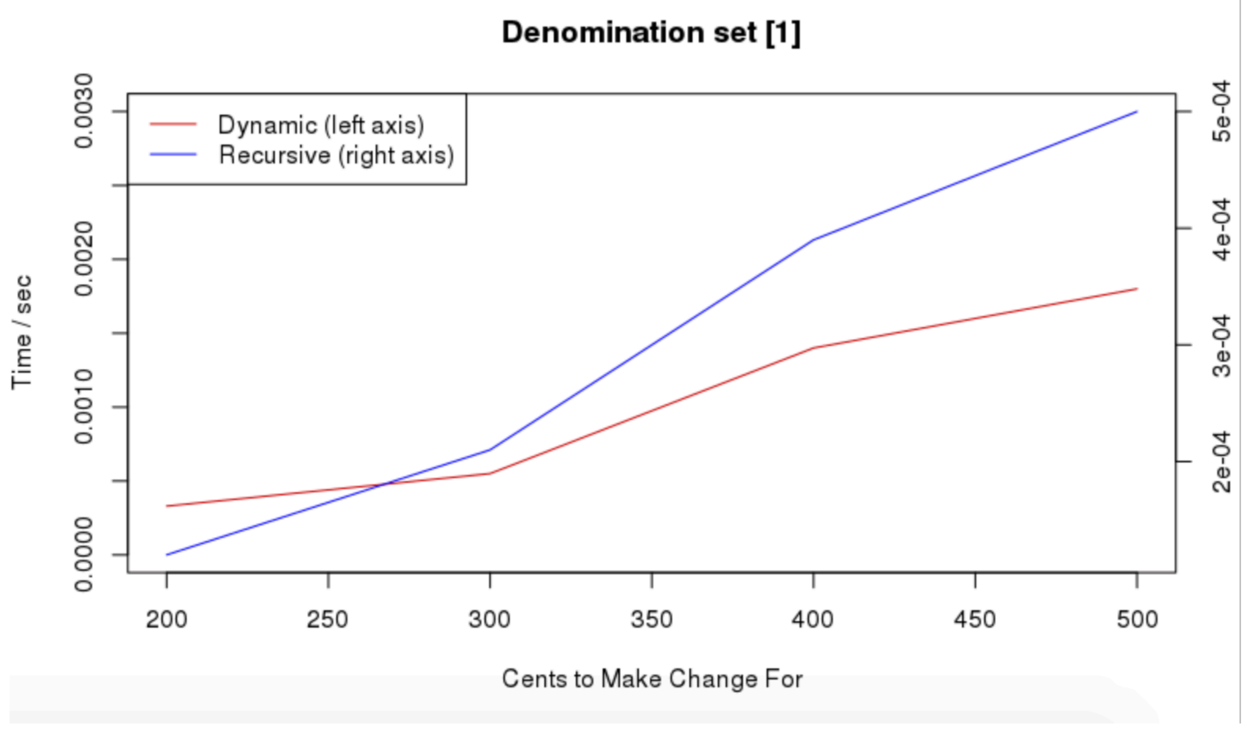
CS 519 - Homework 9 Paul ReFalo 11/26/17

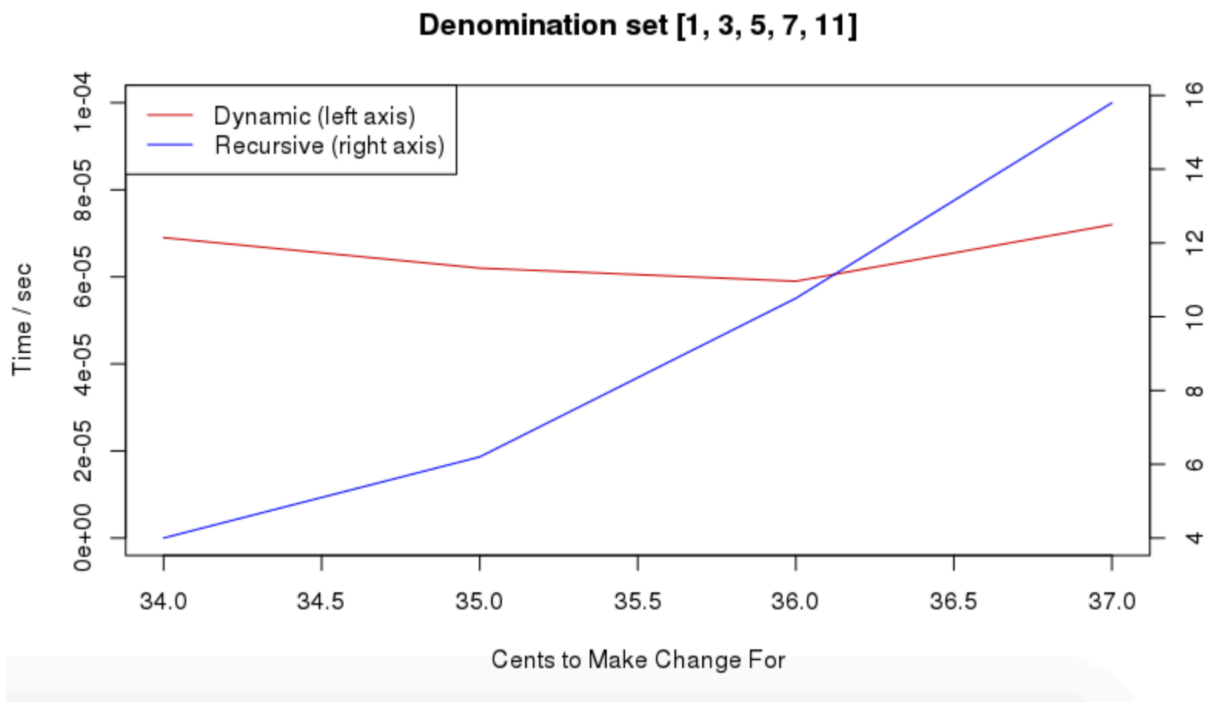
For this assignment I tested the following coin combinations and values. I then graphed the dynamic vs. recursive method times in seconds (note the left and right axis respectively). For only 1 coin, both methods perform well but as the coin combinations to go 5 and then 10 coins the recursive method slows significantly. For 5 and 10 coins values calibrated to ~15 seconds were chosen.

Dynamic vs. Recursive Run Times for Various Denominations

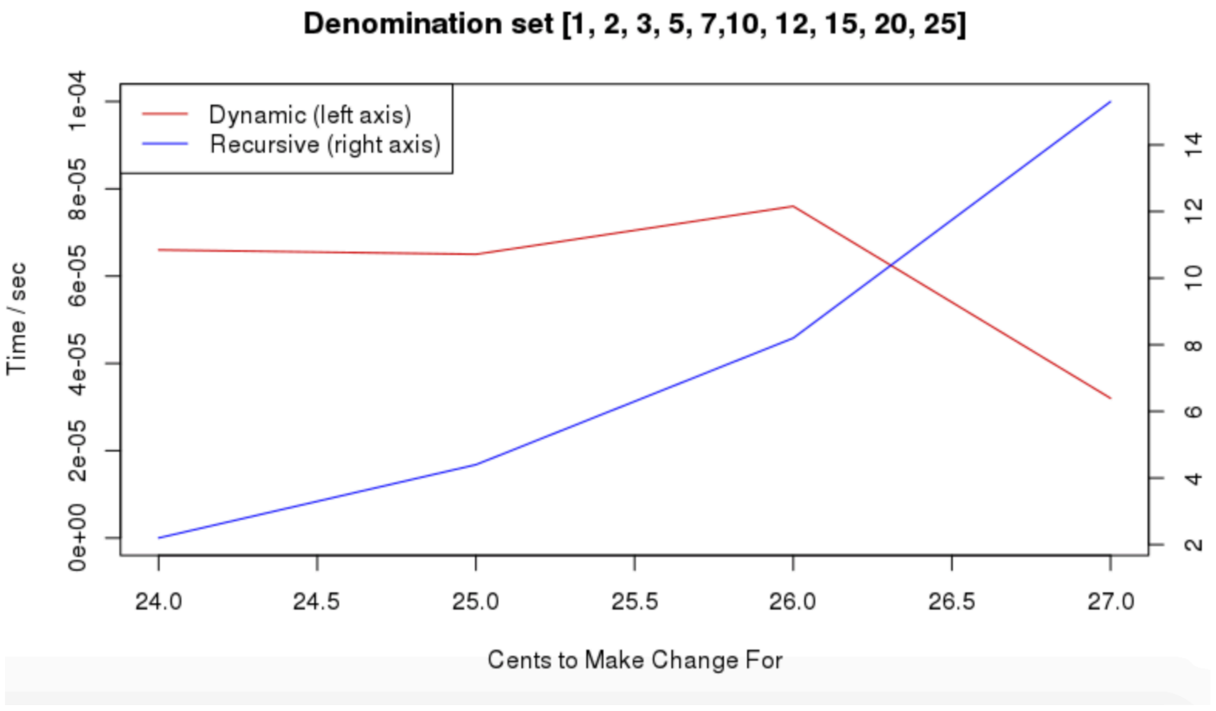
Coins: [1]	Dynamic	Recursive
200 cents	3.3E-04	1.2E-04
300 cents	5.5E-04	2.1E-04
400 cents	1.4E-03	3.9E-04
500 cents	1.8E-03	5.0E-04



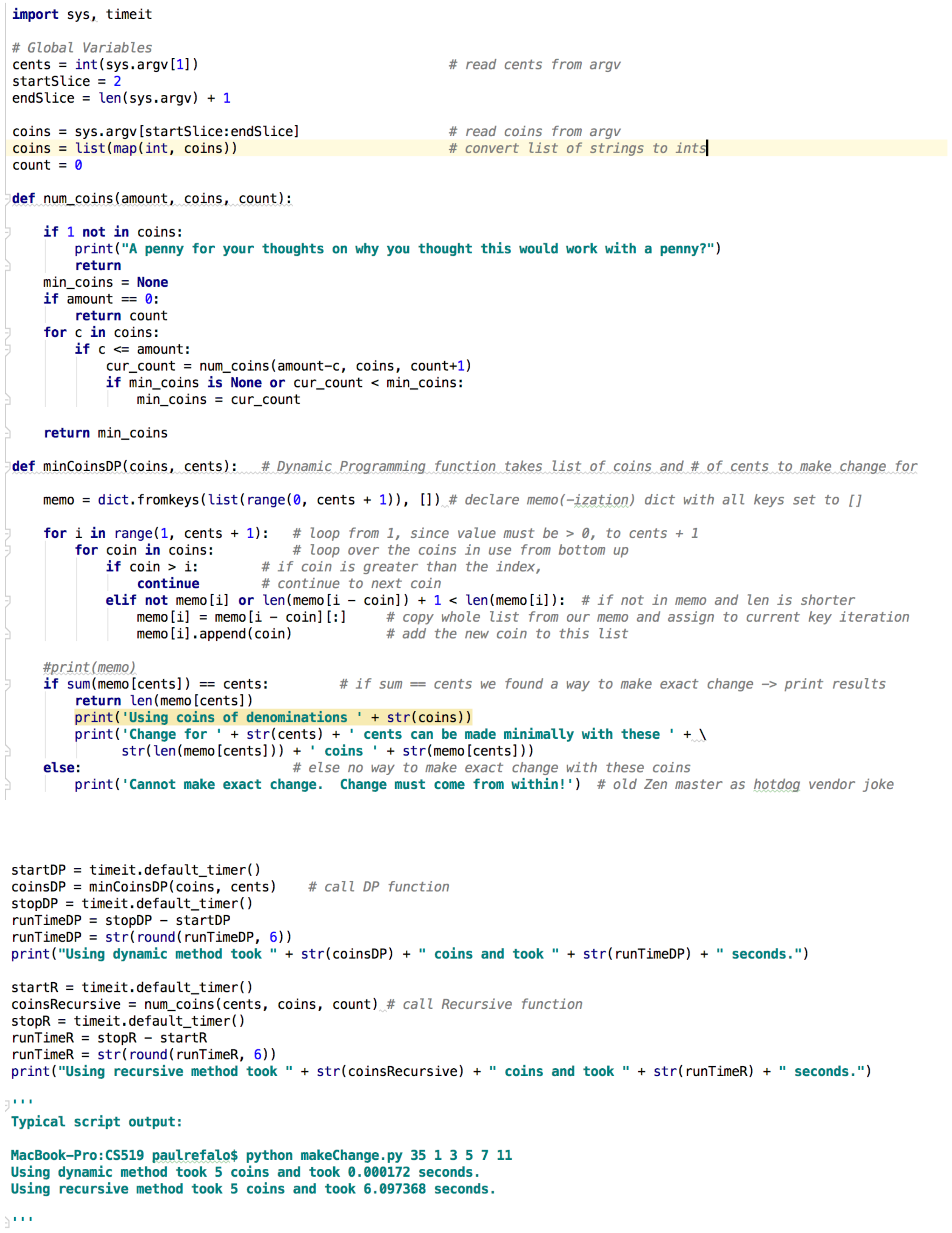
Coins: [1, 3, 5, 7, 11]	Dynamic	Recursive
34 cents	6.9E-05	4.0
35 cents	6.2E-05	6.2
36 cents	5.9E-05	10.5
37 cents	7.2E-05	15.8



Coins: [1, 2, 3, 5, 7, 10, 12. 15, 20, 25]	Dynamic	Recursive
24 cents	6.6E-05	2.2
25 cents	6.5E-05	4.4
26 cents	7.6E-05	8.2
