Assignment 2 – Python

Please add any code files to this document and any code you wrote for optional problems).

**Exercise – Conditionals**

The purpose of this exercise is to understand conditionals. Maria is looking for her dream job, but has some restrictions. She loves California and would take a job there if it paid over 40,000 a year. She hates Massachusetts and demands at least 100,000 to work there. Any other place she’s content to work for 60,000 a year, unless she can work in space in which case she would work for free. The following code shows his basic strategy for evaluating a job offer.

pay = \_\_\_\_\_

location = \_\_\_\_\_

if location == "U.S.S. Enterprise":"

print "So long, suckers! I’ll take it!"

elif location == "Massachusetts":

if pay < 100000:

print "No way"

else:

print "I’ll take it!"

elif location == "California" and pay > 40000:

print "I’ll take it!"

elif pay > 60000:

print "I’ll take it!"

else:

print "No thanks, I can find something better."

For each of the following job offers, write down the output that would be generated. Do this without running the code. It is an important skill to be able to understand what a piece of code does without running it.

1. location = "Massachusetts"

pay = 50000

ans: No Way

1. location = "Iowa"

pay = 50000

ans: No thanks, I can find something better.

1. location = "California"

pay = 50000

ans: I’ll take it!

1. location = "U.S.S. Enterprise"

pay = 1

ans: So long, suckers! I’ll take it!

1. location = "California"

pay = 25000

ans: No thanks, I can find something better.

**Exercise – Understanding loops**

For each of the following fragments of code, write what the output would be. Again, do this without running the code (although feel free to check yourself when you’re done).

1. num = 10

while num > 3:

print num

num = num -1

ans:

10

9

8

7

6

5

4

2. divisor = 2

for i in range(0, 10, 2):

print i/divisor

ans:

0.0

1.0

2.0

3.0

4.0

3. num = 10

while True:

if num < 7:

break

print num

num -= 1

ans:

10

9

8

7

4. count = 0

for letter in ’Snow!’:

print ’Letter #’, count, ’is’, letter

count += 1

ans:

Letter # 0 is S

Letter # 1 is n

Letter # 2 is o

Letter # 3 is w

Letter # 4 is !

**Exercise– Printing**

Write a program using print that, when run, prints out a tic-tac-toe board. Remember to save your program regularly, to keep from losing your work! The purpose of this exercise is to make sure you understand how to write programs using your computing environment; many students in introductory courses experience trouble with assignments not because they have trouble with the material, but because of some weird environment quirk.

Expected output:

| |

--------

| |

--------

| |

Ans:

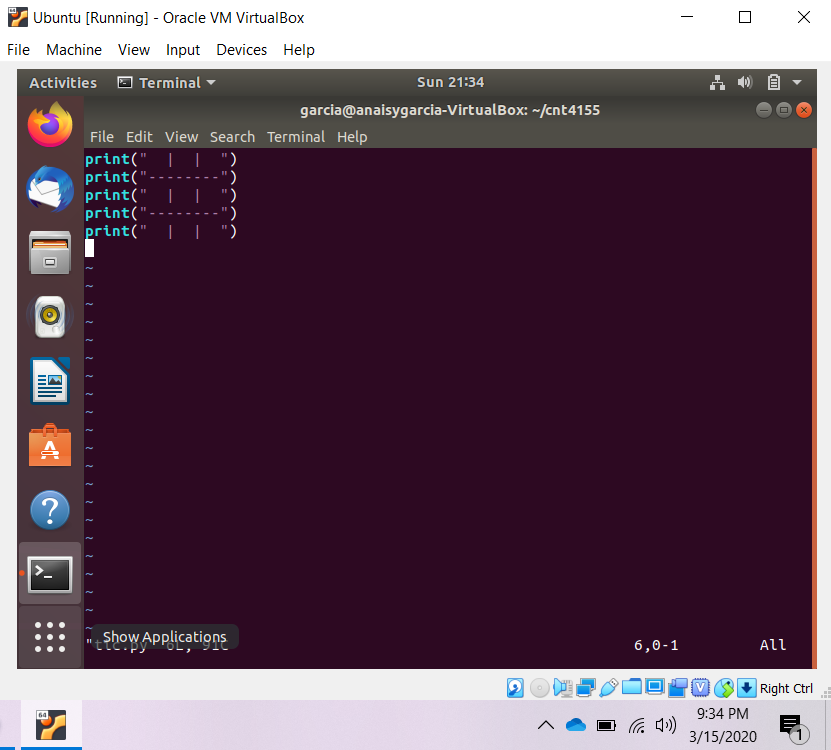
print(" | | ")

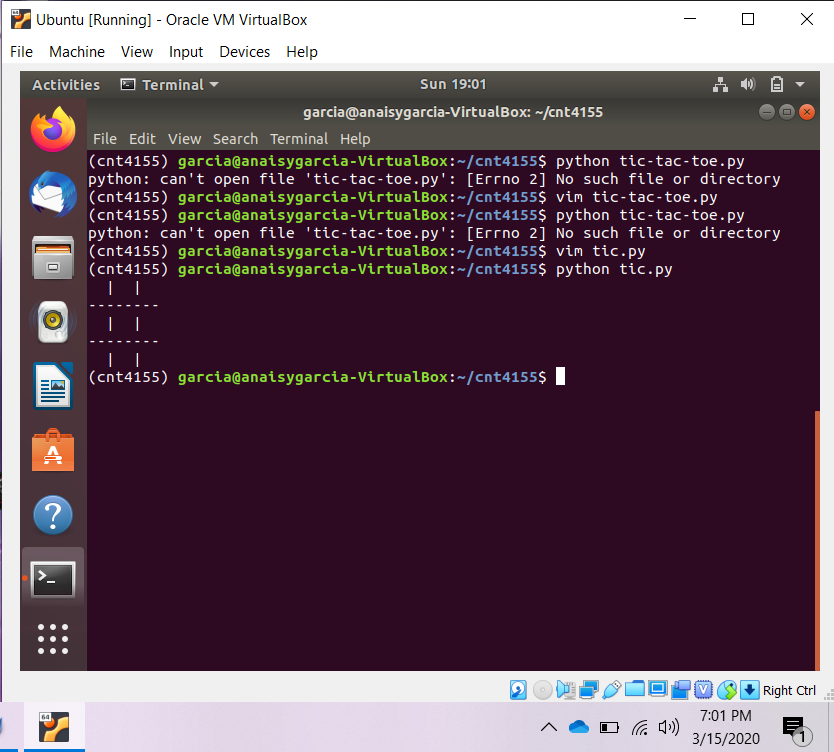
print("--------")

print(" | | ")

print("--------")

print(" | | ")





**Exercise – User input**

In this exercise, we will ask the user for his/her first and last name, and date of birth, and print them out formatted.

