Master of Engineering in Internetworking

INWK 6312

Programming for Internetworking Applications

Lab 3

Lists, Dictionaries, and Tuples

TABLE OF CONTENTS

CHAPTER 1 - PREREQUISITES

Ensure that you have completed the tasks mentioned in Lab 1 and 2.

CHAPTER 2 - INTRODUCTION

This lab is to introduces you to python's built-in data types for a collection of items. First, we will look at lists and the methods on lists objects. Then we look at dictionaries, python's implementation of a hash table and then finally, we learn about tuple an immutable sequence of elements.

CHAPTER 3 - OBJECTIVES OF THE LAB

At the end of this lab you will have learnt the following:

- Create and modify lists
- Perform operations on lists
- Understand map, filter, and reduce
- Create and modify dictionary
- Perform operations on dictionaries
- Create and use tuples
- Combine the 3 key data structures in python

CHAPTER 4 - LISTS

Introduction to lists

list: A sequence of values.
element: One of the values in a list (or other sequence), also called items.

index: An integer value that indicates an element in a list.

nested list: A list that is an element of another list.

list traversal: The sequential accessing of each element in a list.

mapping: A relationship in which each element of one set corresponds to an element of another set. For example, a list is a mapping from indices to elements.

accumulator: A variable used in a loop to add up or accumulate a result.

augmented assignment: A statement that updates the value of a variable using an operator like +=.

reduce: A processing pattern that traverses a sequence and accumulates the elements into a single result.

map: A processing pattern that traverses a sequence and performs an operation on each element.

filter: A processing pattern that traverses a list and selects the elements that satisfy some criterion.

object: Something a variable can refer to. An object has a type and a value.

equivalent: Having the same value.

identical: Being the same object (which implies equivalence).

reference: The association between a variable and its value.

aliasing: A circumstance where two or more variables refer to the same object.

delimiter: A character or string used to indicate where a string should be split.

!!!!!! STRINGS ARE IMMUTABLE & LIST ARE MUTABLE!!!!!
Common Methods List and Strings Share. They return new value (they don't change the sequence)

Operation	Result
x in s	True if an item of *s* is equal to *x*, else False
x not in s	False if an item of *s* is equal to *x*, else True
s + t	the concatenation of *s* and *t*
s * n, n * s	equivalent to adding *s* to itself *n* times
s[i]	*i*th item of *s*, origin 0
s[i:j]	slice of *s* from *i* to *j*
s[i:j:k]	slice of *s* from *i* to *j* with step *k*
len(s)	length of *s*
min(s)	smallest item of *s*
max(s)	largest item of *s*
s.index(x)	index of the first occurrence of *x* in *s*
s.count(x)	total number of occurrences of *x* in *s*

Mutable Sequence Type (List & ByteArray): (They modify the sequence)

Operation	Result
s[i] = x	item *i* of *s* is replaced by *x*
s[i:j] = t	slice of *s* from *i* to *j* is replaced by the contents of the iterable *t*
del s[i:j]	same as s[i:j] = []
s[i:j:k] = t	the elements of s[i:j:k] are replaced by those of *t*
del s[i:j:k]	removes the elements of s[i:j:k] from the list
s.append(x)	same as s[len(s):len(s)] = [x]
s.extend(t) or s += t	for the most part the same as s[len(s):len(s)] = t
s *= n	updates *s* with its contents repeated *n* times
s.count(x)	return number of *i*'s for which s[i] == x
s.index(x[, i[, j]])	return smallest *k* such that s[k] == x and i <= k < j
s.insert(i, x)	same as s[i:i] = [x]
s.pop([i])	same as x = s[i]; del s[i]; return x
s.remove(x)	same as del s[s.index(x)]
s.reverse()	reverses the items of *s* in place
s.sort([cmp[, key[, reverse]]])	sort the items of *s* in place

CHAPTER 5 - REDUCE, MAP AND FILTER

REDUCE:

A processing pattern that traverses a sequence and accumulates the elements into a single result.

```
def add_all(lst):
    total = 0 #Accumulator
    for num in lst:
        total += num
    return total
```

Task 1

*Write a function called **nested_sum** that takes **a nested list of integers** and adds up the elements from all the nested lists.*

Task 2:

reduce (*function*, *iterable*[, *initializer*])

Read documentation about the built-in "reduce" function. (In Python3, Reduce is imported from the "functools" module)

Using the built-in reduce function, write a program to find the sum of even numbers between 100 and 500

MAP:

A processing pattern that traverses a sequence and performs an operation on each element. Study the code snippet below:

```
def capitalize_all(t):
res = []
for s in t:
      res.append(s.capitalize())
return res
```

Built-in Function: Map -- *read documentation about it*

map(function, iterable, ...)

Apply function to every item of iterable and return a generator of the results. If additional iterable arguments are passed, function must take that many arguments and is applied to the items from all iterables in parallel.

