Containers

March 13, 2021

0.1 Iterables

- objects are iterables
- if you can use it in a for loop it is an iterable
- in order be an iterable, I have to be able to obtain an iterator
- iterators are a one time run through unlike lists

[1]:
```
x = list(range(5))
print('x')
print(x)
print("")
print("n")
for n in x:
    print(n)
```

```
x
[0, 1, 2, 3, 4]

n
0
1
2
3
4
```

[13]:
```
class Bacteria:
    def __init__(self, size, color, abilities=[]):
        self.size = size
        self.color = color
        self.abilities = abilities
        self.index = -1

    # send back an iterator if an object has a dunder next
    def __iter__(self):
        return self
        # return (a for a in self.abilities)

    def __len__(self):
        return len(self.abilities)
```

```python
    def __next__(self):
        self.index += 1
        if self.index >= len(self.abilities):
            self.index = -1
            raise StopIteration()
        return self.abilities[self.index]

    def __reversed__(self):
        for i in range(len(self) - 1, -1, -1):
            yield self.abilities[i]

b1 = Bacteria(5, 'red', ['speed', 'stealth'])

for a in b1:
    print(a)

print("")
print(next(b1))
print(next(b1))
```

```
speed
stealth

speed
stealth
```

0.2 Getting, Setrting and Deleting Items

[27]:
```python
class Inventory:
    def __init__(self):
        self._products = ['leds', 'batteries', 'solder']
        self._prices = [1.00, 3.000, 5.00]

    def __repr__(self):
        product_price_pairs = ('{}:${:2f}'.format(*pair)
                               for pair in zip(self._products, self._prices))
        listing = '\n'.join(product_price_pairs)
        return '<Inventory>\n{listing}\n</Inventory>'.format(listing=listing)

    def __setitem__(self, key, value):
        if key in self:
            self._prices[self._products.index(key)] = value
        else:
            self._products.append(key)
            self._prices.append(value)
    def __getitem__(self, key):
        for index, product in enumerate(self._products):
            if product == key:
                return self._prices[index]
        return self.__missing__(key)

    def __contains__(self, key):
        return key in self._products

    def __missing__(self, notfoundKey):
        return "We are currently out of stock"

    def __delitem__(self, key):
        if key in self:
            index = self._products.index(key)
            del self._products[index]
            del self._prices[index]
        else:
            raise KeyError('That product does not exits')

inventory = Inventory()
print(inventory)
print("")
print("inventory['solder']")
print(inventory['solder'])
print("")
inventory['leds'] = 7.00
print("inventory['leds']")
print(inventory['leds'])
```

```
<Inventory>
leds:$1.000000
batteries:$3.000000
solder:$5.000000
</Inventory>

inventory['solder']
5.0

inventory['leds']
7.0
```