Note: There are two functions to get user input. The first, **raw\_input**, turns whatever the user inputs into a string automatically. The second, **input**, preserves type. So, if the user inputs an int, or a float, you will get an int or a float (rather than a string). Be careful though-you still want to use raw\_input if you want a string back, or otherwise the user will have to put quotes around their answer.

Here is an example of what this program should do:

Output:

Enter your first name: Chuck

Enter your last name: Norris

Enter your date of birth:

Month? March

Day? 10

Year? 1940

Chuck Norris was born on March 10, 1940.

To print a string and a number in one line, you just need to separate the arguments with a comma (this works for any two types within a print statement). The comma adds a space between the two arguments. For example, the lines:

mo = ’October’

day = ’20’

year = ’1977’

print mo, day, year

will have the output

October 20 1977

Note that none of the commas are in this output! To do that you want something like this:

print mo, day+’,’, year.

The + sign concatenates two strings, but can only be used on two strings. Using it on a number and a string will cause an error (because it is ambiguous as to what you want the program to do!)

That’s why it’s a great idea to use raw\_input for this problem; if you use input you’d have to convert the int to a string.

Ans:

Fname = input("Enter your first name: ")

Lname = input(“Enter your last name: ")

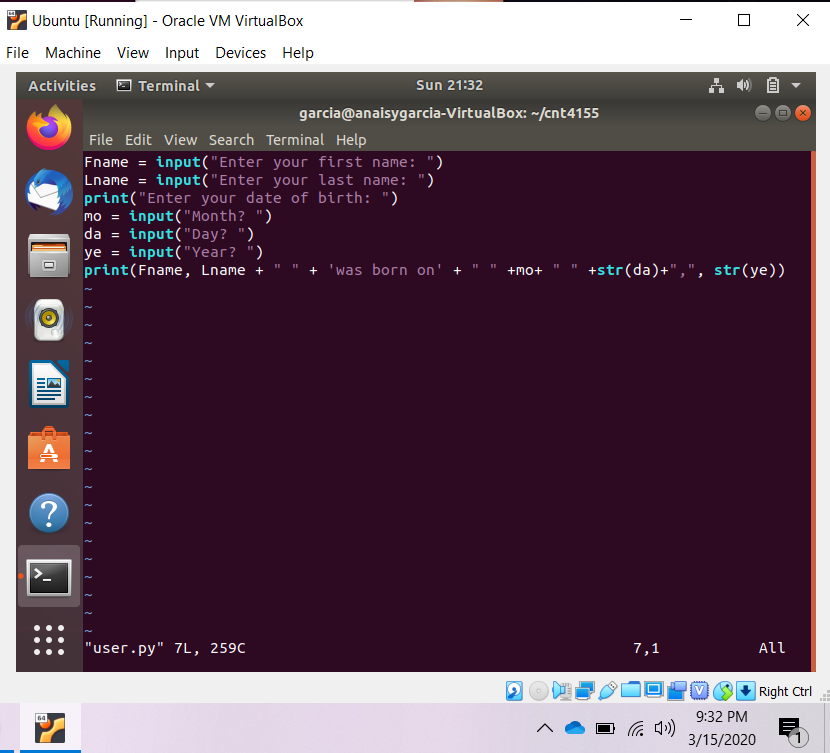
print(“Enter your date of birth: “)

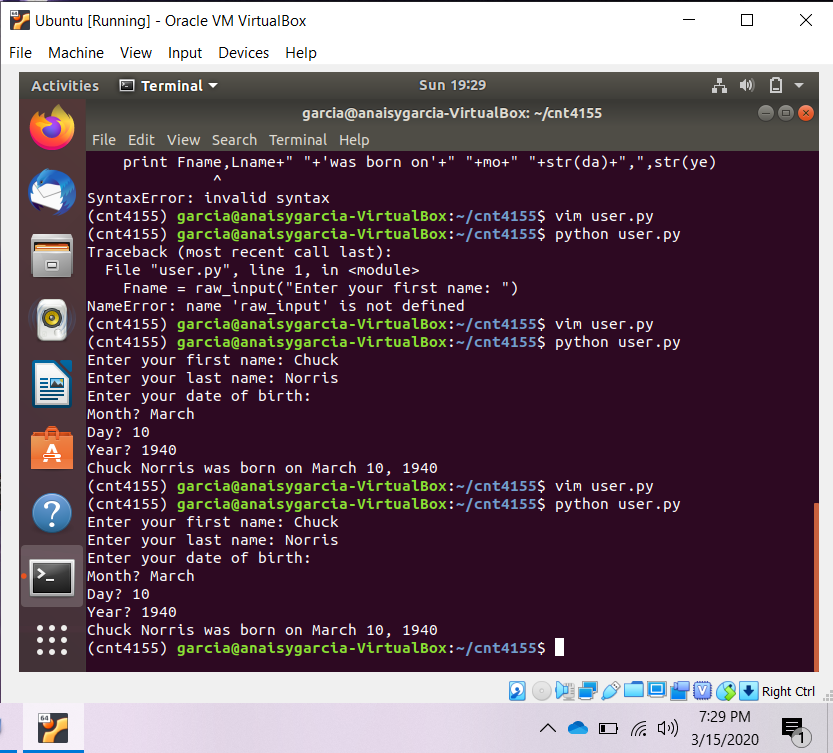
mo=input("Month? ")

da=input('Day? ')

ye=input("Year? ")

print(Fname, Lname + " " + 'was born on' + " " +mo+ " " +str(da)+",",str(ye))





**Exercise – Find The Bug!**

Consider the following program that Juan handed in to the course staff (again, try to do this exercise without running the code):

n = 10

i = 10

while i > 0:

print i

if i % 2 == 0:

i=i/ 2

else:

i=i+1

What do you think this code is doing? Without comments it is hard to guess what Juan’s intention was, so read through it and make a sensible guess as to what it is doing. There’s a lot of mistakes in the code so your guess is as good as ours!

