Assignment 3 – Python

**Exercise – Additional List Practice**

Write a function list\_intersection that takes two lists as parameters. Return a list that gives the intersection of the two lists- ie, a list of elements that are common to both lists. Run the following test cases and make sure your results are the same (the ordering of your outputs does not matter - [3,2] is the same as [2,3]):

>>> list\_intersection([1, 3, 5], [5, 3, 1])

[1, 3, 5]

>>> list\_intersection([1, 3, 6, 9], [10, 14, 3, 72, 9])

[3, 9]

>>> list\_intersection([2, 3], [3, 3, 3, 2, 10])

[3, 2]

>>> list\_intersection([2, 4, 6], [1, 3, 5])

[]

Ans:

Def list\_intersection(a, b):

seta = set(a)

setb = set(b)

result = list()

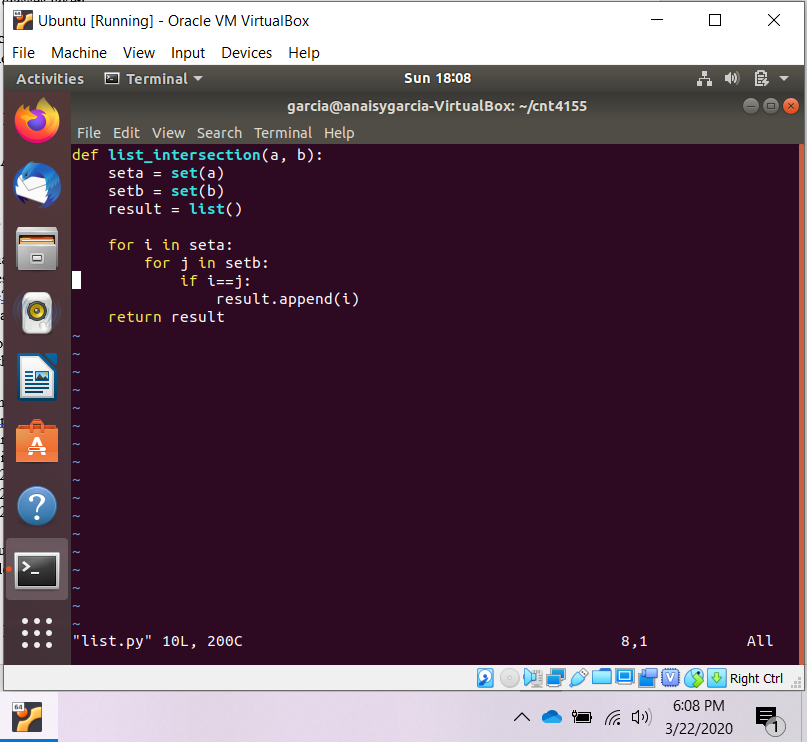
for i in seta:

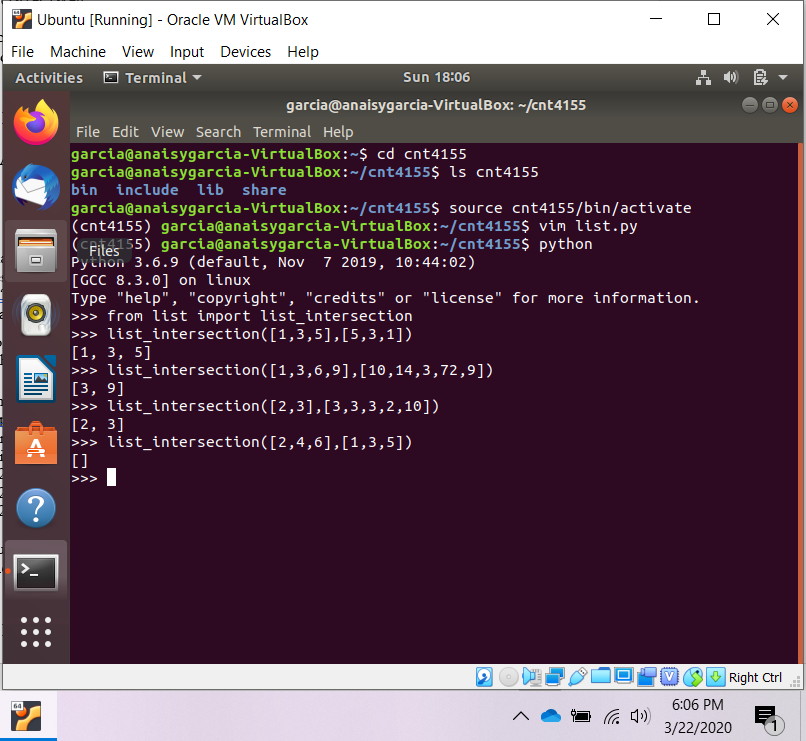
for j in setb:

if i==j:

result.append(i)

return result





**Exercise – An Introduction to Dictionaries**

Quick Reference

D= {}} creates an empty dictionary

D= {key1:value1, ...} creates a non-empty dictionary

D[key] returns the value that mapped to by key. (What if there’s no such key?)

D[key] = newvalue maps newvalue to key. Overwrites any previous value. Remember -newvalue can be any valid Python data structure.

del D[key] deletes the mapping with that key from D.

len(D) returns the number of entries (mappings) in D.

x in D, x not in D checks whether the key x is in the dictionary D.

D.keys() -returns a list of all the keys in the dictionary.

D.values() -returns a list of all the values in the dictionary.

For this exercise, write a dictionary that catalogs the classes you took last term -the keys should be the classnumber, and the values should be the title of the class.

Then, write a function add class that takes 2 arguments -a class number and a description -that adds new classesto your dictionary. Use this function to add the classes you’re taking next term to the dictionary.

Finally, write a function print classes that takes one argument -a Course number (eg ’6’) -and prints out all the classes you took in that Course.

