Week 2 – Programming Exercises

**Problem 1:**

################

# Author: Daniel Burger

# Date: 03/04/2019

# This program creates a dictionary of States and capitols

# in the US, and quizzes the user

#################

import random

states\_capitals = {'Alabama':'Montgomery','Alaska':'Juneau','Arizona':'Phoenix','Arkansas':'Little Rock','California':'Sacramento','Colorad':'Denver','Connecticut':'Hartford','Delaware':'Dover','Florida':'Tallahassee','Georgia':'Atlanta','Hawaii':'Honolulu','Idaho':'Boise','Illinois':'Springfield','Indiana':'Indianapolis','Iowa':'Des Moines','Kansas':'Topeka','Kentucky':'Frankfort','Louisiana':'Baton Rouge','Maine':'Augusta','Maryland':'Annapolis','Massachusetts':'Boston','Michigan':'Lansing','Minnesota':'St. Paul','Mississippi':'Jackson','Missouri':'Jefferson City','Montana':'Helena','Nebraska':'Lincoln','Nevada':'Carson City','New Hampsire':'Concord','New Jersey':'Trenton','New Mexico':'Santa Fe','New York':'Albany','North Carolina':'Raleigh','North Dakota':'Bismarck','Ohio':'Columbus','Oklahoma':'Oklahoma City','Oregon':'Salem','Pennsylvania':'Harrisburg','Rhode Island':'Providence','South Carolina':'Columbia','South Dakota':'Pierre','Tennessee':'Nashville','Texas':'Austin','Utah':'Salt Lake City','Vermont':'Montpelier','Virginia':'Richmond','Washington':'Olympia','West Virginia':'Charleston','Wisconsin':'Madison', 'Wyoming':'Cheyenne'}

correct = 0 #initial score

incorrect = 0 #incorrect score

answer = 'capital' #initial answer to enter loop

#Opening line prompint user how to quit

print('Welcome to U.S. States and Capitals quiz! Enter "quit" as your answer to quit.')

print()

while (answer != 'quit') :

rand\_state = random.choice(list(states\_capitals))

print('What is the capital of ' + rand\_state +'?') #asks user a question using random state generator

print('Number of correct responses: ' + str(correct)) #tells user how many they have gotten right

print('Number of incorrect responses: ' + str(incorrect)) #tells user how many they have gotten wrong

answer = input('Enter answer here: ') #prompts user to answer

rand\_state\_cap = states\_capitals.get(rand\_state) #finds the correct answer

if (rand\_state\_cap == answer): #if answer is correct

correct = correct + 1 #add a point to correct

print()

print("Correct!") #notify the player of their achievement

print()

