**Homework 5**

**Jamie Andrews**

**Problem 1:**

#-----------Homework 6 Problem 1------------

#1. Import turtle before beggining this problem.

#2. Create a def function called draw\_life in which draws out the binary tree.

#3. Create if/else statement that if for else the program draws the binary tree.

#4. Create several statements in which it prints the binary tree.

#5. Create a def function called power in which computes exponents.

#6. Create if/else statement that helps compute the exponents for every condition.

#7. Print out the exponents.

#8. Create a def function called slice\_sum in which the program computes the sum recursively.

#9. Create an if/else statement to perform the task.

#10. Create two lists to correspond to the def function.

#1.

import turtle

#2.

def draw\_life(length,depth):

#3.

if depth==0:

return

else:

t.fd(length/2)

#left side of turtle

draw\_life(length/2,depth-1)

t.bk(length/2)

t.left(60)

t.fd(length/2)

t.right(60)

#right side of turtle

draw\_life(length/2,depth-1)

t.left(60)

t.bk(length/2)

t.right(60)

return

#4.

window = turtle.Screen()

t = turtle.Turtle()

t.color("green")

t.right(120)

draw\_life(160,5)

window.exitonclick()

#5.

def power(x,n):

#6.

if n == 0:

return 1

elif n == 1:

return x

elif n == 2:

return x \* x

elif n % 2 != 0:

return x \* power(x, n-1)

elif n % 2 == 0:

return power(x,n//2) \* power(x,n//2)

#7.

i = power(2,3)

print(i)

#8.

def slice\_sum(lst, begin, end):

#9.

if end==0:

return 0

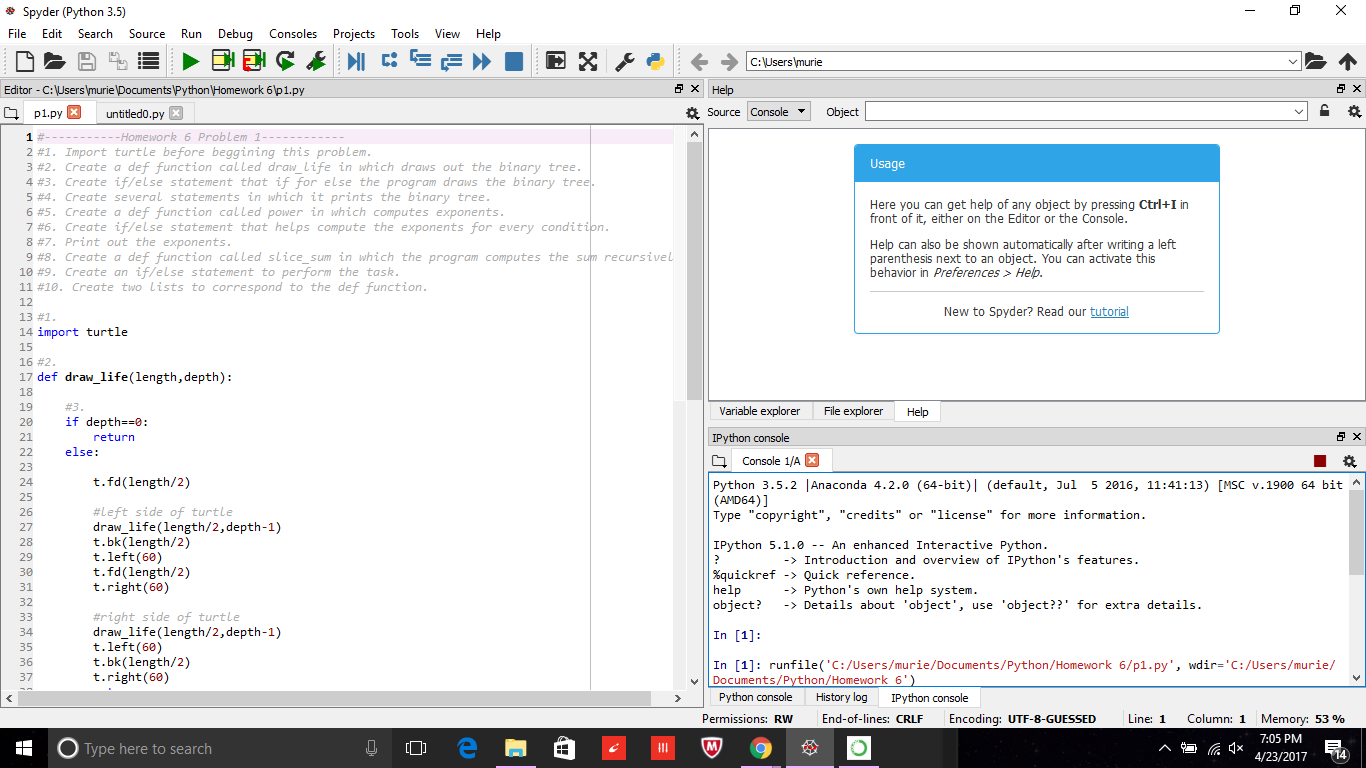
else:

return lst[begin] + slice\_sum(lst,begin+1,end-1)

#10.

X=[0,1,2,3,4,5]

i = [3,2,6,2,1]



**Problem 2:**

# ----------Homework 6 Problem 2--------------

#1. Import itertools before beggining part A

#2. Create class called PrimeIter in which manages the prime integers.

#3. Begin Part A by building the constuctor in the class.

#4. Create a def function in the Class PrimeIter called \_\_next\_\_ which returns the next prime number.

#5. Create a def function in the Class PrimeIter called \_\_iter\_\_ which iterates the prime number.

#6. Begin Part B by Creating another class called PrimeGen which generates prime numbers.

#7. Create a constructor in the class PrimeGen.

#8. Create a def function called genPrime in the class PrimeGen which generates prime numbers.

#9. Create an if statement called \_\_name\_\_ which calles out the 2 classes.

#1.

import itertools

#2.

class PrimeIter:

#3.

def \_\_init\_\_(self):

self.current = 1

#4.

def \_\_next\_\_(self):

self.current = self.current + 1

while 1:

for i in range(2, self.current//2 + 1):

if self.current % i == 0:

self.current = self.current + 1

break # Break current for loop

else:

break # Break the while loop and return

return self.current

#5.

def \_\_iter\_\_(self):

return self

if \_\_name\_\_ == '\_\_main\_\_':

p = PrimeIter()

for x in itertools.islice(p, 10):

print (x)

#6.

class PrimeGen:

#7.

def \_\_init\_\_(self):

self.current = 2

#8.

def genPrime(self, num):

for i in range(num):

while 1:

for j in range(2, self.current//2 + 1):

if self.current % j == 0:

self.current = self.current + 1

break

else:

break

print (self.current)

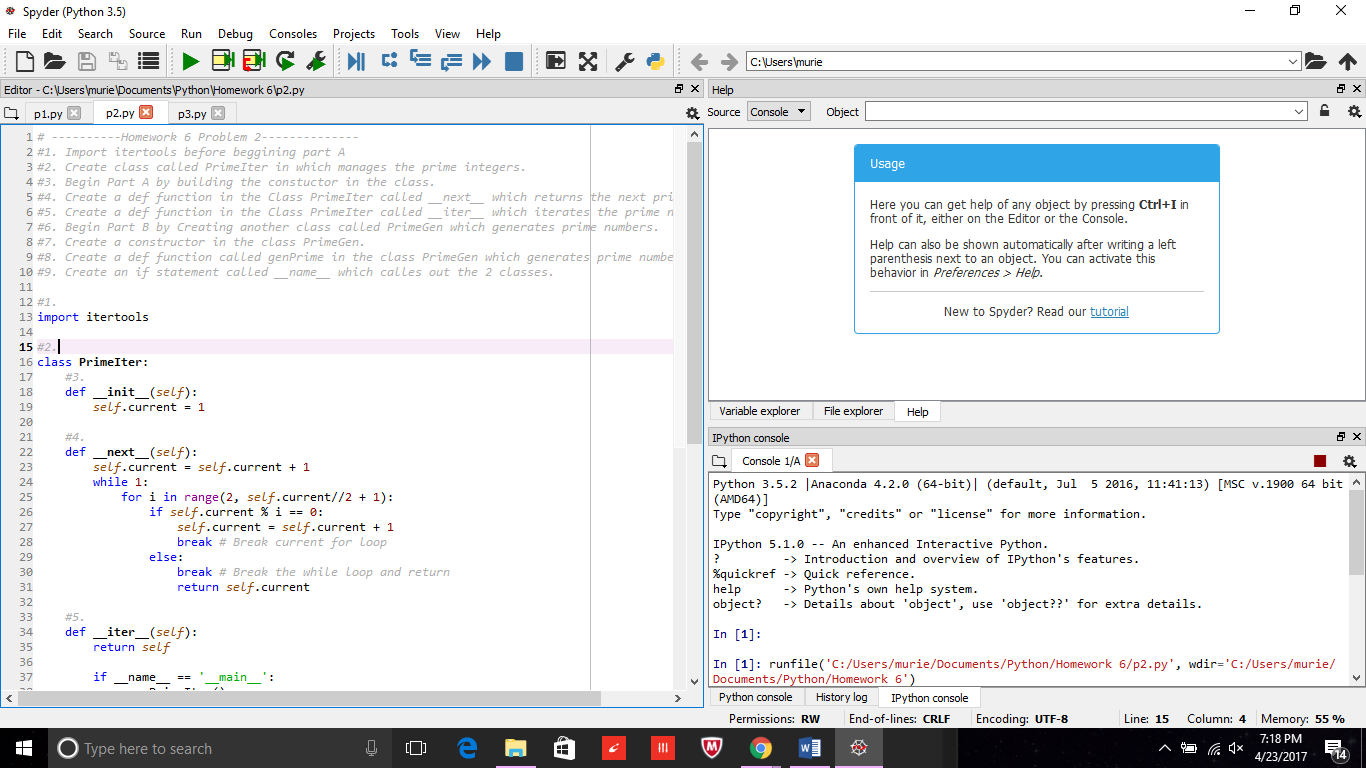
self.current = self.current + 1

#9.

if \_\_name\_\_ == '\_\_main\_\_':

p = PrimeGen()

p.genPrime(10)



**Problem 3:**

#------------Homework 6 Problem 3 --------------

#1. Import the following libaries before doing part A.

#2. Create a def function called gen\_rndtup(n) in which generates random sequence.

#3. Write a while loop that for when true it yeilds the random integer.

#4. Create a def function called answer\_x in which it prints out the answer for the first tuple.

#5. Create a def function called answer\_y in which it prints out the answer for the second tuple.

#6. Create a def function called answer\_z in which it prints out the answer for the third tuple.

#7. Create an if statement called \_\_main\_\_ that calls out def functions answer\_x, answer\_y, and answer\_z

#1.

from itertools import islice

import random

from functools import reduce

#2.

def gen\_rndtup(n):

"""

This generate infinite sequence of tuple(x,y) where 0 < x, y < n

:param n:

:return:

"""

#3.

while True:

yield (random.randint(1, n - 1), random.randint(0, n - 1))

#4.

def answer\_x():

print("Answer x")

n = 7

# create object for generator.

generator\_obj = gen\_rndtup(n)

# use the islice function to obtain 10 tuples

islice\_object = islice(generator\_obj, 10)

# make the filter function using lambda for retrieving tuple like a+b>n/2

filter\_obj = filter(lambda x: x[0] + x[1] > n // 2, islice\_object)

# use start (\*) operator to unpack sequence & print the tuples

print(\*filter\_obj)

#5.

def answer\_y():

print("Answer y")

n = 7

generator\_obj = ((random.randint(1, n - 1), random.randint(0, n - 1)) for i in range(10))

for x in generator\_obj:

if x[0] + x[1] > n // 2:

print(x, end=" ")

print()

#6.

def answer\_z():

print("Answer z")

n = 7

map\_obj = map(lambda x: (random.randint(1, n - 1), random.randint(0, n - 1)), range(10))

filter\_obj = list(filter(lambda x: x[0] + x[1] > n // 2, islice(map\_obj, 10)))

print(filter\_obj)

sum\_of\_tuples = reduce(lambda x, y: (x[0] + y[0], x[1] + y[1]), filter\_obj)

print(sum\_of\_tuples)

#7.

if \_\_name\_\_ == '\_\_main\_\_':

answer\_x()

answer\_y()

answer\_z()