Example output:

>> print\_classes(’6’)

4155 -Introduction to Python

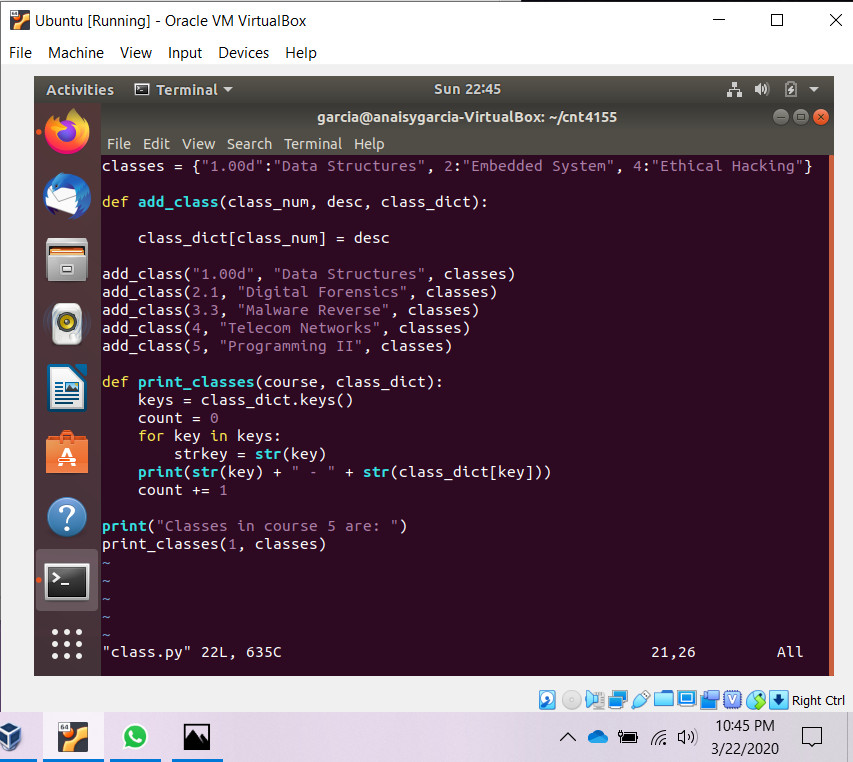
4806 –Ethical Hacking

>> print\_classes(’9’)

No Course 9 classes taken

For this exercise, we suggest using strings everywhere. Be sure to test with Course numbers that you both did and did not take!

Ans:



**Exercise – More About Dictionaries**

NAMES = [’Alice’, ’Bob’, ’Cathy’, ’Dan’, ’Ed’, ’Frank’,

’Gary’, ’Helen’, ’Irene’, ’Jack’, ’Kelly’, ’Larry’]

AGES = [20, 21, 18, 18, 19, 20, 20, 19, 19, 19, 22, 19]

These lists match up, so Alice’s age is 20, Bob’s age is 21, and so on. Write a function combine\_lists that combines these lists into a dictionary (hint: what should the keys, and what should the values, of this dictionary be? ). Then, write a function people that takes in an age and returns the names of all the people who are that age.

Test your program’s functions by running these lines (they are commented at the bottom of the code file; uncomment them to use them):

print ’Dan’ in people(18) and ’Cathy’ in people(18)

print ’Ed’ in people(19) and ’Helen’ in people(19) and\

’Irene’ in people(19) and ’Jack’ in people(19) and ’Larry’in people(19)

print ’Alice’ in people(20) and ’Frank’ in people(20) and ’Gary’ in people(20)

print people(21) == [’Bob’]

print people(22) == [’Kelly’]

print people(23) == []

All lines should print True. The last line is an “edge condition” that we’re testing; your people function should be able to handle this condition (hint: what is this condition?) by simply returning an empty list.

Ans:

def combine\_lists(la, lb):

comb = {}

ind = 0

for elem in la:

comb[elem] = lb[ind]

ind += 1

return comb

NAMES = ["Alice", "Bob", "Cathy", "Dan", "Ed", "Frank", "Gary", "Helen", "Irene", "Jack", "Kelly", "Larry"]

AGES = [20, 21, 18, 18, 19, 20, 20, 19, 19, 19, 22, 19]

dic = combine\_lists(NAMES, AGES)

print(dic)

def people(age):

answer = []

keys = dic.keys()

for elem in keys:

if dic[elem] == age:

answer.append(elem)

return answer

#print ’Dan’ in people(18) and ’Cathy’ in people(18)

#print ’Ed’ in people(19) and ’Helen’ in people(19) and\

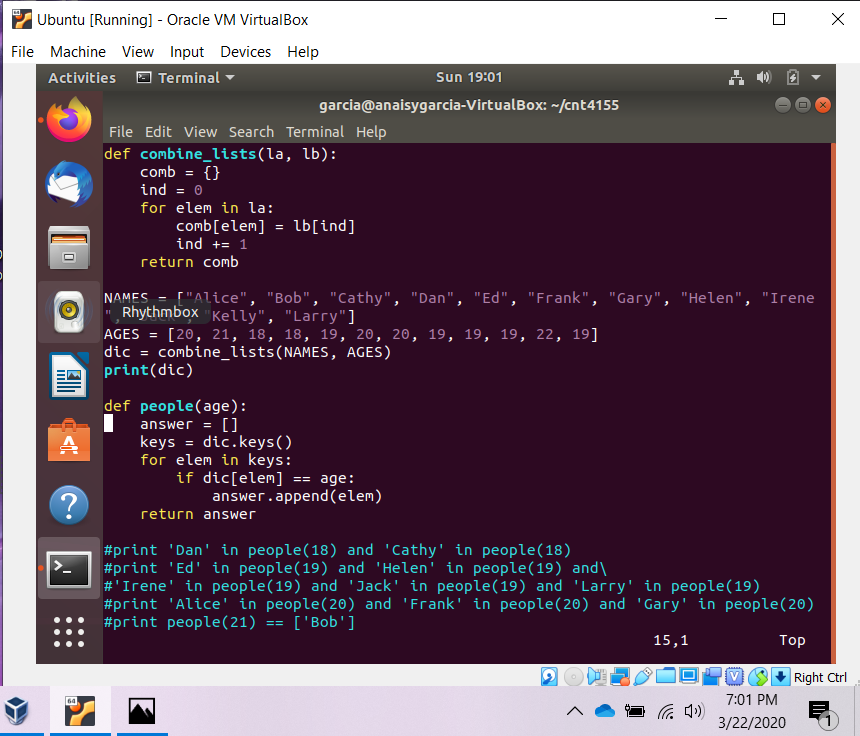
# ’Irene’ in people(19) and ’Jack’ in people(19) and ’Larry’in people(19)