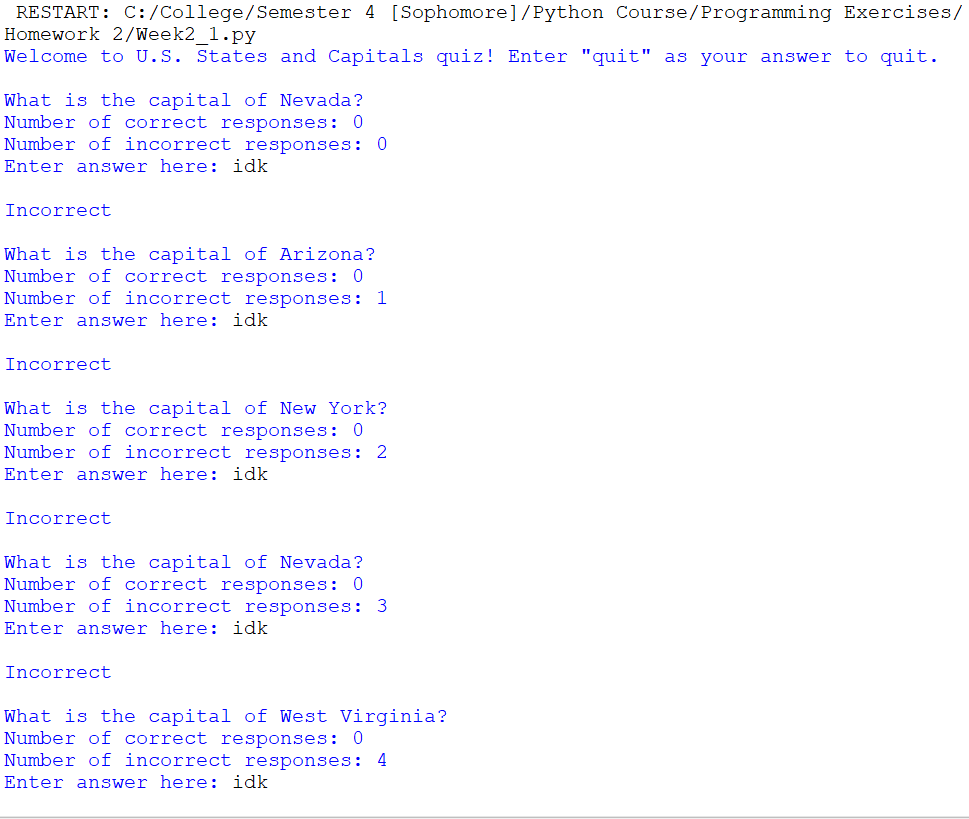
else: #if answer is incorrect

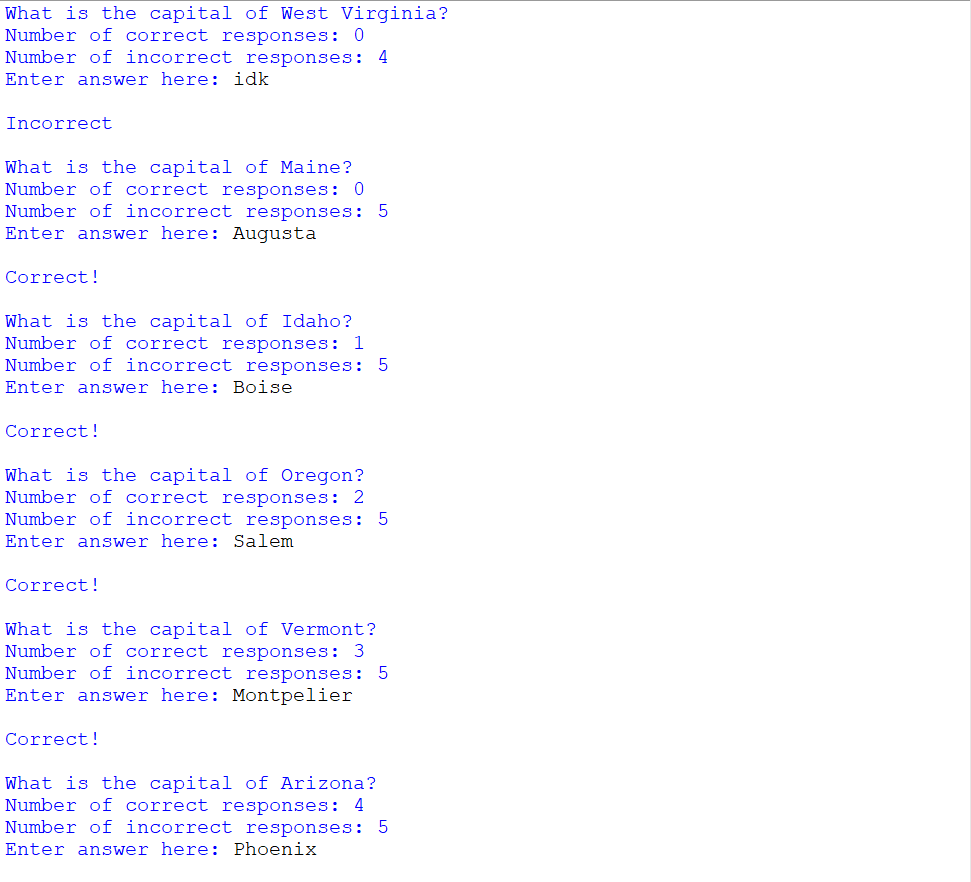
incorrect = incorrect + 1 #add a point to incorrect

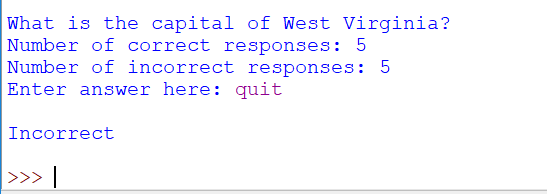
print()

print("Incorrect") #notify the player of their failure

print()







