Lesson 19:

1. Quiz: Growth

sum\_list and mystery scale linearly

1. Quiz: Hash String

The number of string comparisons done to lookup a keyword that is not a key in the hash table may be less than the number needed to lookup a keyword that is a key in the hash table. We should expect the time to lookup most keywords in the hash table will decrease as we increase the number of buckets.

1. Quiz: Is Offered

def is\_offered(courses, course, hexamester):

return True if course in courses\_offered(courses, hexamester) else False

1. Quiz: When Offered

def when\_offered(courses, course):

hexalist = []

for mesters in courses.keys():

if course in courses\_offered(courses, mesters):

hexalist.append(mesters)

return hexalist

1. Quiz: Involved

def involved(courses, person):

descriptions = dict()

for hexamester in courses.keys():

class\_list = list()

for classes in courses[hexamester].keys():

if person in courses[hexamester][classes].values():

class\_list.append(classes)

if class\_list:

descriptions[hexamester] = class\_list

return descriptions

1. Quiz: Refactoring

def bucket\_find(bucket, key):

for entry in bucket:

if entry[0] == key:

return entry

return None

def hashtable\_update(htable, key, value):

entry = bucket\_find(hashtable\_get\_bucket(htable, key), key)

bucket = hashtable\_get\_bucket(htable, key)

if entry:

entry[1] = value

else:

bucket.append([key, value])

def hashtable\_lookup(htable, key):

entry = bucket\_find(hashtable\_get\_bucket(htable, key), key)

if entry[0] == key:

return entry[1]

return None

1. Quiz: Memoization

def cached\_execution(cache, proc, proc\_input):

if proc\_input not in cache:

cache[proc\_input] = proc(proc\_input)

return cache[proc\_input]