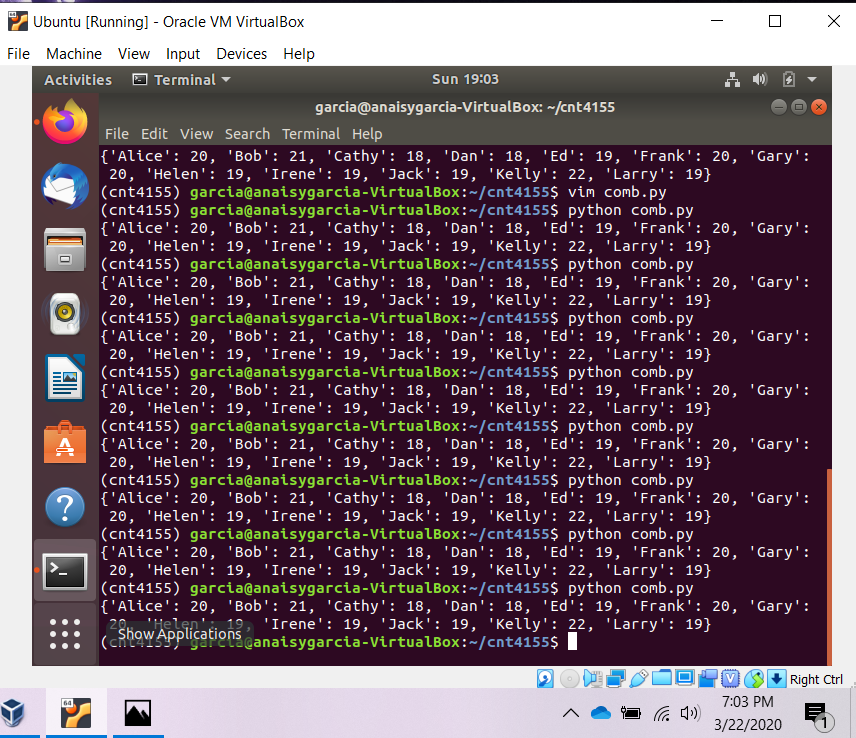
#print ’Alice’ in people(20) and ’Frank’ in people(20) and ’Gary’ in people(20)

#print people(21) == [’Bob’]

#print people(22) == [’Kelly’]

#print people(23) == []





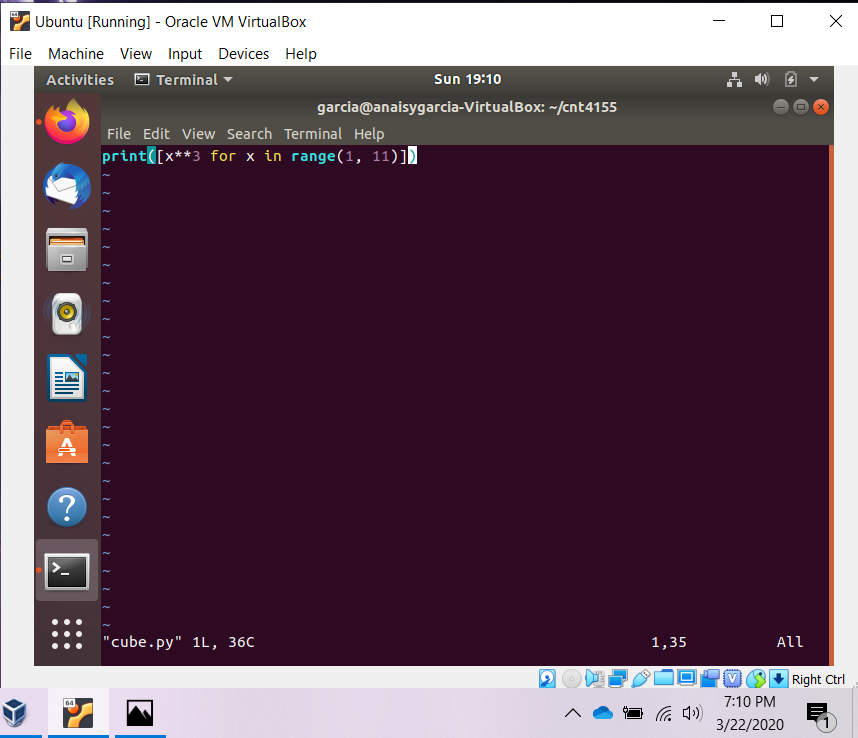
**Exercise – List Comprehensions**

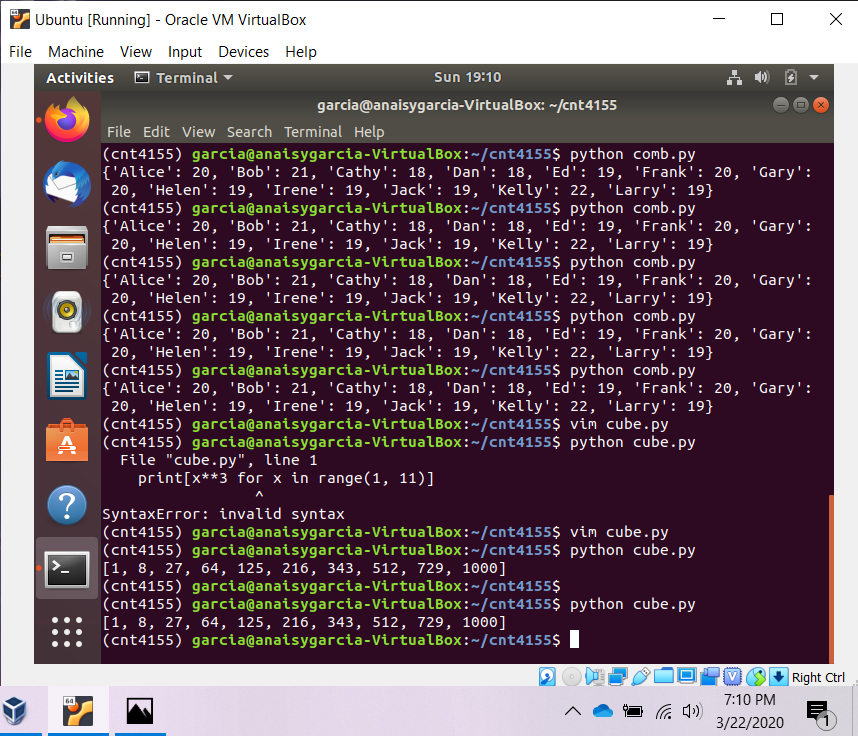
List comprehensions follow naturally from set builder notation and lambda calculus. They are very cool and make your life a lot easier. Don’t worry if you don’t get them, but they can be very useful.

1. Write a list comprehension that prints a list of the cubes of the numbers 1 through 10.

Ans:

print([x\*\*3 for x in range(1, 11)])



