Euler 0018

The Problem:

Moving through a triangle of number (every number chosen yields 2 more numbers to choose), what is the highest sum path in the triangle?

Considerations and Approach:

We will populate a tree data structure with the data from the triangle, and then work from the bottom up to find the best path.

Starting at the bottom of the tree take the value of the nodes themself, then when moving up the tree, take the best value from the left and right children and add it to the sum.

This results in the top node having the best sum.

The Code:

```
triangle_file = open("triangle", "r")

class Node:
    def __init__(self, value):
        self.l_node = None
        self.r_node = None

        self.value = value
        self.l_sum = -1
        self.r_sum = -1
        self.best = -1

#not necessarily the prettiest way to construct a tree... but oh well.
#there is a lot by reference here.
```

```
top_triangle = int(triangle_file.readline())
triangle_tree = Node(top_triangle)
past_line = [triangle_tree]
for line in triangle_file:
    current_line = [Node(int(x)) for x in line.split()]
    for i in range(len(past_line)):
        past_line[i].l_node = current_line[i]
        past_line[i].r_node = current_line[i+1]
    past_line = current_line

#make a function to populate the sums
def pop_sums(node : Node):
    if node.l_node is None:
        node.l_sum = node.value
        node.r_sum = node.value
        node.best = node.value
    else:
        #make sure both below are populated
        if node.l_node.best == -1:
            pop_sums(node.l_node)
        if node.r_node.best == -1:
            pop_sums(node.r_node)

        node.l_sum = node.l_node.best + node.value
        node.r_sum = node.r_node.best + node.value
        node.best = node.l_sum if node.l_sum > node.r_sum else node.r_sum

pop_sums(triangle_tree)

print(triangle_tree.best)
```

Revision #2
Created 2025-11-11 16:17:57 UTC by Maxwell
Updated 2025-11-11 16:36:18 UTC by Maxwell