**Homework 2**

**Jamie Andrews**

**Problem 1:**

1. public Group <Triple> by AWhileLoop () {

Group <Triple> result = new ArrayList <Triple> (limit);

for (int a = 1; a < limit; ++a) {

int aa = a \* a;

int b = a + 1;

int c = b + 1;

while (c <= limit) {

int cc = aa + b \* b;

while (c \* c < cc) {++c;}

if (c \* c == cc && c <= limit) {

result.add (new Triple (a, b, c));

}

++b;

}

}

return result;

}

1. #--------Homework 2 Question 1 by Jamie Andrews --------

#1. Make sure to import numpy as np before typing any code.

#2. Define a def funtion that generates Pythagorean triples, but only gives out the primitive triples.

#3. Define 3 functions np.mat function to set up the Pythagorean numbers.

#4. Call an array function and set array with the 3 np.mat functions.

#5. Call another array function and set array with integers 3, 4, and 5.

#6. Create a while loop in the def function that generates the numbers used in the Pythagorean Theorem.

#7. Create an if statement in the while loop when if is the limit.

#8. Create a yield statement in the while loop function derived from n.

#9. Create a second def function that generates all Pythagorean triples.

#10. Create a for loop in the def function that takes all the primitive triples from the first def function

#11. Create another for loop in the for loop using in range and it should add all the other triples aside from the primitives.

#12. Outside of the def functions, create 2 print statements that prints out a list of the 2 def functions using an exponent value by the 10th power.

# 1.

import numpy as np

# 2.

def gen\_prim\_trips(limit = None):

# 3.

a = np.mat(' 1 2 2; -2 -1 -2; 2 2 3')

b = np.mat(' 1 2 2; 2 1 2; 2 2 3')

c = np.mat('-1 -2 -2; 2 1 2; 2 2 3')

# 4.

abc = np.array([a, b, c])

# 5.

n = np.array([3, 4, 5])

# 6.

while n.size:

n = n.reshape(-1, 3)

# 7.

if limit:

n = n[n[:, 2] <= limit]

# 8.

yield from n

n = np.dot(n, abc)

# 9.

def gen\_all\_trips(limit):

# 10.

for prim in gen\_prim\_trips(limit):

i = prim

# 11.

for \_ in range(limit//prim[2]):

