1. WWPD The beginning of the end (12 points)

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. If an error occurs, write “Error” and if nothing is display, write “Nothing”. You answers must fit within the boxes provided.

Hint: No answer requires more than 4 lines. (It’s possible that all of them require even fewer.) The first two rows have been provided as examples. Recall: The interactive interpreter displays the value of a successfully evaluated expression, unless it is None.

Assume that you have started a python3 interpreter and executed the following statements:

```python
they = lambda x: return len(x())

def cant(y, x):
    return lambda: [y(x + 1) for i in range(x) if x % 2 == 0]

def stop(naruto_running):
    if naruto_running > 0:
        print('SASUKEEE')
    return 2019

of = ['high', 'school']
of = of + [of[0] + of[1]]
us = len(of)
def all(in_this, together):
    if 'highschool' in in_this:
        return 2
    if together % 7 == 3:
        return 3
    else:
        print('musical')

class Clown:
    fles = 1
    def __init__(self, se, lf):
        self.se = lf
        self.lf = self
        self.fles = Clown.fles
        Clown.fles += self.fles

stop = Clown(Clown, Clown)
no = Clown(stop, Clown)
why = Clown(no, stop)
```
<table>
<thead>
<tr>
<th><strong>Expression</strong></th>
<th><strong>Output</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>us</td>
<td>3</td>
</tr>
<tr>
<td>they</td>
<td>Function</td>
</tr>
<tr>
<td>(y,y) for y in [ x for x in range(3) ]</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>they(cant(stop, all(of, us)))</td>
<td>Error</td>
</tr>
<tr>
<td>sleep = lambda zzz: zzz * 8 cant(sleep, len('well'))()</td>
<td>[40, 40, 40, 40]</td>
</tr>
<tr>
<td>Clown.fles</td>
<td>8</td>
</tr>
<tr>
<td>why.se == why.lf</td>
<td>True</td>
</tr>
<tr>
<td>assert isinstance(stop, Clown)</td>
<td>Nothing</td>
</tr>
</tbody>
</table>
2. Environment Diagram (Sophia)

```python
x = [100, lambda budget: x[0]]
def how(vacation):
    return vacation // 10

def food(money, budget):
    money = x[0]
    work = x[1]
    def how(food):
        return how is your(food)
    def your(fav):
        while favorite == 100:
            x = lambda fav: work(money) - 2
            favorite = x(fav)
        x.append(5)
    return how

vacationPrice, discount = how(x[0]), 50
is_food = food(vacationPrice, discount)(100)
```

6 frames, Python Tutor
3. **(10 points) SQL: CS88Rising**

We have compiled information on popular artists that are a part of the popular music group, 88rising. Knowledge of the group is not required to answer the following questions. Given these following two tables, complete the following SQL queries so that they return the requested data. (“Hard coding” answers to return only the exact data will not receive credit.)

Table: **Artists**

<table>
<thead>
<tr>
<th>stage_name</th>
<th>real_name</th>
<th>place_of_birth</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joji</td>
<td>George Miller</td>
<td>USA</td>
<td>27</td>
</tr>
<tr>
<td>NIKI</td>
<td>Nicole Zefanya</td>
<td>Indonesia</td>
<td>20</td>
</tr>
<tr>
<td>Rich Brian</td>
<td>Brian Soewarno</td>
<td>Indonesia</td>
<td>20</td>
</tr>
<tr>
<td>Higher Bros.</td>
<td>更高兄弟</td>
<td>China</td>
<td>25</td>
</tr>
<tr>
<td>Oski Bear</td>
<td>Oski</td>
<td>USA</td>
<td>79</td>
</tr>
</tbody>
</table>

Table: **Songs**

<table>
<thead>
<tr>
<th>title</th>
<th>artist</th>
<th>length (in seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>History</td>
<td>Rich Brian</td>
<td>307</td>
</tr>
<tr>
<td>Head in the Clouds</td>
<td>Joji</td>
<td>178</td>
</tr>
<tr>
<td>Indigo</td>
<td>NIKI</td>
<td>173</td>
</tr>
<tr>
<td>Hold Me Down</td>
<td>Higher Bros.</td>
<td>223</td>
</tr>
<tr>
<td>Sanctuary</td>
<td>Joji</td>
<td>180</td>
</tr>
</tbody>
</table>

