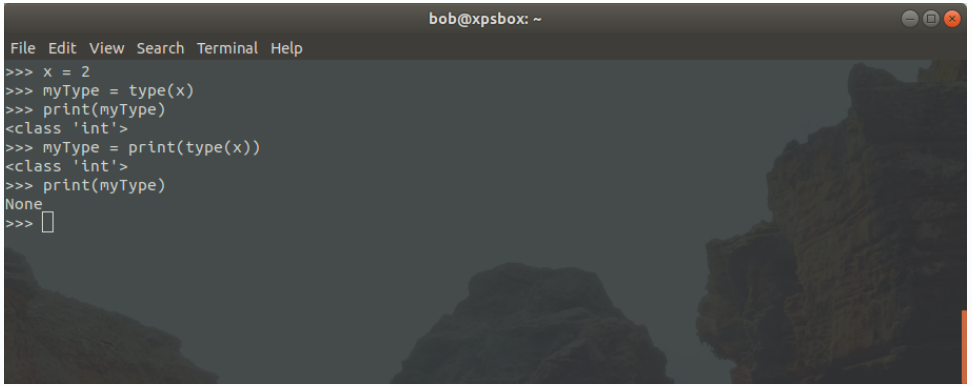
EGM722 – Programming for GIS and Remote Sensing

Week 1, Part 4: Functions

What is a function?

- Function: a named sequence of statements that performs a computation
  - Take in arguments
  - (sometimes) return results
- “Fruitful”: returns a result
- “Void”: returns None (no value)
- Example: type()
  - Takes object as argument, returns type



Why bother?

- Ease of readability
- Eliminates repetitive code
- Can debug (troubleshoot) one part of code at a time
  - Updates become much easier, too
- Can reuse them in other scripts

Anatomy of a function

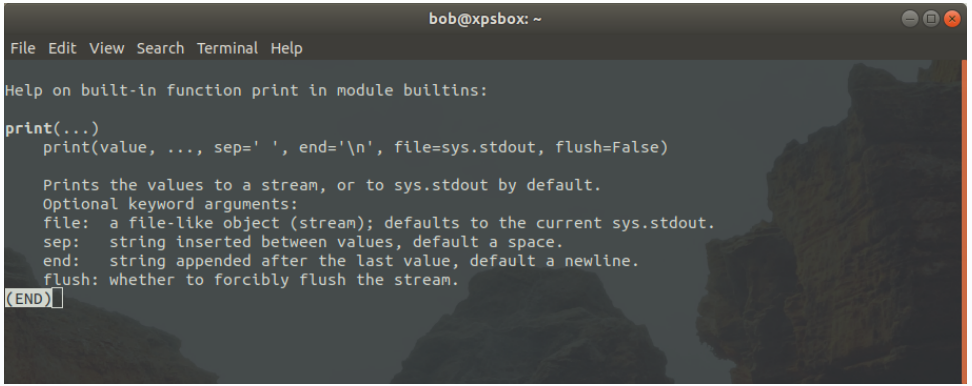
- Names follow the same rules as variable names
- First line: header
  - Starts with a definition statement
  - Ends with :
- Body is indented
- Good practice: writing a docstring that tells about the function
- Call functions with arguments (or not)
- Parameter: a variable that has an argument assigned to it

```python
def myFunction(parameter1, parameter2):
    print('Argument 1: ' + parameter1)
    print('Argument 2: ' + parameter2)
```

```
bob@xpsbox: ~
File  Edit  View  Search  Terminal  Help

Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
(END)
```

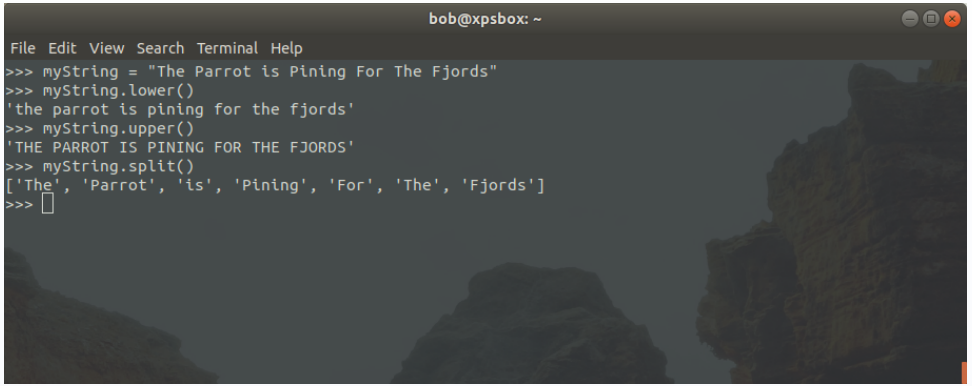
Keyword arguments

- Most arguments identified by position (positional parameters)
- Some identified by keyword (optional parameters)
- Example: print()
  - Positional: value(s)
  - Keyword



