3) see **Chapter3-ShortAnswer3.py** for the implementation.
Here is the running time for each of four requested operations:

operations	(a) the objects are stored in a Python list in the order that they are added	(b) the objects are stored in a Python list in order by name	(c) the objects are stored in a Python dictionary indexed by name
add(person)	Θ(1) we discussed that addition of a value to the Python's list is on average Θ(1) operation (even though sometimes we have to allocate new space and copy elements there)	Θ(log n) since the elements are ordered, we need to find a position to insert the new record, search can be done with *log n* time (recall binary search on sorted arrays).	Θ(1) almost all basic operations on dictionaries are Θ(1), since hash tables with hashing function are used.
remove(name)	Θ(n) operations of insertion and deletion are Θ(n) for Python's lists + we need to find a record with that name, and since elements are no sorted it is also Θ(n), which results is Θ(n)	Θ(n) first we will need to locate the element with the given name (Θ(log n) operation), then we will need to delete is (Θ(n) operation on Pyton's lists), hence the result is Θ(n)	Θ(1) using hashing function the record will be accessed in constant time, and deleted
lookup(name)	Θ(n) search operation on an unsorted array/list is Θ(n)	Θ(log n) search in a sorted array is Θ(log n) operation	Θ(1) using hashing function the record will be accessed in constant time
list_all	Θ(n log n) sorting unsorted list takes at least (*based on what we learned so far*) Θ(n log n) time	Θ(1) the elements are already sorted	Θ(n log n) copying elements into a list is Θ(n) + sorting is at least Θ(n log n), hence overall we have Θ(n log n)

About operations and their cost on Python's list see pages 99
About operations and their cost on dictionaries see pages 94-95