1. Draw box-and-pointer diagrams for the following:
   ```python
   >>> a = [1, 2, 3]
   >>> a
   Solution: [1, 2, 3]
   >>> a[2]
   Solution: 3
   >>> b = a
   >>> a = a + [4, 5]
   >>> a
   Solution: [1, 2, 3, 4, 5]
   ```
2. Draw the environment diagram that results from running the code.

```python
def reverse(lst):
    if len(lst) <= 1:
        return lst
    return reverse(lst[1:]) + [lst[0]]
```

```python
lst = [1, [2, 3], 4]
rev = reverse(lst)
```

**Solution:** https://goo.gl/6vPeX9
3. Write a function that takes in a list `nums` and returns a new list with only the primes from `nums`. Assume that `is_prime(n)` is defined. You may use a `while` loop, a `for` loop, or a list comprehension.

```python
def all_primes(nums):
    result = []
    for i in nums:
        if is_prime(i):
            result = result + [i]
    return result

List comprehension:
return [x for x in nums if is_prime(x)]
```
4. The following is an Abstract Data Type (ADT) for elephants. Each elephant keeps track of its name, age, and whether or not it can fly. Given our provided constructor, fill out the selectors:

```python
def elephant(name, age, can_fly):
    ""
    Takes in a string name, an int age, and a boolean can_fly. Constructs an elephant with these attributes.
    >>> dumbo = elephant("Dumbo", 10, True)
    >>> elephant_name(dumbo)
    "Dumbo"
    >>> elephant_age(dumbo)
    10
    >>> elephant_can_fly(dumbo)
    True
    ""
    return [name, age, can_fly]
def elephant_name(e):
    Solution:
    return e[0]
def elephant_age(e):
    Solution:
    return e[1]
def elephant_can_fly(e):
    Solution:
    return e[2]
```
5. This function returns the correct result, but there’s something wrong about its implementation. How do we fix it?

```python
def elephant_roster(elephants):
    """
    Takes in a list of elephants and returns a list of their names.
    """
    return [elephant[0] for elephant in elephants]
```

**Solution:**
elephant[0] is a Data Abstraction Violation (DAV). We should use a selector instead.

6. Fill out the following constructor for the given selectors.

```python
def elephant(name, age, can_fly):
    return [[name, age], can_fly]

def elephant_name(e):
    return e[0][0]
def elephant_age(e):
    return e[0][1]
def elephant_can_fly(e):
    return e[1]
```

7. How can we write the fixed `elephant_roster` function for the constructors and selectors in the previous question?

**Solution:** No change is necessary to fix `elephant_roster` since using the `elephant` selectors “protects” the roster from constructor definition changes.
8. (Optional) Fill out the following constructor for the given selectors.

```python
def elephant(name, age, can_fly):
    """
    >>> chris = elephant("Chris Martin", 38, False)
    >>> elephant_name(chris)
        "Chris Martin"
    >>> elephant_age(chris)
        38
    >>> elephant_can_fly(chris)
        False
    """
    def select(command)
        if command == "name":
            return name
        elif command == "age":
            return age
        elif command == "can_fly":
            return can_fly
        return "Breaking abstraction barrier!"

    return select

def elephant_name(e):
    return e("name")
def elephant_age(e):
    return e("age")
def elephant_can_fly(e):
    return e("can_fly")
```