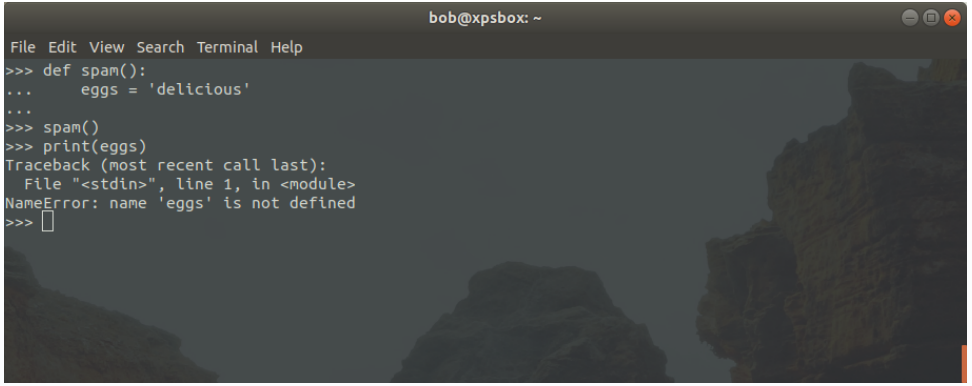
Methods

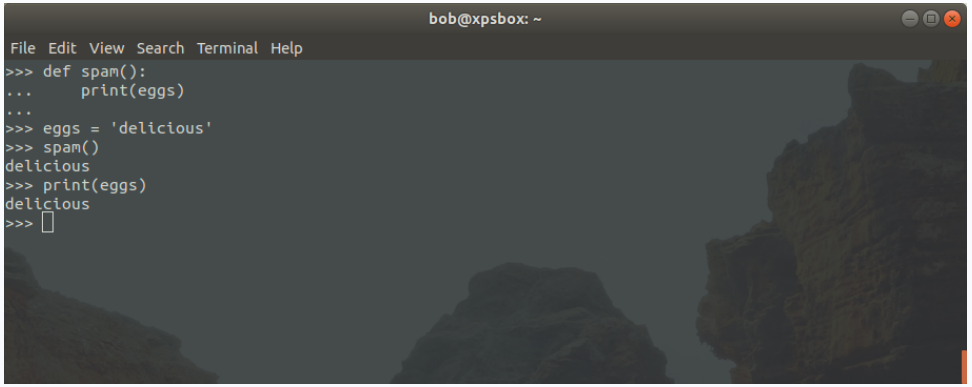
- Method: a function that operates directly on an object
- Examples:
  - str.lower(): returns the string object in lowercase
  - str.upper(): returns the string object in uppercase
  - str.split(): splits the string on a separator (default: whitespace)



Scope

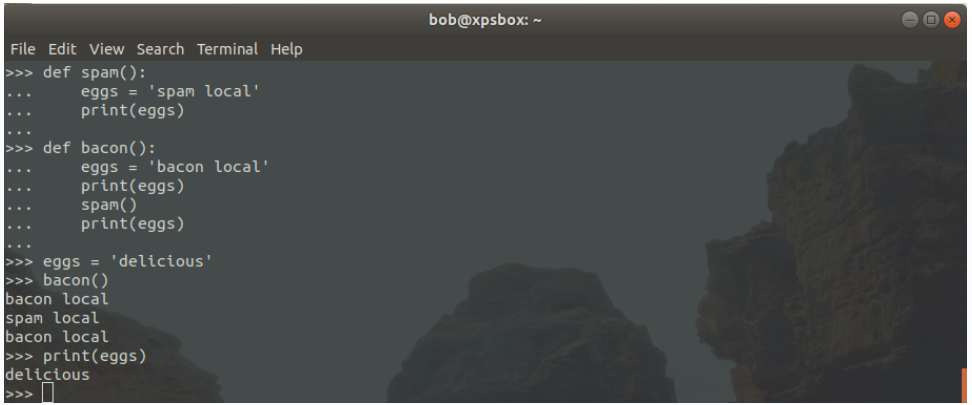
- Within function, variables + parameters exist in function's local scope
  - Local variables
- When function finishes, local variables are forgotten
- Variables outside all functions exist in global scope
  - Global variables
- Code in global scope cannot access local scope
  - But local scope can access global scope





Scope

- Technically, we can use the same variable name in multiple scopes
- As a general rule, don't do this:
  - Quickly gets confusing
  - Debugging is much more difficult
  - Some IDEs will yell at you



Docstrings and help

- Good practice to include docstrings when writing functions
  - Describe what function does
  - Describe parameters, returned values (if any)
  - Both future you, and other programmers, will thank you
- When you call help(), it's actually reading the docstring
  - No docstring, no help

```python
def reproject_geometry(src_data, src_crs, dst_crs):
    """
    Reproject a geometry object from one coordinate system to another.

    :param src_data: geometry object to reproject
    :param src_crs: proj4 description of source CRS
    :param dst_crs: proj4 description of destination CRS
    :type src_data: shapely geometry
    :type src_crs: str, dict
    :type dst_crs: str, dict

    :returns dst_data: reprojected data.
    """
    # unfortunately this requires pyproj>1.95, temporary fix to avoid shambling dependencies in mmaster_environment
    src_proj = pyproj.Proj(src_crs)
    dst_proj = pyproj.Proj(dst_crs)

    project = partial(pyproj.transform, src_proj, dst_proj)
    return transform(project, src_data)
```

```
bob@xpsbox: ~
File  Edit  View  Search  Terminal  Help

Help on function reproject_geometry in module pymmaster.mmaster_tools:

reproject_geometry(src_data, src_crs, dst_crs)
    Reproject a geometry object from one coordinate system to another.

    :param src_data: geometry object to reproject
    :param src_crs: proj4 description of source CRS
    :param dst_crs: proj4 description of destination CRS
    :type src_data: shapely geometry
    :type src_crs: str, dict
    :type dst_crs: str, dict

    :returns dst_data: reprojected data.
(END)
```

Summary

- Functions are a useful way to:
  - Clean up code
  - Help with debugging/rewriting
  - Reuse code
- Variables in python have scope: either local or global
- When writing functions, important to document them