1. Draw a table that shows the value of the variables n and i during the execution of the program. Your table should contain two columns (one for each variable) and one row for each iteration. For each row in the table, write down the values of the variables as they would be at the line containing the print statement.

Ans:

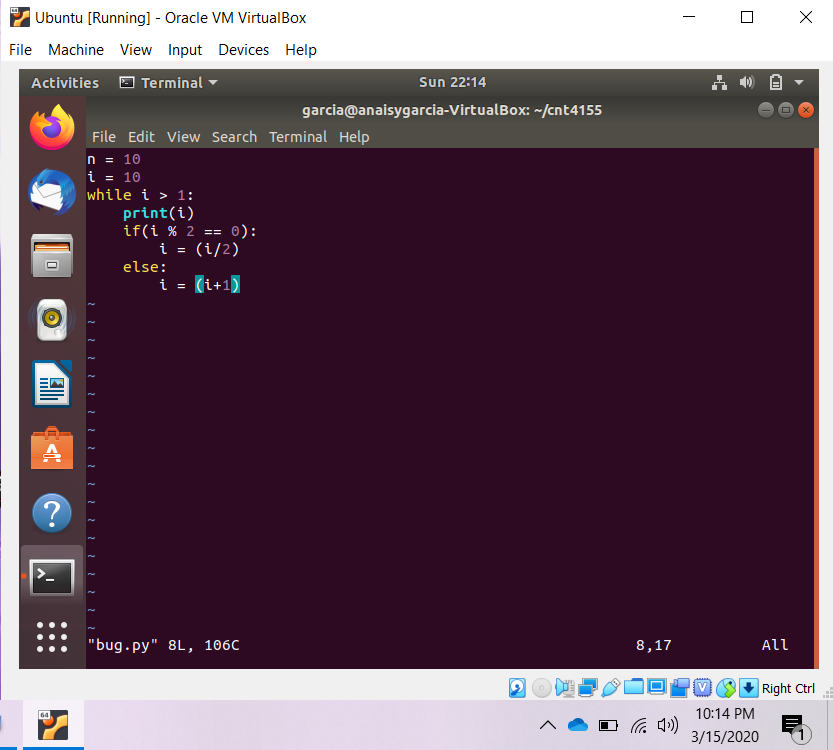
|  |  |  |
| --- | --- | --- |
| i | n | output |
| 10 | 10 | 10 |
| 5 | 10 | 5.0 |
| 6 | 10 | 6.0 |
| 3 | 10 | 3.0 |
| 2 | 10 | 4.0 |
| 1 | 10 | 2.0 |

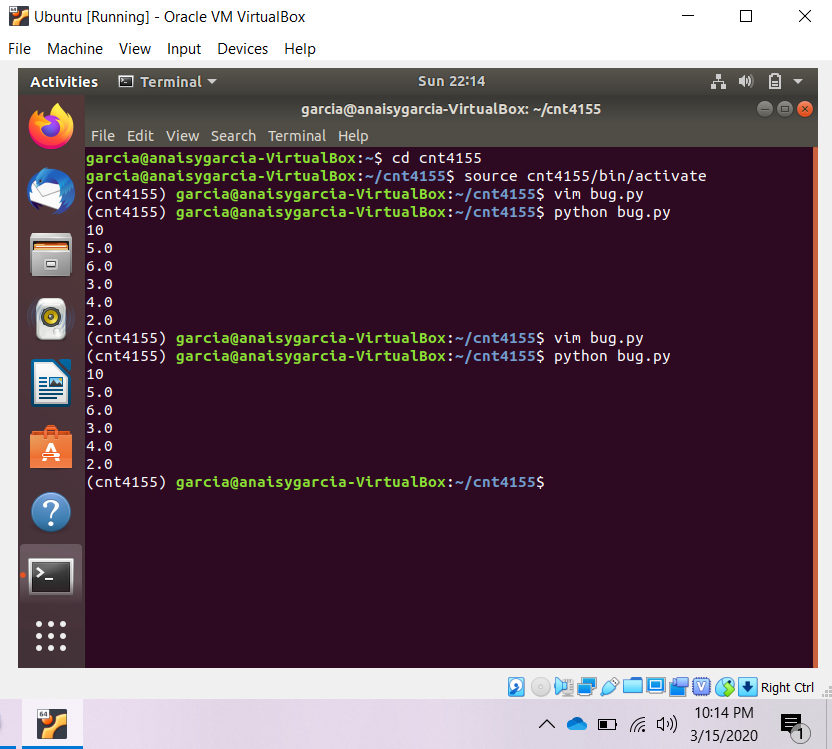
1. Juan made a lot of mistakes. State what you think Juan was trying to do and suggest one or more ways he could fix his code (there’s a few good answers for this depending on what you think the code should be doing).

Ans:

Juan is creating a program that checks if a number is even then divides by two until the number is zero. If there is a negative number it increases by 1.

My suggestion to fix this code is:





**Exercise – Writing Simple Methods**

In this problem you’ll be asked to write a few simple methods (method is an interchangeable term for ‘function’). Be sure to test your functions well, including at least 3 test cases for each method.

1. Write a method is\_divisible that takes two integers, m and n. The method returns True if m is divisible by n, and returns False otherwise. Test cases for this function to verify it functions correctly.

Ans:

def is\_divisible(m, n):

    if(m % n == 0):

           return True

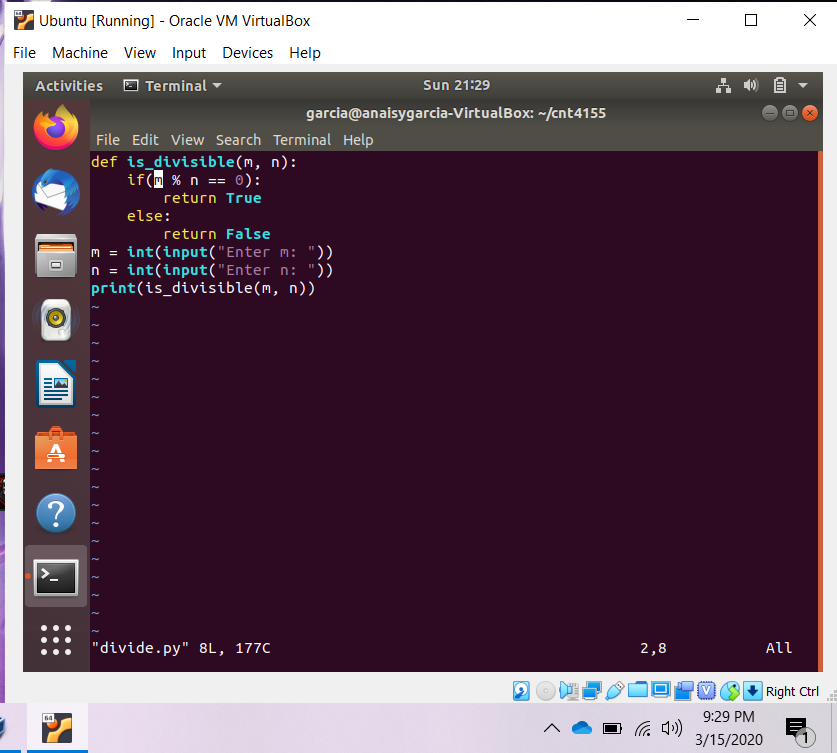
    else:

