Chapter 6: Variable Scope

Dona Hertel

CS161 – Winter 2013

Portland State University

Glossary

- **Variable Scope:** The portion of the code where a variable is known to the python interpretor and can be used without getting an “not found” error.

- **Namespace:** At a particular point in the code, the list of variable and function names that are known to the interpretor.
  - Extra Aside: Can use the builtin function dir() to get a list of the current namespace.

Variable Scope

- When you first assign a value to a variable, it is 'defined' and you can use it's value from then on.
- The following code doesn't work because i is not assigned before the while loop conditional:

```
while(i < 20):
    i = i + 1
    print(“HELLO”)
```

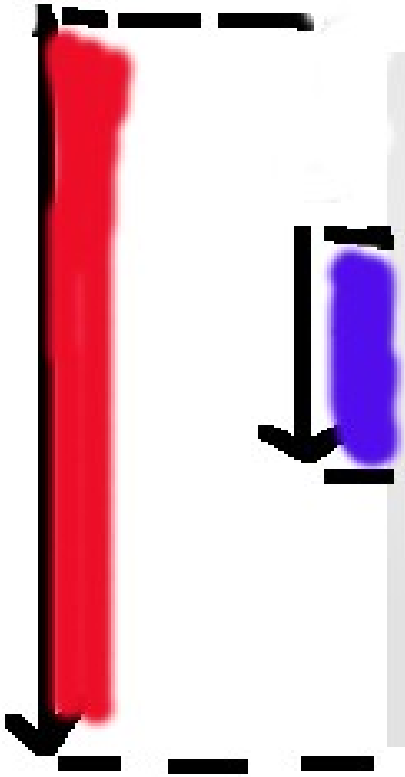
Variable scope inside the function

- Parameters and variables assigned within a function body only have a scope to the end of that function body. They do not exist outside the function.
- The following code will cause an error if 'i' and 'times' are not defined elsewhere outside of printMessage().

```
def printMessage(message,times):
    i = 0
    for j in range(times):
        print(message,i)
...
print(i, times)
```

global and local namespaces overlap....

- global namespace
  - times = 5
  - printMessage
- local namespace
  - times = 2
  - message = "HELLO"
  - i = 0



```
times = 5

def printMessage(message,times):
    for i in range(times):
        print(message)


printMessage(“HELLO”,2)
```

Variable Scope Example

- (See moodle site for a zip file of code examples):
  - scope_example1.py

Global statement

- Global variables can be read inside a function but ***cannot be changed***. Use the **global statement** to change a global variable inside a function.

```
i = 0
....
def change_i():
   i = 2
....
change_i()
print(i)
```

This will print 0.

```
i = 0
....
def change_i():
   global i
   i = 2
...
change_i()
print(i)
```

This will print 2

Global Statement Example

- (See moodle site for a zip file of code examples):
  - global_statement.py

Variable Scope outside the file

- When variable and function names are imported into a file, they act like global variables. This can create a large amount of names in your namespace that you are not aware of.
- To cut down on 'namespace pollution', a good coding practice is to only import what is needed into the file.

```
import sys                              # good
from sys import exit, exc_info          # good
from sys import *                       # not a good idea but
                                        # acceptable under
                                        # some circumstances
```

A Word of Caution About Changing Global Variables Inside A Function

- Changing a global variable can lead to unexpected behavior. (see global_statement.py example).
- **So don't change global variables unless you have no other choice.**
- NOTE: It's okay to set up a global variable that is only read but not changed. These are called **constant variables.**

Import Example

- (See moodle site for a zip file of code examples):
  - import1.py
  - my_module.py