *= i
    return product
```

The calling program should use the value:

```
fact = factorial(5)
```

Or

```
sum = fact(7) + fact(2)
```

or...

The value “None”

Python defines a special value “None” which is of type “Nonetype”

If a function does not otherwise return a value, it returns “None”

- Functions do not have to contain a return statement. Any function without a return statement returns “None.”
- If a function’s return statement is not executed, the function returns “None.”
- You can explicitly tell the function to return “None.”

A common error is to have a function return None when you expected it to return something else.

- You’ll see an error message like:
- `Error; type ‘Nonetype’ is not iterable`