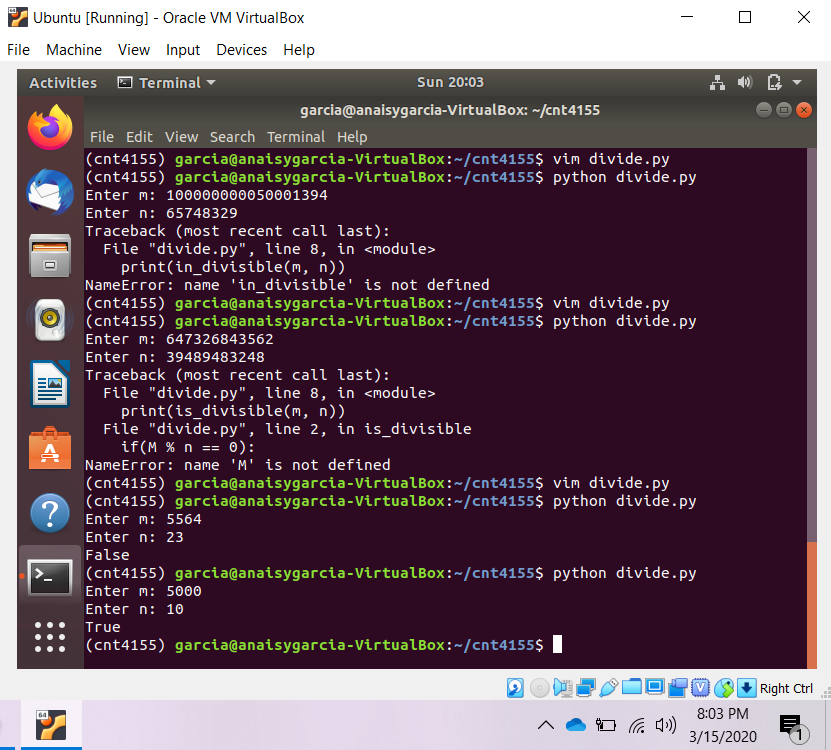
           return False

m = int(input("Enter m: "))

n = int(input("Enter n: "))

print(is\_divisible(m, n))





2. Imagine that Python doesn’t have the != operator built in. Write a method not equal that takes two parameters and gives the same result as the != operator. Obviously, you cannot use != within your function! Test if your code works by thinking of examples and making sure the output is the same for your new method as != gives you.

Ans:

def not\_equal(x, y):

if(x == y):

return False

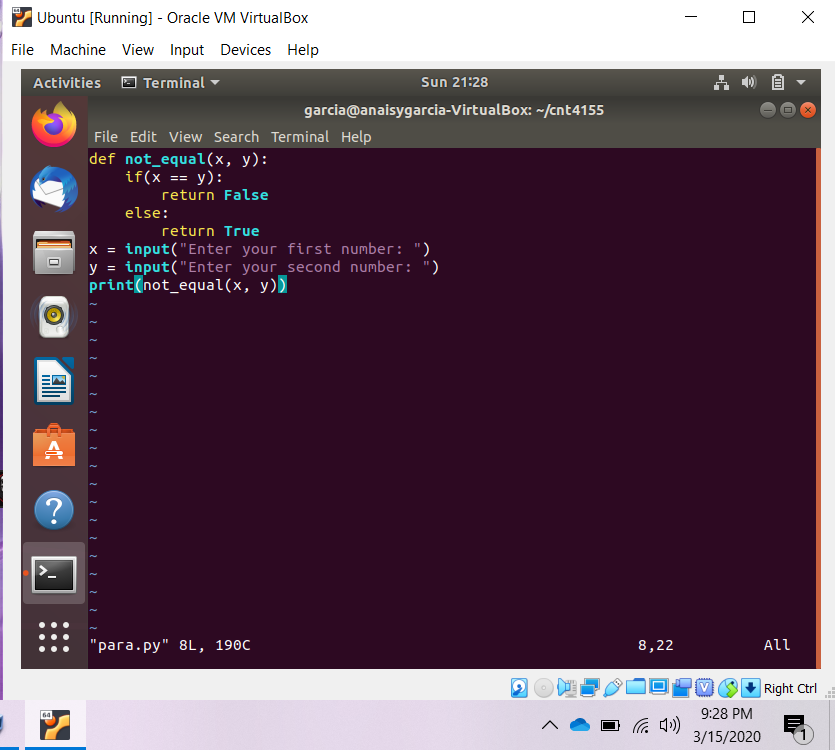
else:

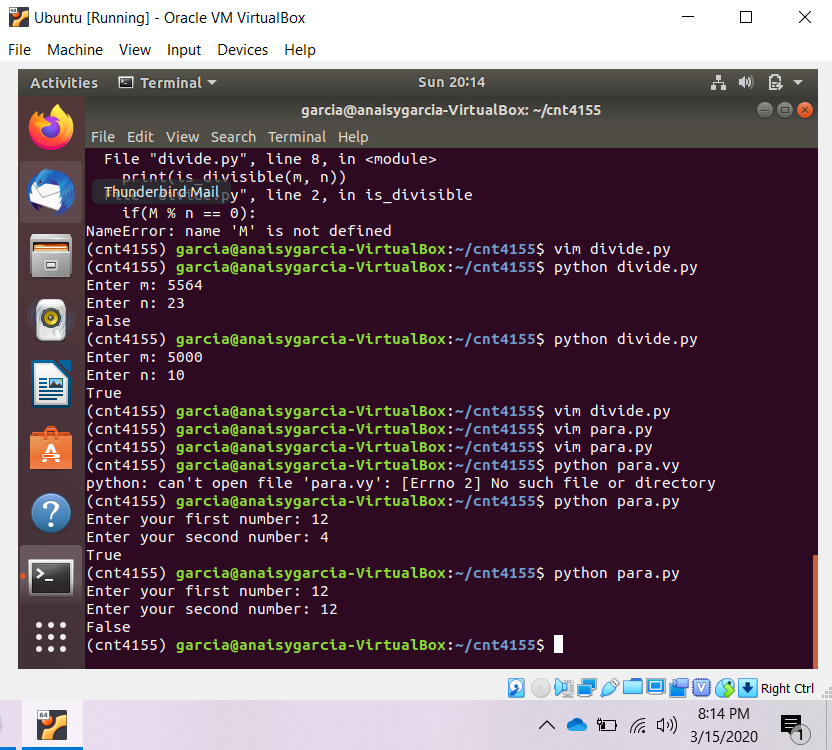
return True

x = input("Enter your first number: ")

y = input("Enter your second number: ")

print(not\_equal(x, y))





3. Write a method rand\_divis\_3 that takes no parameters, generates and prints a random number, and finally returns True if the randomly generated number is divisible by 3, and False otherwise. For this method we’ll use a new module, the random module. At the top of your code, underneath import math, add the line import random. We’ll use this module to generate a random integer using the function randint, which works as follows:

random.randint(lo, hi)

where lo and hi are integers that tell the code the range in which to generate a random integer (this range is inclusive). 0 to 100 is probably a decent range

Ans:

import random

def rand\_divis\_3():

x = random.randint(0,100)

print(x)

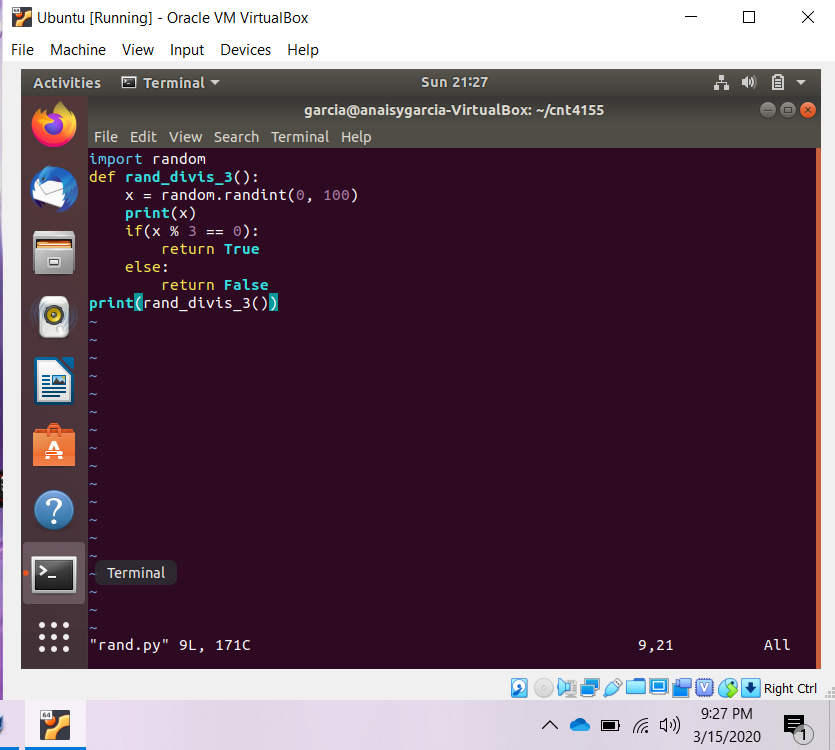
if(x % 3 == 0):

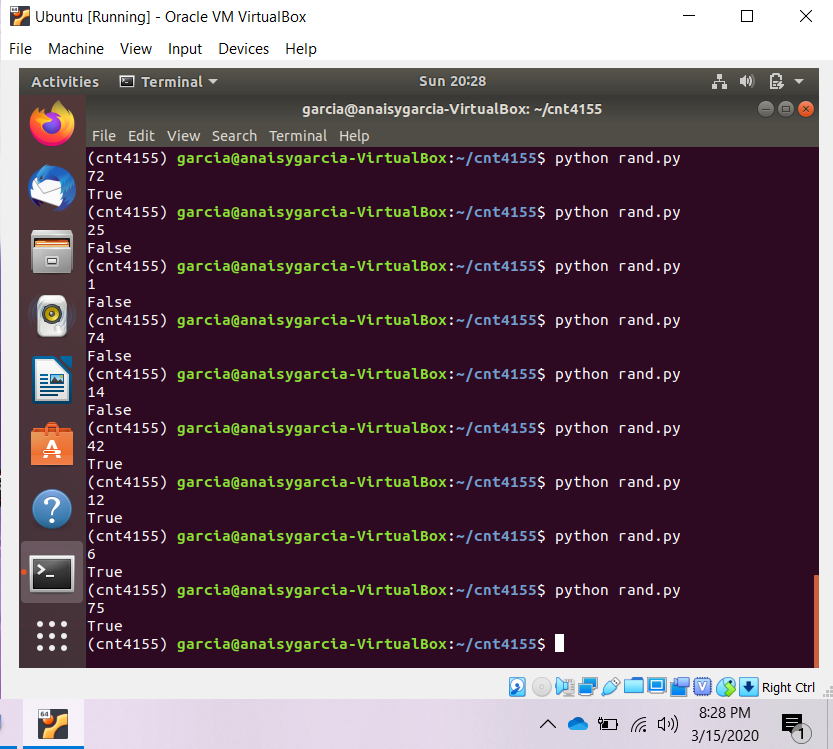
return True

else:

return False

print(rand\_divis\_3())





4. Write a method roll dice that takes in 2 parameters -the number of sides of the die, and the number of dice to roll -and generates random roll values for each die rolled. Print out each roll and then return the string “That’s all!” An example output:

>>> roll\_dice(6, 3)

4

1

6

That’s all!