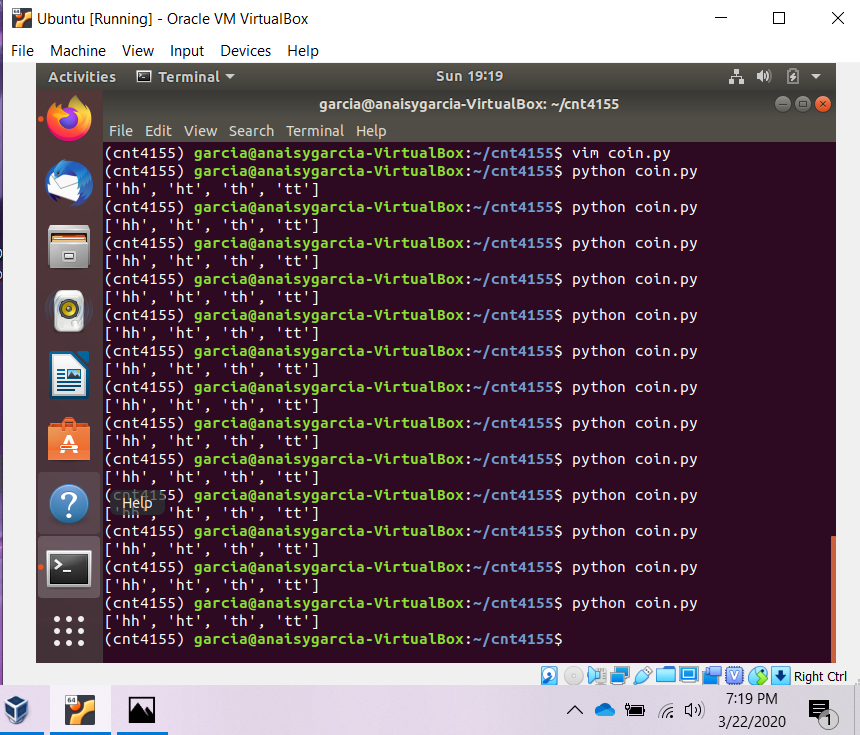


1. Write a list comprehension that prints out the possible results of two coin flips (one result would be ’ht’). (Hint -how many results should there be?)

Ans:

print([x+y for x in [‘h’, ‘t’] for y in [‘h’, ‘t’]])





1. Write a function that takes in a string and uses a list comprehension to return all the vowels in the string.

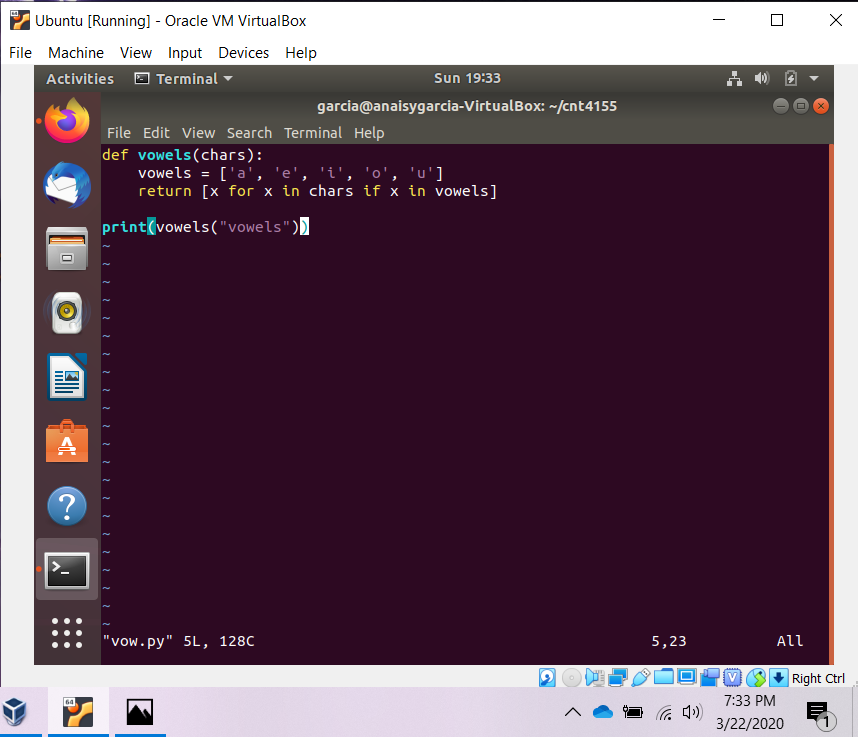
Ans:

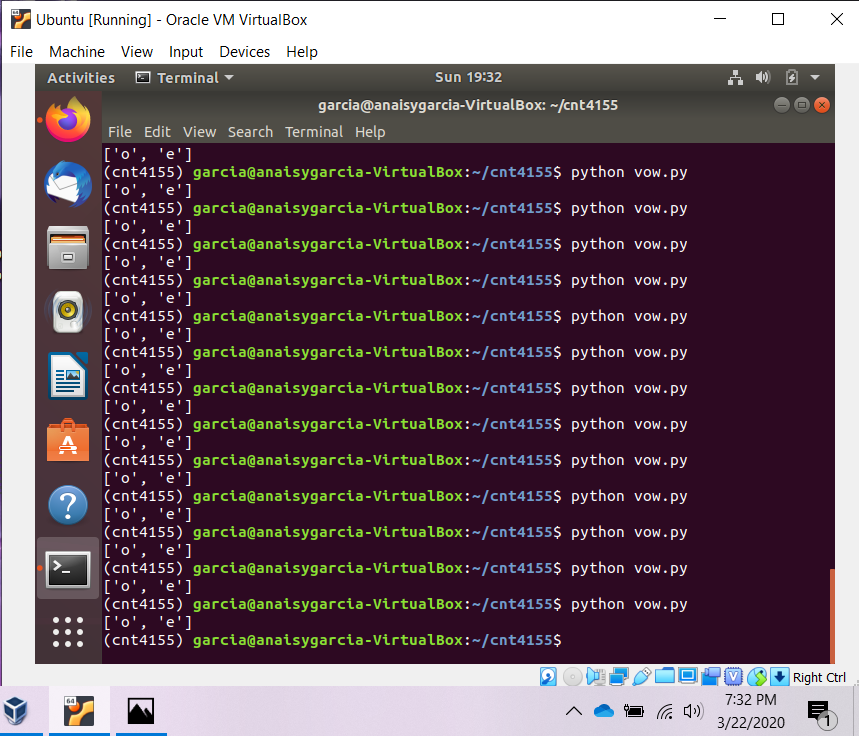
def vowels(chars):

vowels = [‘a’, ‘e’, ‘i’, ‘o’, ‘u’]

return [x for x in chars if x in vowels]

print(vowels(“vowels”))





4. Run this list comprehension in your prompt:

[x+y for x in [10,20,30] for y in [1,2,3]]

Figure out what is going on here, and write a nested for loop that gives you the same result. Make sure what is going on makes sense to you!