27 cents	3.2E-05	15.3



makeChange.py script



```python
import sys, timeit

# Global Variables
cents = int(sys.argv[1])                                    # read cents from argv
startSlice = 2
endSlice = len(sys.argv) + 1

coins = sys.argv[startSlice:endSlice]                       # read coins from argv
coins = list(map(int, coins))                               # convert list of strings to ints
count = 0

def num_coins(amount, coins, count):

    if 1 not in coins:
        print("A penny for your thoughts on why you thought this would work with a penny?")
        return
    min_coins = None
    if amount == 0:
        return count
    for c in coins:
        if c <= amount:
            cur_count = num_coins(amount-c, coins, count+1)
            if min_coins is None or cur_count < min_coins:
                min_coins = cur_count

    return min_coins

def minCoinsDP(coins, cents):   # Dynamic Programming function takes list of coins and # of cents to make change for

    memo = dict.fromkeys(list(range(0, cents + 1)), []) # declare memo(-ization) dict with all keys set to []

    for i in range(1, cents + 1):   # loop from 1, since value must be > 0, to cents + 1
        for coin in coins:          # loop over the coins in use from bottom up
            if coin > i:        # if coin is greater than the index,
                continue        # continue to next coin
            elif not memo[i] or len(memo[i - coin]) + 1 < len(memo[i]):  # if not in memo and len is shorter
                memo[i] = memo[i - coin][:]         # copy whole list from our memo and assign to current key iteration
                memo[i].append(coin)                # add the new coin to this list

    #print(memo)
    if sum(memo[cents]) == cents:           # if sum == cents we found a way to make exact change -> print results
        return len(memo[cents])
        print('Using coins of denominations ' + str(coins))
        print('Change for ' + str(cents) + ' cents can be made minimally with these ' + \
            str(len(memo[cents])) + ' coins ' + str(memo[cents]))
    else:                           # else no way to make exact change with these coins
        print('Cannot make exact change.  Change must come from within!')  # old Zen master as hotdog vendor joke



startDP = timeit.default_timer()
coinsDP = minCoinsDP(coins, cents)    # call DP function
stopDP = timeit.default_timer()
runTimeDP = stopDP - startDP
runTimeDP = str(round(runTimeDP, 6))
print("Using dynamic method took " + str(coinsDP) + " coins and took " + str(runTimeDP) + " seconds.")

startR = timeit.default_timer()
coinsRecursive = num_coins(cents, coins, count) # call Recursive function
stopR = timeit.default_timer()
runTimeR = stopR - startR
runTimeR = str(round(runTimeR, 6))
print("Using recursive method took " + str(coinsRecursive) + " coins and took " + str(runTimeR) + " seconds.")

'''
Typical script output:

MacBook-Pro:CS519 paulrefalo$ python makeChange.py 35 1 3 5 7 11
Using dynamic method took 5 coins and took 0.000172 seconds.
Using recursive method took 5 coins and took 6.097368 seconds.

'''
```