# Programming, Probability, and the Modern Mathematics Classroom Exercises — Part 5

Manan Shah
Mathematician-At-Large

June 6, 2013

Please make sure to have read the blog post with the same topic name on the Math Misery website, otherwise this will be out of context.

## Some reminders

- Have students to comment the code, line by line.
- Let students tweak, modify, and otherwise alter the code.
- Don't worry about programming formalities.
- Have fun, relax, and learn.

## The Python dictionary object

```
>>> ##more about Python sets, dictionaries, and lists
>>> ##both sets and dictionaries require "hashable" objects
>>> d = {}
>>> type(d)
<class 'dict'>
>>> s = set()
>>> type(s)
<class 'set'>
>>> ##we can't add lists as keys to dictionaries or as elements to sets
>>> d[[1,2,3]] = "a list"
Traceback (most recent call last):
  File "<pyshell#9>", line 1, in <module>
    d[[1,2,3]] = "a list"
TypeError: unhashable type: 'list'
>>> s.add([1,2,3])
Traceback (most recent call last):
  File "<pyshell#10>", line 1, in <module>
    s.add([1,2,3])
TypeError: unhashable type: 'list'
>>> ##however, tuples can be used instead
>>> d[(1,2,3)] = "a tuple"
>>> d
{(1, 2, 3): 'a tuple'}
>>> s.add((1,2,3))
>>> s
{(1, 2, 3)}
>>> ##there are some technical nuances as to what objects can be used as keys
>>> ##for a dictionary, and what objects can be used as elements of a set
>>> ##the interested reader should reference Python's Glossary on the
>>> ##definition of "hashable" vs "immutable"
>>> ##there are ways to make lists hashable and have them be dictionary keys
>>> ##but that's a bit out of scope here.
>>> ##still, students should be made aware of this nuance.
>>> ##here is a quick and dirty example of what happens
>>> mystr = "hello"
>>> mystr.__hash__()
-1267296259
>>> mystr += " there"
>>> mystr
'hello there'
>>> mystr.__hash__()
-1368785827
>>> s = set()
>>> s.add(mystr)
>>> mystr += " this change again"
>>> mystr
'hello there this change again'
>>> s
{'hello there'}
>>> for element in s:
        print("'" + str(element) + "' has a hash of " + str(element.__hash__()))


'hello there' has a hash of -1368785827
>>> mystr.__hash__()
-1827124077
>>> mylist = [1,2,3]
>>> mylist.__hash__()
Traceback (most recent call last):
  File "<pyshell#160>", line 1, in <module>
    mylist.__hash__()
TypeError: 'NoneType' object is not callable
>>> ##notice that the list object does not have a hash method
>>> ##now we will take a small diversion and go back to strings
>>> ##and then tie [pun!] them back to together with sets
>>> sentence1 = "this is a sentence"
>>> sentence2 = "this is another sentence"
>>> ##if we want to get the words from each sentence, we may via 'split'
>>> words1 = sentence1.split()
>>> words1
['this', 'is', 'a', 'sentence']
>>> words2 = sentence2.split()
>>> words2
['this', 'is', 'another', 'sentence']
>>> ##let's say that we wanted to keep track of what words were used
>>> ##in each sentence. for this we could use a set.
>>> set1 = set()
>>> set2 = set()
>>> for word in words1:
        set1.add(word)


>>> for word in words2:
        set2.add(word)


>>> set1
{'this', 'a', 'is', 'sentence'}
>>> set2
{'this', 'is', 'another', 'sentence'}
>>> ##what if we wanted to know what these two sets had in common?
>>> ##this is called an intersection in set theory and in Python!
>>> set1.intersection(set2)
{'this', 'is', 'sentence'}
>>> set2.intersection(set1)
{'this', 'is', 'sentence'}
>>> set1
{'this', 'a', 'is', 'sentence'}
>>> set2
{'this', 'is', 'another', 'sentence'}
>>> ##what about a set difference?
>>> set1.difference(set2)
{'a'}
>>> set2.difference(set1)
{'another'}
>>> ##so notice that for set difference the order matters
>>> ##while for an intersection it didn't matter
>>> ##now what if we wanted to know what a "universe" set would look like?
>>> ##for this we use a union
>>> set1.union(set2)
{'a', 'sentence', 'this', 'is', 'another'}
>>> set2.union(set1)
{'a', 'sentence', 'this', 'is', 'another'}
>>> ##now let's get a bit more nuanced and try to keep track of each letter
>>> ##in the sentence
>>> sentence1
'this is a sentence'
>>> sentence2
'this is another sentence'
>>> ##there are many ways to do this, and the instructor should let students
>>> ##figure this out
>>> ##here is one way
>>> letterset = set()
>>> letters = list("abcdefghijklmnopqrstuvwxyz")
>>> for letter in letters:
        letterset.add(letter)


>>> letters1 = set()
>>> letters2 = set()
>>> for character in sentence1:
        if character in letterset:
                letters1.add(character)


>>> letters1
{'a', 'c', 'e', 'i', 'h', 'n', 's', 't'}
>>> for character in sentence2:
        if character in letterset:
                letters2.add(character)


>>> letters2
{'a', 'c', 'e', 'i', 'h', 'o', 'n', 's', 'r', 't'}
>>> ##notice how we ignored spaces and only captured letters
>>> ##CAUTION: letterset only had lowercase letters in it!
>>> letters1.intersection(letters2)
{'a', 'c', 'e', 'i', 'h', 'n', 's', 't'}
>>> letters2.intersection(letters1)
{'a', 'c', 'e', 'i', 'h', 'n', 's', 't'}
>>> letters1.difference(letters2)
set()
>>> ##the set is empty! what does that mean in terms of the set difference?
>>> letters2.difference(letters1)
{'r', 'o'}
>>> ##this set is non-empty! why?
>>> letters1.union(letters2)
{'a', 'c', 'e', 'i', 'h', 'o', 'n', 's', 'r', 't'}
>>> ##here's something a bit more fancy
>>> (letters1.union(letters2)).difference(letters1)
{'r', 'o'}
>>> letterset.difference(letters1.union(letters2))
{'b', 'd', 'g', 'f', 'k', 'j', 'm', 'l', 'q', 'p', 'u', 'w', 'v', 'y', 'x', 'z'}
```

## Summary

In this part, we discussed Python sets in more detail. We went over some set operations, like `intersection`, `difference`, and `union`. We also did a little bit of string-handling. We gave a very light introduction to the Python `__hash__` method as well.

While, there wasn't really a discussion of probability problems, we did give an introduction to some concepts from set theory. Instructors are encouraged to go over this with students by tieing in the typical Venn diagrams and writing small programs to do some of the Venn diagram like calculations.

If you need help, have questions, or would like to set up a workshop at your school get in touch with me at help@mathmisery.com.