## Boolean Expression Evaluation

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>True or 4 == 3</td>
<td></td>
<td>False</td>
</tr>
<tr>
<td>3 &lt; 4 &lt;= 10</td>
<td></td>
<td>True</td>
</tr>
<tr>
<td>not(4 == 5) or 6 &gt; 5 and 4 &gt; 5</td>
<td></td>
<td>True</td>
</tr>
<tr>
<td>6 + 2 &gt; 10 &lt; 4 * 2</td>
<td></td>
<td>True</td>
</tr>
<tr>
<td>True or False</td>
<td></td>
<td>True</td>
</tr>
<tr>
<td>True and False</td>
<td></td>
<td>True</td>
</tr>
<tr>
<td>not(True) or True or not(False)</td>
<td></td>
<td>True</td>
</tr>
<tr>
<td>7 + 3 / 2 &gt;= 8</td>
<td></td>
<td>True</td>
</tr>
</tbody>
</table>
Code Tracing

What is printed when the following code is executed?

```python
l = ["open", "close", "in", "out", "up", "down"]
for i in range(0,6,2):
    print l[i]
```
List Practice

What will be the output of the following sequence of code?

```python
mylist = ["Hello", 88.5, 100, "cs190"]
print len(mylist)
print mylist[0]
print mylist[len(mylist)]
print mylist[2:]
print mylist[0:2]
mylist.append(140)
print mylist
mylist[0] = 4
mylist[3] = 200
print mylist
mylist.sort()
print mylist
mylist.reverse()
print mylist
```
def fun1(x):
    print "Fun1 x:", x
    return x * 2
print "Start"
y = fun1(10)
if (5 > y):
    print y
elif (15 > y):
    print y + 10
elif (25 > y):
    print y + 100
elif (35 > y):
    print y + 1000
else:
    print y + 10000
print "End"
Write a function called two which takes a list of numbers as an input parameter. This function should return True if 2 appears as either the first or last element in the array. The list will be length 1 or more. Examples of function calls:

two([1, 2, 2]) → True
two([2, 1, 2, 3]) → True
two([13, 6, 1, 2, 3]) → False
Function Writing(2)

Write a function called stringTimes that takes two input parameters:
- A string
- A non-negative integer representing the number of times a string should be copies.

The function should return a larger string that is n copies of the original string. You **may not** use the * operator to accomplish this repetition.

Examples:

```
stringTimes('Hi', 2) → 'HiHi'
stringTimes('Hi', 3) → 'HiHiHi'
stringTimes('Hi', 1) → 'Hi'
```
Function Writing(3)

Write a function called sumEvens that takes a non-negative integer as an input parameter. The function should return the sum of the even numbers between 0 and the input parameter. If the input parameter is even, it should be included in the sum. Examples:

sumEvens(2) → 2
sumEvens(3) → 2
sumEvens(6) → 12