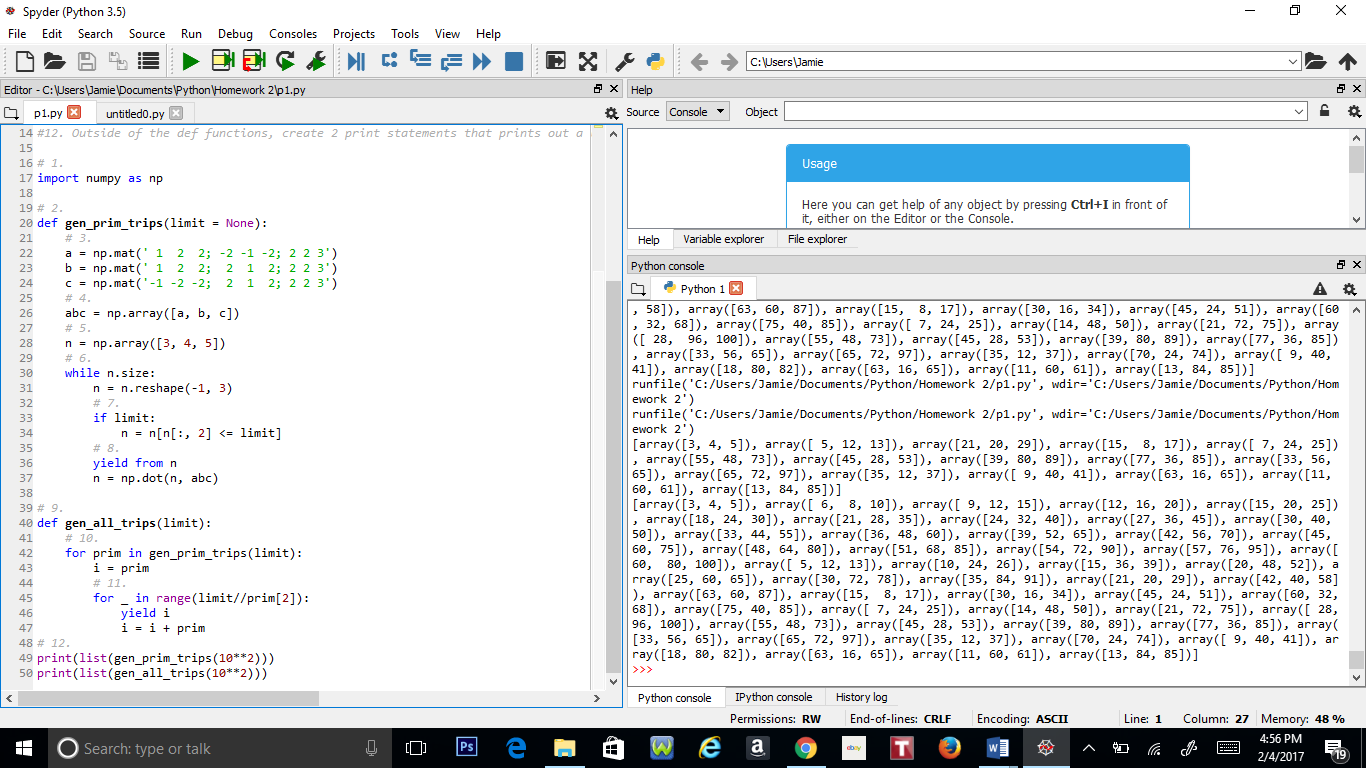
yield i

i = i + prim

# 12.

print(list(gen\_prim\_trips(10\*\*2)))

print(list(gen\_all\_trips(10\*\*2)))

****

**Problem 2:**

1. Public Group<Duplicate> by AWhileLoop(s, n){

a = 0;

length = length(s);

for(int = i; i < length; ++i){

for(int = j; j < length; ++j){

if i != j {

if s(i, i + n) == s(j, j + n){

a = s(i, i + n);

return a;

}

}

}

}

}

1. Public Group<Max\_Duplicate> by AWhileLoop(s){

b = length(s) – 1;

while (b >=0){

c = string’ ’;

if c != 0{

for(int = i; i < length; ++i){

for(int = j; j < length; ++j){

if i != j {

if s(i, i + n) == s(j, j + n){

a = s(i, i + n);

return a;

}

}

}

}

}

else{

b = b – 1;

return c;

}

}

}

1. #------- Homework 2 Problem 2 by Jamie Andrews -------

# --------------------Part A-------------------------

#1. Create a def function that finds the duplicate string.

#2. Assign a variable to 0.

#3. Create a nested for loop that searches for anything in range within the string.

#4. Create a nested if statement that finds the duplicate in the string and returns the duplicate.

#1.

def find\_dup\_str(s, n):

#2.

a = 0

#3.

for i in range(len(s)-1):

for j in range(len(s)-1):

#4.

if i != j:

if s[i:i+n] == s[j:j+n]:

a = s[i:i+n]

return a

#--------------------Part B--------------------------

#1. Create a def function that finds the most occurring duplicate.

#2. Set a variable and assign it to be the length of the string.

#3. Create a while loop stating that the variable is greater than 0.

#4. In the while loop assign a variable that is equal to an empty set.

#5. Create an if/else statement for what happens if c has duplicates and does not equal to 0.

#6. Repeat steps 3 & 4 in Part A.

#7. If else, then return the empty set.

#8. Declare 2 separate strings, one for each def function and length number.

#9. Print out the 2 def functions.

