Audit Access Expires 5 de ago de 2020
You lose all access to this course, including your progress, on 5 de ago de 2020.

Exercise: int set

Finger Exercises due Aug 5, 2020 20:30 -03 *Completo*

Exercise: int set

5/5 points (graded)
ESTIMATED TIME TO COMPLETE: 10 minutes

Consider the following code from the last lecture video:

```
class intSet(object):
    """An intSet is a set of integers
    The value is represented by a list of ints, self.vals.
    Each int in the set occurs in self.vals exactly once."""

    def __init__(self):
        """Create an empty set of integers"""
        self.vals = []

    def insert(self, e):
        """Assumes e is an integer and inserts e into self"""
        if not e in self.vals:
            self.vals.append(e)

    def member(self, e):
        """Assumes e is an integer
           Returns True if e is in self, and False otherwise"""
        return e in self.vals

    def remove(self, e):
        """Assumes e is an integer and removes e from self
           Raises ValueError if e is not in self"""
        try:
            self.vals.remove(e)
        except:
            raise ValueError(str(e) + ' not found')

    def __str__(self):
        """Returns a string representation of self"""
        self.vals.sort()
        return '{' + ','.join([str(e) for e in self.vals]) + '}'
```

Your task is to define the following two methods for the `intSet` class:

1. Define an `intersect` method that returns a new `intSet` containing elements that appear in both sets. In other words,

   ```
   s1.intersect(s2)
   ```

   would return a new `intSet` of integers that appear in both `s1` and `s2`. Think carefully - what should happen if `s1` and `s2` have no elements in common?

2. Add the appropriate method(s) so that `len(s)` returns the number of elements in `s`.

   Hint: look through the Python docs to figure out what you'll need to solve this problem.

```
class intSet(object):
    """An intSet is a set of integers
    The value is represented by a list of ints, self.vals.
    Each int in the set occurs in self.vals exactly once."""

    def __init__(self):
        """Create an empty set of integers"""
        self.vals = []

    def insert(self, e):
        """Assumes e is an integer and inserts e into self"""
        if not e in self.vals:
            self.vals.append(e)

    def member(self, e):
```

Press ESC then TAB or click outside of the code editor to exit

Correta

```
class intSet(object):
    """An intSet is a set of integers
    The value is represented by a list of ints, self.vals.
    Each int in the set occurs in self.vals exactly once."""

    def __init__(self):
        """Create an empty set of integers"""
        self.vals = []

    def insert(self, e):
        """Assumes e is an integer and inserts e into self"""
        if not e in self.vals:
            self.vals.append(e)

    def member(self, e):
        """Assumes e is an integer
           Returns True if e is in self, and False otherwise"""
        return e in self.vals

    def remove(self, e):
        """Assumes e is an integer and removes e from self
           Raises ValueError if e is not in self"""
        try:
            self.vals.remove(e)
        except:
            raise ValueError(str(e) + ' not found')

    def intersect(self, other):
        """Assumes other is an intSet
           Returns a new intSet containing elements that appear in both sets."""
        # Initialize a new intSet
        commonValueSet = intSet()
        # Go through the values in this set
        for val in self.vals:
            # Check if each value is a member of the other set
            if other.member(val):
                commonValueSet.insert(val)
        return commonValueSet

    def __str__(self):
        """Returns a string representation of self"""
        self.vals.sort()
        return '{' + ','.join([str(e) for e in self.vals]) + '}'

    def __len__(self):
        """Returns the length of the set.
           This method is called by the `len` built-in function."""
        return len(self.vals)
```

Test results

Hide output

CORRECT

Test: intersect 1

Output:

```
setA: {-19,-12,-11,-7,-6,-1,0,12,13,18}
setB: {-12,-11,-7,-5,-4,-3,11,19}
setA.intersect(setB): {-12,-11,-7}
```

Test: intersect 2

Output:

```
setA: {-19,-18,-3,1,2,3,11,13,19}
setB: {-19,-15,-11,-6,10,11,14,18}
setA.intersect(setB): {-19,11}
setB.intersect(setA): {-19,11}
```

Test: intersect 3

Output:

```
setA: {-18,-12,-4,-2,0,1,2,3,17}
setB: {-20,-16,-14,-8,-5,-3,6,11,16}
setA.intersect(setB): {}
```

Test: intersect 4

Output:

```
setA: {}
setB: {}
setA.intersect(setB): {}
```

Test: len 1

Output:

```
setA: {-9,1,4,9,14,15}
len(setA): 6
```

Test: len 2

Output:

```
setA: {}
len(setA): 0
```

Test: len 3

Output:

```
setA: {0,7}
len(setA): 2
```

Test: len 4

Output:

```
setA: {-18,-15,-2,3,8,13}
len(setA): 6
```

Hide output

Enviar

Answers are displayed within the problem

Exercise: int set

Topic: Lecture 9 / Exercise: int set

Ocultar discussão

Add a Post

Show all posts por atividade recente

? Connection between __str__ and print?
I understand that __str__ creates a string. But how is it that 'print' calls on the __str__ method? We've ... 3