The goal of this worksheet is to reinforce some common principles of algorithm running time analysis that we have been using all term. Each code snippet/function/method below has some input variables, and performs some operations. For each one, write down beside it its Big-Oh worst-case running time, as well as a brief explanation.

Remember that a **constant time** operation is one whose running time does not depend on the size of the input. All Python arithmetic and comparison operations, variable/attribute access and assignments, and `print` and `return` all take constant time.

<table>
<thead>
<tr>
<th># n and m are numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>print(n)</code></td>
</tr>
<tr>
<td><code>print(m)</code></td>
</tr>
<tr>
<td><code>print(n + m)</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># A is a list of numbers of length r</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>s = 0</code></td>
</tr>
<tr>
<td><code>y = 10</code></td>
</tr>
<tr>
<td><code>for x in A:</code></td>
</tr>
<tr>
<td><code>s += x</code></td>
</tr>
<tr>
<td><code>print(x)</code></td>
</tr>
<tr>
<td><code>print(s * y)</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># A is a list of numbers of length r</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>i = 0</code></td>
</tr>
<tr>
<td><code>s = 0</code></td>
</tr>
<tr>
<td><code>while i &lt; len(A):</code></td>
</tr>
<tr>
<td><code>s += A[i]</code></td>
</tr>
<tr>
<td><code>print(A[i])</code></td>
</tr>
<tr>
<td><code>i += 10</code></td>
</tr>
<tr>
<td><code>print(s)</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># A is a list of length n, B is a list of length m</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>for x in A:</code></td>
</tr>
<tr>
<td><code>print(x)</code></td>
</tr>
<tr>
<td><code>for y in B:</code></td>
</tr>
<tr>
<td><code>print(10)</code></td>
</tr>
</tbody>
</table>
# A is a list of length n, B is a list of length m
for x in A:
    print(x)

for y in B:
    print(10)

# A is a list of length n
for x in A:
    if x == 1:
        print(x)
    else:
        print(x * 2)
        print(x * 3)

# a and b are positive numbers
if a > b:
    for i in range(b):
        print(a + i)
else:
    for i in range(a):
        print(b + i)

# A is a list of length p
for x in A:
    for i in range(10):
        print(x * i)

# A is a list of length n, k is a positive number
for x in A:
    for _ in range(k * k):
        print(x)
# x is a positive number
i = 1
while i < x:
    print(i)
    i *= 2

# A is a list of length n
def find_duplicates(A):
    i = 0
    while i < len(A):
        j = i + 1
        while j < len(A):
            if A[i] == A[j]:
                print('Duplicate found')
                return True
            j += 1
        i += 1
    return False

# A is a list of length n, k is a positive number
for i in range(k):
    A.append(i)

# A is a list of length n, k is a positive number
for i in range(k):
    A.insert(0, i)