Ans:

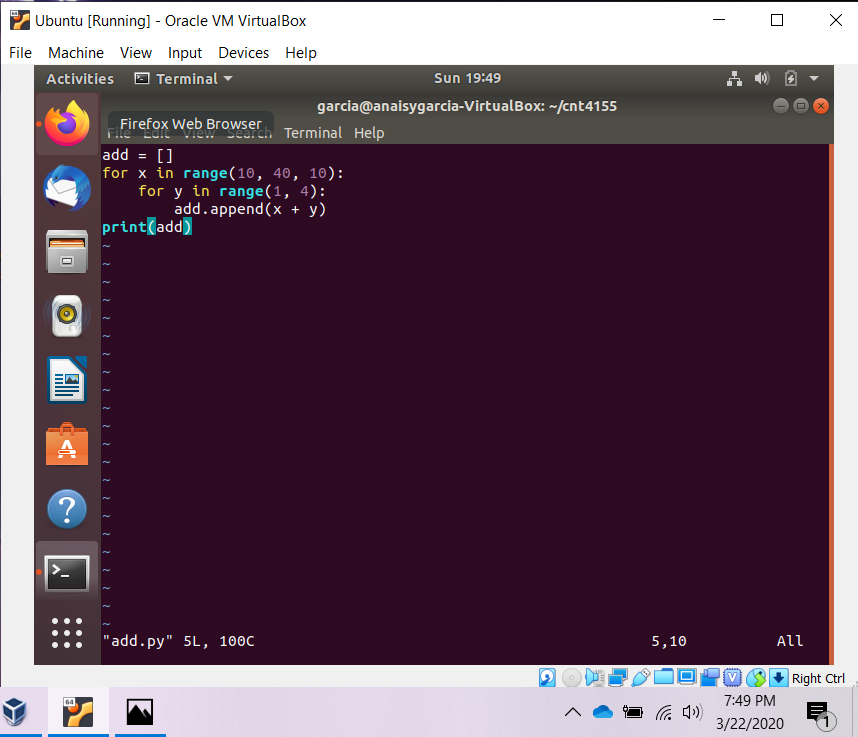
add = []

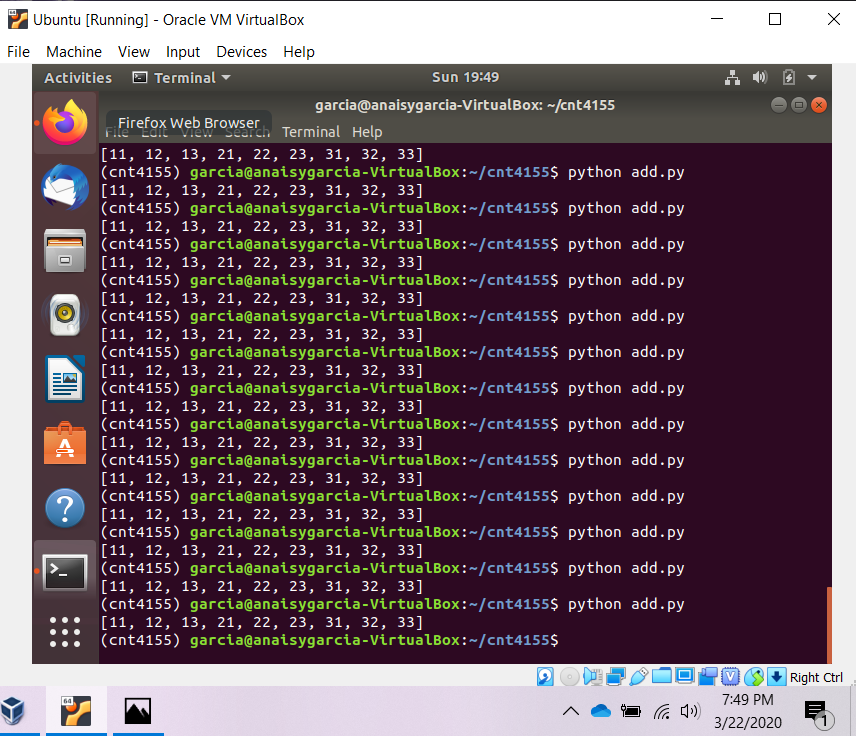
for x in range(10, 40, 10):

for y in range(1, 4):

add.append(x + y)

print(add)





**Exercise – List Comprehension Challenges (tricky!)**

1. Write a function that takes in a list of elements of different types and uses a list comprehension to return all the elements of the list of type int. Note: The function **isinstance** will be of help here. Google “Python isinstance” and see if you can figure out what it does, or type help(isinstance) at the Python shell.

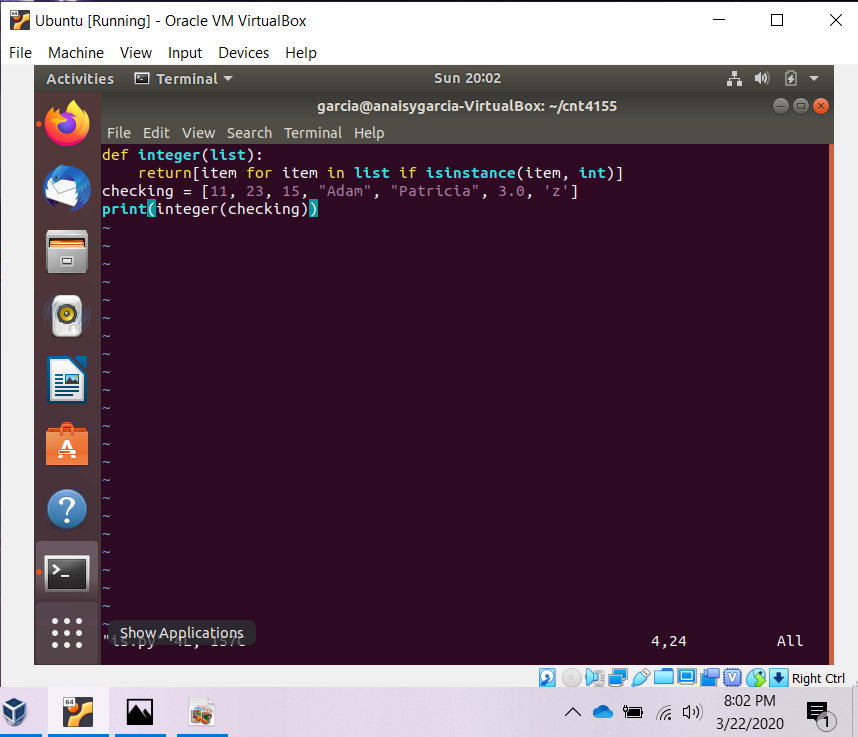
Ans:

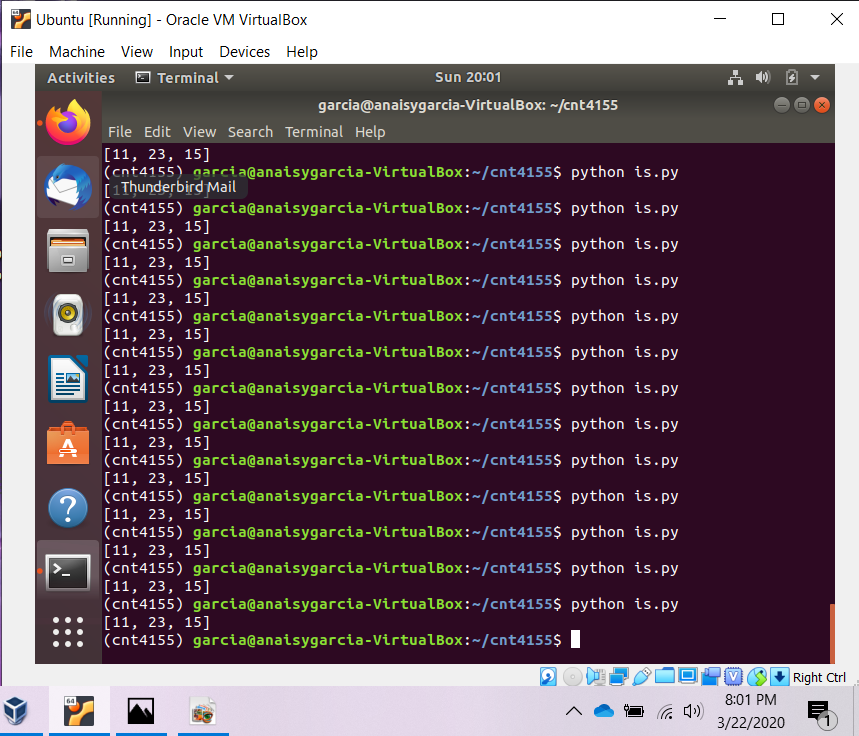
def integer(list):

return [item for item in list if isinstance(item, int)]

checking = [11,23,15,"Adam","Patricia",3.0,'z']

print(integer(checking))

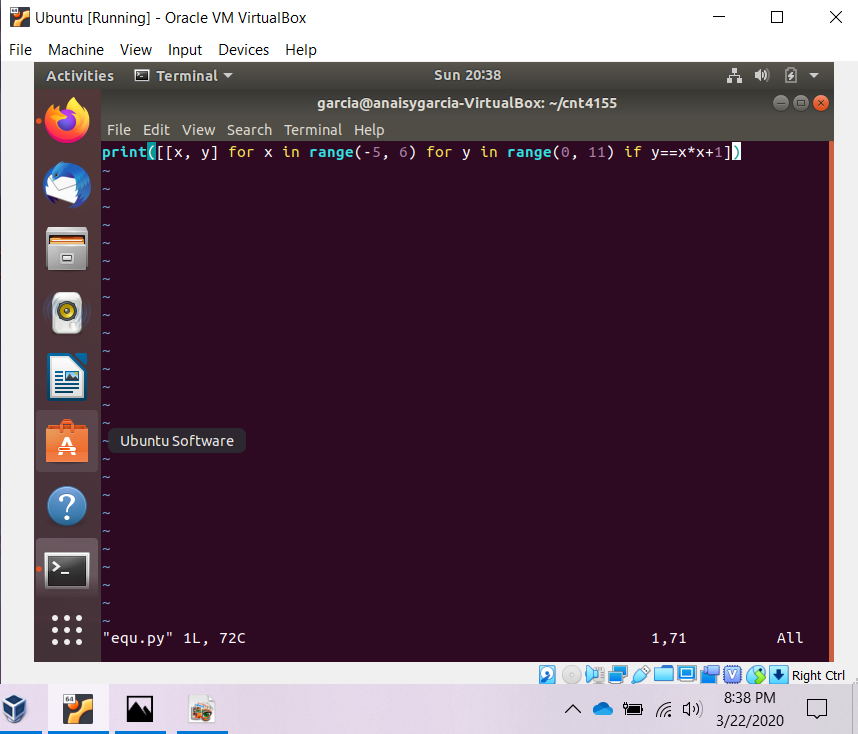




1. Write a list comprehension which solves the equation y = x2 +1. Your solution should print out a list of [x,y] pairs; use the domain x ∈ [−5, 5] and the range y ∈ [0, 10].

Ans:

print([[x, y] for x in range(-5, 6) for y in range(0, 11) if y == x\*x+1])

