Activity: `getOddWeights(g)`

Given a directed, weighted graph, return a list of all the odd weights in the graph.

For example, given the following graph:

```
g = {
    "A": [ ["B", 4], ["C", 7] ],
    "B": [ ["C", 1], ["D", 3] ],
    "C": [ ],
    "D": [ ["B", 1] ],
}
```

`getOddWeights(g)` would return `[7,1,3,1]`

Solution: getOddWeights(g)

```
def getOddWeights(g):
    result = []
    for node in g:
        for edge in g[node]:
            if edge[1] % 2 != 0:
                result.append(edge[1])
    return result
```

Activity: Compute the Big-O

Consider the following function. What is its Big-O runtime in the worst case?

```
def example(s):
    result = ""
    for i in range(len(s)//2, len(s)):
        result = s[i] + result

    for j in range(len(s)//2):
        if s[j].isupper():
            result = result + s[j].lower()
        else:
            result = result + s[j]
    return result
```

Answer: O(n). Each loop does n/2 iterations, and each loop body does O(1) work. The calls to isupper and lower do run in linear time wrt the length of the string, but they're called on single characters, so it is still constant work one.

Activity: What does this print?

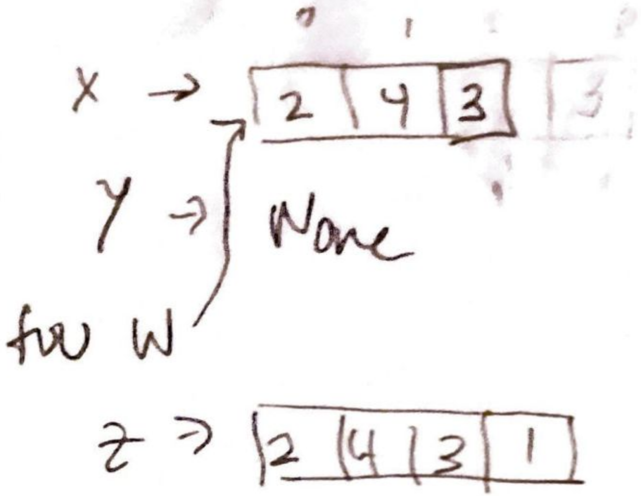
```
def foo(W):
    W.pop(2)
    return W + [1]

X = [2, 4, 6]
Y = X.append(3)
Z = foo(X)

print("X:",X)
print("Y:",Y)
print("Z:",Z)
```

Hints:
Is `Y` aliased to `X`? **No**

Is `W` aliased to `X`? **Yes**

Will `lst + <anything>` make a new list? **Yes**

Does `lst.append` make a new list? **No - append is a mutating method**



X: [2,4,3]
Y: None
Z: [2,4,3,1]