A. Write a SQL query that retrieves the `stage_name` of all artists who are not from the USA.

```
SELECT Stage_Name FROM Artists WHERE Place_of_Birth != USA
```

B. Write a SQL query that lists the `real_name` and `place_of_birth` of all artists in descending order based on their age.

```
SELECT Real_Name, Place_of_Birth FROM Artists ORDER BY Age DESC
```

C. Write a SQL query that lists the `title` and `real_name` of a song’s artist in descending order based on the length of the respective song.

```
SELECT s.Title, a.Real_Name FROM Artists as a, Songs as s WHERE a.Stage_Name = s.Artist ORDER BY s.Length DESC
```

D. Write a SQL query that gets all the `title` of all unique pairs of songs where both their respective artists are younger than 27.

```
SELECT a.Title, b.Title FROM Songs as a, Songs as b, Artists as c, Artists as d WHERE c.Stage_Name = a.Artist and d.Stage_Name = b.Artist and a.Title < b.Title and c.Age < 27 and d.Age < 27
```
4. (6 points) Lists of Lists of Lists of Lists...
Write a remove function that takes in a linked list and a number and returns a new linked
list with that number removed in the new linked list. If the element does not exist in the
linked list, return a copy of the original linked list.

```python
>>> lnk_lst = Link(1, Link(2, Link(3, Link(4, Link(5)))))
>>> remove(lnk_lst, 2)
Link(1, Link(3, Link(4, Link(5))))
>>> lnk_lst
Link(1, Link(2, Link(3, Link(4, Link(5))))))

def remove(linked_lst, num):
    if linked_list == Link.empty:
        return Link.empty
    if linked_list.first == num:
        return remove(linked_list.rest, num)
    else:
        return Link(linked_list.first, remove(linked_list.rest, num))
```

What is the runtime of this function with respect to the length of the list? (Bubble one)
○ Constant ○ Logarithmic  ○ Linear ○ Quadratic ○ Exponential

```python
class Link:
    """Reference of the Link class."
    
    empty = ()

    def __init__(self, first, rest=empty):
        assert rest is Link.empty or isinstance(rest, Link)
        self.first = first
        self.rest = rest

    def __len__(self):
        return 1 + len(self.rest)

    def __repr__(self):
        if self.rest:
            rest_str = ', ' + repr(self.rest)
        else:
            rest_str = ''
        return 'Link({0}{1})'.format(self.first, rest_str)
```
5. (6 points) Get Your Numbers in a Row
Complete the isStrictlyIncreasing function and its helper function below to return True if a number has strictly increasing digits from left to right and False otherwise. You may assume a single digit number is specified to be strictly increasing and that the number is always nonnegative.

Use the following lines of code, which are currently out of order as a basis for your solution. You may need to fill in the blanks.

```python
def isStrictlyIncreasing(num):
    """
    >>> isStrictlyIncreasing(8)
    True
    >>> isStrictlyIncreasing(21)
    False
    >>> isStrictlyIncreasing(133)
    False
    >>> isStrictlyIncreasing(569)
    True
    """
    def helper(n, prev):
        """
        if n == 0:
            return True
        if prev > n % 10:
            return helper(n // 10, n % 10)
        else:
            return False
        return helper(n // 10, n % 10)
```

```python
return True
return helper(___________, ____________)
return False
if n == __________:
else:
    if prev ___ __________:
```
6. (15 points) One Hungry Artist
You are a famous artist and for your newest exhibit you decided to tape a banana to a wall. The banana starts out unripe, but when it ripens someone comes and eats your banana, and then you can replace banana back with an unripe banana once it is eaten. The banana in the artpiece can only be “UNRIPE”, “RIPE”, or “EATEN”.

def create_art(state):
    """Create the art""
    return { 'banana?': [state] }

Use the following set of doctests to understand the behavior of the functions on the next page.