**Problem 2:**

################

# Author: Daniel Burger

# Date: 03/04/2019

# This program returns a list of all prime factors in

# a number in ascending order when given any positive integer

#################

def main():

input\_num = input('Enter a number: ') #asks user to enter number

num = int(input\_num) #typecasts input to an int to be used for math

prime\_factors\_list = prime\_factors(num) #calls function to return list

print(prime\_factors\_list) #prints answer

def prime\_factors(n):

prime\_nums = [] #creates empty list to store prime factors

exit\_loop = 1 #will become 0 when loop should exit

while exit\_loop:

if (((n%2) == 0) and (n != 2)): #if number is divisible by 2

n = n / 2 #then divide it by 2

prime\_nums.append(2) #and add 2 to the list of prime factors

continue

if (((n%3) == 0) and (n != 3)): #if number is divisible by 3

n = n / 3 #then divide it by 3

prime\_nums.append(3) #and add 3 to the list of prime factors

continue

if (((n%5) == 0) and (n != 5)): #if divisible by 5

n = n / 5 #then divide by 5

prime\_nums.append(5) #add 5 to list

continue

if (((n%7) == 0) and (n != 7)): #if divisible by 7

n = n / 7 #divide by 7

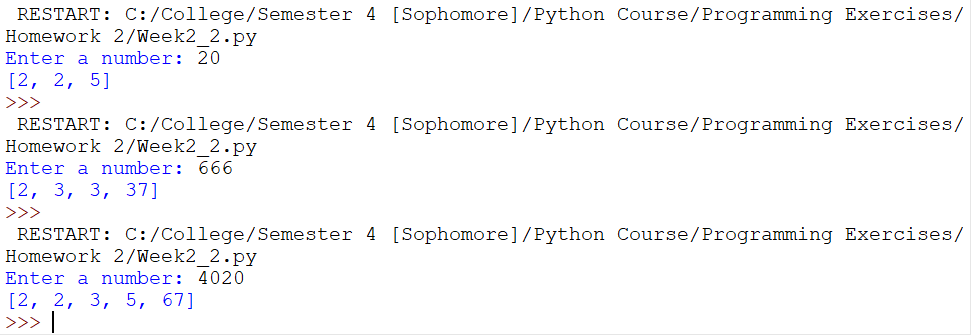
prime\_nums.append(7) #add 7 to list

continue

prime\_nums.append(int(n)) #add last prime number to list

exit\_loop = 0 #allows while loop to exit

return prime\_nums #returns answer in ascending order (list was built in ascending order)

main()

**Problem 3:**

################

# Author: Daniel Burger

# Date: 03/04/2019

# This program determines all natural numbers less than

# or equal to n which are coprime to n as well as the

# result of the Euler's phi function when run on n

#################

def main():

n = int(input('Enter a natural number: ')) #takes input from user and typecasts it to int

coprime\_nums, phi = totient(n) #calls totient function to get answers

#prints output to user

print(coprime\_nums, end='')

print(',' + str(phi))

def totient(n):

all\_nums = list(range(1,n)) #list of numbers <= n

#print(all\_nums)

for i in range(1,n): #for every number in all\_nums

coprime = are\_coprime(i,n)

if not coprime: #if the numbers are not coprime

all\_nums.remove(i) #remove the number from the list

return all\_nums, len(all\_nums)

#Determines if two numbers are coprime

def are\_coprime(x,y):

x\_primes = prime\_factors(x) #finds prime factors of x

y\_primes = prime\_factors(y) #finds prime factors of y

x\_primes\_set = set(x\_primes) #converts to set

y\_primes\_set = set(y\_primes) #converts to set

difference = x\_primes\_set - y\_primes\_set #finds differences in set

if (difference == x\_primes\_set): #if true, both sets share no values, aka coprime

return 1

else: #not coprime

return 0

def prime\_factors(n):

prime\_nums = [] #creates empty list to store prime factors

exit\_loop = 1 #will become 0 when loop should exit

while exit\_loop:

if (((n%2) == 0) and (n != 2)): #if number is divisible by 2

n = n / 2 #then divide it by 2

prime\_nums.append(2) #and add 2 to the list of prime factors

continue

if (((n%3) == 0) and (n != 3)): #if number is divisible by 3

n = n / 3 #then divide it by 3

prime\_nums.append(3) #and add 3 to the list of prime factors

continue

if (((n%5) == 0) and (n != 5)): #if divisible by 5

n = n / 5 #then divide by 5

prime\_nums.append(5) #add 5 to list

continue

if (((n%7) == 0) and (n != 7)): #if divisible by 7

n = n / 7 #divide by 7

prime\_nums.append(7) #add 7 to list

continue

prime\_nums.append(int(n)) #add last prime number to list

exit\_loop = 0 #allows while loop to exit

return prime\_nums #returns answer in ascending order (list was built in ascending order)

main()

