gist

Jalanorian / gist:b9abc57e54bdb70d1476
Created just now

gistfile1.py Python (/Jalanorian/b9abc57e54bdb70d1476/raw/454843c1e8d5d7ad436c8d177a6f4938fd940b5f/gistfile1.py)

```
1  from random import randint
2  class DiceDict(object):
3    """
4          List to simulate dictionary. Initializes as empty, and is formatted
5          with L[0] = key, L[1] = value.
6          """
7          def __init__(self):
8                  """
9                  Dict is initialized with no components. Format:
10                 List to simulate the dictionary.
11                 1st column: "key"
12                 2nd column: "value"
13                 -1st column in 2nd column: number of dice of that type
14                 -2nd column in 2nd column: Die object
15
16                 Args:
17                         Nothing
18                 """
19                 self.List = []
20
21         def add_or_increment_die(self, dieNumSides):
22                 """
23                 Takes one int and, if the die corresponding to that number is present in the list,
24                 increments the number of those dice, otherwise adds an instance of that die to the poke
25                 and sets the quantity to 1 - all internally.
26                 Args:
27                         dieNumSides - int for number of sides on that die
28                 Returns:
29                         Nothing
30                 """
31                 if len(self.List) > 0:
32                         for i in range(len(self.List)):
33                         if (self.List[i])[0] == dieNumSides:
34                                 self.List[i][1][0] += 1
35                                 break
36                         else:
37                                 key = dieNumSides
38                                 value = [1, Die(dieNumSides)]
39                                 self.List.append([key, value])
40                 else:
41                         key = dieNumSides
42                         value = [1, Die(dieNumSides)]
43                         self.List.append([key, value])
44
45         def values(self):
46                 """
47                 Emulates the .values() function of the built-in Python dictionary.
48                 Args:
49                         Nothing
50                 Returns:
51                         A list of the contents of the second column of the "dictionary" list.
52                 """
53                 return [sublist[1] for sublist in self.List]
54
55
56 class Poke(object):
57         """
58         An abstract bag that "holds" dice. Implements DiceDict to store information
59         and instances of Die class.
60         """
61
62         def __init__(self):
63                 """
64                 Initializes empty with a new DiceDict and a variable to keep track of
65                 how many dice it holds (used later for quicker randomized dice
```

```
                how many dice it holds (used later for quicker randomized dice
                selection).

                Args:
                        Nothing
                """
                self.diceDict = DiceDict()
                self.totalNumDice = 0

        def add_die(self, numDieSides):
                """
                Called with one argument, the side of the die to be added, it will then
                call the DiceDict to add it to the list and also increment its running
                total of the currently held number of dice. Exception catching is
                performed at the level of the poke - only integers in the allowed
                list are passed on to the Die method.
                Args:
                        numDieSides - int for number of sides of die to be added
                Returns:
                        Nothing
                """
                allowedDice = [4, 6, 8, 12, 20]
                if numDieSides in allowedDice:
                        self.diceDict.add_or_increment_die(numDieSides)
                        self.totalNumDice += 1
                else:
                        print "Invalid number of sides for a die in this poke."

        def pick_die(self):
                """
                Randomly returns one Die instance from the poke implementing randint from random module.
                Args:
                        Nothing
                Returns:
                        Randomly selected die from poke's contents
                """
                r = randint(1,self.totalNumDice)
                for eachDieType in self.diceDict.values():
                        if r > eachDieType[0]:
                                r -= eachDieType[0]
                        else:
                                return eachDieType[1]

        def sample_poke(self):
                """
                Calls pick_die to randomly select a die from the poke and then calls
                roll() on that Die instance to obtain the numerically rolled value,
                returning that to the caller.
                Args:
                        Nothing
                Returns:
                        Int for the value of the rolled random die
                """
                return self.pick_die().roll()

        def print_poke(self):
                """
                Prints the contents of the poke to the console, omitting any slots of
                the poke that have no dice and therefore haven't been initialized in
                the DiceDict yet.
                Args:
                        Nothing
                Returns:
                        Nothing
                """
                for dieType, value in sorted(self.diceDict.List):
                        print str(value[0]), str(dieType) + "-sided dice"


class Die(object):
        """
        A simple class to represent a die, upon which can be called only the roll
        method. Uses instance variable for sides and randint.
        """
        def __init__(self, sides):
                """
                Initializes with a simple internal value of the number of sides of that
                instance of the Die class.
```

```
143                 Args:
144                         sides - number of sides for the new Die object
145                 """
146                 self.sides = sides
147
148         def roll(self):
149                 """
150                 Returns a random integer value based on the number of sides on that die.
151                 Args:
152                         Nothing
153                 Returns:
154                         randomly selected integer in [1, self.sides] range
155                 """
156                 return randint(1,self.sides)
157
158 def main():
159         # Initalize the first (empty) poke
160         poke = Poke()
161
162         # A list of the possible dice being used in this particular test.
163         possDice = [4,6,8,12,20]
164
165         # A testing of the Exploration case of 2 tetra-, 0 hexa-, 3 octa-, 1 dodeca-,
166         # and 4 icosa-hedrons. Also testing the catching of unintended inputs.
167         # Using the list of possible dice to make assignment easier.
168         poke.add_die(possDice[0])
169         poke.add_die(possDice[0])
170         poke.add_die(possDice[2])
171         poke.add_die(possDice[2])
172         poke.add_die(possDice[2])
173         poke.add_die(possDice[3])
174         poke.add_die(possDice[4])
175         poke.add_die(possDice[4])
176         poke.add_die(possDice[4])
177         poke.add_die(possDice[4])
178         poke.add_die("fas")
179         poke.add_die(17)
180
181         # Testing of the print_poke() poke method and double-checking the
182         # implementation of add_die()
183         poke.print_poke()
184         # Variables for Observed Expected Values
185         total = 0.0
186         testSize = 1
187         # Iterating through for the three (3) cases for Observed Expected Values
188         # and printing the results to the console.
189         for i in range(3):
190                 total = 0.0
191                 testSize *= 100
192                 for i in range(testSize):
193                         total += poke.sample_poke()
194
195                 print str(testSize) + " Total: " + str(total)
196                 print "Average: " + str(total / testSize)
197
198
199         # Testing again with another poke (POKE)
200         POKE = Poke()
201
202         for i in range(1000):
203                 r = randint(0,4)
204                 POKE.add_die(possDice[r])
205
206         POKE.print_poke()
207
208 if __name__ == '__main__':
209         main()
```