Ans:

import random

def roll\_dice(x, y):

for I in range(0, y, 1):

ran = random.randint(1, x)

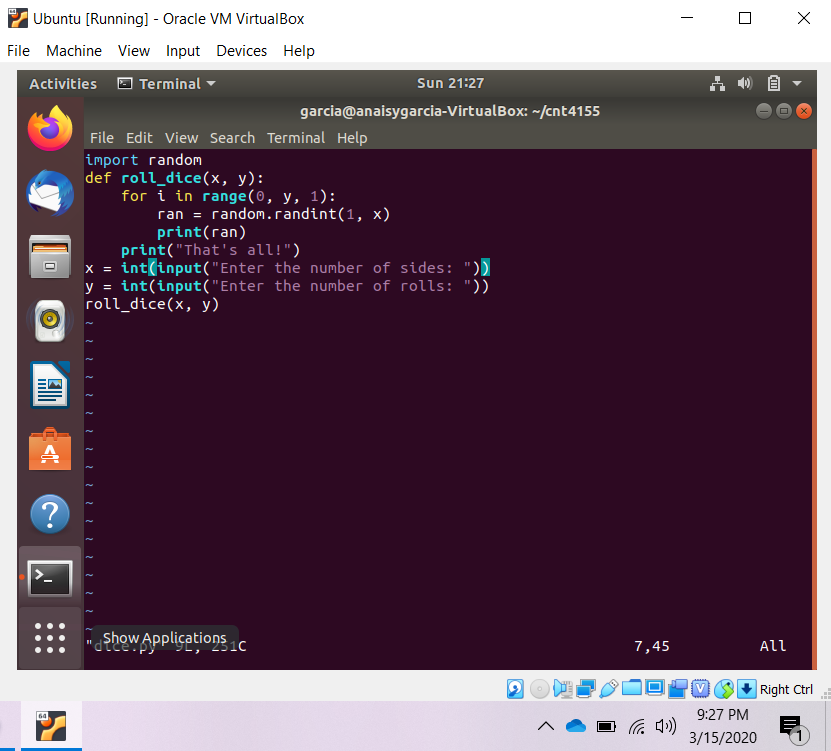
print(ran)

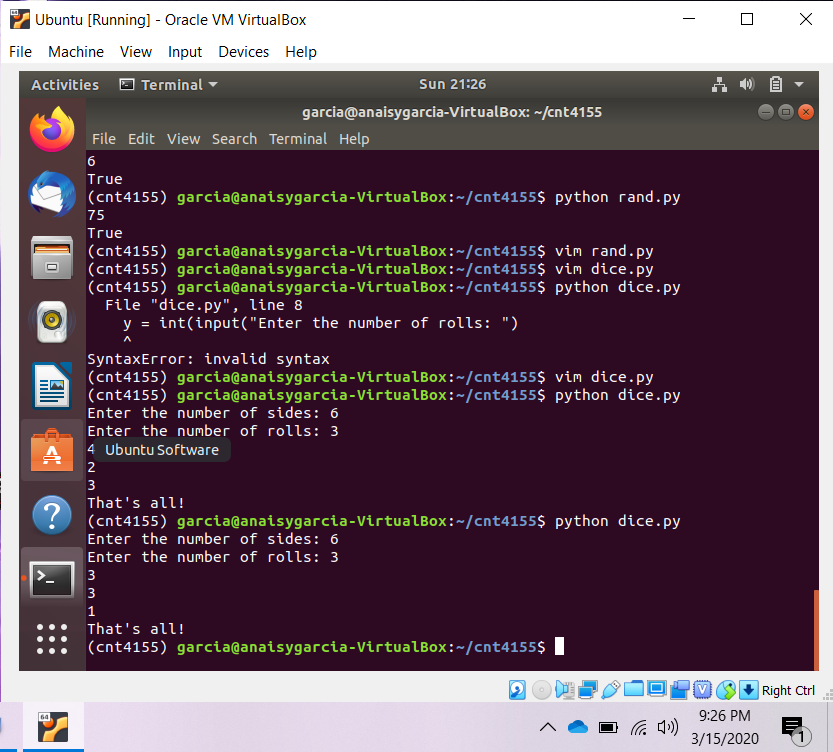
print(“That’s all!”)

x = int(input(“Enter the number of sides: “))

y = int(input(“Enter the number of rolls: “))

roll\_dice(x, y)





1. Now make a new function cumulative sum that modifies sum all so that instead of returning the sum of all the elements, it 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 [4, 3, 6] is [4, 7, 13].

Ans:

def mod(x):

x[0] = int(x[0])

For i in range(0, len(x)-1):

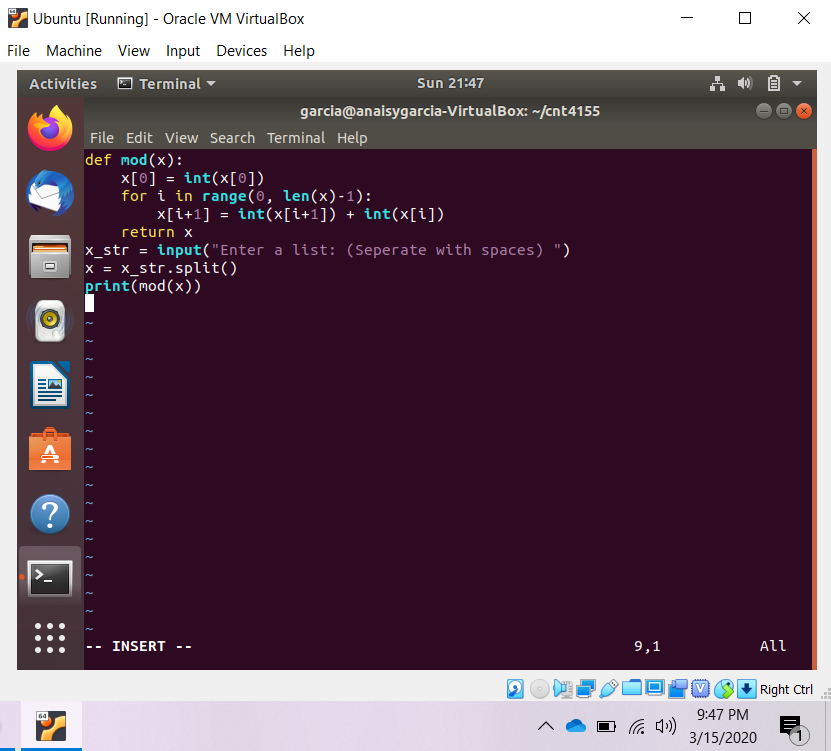
x[i+1] = int(x[i+1]) + int(x[i])