>>> art = create_art("UNRIPE")
>>> is_unripe(art)
True
>>> set_banana("RIPE", art)
>>> is_ripe(art)
True
>>> is_unripe(art)
False
>>> is_eaten(art)
False
>>> eat_banana(art) # We eat our ripe banana
>>> is_ripe(art)
False
>>> is_eaten(art)
True
>>> ripen_banana(art) # Does nothing
>>> exhibit = generator(art)
>>> next(exhibit)
"There is no banana."
>>> next(exhibit)
"There is no banana."
>>> replace_banana(art)
>>> next(exhibit)
"Eww, unripe banana."
>>> ripen_banana(art)
>>> next(exhibit)
"The banana looks so delicious."
>>> set_banana("Broken Exhibit", art) # set to a not expected value
>>> next(exhibit) # cause a StopIteration Exception
StopIteration: Unknown banana state
Fill out the rest of the functions to get the corresponding doctest to have the right output.

```python
def set_banana(banana, art):
    "Update the art"
    art[‘banana?’][0] = banana

def is_ripe(art):
    "Return whether the art’s internal state is ‘RIPE’"
    return art[‘banana’][0] == ‘RIPE’

def is_unripe(art):
    "Return whether the art’s internal state is ‘UNRIPE’"
    return art[‘banana’][0] == ‘UNRIPE’

def is_eaten(art):
    "Return whether the art’s internal state is ‘EATEN’"
    return art[‘banana’][0] == ‘EATEN’
```

For the next set of functions, you must not violate the abstraction barrier! Any abstraction violations will have points deducted.

```python
def ripen_banana(art):
    "Only ripen banana if it is unripe"
    if is_unripe(art):
        set_banana(“RIPE”, art)

def replace_banana(art):
    "Only replace banana if it is eaten"
    if is_eaten(art):
        set_banana(“UNRIPE”, art)

def eat_banana(art):
    "Only eat banana if it is ripe"
    if is_ripe(art):
        set_banana(“EATEN”, art)
```
def generate_art(art):
    ""
    Yield the corresponding string based on the state of the banana. 
    Raise an Exception if the art is in an unknown state. 
    (Remember, you may not need all lines).
    ""
    while True:
        if is_eaten(art):
            yield "There is no banana"
        elif is_ripe(art):
            yield "The banana looks so delicious"
        elif is_unripe(art):
            yield "Eww unripe banana"
        else:
            raise StopIteration("Unknown banana state")
It's dessert time again! Fill in the Chocolate and Cake classes based on the doctests. Both these classes should inherit from our base Dessert class. Be sure to take advantage of inheritance whenever possible. You may not need all lines provided.

```python
>>> cake = Cake('cake', 2, 2, 'chocolate')
>>> cake.eat()
Oh no! There is no more cake
>>> cake.bake()
Adding chocolate
Baking...
>>> cake.eat()
Yum, I ate 2 pieces of cake
>>> cake.eat()
Oh no! There is no more cake
>>> chocolate = Chocolate('chocolate', 5, 3)
>>> chocolate.eat()
Yum, I ate 3 pieces of milk chocolate
>>> chocolate.eat()
Oh no! There is no more milk chocolate
>>> chocolate.happiness
False
```

class Dessert:
    def __init__(self, name, portions_remaining, portion_size):
        self.name = name
        self.happiness = True
        self.portions_remaining = portions_remaining
        self.portion_size = portion_size

    def eat(self):
        if self.portions_remaining >= self.portion_size:
            self.portions_remaining -= self.portion_size
            print("Yum, I ate " + str(self.portion_size) + " pieces of " + self.name)
        else:
            print("Oh no! There is no more " + self.name)
class Chocolate(Dessert):
    def __init__(self, name, portions_remaining, portion_size):
        super().__init__(name, portions_remaining, portion_size)

    def eat(self):
        if (self.portions_remaining < self.portion_size):
            self.happiness = False
        Dessert.eat(self)

class Cake(Dessert):
    def __init__(self, name, portions_remaining, portion_size, flavor):
        Dessert.__init__(self, name, 0, portion_size)
        self.portions_to_be_made = portions_remaining
        self.flavor = flavor

    def bake(self):
        print("Adding " + self.flavor)
        print("Baking ...")
        self.portions_remaining = self.portions_to_be_made