CS152: Data Structure

Week 1.2 Knowledge Check

Write the full name of all collaborators inside this box

Discuss these qustions and write your answers in the space provided below it.

1. How can you tell if a `def` statement in Python defines a function or class method?

2. What are constructors used for? What special name does Python use for them?

Discuss these questions and tick the option you think is correct

3. The `==` operation for two lists must:
   (a) Compare pairs of items at each position for equality
   (b) Merely verify that each item in one list is also in the other list

4. The `==` operation for two sets must:
   (a) Compare pairs of items at each position for equality
   (b) Verify that the sets are of the same size and that each item in one set is also in the other set

Short pair programming problem:

Statisticians would like to have a set of functions to compute the median and mode of a list of numbers. The median is the number that would appear at the midpoint of a list if it were sorted. The mode is the number that appears most frequently in the list. Define these functions in a module named `stats.py`. Also include a function named `mean`, which computes the average of a set of numbers. Each function expects a list of numbers as an argument and returns a single number.

A version of the solution

```
"""
File: stats.py

Defines functions to compute the mean, median, and mode
of a list of numbers.
"""

def mean(lyst):
    """Returns the mean of a list of numbers."""
    sum = 0
    for number in lyst:
        sum += number
    if len(lyst) == 0:
        return 0
    else:
        return sum / len(lyst)

def mode(lyst):
    """Returns the mode of a list of numbers."""
    # Obtain the set of unique numbers and their
    # frequencies, saving these associations in
    # a dictionary
    theDictionary = {}
    for number in lyst:
        freq = theDictionary.get(number, None)
        if freq == None:
            # number entered for the first time
            theDictionary[number] = 1
        else:
            # number already seen, increment its freq
            theDictionary[number] = freq + 1

    # Find the mode by obtaining the maximum freq
    # in the dictionary and determining its key
    if len(theDictionary) == 0:
        return 0
    else:
        theMaximum = max(theDictionary.values())
        for key in theDictionary:
            if theDictionary[key] == theMaximum:
                return key

def median(lyst):
    """Returns the median of a list of numbers."""
    # Create a copy of lyst before sorting
    numbers = []
    for number in lyst:
        numbers.append(number)
    # Sort the list and return the number at its midpoint
    numbers.sort()
    if len(numbers) == 0:
        return 0
```

```python
    else:
        midpoint = len(numbers) // 2
        if len(numbers) % 2 == 1:
            return numbers[midpoint]
        else:
            return (numbers[midpoint] + numbers[midpoint - 1]) / 2

def main():
    """Tests the functions."""
    lyst = [3, 1, 7, 1, 4, 10]
    print("List:", lyst)
    print("Mode:", mode(lyst))
    print("Median:", median(lyst))
    print("Mean:", mean(lyst))

if __name__ == "__main__":
    main()
```