Weifan Lin

Csc342 Section G

02/11/2015

Homework

Title:

Summary on Local Variables

Objective:

The goal of this lecture was to understand how local variables are used in computer programming.

Definition:

A variable that is declared inside a function definition is called local variable. Every time when the function is executed the local variable(s) are created and destroyed. Local variables can only be accessed by the code inside the function, and they are not accessible from the code outside the function.

Functionality:

A local variable is used as a method to assign a temporary value to an object. In a running program, when a function is called, variable(s) inside the function (local variable) are created and they have been assigned temporarily with the values that are inside the function and they don’t possess these values when the function is ended.

Design:

In C:

```
1   #include <stdio.h>
2   int main()
3   {
4   int a=90;
5       {
6       int b = 100;
7       printf("%d %d",a,b);
8       }
9   printf("%d %d",a,b);
10  }
```

Result:

```
Stefans-iMac:G Lin$ gcc localvariable.c -o c
localvariable.c:9:18: error: use of undeclared identifier 'b'
printf("%d %d",a,b);
                 ^
1 error generated.
```

Inside the scope from line 5 to 8, when local variable b is declared to be 9, it is only accessible inside the scope. In line 9, b is undeclared because any code outside of the scope from line 5 to 8 cannot access to b as we can see the error when we compile the code.

In C++:

```cpp
#include <iostream>
using namespace std;

void some_function_1()
{
    //local variables a b c in some_function_1
    int a = 10;
    int b = 20;
    int c = 30;
    cout << a << " "<< b << " "<< c <<endl;
}
void some_function_2()
{
    //local variables a b c in some_function_2
    int a = 50;
    int b = 60;
    int c = 70;

    cout << a << " "<< b << " "<< c <<endl;
}
int main()
{
    int a = 1;
    int b = 2;
    int c = 3;
    cout << a << " "<< b << " "<< c <<endl;
    some_function_1();
    some_function_2();
    cout << a << " "<< b << " "<< c <<endl;
}
```

Result:

```
Stefans-iMac:G Lin$ ./localvariable
1 2 3
10 20 30
50 60 70
1 2 3
```

In the main function, we have three local variables: a=1, b=2, c=3. So the first time printing a,b,and c is 1,2,3. Then we called the function some_function_1, in some_function_1 the local variable a,b,c are assigned to 10,20,30, so printing a,b,c will

result in 10,20,30 before some_function_1 ended. Same thing happened when we called some_function_2, a,b,c changed into the values inside some_function_2 and printed the results and the function ended. Lastly, we printed a,b,c again, but this time the result is same as the first time because it's printing inside the main function so the values of a,b,c are referred to the local variable inside the main function but not in some_function_1 or some_function_2.

In Java:

```java
public class localvariable
{
    static int a = 10;
    static int b = 20;

    public void function1()
    {
        int a = 0;
        int b = 1;
        System.out.println(a + " " + b);
    }

    public static void main(String args[])
    {
        localvariable fun = new localvariable();
        fun.function1();
        System.out.println(a + " " + b);
    }
}
```

Result:

```
stefan@Ubuntu:~/Desktop$ java localvariable
0 1
10 20
```

We have two static variables a and b. Inside the function function1 there are two local variables a and b have values 0 and 1. In the main function, we created a new instance fun and called fun.function1. The program first printed 0 and 1 as local variable in function1. Then it printed 10 and 20 from line 17.

In Python:

```
1  def function0():
2      y = 0
3      print x, y
4
5  def function2():
6      x = 5
7      print x, y
8
9  x = 1
10 y = 2
11 function0()
12 function2()
13 print x, y
```

Result:

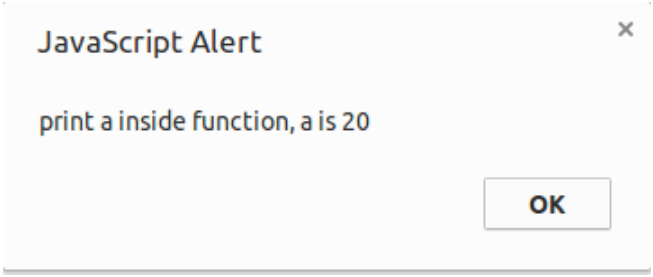
```
stefan@Ubuntu:~/Desktop$ python localvariable.py
1 0
5 2
1 2
```

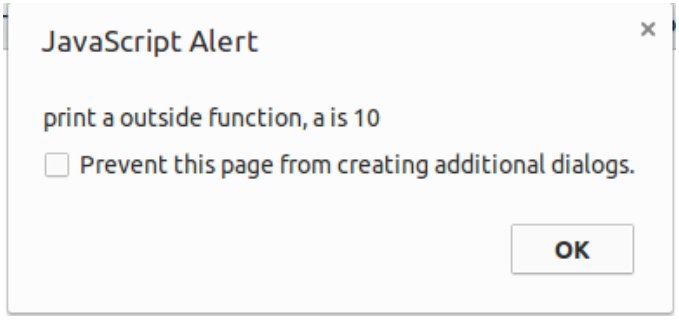
x = 1 and y = 2 are global variables in Python. In line 11 we called the function function0, we have 1 and 0 as results, because y value has been changed to 0 but x value remains the same because there is no x assignment in the function0. Similarly we called function2 in line 12, we have 5 and 2 as results, y remains the same as global variable and x has been changed to 5 as local variable in function2. Lastly, we have 1 and 2 referred to global variables.

In JavaScript:

```
<!DOCTYPE html>
<html>
<script>
var a = 10;

function1();

function function1()
    {
    var a = 20;
    alert("print a inside function, a is " + a);
    }

alert("print a outside function, a is " + a);
</script>
<body>

</body>
</html>

```

Result:





Inside of script tags we have a global variable a = 10. When we called funcion1, a changed into value of 20 as local variable inside function1. When function1 ended a has value of 10 as global variable.