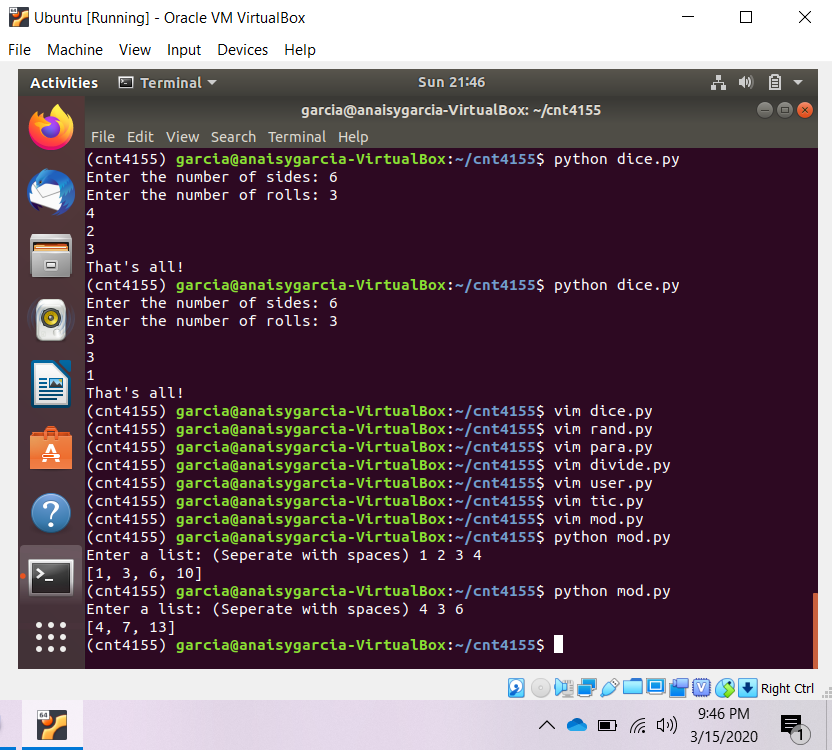
return x

x\_str = input(“Enter a list: (Separate with spaces)”)

x = x\_str.split()

print(mod(x))





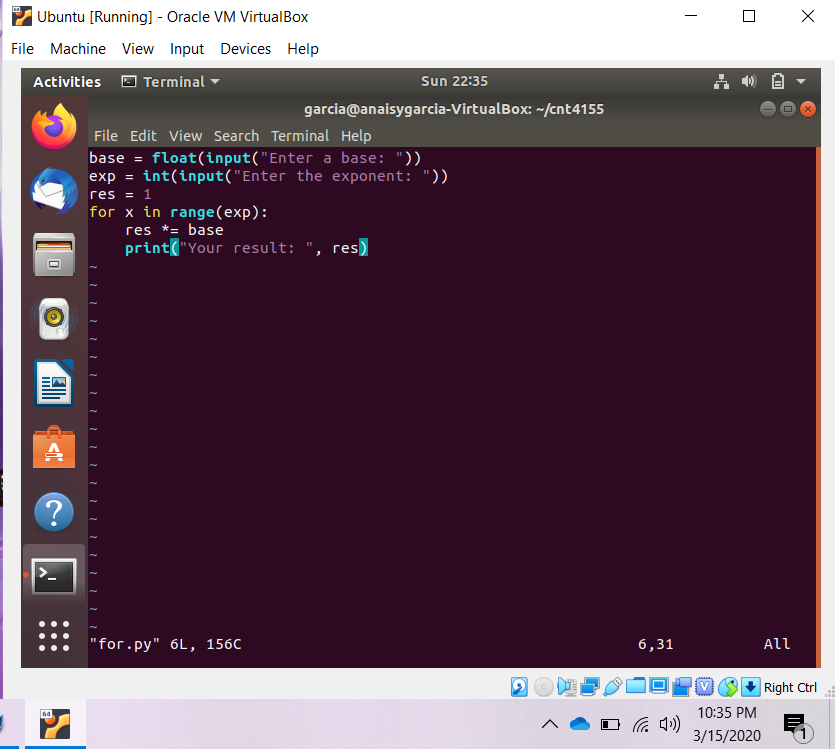
**Exercise – For & While Loops**

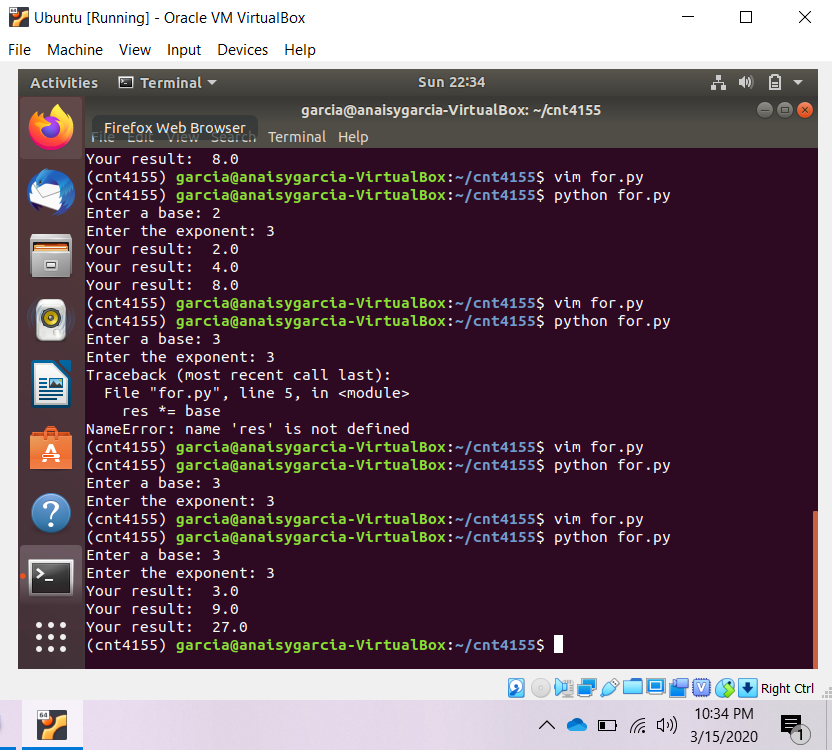
Create a new file called loops.py and use it for all parts of this exercise. Remember the difference between input and raw input?