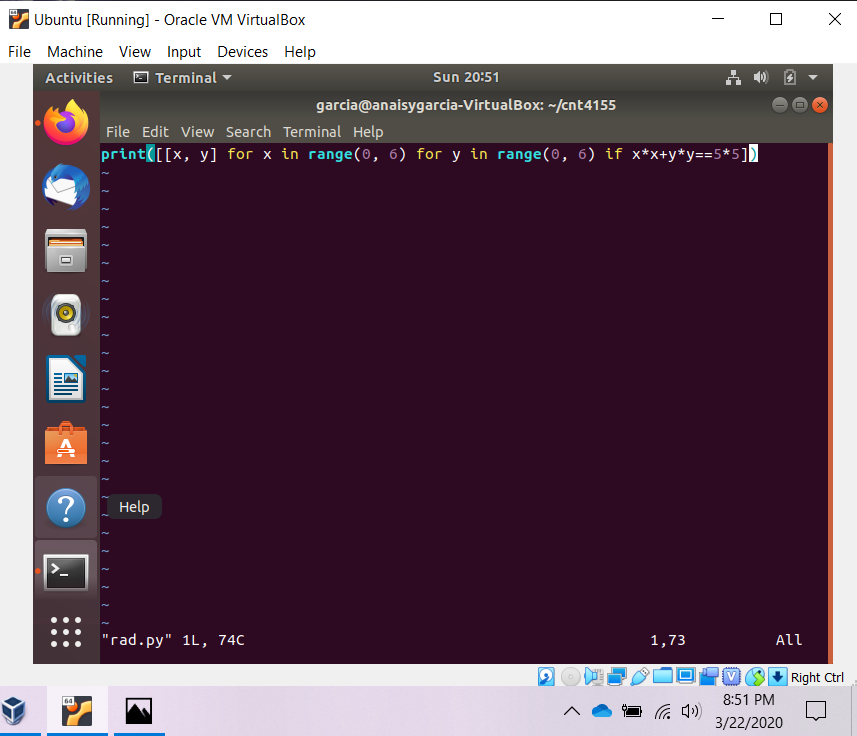


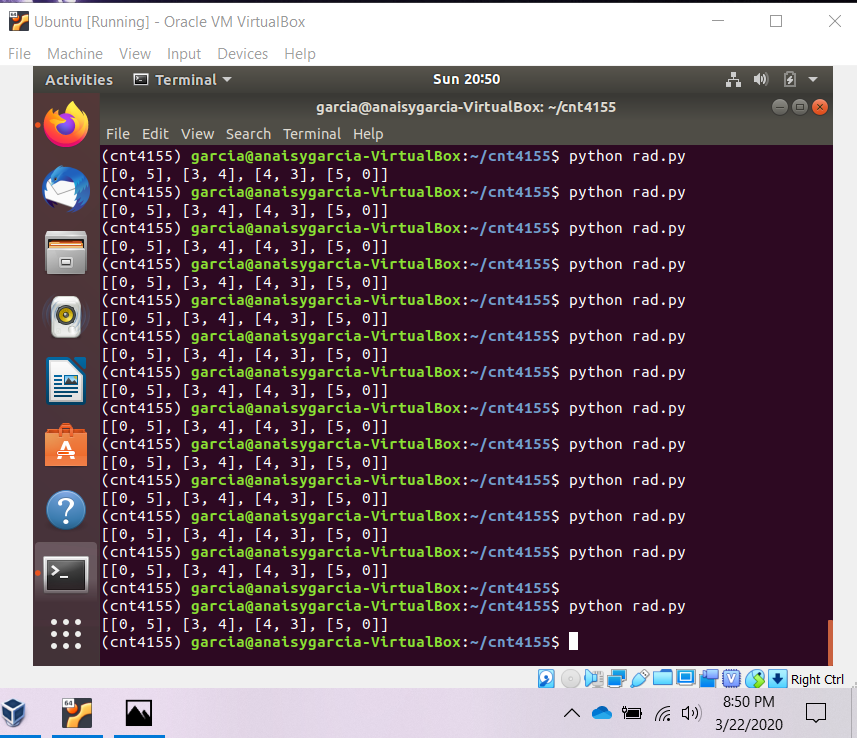


1. Similarly, write a list comprehension that finds the integer solutions [x, y] for a circle of radius 5.

Ans:

print([[x,y] for x in range(0,6) for y in range(0,6) if x\*x+y\*y==5\*5])





**Exercise – Generator Function and Expression**

Create and demonstrate the use of the developed generator function and expression.

1. Write a generator function that will take a number n and generate all of the combinations using the sequence of numbers, ex. N = 3, (0, 1, 2) and create all combination (0,0) (0,1) (0,2) (1,1) (1,2) (2,2) N! = 6 and show its operation in using it in a list and print its generation.

Ans:

def generator(n):

gen = []

for i in range(n):

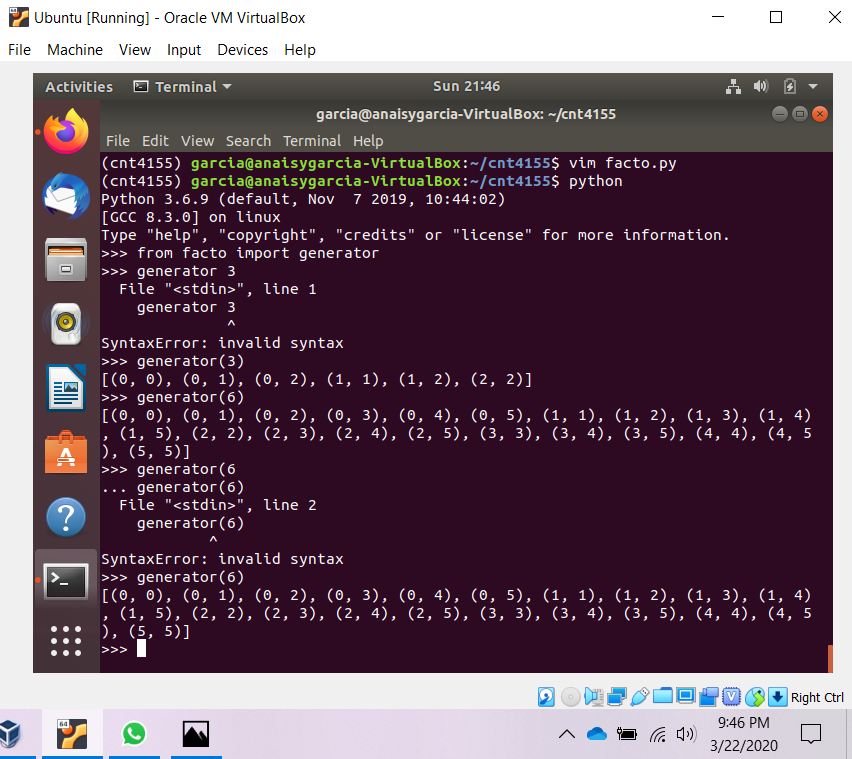
for j in range(n):

if (i, j) and (j, i) not in gen:

gen.append((i,j))

print(gen)





1. Create an example for a generation expression that will compute the cute of all of the odd numbers between 1 and 100, show how you would use this generator.

Ans:

def takecube():

c = 0

for i in range(1, 101):

if i%2!=0:

c = c+i\*\*3

else:

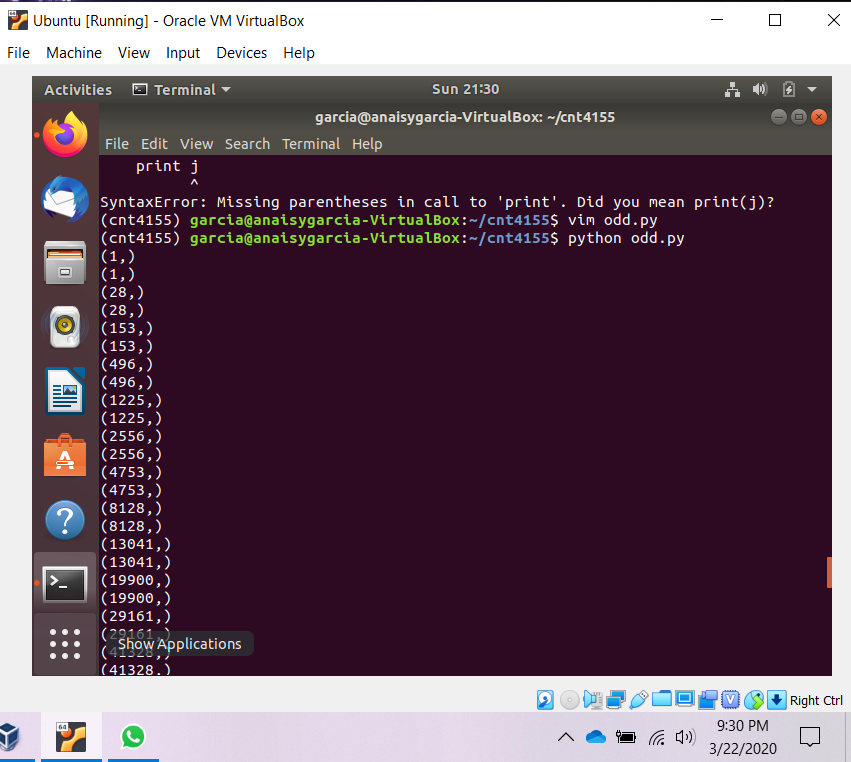
pass

yield c,

for j in takecube():

print(j)





**Exercise – Finding Bugs**

The following set of instructions was given to Juan, and he produced the code below. Find at least three bugs he made, and say how to fix them.

