Week 13

# All about Python SETS (a built-in Python container type)

''' A set stores unordered collections of items; no duplicates allowe. Items are immutable objects.

Which operations can we do on sets?

membership, intersection, union, symmetric difference ... etc.

...

def wait():
    x = input()

def main():

    living = {"cats", "dogs", "people", "bugs", "birds", "dolphins", "sharks", "rabbits", "mice"}

    # Note that when you type "living" (in the shell) or print "living" in the program, you will see this set, BUT Python DOES ITS OWN ORDERING OF ITEMS!
    
    wait()

    print(living)  # if you do this in the shell you do not have to call "print"

    wait()

    print(type(living))  # should say <class 'set'>

    wait()

    # duplicates are ignored

    pets = {"cats", "dogs", "cats", "giraffes", "fleas", "fleas"}

    wait()

    print(pets)

    wait()
#Question: What is a good way to remove duplicate items in a list?

#Answer: Use the SET constructor as follows.

# Suppose Mrs. Smith has 7 children:

# --- 1 set of twins, age 2
# --- 1 set of triplets, age 5
# --- 1 boy, age 7
# --- 1 girl, age 10

# list of children ages

ages = [10, 7, 2, 2, 3, 3, 3]  # note the order --->

print(ages)

#now do this to remove duplicates

ages = list(set(ages))  #we make a set from the list, 
# and make that set

print(ages)  #Python makes a set

#Question: How can we create an empty set?

living = {}  # but doesn't this look like an empty
dictionary?!!!

print(type(living))  # Yes! We created an empty dictionary, not a set!

# A Problem! Curly braces are used for both sets *and* dictionaries. But a
dictionary has
# <key:value> pairs with colons, whereas items in a set have no colons.

# So we have a way to tell sets from dictionaries. But how do we create an
empty set?
# Answer: Make EXPLICIT use of the SET constructor
living = set ()  # use the set constructor
print(living)  # prints "set()" to show it's an empty set
wait()

print(type(living))  # should say it's of type 'set'

# Which operators can we use on sets? Here are some examples.

# First, make some set

living = {'cats', 'dogs', "people", "bugs", "birds", "dolphins", "sharks", "rabbits", "mice"}

print(living)

wait()

#Test for membership, get size of set

print("cats" in living)  # should says True
print("pencils" in living)  # should say False, because "pencils" is not in the set

print(len(living))  # tells how many items are in this set

wait()

# Comparison of sets

animals = {'pigs', "elephants", "lions", "tigers"}

print(living == animals)  # different sets, so False
print(living != animals)  # not the same, so True

wait()

# If set A is a subset of set B, the A <= B

# If set A is a PROPER subset of B, then A < B (i.e., B has stuff that is in A and even more)
bigcats = \{"lions", "tigers"\}   # a proper subset of animals

print(bigcats < animals)       #should be true
print(animals <= animals)      #should be true
print(animals < bigcats)       #should be false

wait()

# Mathematical operations on sets:

# A union B is is "A | B"
# A intersection B is "A & B"
# A - B is "A - B"
# (A-B) union (B-A) is "A ^ B"

print(animals | bigcats)       # bigcats is subset, so union
is still set "animals"

print(animals - bigcats)       # remove lions and tigers from
set animals

print(animals & bigcats)       # intersection is set with
lions and tigers

print(animals ^ bigcats)       # stuff in set A and in set B
but not in both
"elephants" and "pigs" because B is subset of A

#Besides operators, sets also have a number of methods

wait()

print(bigcats)

bigcats.add('panthers')        #add one item to set

print(bigcats)

bigcats.remove("lions")       #remove one item from set

print(bigcats)

bigcats.clear()                #empty out the set

print(bigcats)                 #should see an empty set
# Homework: write a simple function "sunion" that takes 3 sets and returns the union of all three

From: http://courses.cs.purdue.edu/ - Computer Science Courses

Permanent link: http://courses.cs.purdue.edu/cs17700:fall15:sets

Last update: 2016/04/05 09:56