Assignment

Create @size decorator that is used the way as shown below.

The decorator converts a function to a descriptor. If executed on an instance, the descriptor returns the value of the function. If called on a class, the descriptor returns a sum of the original functions over all instances of the class. To retrieve all instances the descriptor uses the function provided as a parameter to the decorator.

```
class Cache:
    def __init__(self):
        Cache._all_caches.append(self)
        self._storage = dict()

    _all_caches = []

    def set(self, key, value):
        self._storage[key] = value

    def get(self, key):
        self._storage[key]

    def _all_instances():
        return Cache._all_caches

    @size(all_instances = _all_instances)
    def entries_count(self):
        return len(self._storage)
```

```
a = Cache()
b = Cache()

a.set("a", 1)
a.set("b", 2)
b.set("c", 3)

print(a.entries_count)      # prints 2
print(b.entries_count)      # prints 1
print(Cache.entries_count)  # prints 3
```

Assignment

Create class Struct which is used as shown below. The attributes listed in the class are the only ones that are permitted in the struct. The values defined in the class declaration are defaults to be used if the value of a respective attribute is not specified when creating an instance of the struct.

```
class Point(Struct):
  x = 3.5
  y = 4.5


p = Point(y = 8)
print(p) # prints Point(y=8) -- value of x is not shown because it is the default one
p.y = 2
p.x = 3
print(p) # prints Point(x=3, y=2) -- both values are shown because neither is the same as the default

# This causes the following error ... AttributeError: 'Point' object has no attribute 'z'
# p.z = 3
```

Hints:

- Class Struct has a custom meta-class that takes care of correct interpretation of the class content and saving the default values to the newly created class
- The meta-class provides __init__ and __repr__ methods for the created class
- The created class uses __slots__ instead of __dict__ in order to specify which attributes are permitted. (Alternatively, the metaclass may provide __getattribute__ and __setattr__ to the newly created class.)
- You may need built-in methods setattr and getattr to read/set attributes of an instance

Department of Distributed and Dependable Systems