0. (1 Point) On a scale of Brian to Sumukh, how do you feel? Brian ______________________ Sumukh
01. (4 Points) PUMPKIN FUN

For each of the expressions in the table below, write the output displayed by the interactive Python interpreter when the expression is evaluated. The output may have multiple lines. Expressions are evaluated in order, and expressions may affect later expressions.

Whenever the interpreter would report an error, write Error. If execution would take forever, write Forever. Assume that you have started Python 3 and executed the following statements:

class Patch:
    pop = [0, 1, 2, 3, 4]
    best = 3

    def harvest(self, i):
        if self.pop.pop(i):
            print('Goodbye', i, ':(')
            self.steal(i-1)

    def steal(self, i):
        print(self.pop)
        if i % 2 == 0:
            self.pop.append(self.pop[:i])
        self.harvest(i)

<table>
<thead>
<tr>
<th>Expression</th>
<th>Interactive Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>p = Patch()</td>
<td></td>
</tr>
<tr>
<td>q = Patch()</td>
<td></td>
</tr>
<tr>
<td>p.harvest(2)</td>
<td></td>
</tr>
<tr>
<td>q.harvest(1)</td>
<td></td>
</tr>
<tr>
<td>p.pop = [0, 1, 2, 3]</td>
<td></td>
</tr>
<tr>
<td>Patch.steal(q)</td>
<td></td>
</tr>
<tr>
<td>p.harvest(len(q.pop))</td>
<td></td>
</tr>
</tbody>
</table>
02. (10 Points) TO BRIAN OR TO SUMUKH

Fill in the environment diagram that results from executing the code below until the entire program is finished, an error occurs, or all frames are filled. You may not need to use all of the spaces or frames.

A complete answer will:

• Add all missing names and parent annotations to all local frames.
• Add all missing values created or referenced during execution.
• Show the return value for each local frame.

Remember: Do not add a new frame when calling a built-in function (such as abs). The built-in abs function is always written as func abs(...) [parent=Global].

def dilemma(denero):
    def to(sumukh):
        nonlocal denero
        def orto(brian):
            denero = sumukh[2]
            brian[2] = sumukh[1:]
            return sumukh
        sumukh[0] = [1, 2]
        denero = orto(sumukh)
        return denero
        return to([denero]*3)

dilemma(2)
03. (15 Points) TREES ARE FOR KIDS

(a) (8 points) Implement max_function, which returns the maximum value of f(x, y), where x is an entry and y is one of its children. Both the Tree class and the tree data abstraction appear on the midterm 2 study guide.

Warning: Do not violate the tree data abstraction! (Exams are flammable.)

```python
def max_function(t, f):
    """Returns the maximum value of f(x, y) where x is the value of an entry and y is the value of that entry's immediate child. 0 is the smallest possible value.

>>> dist = lambda x, y: pow(x**2 + y**2, 1/2)
>>> t = tree(3, [tree(4, [tree(5), tree(6)]),
...    tree(2, [tree(1), tree(3), tree(7, [tree(8)])]))])
>>> max_function(t, dist)  # sqrt( 7^2 + 8^2 )
10.63014581273465
>>> diff = lambda x, y: y-x
>>> max_function(t, diff)  # 7 - 5
5
""
    if ________________:
        return ___
    children = ________________________________
    return max(__________________________)
```

(b) (5 points) Using only parentheses, square brackets, “tree,” and “lambda”, complete the following so that max_function(hug, denero) yields 2015. Only one of the four items above may go in each blank.

denero = _______ x, y: x_ _ _0_ ___ y
hug = tree(______: _2015_, ______(1)_)  
max_function(hug, denero)

(c) (2 points) Compute the runtime of max_function with respect to the number of nodes in the tree, assuming the definition of f below is passed in as max_function(t, f):

```python
def f(x, y):
    for i in range(x):
        print(i)
```

Runtime:
Implement \texttt{is\_uniform}, which takes in a tree and determines whether or not the tree is uniformly wide at each level. The width of a tree at level \(i\), is the sum of all entries at depth \(i\). Note that this solution implements \textit{breadth-first search}, where the tree is traversed and consequently processed a level at a time.

```python
def is_uniform(t):
    ""
    
    >>> t = tree(3, [tree(4, [tree(5), tree(6)]), tree(2, ...
    ... [tree(1), tree(3), tree(7, [tree(8)])]])
    >>> is_uniform(t)
    False
    >>> t = tree(8, [tree(6, [tree(3), tree(5)]), tree(2), tree(0)])
    >>> is_uniform(t)
    True
    >>> t = tree(8, [tree(6, [tree(2), tree(1)]), tree(2, ...
    ... [tree(2), tree(3), tree(0, [tree(8)])]])
    >>> is_uniform(t)
    True
    ""

    queue, curr, vals, width = ___, ___, ___, __________
    while _________________:
        t = _____________
        curr = _______________
                      _______________
        if not __________:
            if _________________:
                return False
        queue, curr, vals = ______, ___, ______
    return _________________
```
BONUS. (0 Points) CINDY’S SOCKS AND SANDALS

Cindy is starting a new trend in footwear, but even divas need breaks. Using the following rules, determine if Cindy can and should wear sandals based on the weather:

- Cindy does not wear socks and sandals within 2 days of a rainy day.
- Nathan steals Cindy’s socks and sandals every other day after a rainy day.
- Paul performs a rain dance after six days without rain, to invoke another rainstorm.
- Assume the first day is a rainy day.

```python
def siri_generator():
    """
    >>> siri = siri_generator()
    >>> siri()
    'Just rained.'
    >>> siri()
    'The Nathan attacked!'
    >>> siri(4)
    'Paul danced.'
    >>> siri(3)
    'Cindy wore socks and sandals.'
    """
    data = 
    def siri(days=1):
        nonlocal data
        msg = ''
        data +=
        rain, nathan, paul = 
        if 
            msg, paul, nathan, rain = 
        elif 
            msg, nathan = 
        elif 
            msg = 
        else:
            msg = 
        data =
        return msg
    return siri
```