Programming Fundamentals Using Python

2018

Problem Set 7

Most recent updated: July 12, 2018

Objectives

1. Python's built-in data structure

Note: Solve the programming problems listed using your favorite text editor. Make sure you save your programs in files with suitably chosen names, **and try as much as possible to write your code with good style (see the style guide for python code)**. In each problem find out a way to test the correctness of your program. After writing each program, test it, debug it if the program is incorrect, correct it, and repeat this process until you have a fully working program. Show your working program to one of the cohort instructors.

Problems: Cohort sessions

1. Compare the similarities and differences in doing the following operations for List, Dictionary, Set, and Tuple.

   (a) accessing the value of an item

   (b) modifying the value of an item

   (c) adding an item into the collection

   (d) finding an item

   (e) traversing items one at a time

2. Design an experiment to find out the computation time of various operations of Python's built-in data structure. You will be assigned to one of the following tasks.

   (a) *List* Design an experiment to find the computation time of the following List operation:

   - indexing, e.g. `list_a[x]`
   - assignment, e.g. `list_a[x] = 4`
   - append, e.g. `list_a.append(4)`
   - get number of items, e.g. `len(list_a)`

   (b) *List* Design an experiment to find the computation time of the following List operation:

   - extend, e.g. `list_a.extend([4,5,7])`
   - insert, e.g. `list_a.insert(pos, 10)`
   - concatenation (+), e.g. `list_a = list_b + list_c`
   - finding item, e.g. `3 in list_a`

   (c) *Tuple* Design an experiment to find the computation time of the following Tuple operation:

   - indexing, e.g. `tuple_a[x]`
   - concatenation (+), e.g. `tuple_a = tuple_b + tuple_c`
   - finding item, e.g. `3 in tuple_a`
   - get number of items, e.g. `len(tuple_a)`

   (d) *Dictionary* Design an experiment to find the computation time of the following Dictionary operation:

   - getting value, e.g. `dict_a[key]`

- assignment, e.g. `dict_a[key] = value`
- concatenation (+), e.g. `dict_a = dict_b + dict_c`
- finding item, e.g. `3 in dict_a`
- get number of items, e.g. `len(dict_a)`

(e) *Set* Design an experiment to find the computation time of the following Set operation:

- indexing, e.g. `set_a[key]`
- concatenation (+), e.g. `set_a = set_b + set_c`
- finding item, e.g. `3 in set_a`
- get number of items, e.g. `len(set_a)`

(f) *Heapq* Design an experiment to find the computation time of the following Heapq (`https://docs.python.org/3.6/library/heapq.html`) operation:

- building heap, e.g. `heapq.heapify(list_a)`
- push, e.g. `heapq.heappush(heap,item)`
- pop, e.g. `heapq.heappop(heap)`
- get the smallest n items, e.g. `heapq.nsmallest(n, list_a)`

End of Problem Set 7.