Be sure to test your code for each part before moving on to the next part.

1. Write a program using a for loop that calculates exponentials. Your program should ask the user for a base base and an exponent exp, and calculate baseexp.

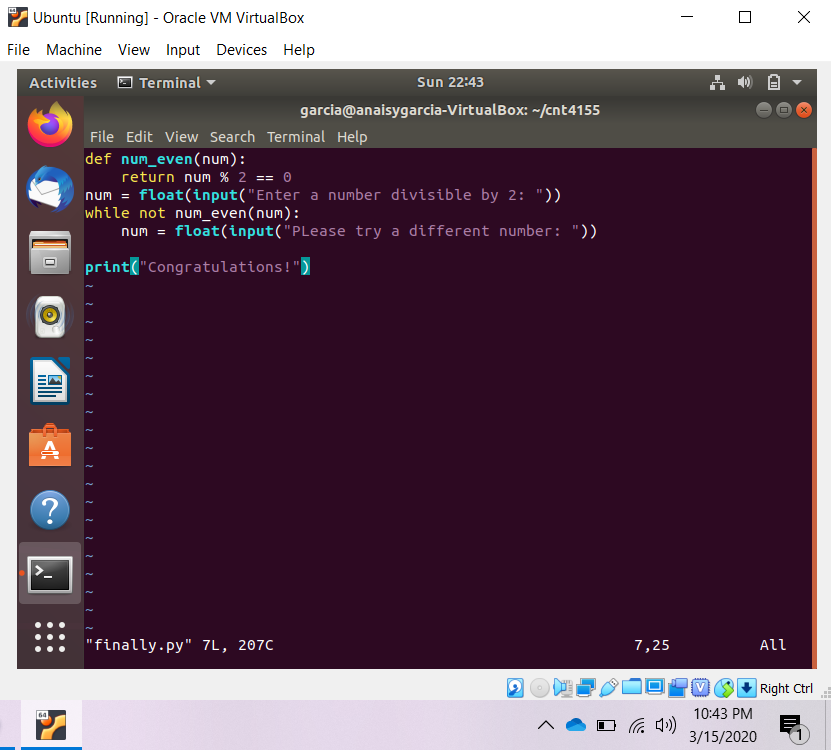
Ans:

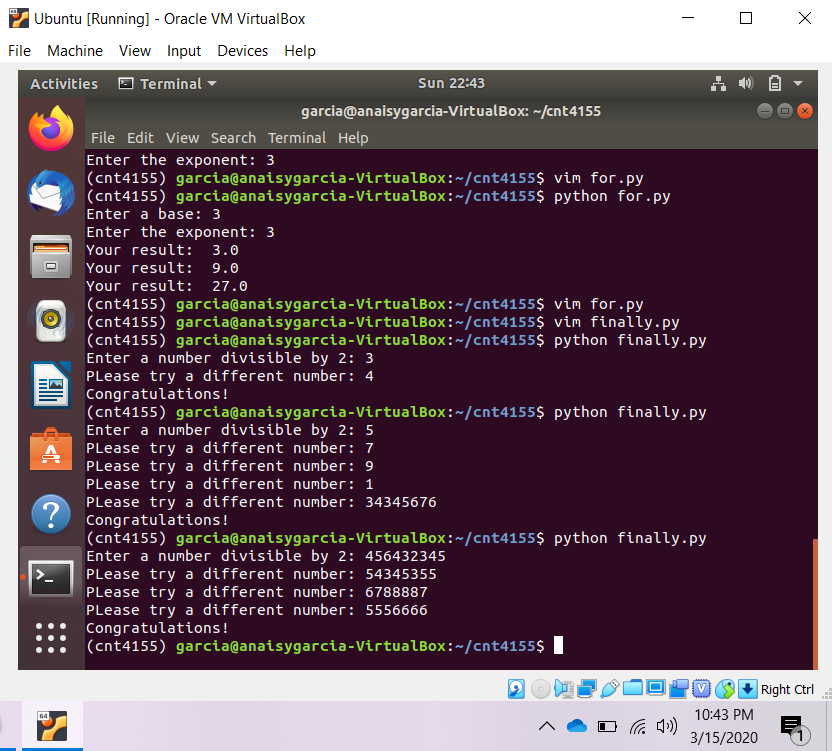




1. Write a program using a while loop that asks the user to enter a number that is divisible by 2. Give the user a witty message if they enter something that is not divisible by 2-and make them enter a new number. Don’t let them stop until they enter an even number! Print a congratulatory message when they \*finally\* get it right.

Ans:





**Exercise Main – Secret Messages**

The goal of this exercise is to write a cyclic cipher to encrypt messages. This type of cipher was used by Julius Caesar to communicate with his generals. It is very simple to generate but it can actually be easily broken and does not provide the security one would hope for.

The key idea behind the Caesar cipher is to replace each letter by a letter some fixed number of positions down the alphabet. For example, if we want to create a cipher shifting by 3, you will get the following mapping:

Plain: ABCDEFGHIJKLMNOPQRSTUVWXYZ

Cipher: DEFGHIJKLMNOPQRSTUVWXYZABC

To be able to generate the cipher above, we need to understand a little bit about how text is represented inside the computer. Each character has a numerical value and one of the standard encodings is ASCII (American Standard Code for Information Interchange). It is a mapping between the numerical value and the character graphic. For example, the ASCII value of ’A’ is 65 and the ASCII value of ’a’ is 97. To convert between the ASCII code and the character value in Python, you can use the following code, also remember that python stores strings in Unicode:

letter = ’a’

# converts a letter to ascii code ascii\_code = ord(letter)

# converts ascii code to a letter letter\_res = chr(ascii\_code)

print ascii\_code, letter\_res

Create a file called cipher.py. Start your program by asking the user for a phrase to encode and the shift value. Then begin the structure of your program by entering in this loop:

encoded\_phrase = ’’

for c in phrase:

encoded\_phrase = encoded\_phrase + c

Hint: Note that the ASCII value of ’A’ is 65 and ’a’ is 97, not 0. So you will have to think how to use the modulus operator to achieve the desired result. Apply the cipher separately to the upper and lower case letters.

Here is what you program should output:

Enter sentence to encrypt: Mayday! Mayday!

Enter shift value: 4

The encoded phrase is: Qechec! Qechec!

When you are done, print a copy of the file and turn it in.

Ans:

