PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

Closures

Prof. Sindhu R Pai
PCPS Theory Anchor - 2024
Department of Computer Science and Engineering

Functions - Closure

A closure is a nested function which has access to a free variable from an enclosing function that has finished its execution.

Three characteristics of a Python closure are:
- It is a nested function
- It has access to a free variable in outer scope
- It is returned from the enclosing function

A free variable - a variable that is not bound in the local scope.

Closures with immutable variables such as numbers and strings - use the nonlocal keyword.

Functions - Closure

1. Example:

```
def  outer(msg): # This is the outer enclosing function
   def inner():   # This is the nested function
      print(msg)
   return inner  # returns the nested function

# Now let's try calling this function.
different = outer(“This is an example of closure")
different () #refers to inner()
```

Output:

```
This is an example of closure
```

2. Example:

```
def f1(): #outer function
        def f2(): #inner function
                print ("Hello")
                print ("world")
        print('f2=',id(f2))
        return f2
c=f1() #refers to f2()
c()
print('c=',id(c)) #id of f2() and c() are same
```

Output:

```
f2= 2908823806840
Hello
world
c= 2908823806840
```

Functions - Closure

3. Example:

```
def division(y): #outer function
    def divide(x): #inner function
        return x/y
    return divide

d1=division(2) #refers to divide
d2=division(3) #refers to divide

print(d1(20))
print(d2(96))
```

Output:

```
10.0
32.0
```

Functions - Closure

4. **Example**:

```
def f1(): #outer function
        def f2(): #inner function
                print ("Hello")
                print ("world")
        return f2
c=f1() #refers to f2()
del f1
c() #still works
```

Output:

```
Hello
world
```

Functions - Closure

5. **Example**:

```
def outer(msg):
    text = msg        #text is having the scope of outer function
    def inner():
        print(text)     #using non-local variable text
    return inner      #return inner function


func = outer('Hello')
func()
```

Output:

```
Hello
```

Functions - Closure

6. **Example**:

```
def outerfunc(x):
        def innerfunc():
                print(x)
        return innerfunc

myfunc=outerfunc(7)
myfunc() #refers to innerfunc()
del outerfunc
myfunc() #still refers to innerfunc() retaining the value of enclosing scope of x
```

Output:

```
7
7
```

We are assigning the function outerfunc() to the variable myfunc. Even if we delete outerfunc() from the memory, the function outerfunc() can be called, using the referred variable myfunc.

Functions - Closure

7. **Example**:

```
def f1():    #outer function
   x=0
   def f2(): #inner function
      nonlocal x     # x - that belongs to scope of outer function is made non-local
      x=x+1
      return x
   return f2

func = f1()
retval = func()
print ("x=", retval)
retval = func()
print ("x=", retval)
```

Output:

```
x= 1
x= 2
```

Function Closure vs. Nested function

- Not all nested functions are closures.
- For a nested function to be a closure, the following conditions need to be satisfied:

   1. The inner function has access to the non-local variables or local variables of the outer function.

   2. The outer function must return the inner function.

Function Closure vs. Nested function

Example: Nested Function but not Closure (When msg is passed to inner(), msg ends up belonging to inner() function's local scope. So, the 1st condition is not satisfied)

```
def  outer(msg):          # This is the outer enclosing function
   def inner(m=msg):    # This is the nested function
      print(m,"World")
   return inner           # returns the nested function

different = outer(msg="Hello")
different()               #refers to inner()
```

Output

```
Hello World
```

Function Closure: Summary

- A function object that remembers values in enclosing scopes even when the variable goes out of scope.

- Python closures help avoiding the usage of global values and provide some form of data hiding. They are used in Python decorators.

THANK YOU

Department of Computer Science and Engineering

Dr. Shylaja S S, Director, CDSAML & CCBD, PESU
Prof. Sindhu R Pai – sindhurpai@pes.edu
Prof. Sowmya Shree P