Instructions: Write a negate function that takes a number and returns the negation of that number. Also write a large\_num function that takes a number, and returns True if that number is bigger than 10000, and False otherwise. Additionally, write some code to test your functions.

def negate(num):

return –num

def large\_num(num):

res = (num > 10000)

negate(b)

neg\_b = num

print ’b:’, b, ’neg\_b:’, neg\_b

big = large\_num(b)

print ’b is big:’, big

Bugs:

1. b is undefined
2. nothing is returned by the function large\_num()
3. the negate() function should be equated to neg\_b, not neg\_b = num which are both undefined

Fixed Code:

def negate(num):

return -num

def large\_num(num):

res = (num > 10000)

return res

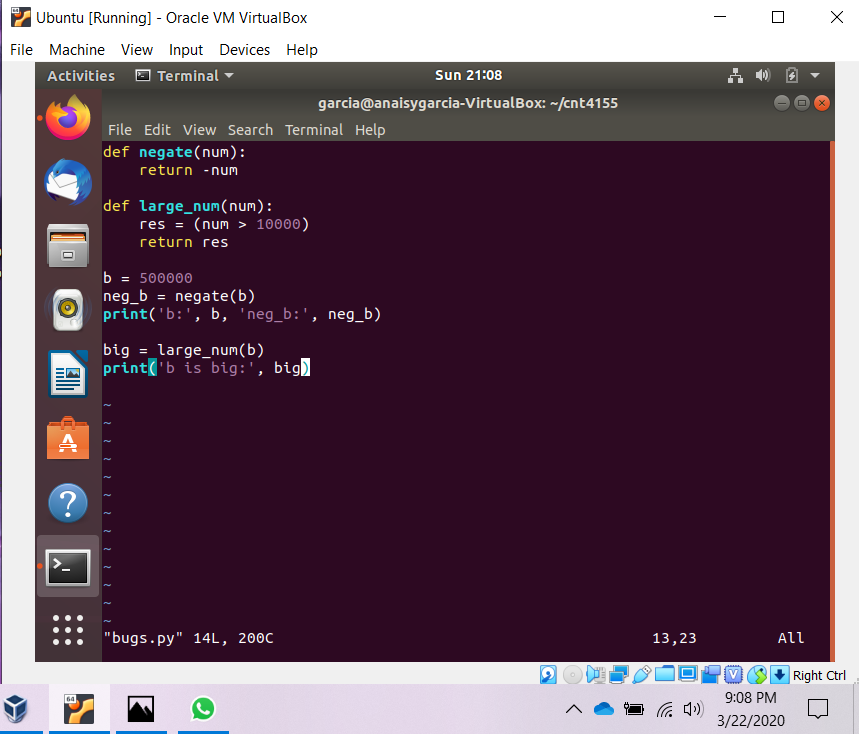
b = 500000

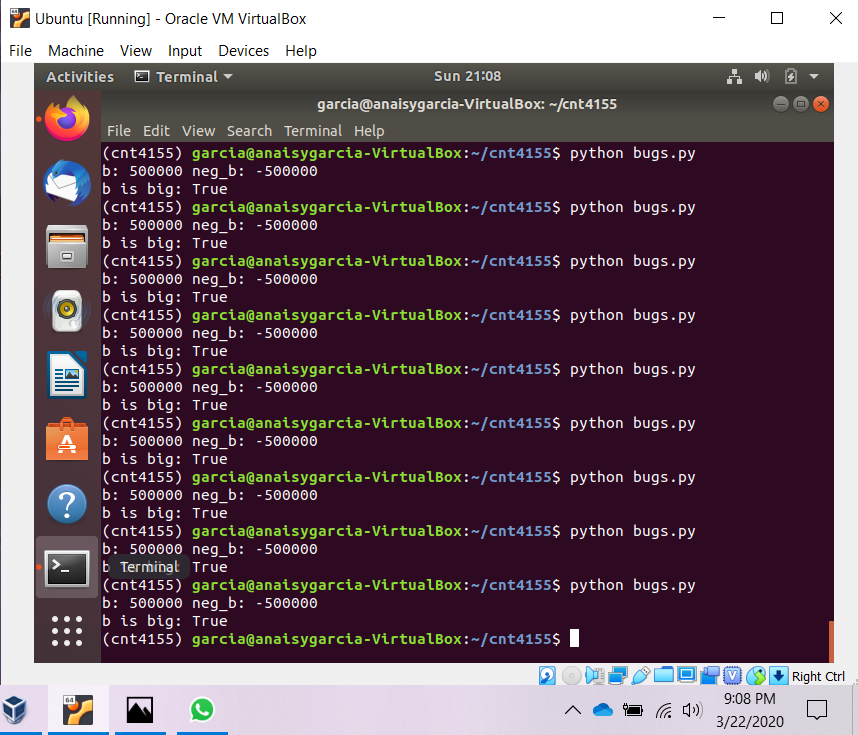
neg\_b = negate(b)

print ('b:', b, 'neg\_b:', neg\_b)

big = large\_num(b)

print ('b is big:', big)





**Exercise – Mystery Program**

Now, in another assignment Juan turned in the following uncommented code. Help us figure out what it does!

1 print "Think of a number between 1 and 100, but don’t tell me what you choose."

2 min\_n = 1

3 max\_n = 100

4 right\_answer = False

5

6 while not right\_answer:

7 mid\_n = (max\_n + min\_n + 1)/2

8 answer = raw\_input(’Is it ’ + str(mid\_n) + ’? ’)

9 if answer[0] == ’y’:

10 right\_answer = True

11 elif answer.startswith(’higher’):

12 min\_n = mid\_n + 1

13 elif answer.startswith(’lower’):

14 max\_n = mid\_n -1

15 else:

16 print "Sorry, I don’t understand your answer."

17

18 print ’Woohoo! I got it!’

1. The while loop exits when the variable right answer is True. What will cause right answer to be true?

When the user enters a string starting with a y, it will set right\_answer to true.

1. How many times will the program print out ’Woohoo! I got it!’?

The program will print this statement once only after the loop ends.

1. What are we using the variable answer for?

The variable answer stores the user response.

1. The program makes a guess in line 8. What user responses will be understood by the program after it makes its guess?

There are three types of responses understood by the program which are ones that start with the letter y, ones that have the prefix higher, and ones with the prefix lower.

1. If the program gets the response ’higher’, what does that tell it about its guess?

What it says about the guess is that it is lower than the answer and it needs to guess higher.

1. What are the variables min n, max n and mid n used for?

The max n stores the number of the upper limit of the range. The min n stores the lower limit of the range. The mid n is the middle number of the range.

*This is an example of binary search, a simple but important algorithm in computer science. If you’re curious, or confused, read the Wikipedia article on binary search to find out more and get a good explanation of what’s going on here.*