Instruction

Code for Bogosort

Python

You'll need a `load.py` file saved in the same directory as the `bogosort.py` file, with the following contents:

```
# You call this function with the path to a file you want to load.
# It loads the file contents, converts each line from a string to
# an integer, and returns them all as a Python list.
def load_numbers(file_name):
  numbers = []
  with open(file_name) as f:
    for line in f:
      numbers.append(int(line))
  return numbers
```

Here's the code for `bogosort.py` itself:

```
# The function that randomizes the order of the list is kept in the
# "random" module, so we need to import that here.
import random
# The sys.argv list gives us the command line arguments to the
# script. To use it, we also need to import the "sys" module.
import sys
# Here's where we import the load_numbers function from above.
from load import load_numbers

# And here, we pass the first command line argument (which should be
# the path to a file) to load_numbers, and store the returned list of
# numbers in a variable.
numbers = load_numbers(sys.argv[1])

# Bogosort just randomly rearranges the list of values over and over,
# so the first thing it's going to need is a function to detect when
```

```
# the list is sorted. We'll write an is_sorted function that takes a
# list of values as a parameter. It will return True if the list
# passed in is sorted, or False if it isn't.
def is_sorted(values):
  # We'll loop through the numeric index of each value in the list,
  # from 0 to one less than the length of the list. Like many
  # languages, Python list indexes begin at 0, so a list with a
  # length of 5 has indexes going from 0 through 4.
  for index in range(len(values) - 1):
    # If the list is sorted, then every value in it will be less than
    # the one that comes after it. So we test to see whether the
    # current item is GREATER than the one that follows it.
    if values[index] > values[index + 1]:
      # If it is, it means the whole list is not sorted, so we return
      # False.
      return False
  # If we get down here, it means the loop completed without finding
  # any unsorted values. (Python uses whitespace to mark code blocks,
  # so un-indenting the code like this marks the end of the loop.)
  # Since all the values are sorted, we can return True.
  return True

# Now we need to write the function that will actually do the
# so-called sorting.  The bogo_sort function will also take the list
# of values it's working with as a parameter.
def bogo_sort(values):
  # We'll call our is_sorted function to test whether the list is
  # sorted. We'll keep looping until is_sorted returns True.
  while not is_sorted(values):
    # Python has a ready-made function that randomizes the order of
    # elements in a list. Since the list isn't sorted, we'll call
    # that function here. And since this is inside the loop, it will
    # be randomized over and over until our is_sorted function
    # returns True.
    random.shuffle(values)
  # If the loop exits, it means is_sorted returned True, and the list
  # is sorted.  So we can now return the sorted list.
  return values

# Finally, we need to call our bogo_sort function, pass it the list
```

```
# we loaded from the file, and print the sorted list it returns.
print(bogo_sort(numbers))
```

To run this, save the above code to two files in the same directory, and ensure a file named `5.txt` is saved in a subdirectory named `numbers`. Then open a terminal/console, change to the directory where `bogosort.py` is saved, and run this command:

```
python bogosort.py numbers/5.txt
```