Task 3

Using the built-in map function, write a program that takes in a list of letter in lowercase and returns another list of the letters in uppercase.

FILTER:

A processing pattern that traverses a list and selects the elements that satisfy some criterion. Study the code snippet below:

```
def only_upper(t):
    res = []
    for s in t:
        if s.isupper():
           res.append(s)
    return res
```

Built-in Function: filter -- *read documentation about it*

filter(function, iterable):

Constructs a list from those elements of iterables for which function returns true. iterables may be either a sequence, a container which supports iteration, or an iterator.

Task 4

Given a list of number, return a list of the numbers that are multiples of 5. (Use the Built-in filter functions)

Task 5

Two words are anagrams if you can rearrange the letters from one to spell the other. Write a function called is_anagram that takes two strings and returns True. if they are anagrams.

CHAPTER 6 - DICTIONARY

Introduction to Dictionaries

dictionary: A mapping from a set of keys to their corresponding values.
key-value pair: The representation of the mapping from a key to a value.
item: Another name for a key-value pair.
key: An object that appears in a dictionary as the first part of a key-value pair.
value: An object that appears in a dictionary as the second part of a key-value pair. This is more specific than our previous use of the word "value."
implementation: A way of performing a computation.
hashtable: The algorithm used to implement Python dictionaries.
hash function: A function used by a hashtable to compute the location for a key.
hashable: A type that has a hash function. Immutable types like integers, floats and strings are hashable; mutable types like lists and dictionaries are not.
lookup: A dictionary operation that takes a key and finds the corresponding value.
reverse lookup: A dictionary operation that takes a value and finds one or more keys that map to it.
singleton: A list (or other sequence) with a single element.

call graph: A diagram that shows every frame created during the execution of a program, with an arrow from each caller to each callee.
histogram: A set of counters.
memo: A computed value stored to avoid unnecessary future computation.
flag: A boolean variable used to indicate whether a condition is true.
declaration: A statement like global that tells the interpreter something about a variable.

Task 6

Write a function that reads the words in "words.txt" and stores them as keys in a dictionary. It doesn't matter what the values are. Then you can use the in operator as a fast way to check whether a string is in the dictionary.

Task 7

Dictionaries have a method called 'get' that takes a key and a default value. If the key appears in the dictionary, 'get' returns the corresponding value; otherwise it returns the default value. For example:

```
>>> h = dict(a=1)
>>> print(h)
 {'a': 1}
>>> h.get('a', 0)
 1
>>> h.get('b', 5)
0
>>> print(h)
 {'a':1, 'b':5}
```

Also, read the documentation of the dictionary method 'setdefault'. Test it in your python interpreter and study how it works. It is similar to the "get" method but behaves in a different way. Know the difference!

CHAPTER 7 - TUPLE

Important topics – Tuple

tuple: An immutable sequence of elements.

tuple assignment: An assignment with a sequence on the right side and a tuple of variables on the left. The right side is evaluated and then its elements are assigned to the variables on the left.

gather: The operation of assembling a variable-length argument tuple.

scatter: The operation of treating a sequence as a list of arguments.

DSU: Abbreviation of "decorate-sort-undecorate," a pattern that involves building a list of tuples, sorting, and extracting part of the result.

data structure: A collection of related values, often organized in lists, dictionaries, tuples, etc.

shape (of a data structure): A summary of the type, size and composition of a data structure

Task 8

Write a function called 'sumal'l that takes any number of arguments and returns their sum. Use the code snippet below as template:

```
def sumall(*args, **kwargs):
    #write your code here
```

Task 9

Write a function called 'most_frequent' that takes a string and prints the letters in decreasing order of frequency.

HOMEWORK

Q1

Write a program that reads a word list from a file (see Section 9.1) and prints all the sets of words that are anagrams.

Here is an example of what the output might look like:

['deltas', 'desalt', 'lasted', 'salted', 'slated', 'staled']

['retainers', 'ternaries']

['generating', 'greatening']

['resmelts', 'smelters', 'termless']

Hint: you might want to build a dictionary that maps from a set of letters to a list of words that can be spelled with those letters. The question is, how can you represent the set of letters in a way that can be used as a key?

1. *Modify the previous program so that it prints the largest set of anagrams first, followed by the second largest set, and so on.*

Q2

The (so-called) Birthday Paradox:

1. *Write a function called has_duplicates that takes a list and returns True if there is any element that appears more than once. It should not modify the original list.*
2. *If there are 23 students in your class, what are the chances that two of you have the same birthday? You can estimate this probability by generating random samples of 23 birthdays and checking for matches. Hint: you can generate random birthdays with the randint function in the random module.*

Q3

Write a function that takes a list of numbers and returns the cumulative sum; that is, a new list where the ith element is the sum of the first i+1 elements from the original list. For example, the cumulative sum of [1, 2, 3] is [1, 3, 6].