data_loader_two_by_two.py

```
import numpy as np

def get_data_set():
    """this creates a list of size n_examples of np arrays, each of size 2x2
    Generators (functions that use yield instead of return) are useful because they only get the subset of data that you want as opposed to loading the whole set to memory, and only give you the subset once, when called

    To use this, do import data_loader_two_by_two as dat; train_generator, evaluation_generator = get_data_set(); new_train_example=next(train_generator())
    """
    examples = []
    n_examples=1000
    for i in range(n_examples):
        array_2d = np.random.rand(2,2)#sample unifrm data in range (0,1) of size 2x2
        examples.append(array_2d)
        #its generating a 2x2 array for each example, but when training, this will be flattened as a 1d aray for each example

    def training_set():
        #start with an infinite loop, so that you can keep calling next (i.e. set = train_set(); set.next() ) until you run out of training examples
        while True:
            index = np.random.choice(len(examples))#index of one of the items in our examples
            yield examples[index]

    def evaluation_set():
        #start with an infinite loop, so that you can keep calling next (i.e. set = train_set(); set.next() ) until you run out of training examples
        while True:
            index = np.random.choice(len(examples))#index of one of the items in our examples
            yield examples[index]

    return training_set, evaluation_set
```