

Scripting Languages G22.3033-002 Summer 2008

Example solutions for hw10

Assigned Th 7/24/2008, due Fr 8/1 at 9pm. 50 points.

<http://www.cs.nyu.edu/courses/summer08/G22.3033-002/>

Instructions for example solutions

These are example solutions. Please keep in mind that often, there is not just one correct solution to a question. If you come up with different answers, then it may be that both your answers and these answers here are correct. Of course, these answers here may also contain mistakes. If you spot a mistake, please let me know so I can correct it.

Example solutions for Homework 10

solutions-hw10-1 Iterating with Coroutines

a.

```
preorder:
1
2
4
5
3
6
7
```

b.

```
#!/usr/bin/env python
class Tree:
    def __init__(self, value, left=None, right=None):
        self.value = value
        self.left = left
        self.right = right
    def preorder(self):
        yield self.value
        if self.left:
            for v in self.left.preorder():
                yield v
        if self.right:
            for v in self.right.preorder():
                yield v
    def inorder(self):
        if self.left:
            for v in self.left.inorder():
                yield v
        yield self.value
        if self.right:
```

```

        for v in self.right.inorder():
            yield v

my_tree = Tree(1,
               Tree(2,Tree(4),Tree(5)),
               Tree(3,Tree(6),Tree(7)))

def print_preorder(tree):
    print 'preorder:'
    for y in my_tree.preorder():
        print y

print_preorder(my_tree)

def print_inorder(tree):
    print 'inorder:'
    for y in my_tree.inorder():
        print y

print_inorder(my_tree)

```

solutions-hw10-2 Iterating with Block Parameters

a.

```

preorder:
1
2
4
5
3
6
7

```

b.

```

#!/usr/bin/env ruby
class Tree
  attr_accessor :value, :left, :right
  def initialize(value, *leftRight)
    @value = value
    if 0 < leftRight.length()
      @left = leftRight[0]
      @right = leftRight[1]
    end
  end
  def preorder()
    yield @value
    if @left
      @left.preorder { |v| yield v }
    end
  end
end

```

```

        end
        if @right
            @right.preorder {|v| yield v }
        end
    end
end

$my_tree = Tree.new(1,
                    Tree.new(2, Tree.new(4), Tree.new(5)),
                    Tree.new(3, Tree.new(6), Tree.new(7)))

def print_preorder(tree)
    puts 'preorder:'
    tree.preorder {|y| puts y.to_s + "\n" }
end

print_preorder($my_tree)

def print_sum(tree)
    puts "sum:"
    sum = 0
    tree.preorder {|y| sum += y }
    puts sum
end

print_sum($my_tree)

```

solutions-hw10-3 Coroutines vs. Block Parameters

- a. Method `Tree.preorder` in the Python example above is a coroutine. Every time the co-routine yields, it returns control to the caller. When the caller calls it again, the coroutine resumes where it left off.

Method `Tree.preorder` in the Ruby example above takes a block as a parameter. Every time the Ruby method `Tree.preorder` yields, it calls the block. When the block returns, the method continues just like after a normal call.

The difference is that the Python coroutine uses `yield` to return to its caller, whereas the Ruby method uses `yield` to call its block.

- b. From the user's perspective, Python's coroutines more completely hide the complexity of what's going on. Another advantage that is not illustrated by this example is that a coroutine call need not happen in a loop, it can be called from multiple places in regular computation.
- c. Most scripting languages already provide closures in one form or another. Block parameters are a simple variation on that. This makes it easier to implement the script interpreter. Also, if the user already understands closures, block parameters are easier to understand.

solutions-hw10-4 Iterating with Closure Parameters (PHP)

```
#!/usr/bin/env php
<?php
class Tree {
    var $value, $left, $right;
    function __construct($value, $left=null, $right=null) {
        $this->value = $value;
        $this->left  = $left;
        $this->right = $right;
    }
    function preorder($closure) {
        $closure($this->value);
        if ($this->left):
            $this->left->preorder($closure);
        endif;
        if ($this->right):
            $this->right->preorder($closure);
        endif;
    }
}

$myTree = new Tree(1,
                  new Tree(2, new Tree(4), new Tree(5)),
                  new Tree(3, new Tree(6), new Tree(7)));

function printPreorder($tree) {
    echo "preorder:\n";
    $tree->preorder(create_function('$v', 'echo "$v\n";'));
}

printPreorder($myTree);
?>
```