#1.

def find\_max\_dup(s):

#2.

b = len(s) - 1

#3.

while b >= 0:

#4.

c = set()

#5.

if c != 0:

#6.

for i in range(len(s)-1):

for j in range(len(s)-1):

if i != j:

if s[i:i+n] == s[j:j+n]:

a = s[i:i+n]

return a

#7.

else:

b = b - 1

return c

#8.

s = 'abcdefbcdgh'

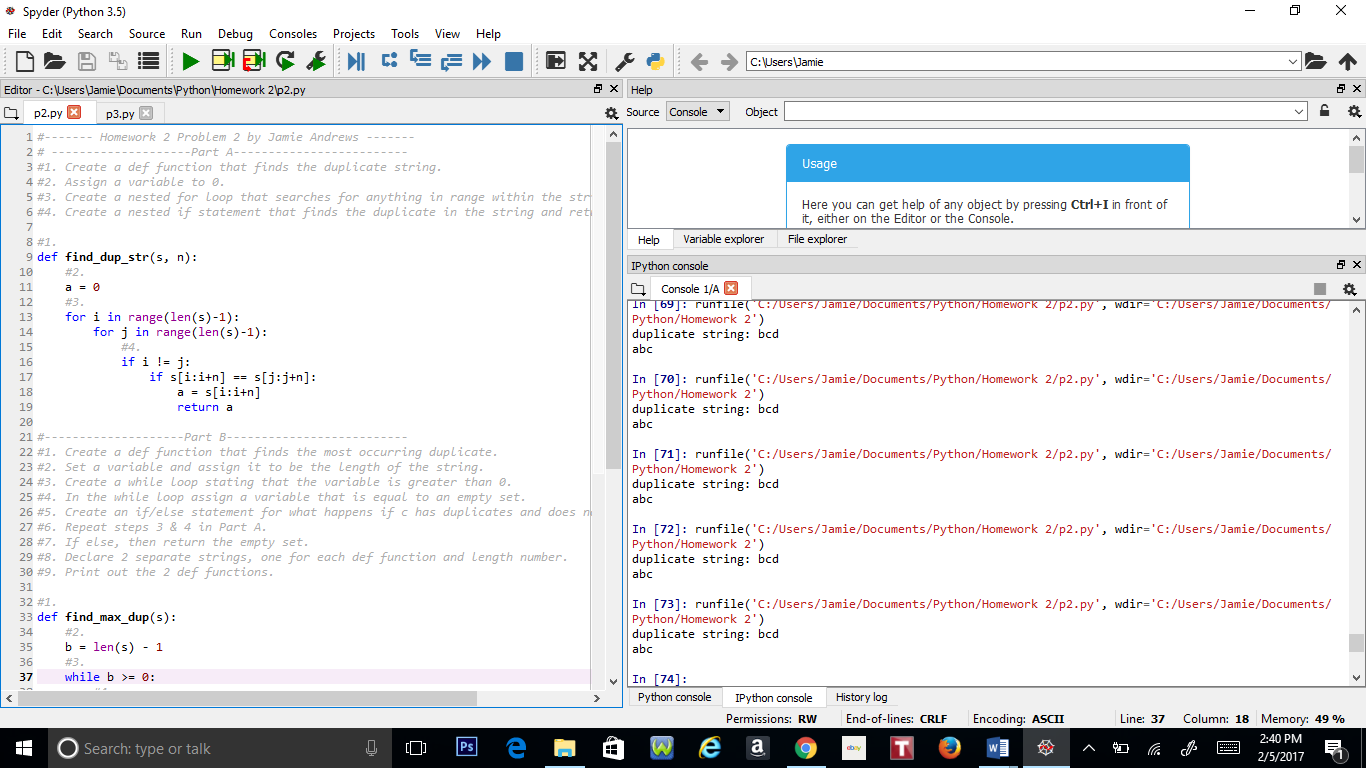
d = 'abcdefgheabcd'

n = 3

#9.

print("duplicate string:", find\_dup\_str(s,n))

print(find\_max\_dup(d))



**Problem 3:**

#-------Homework 2 Question 3 by Jamie Andrews-------

#1. Import math and pylab before typing any code.

#2. Declare 2 variables and set is as arrays.

#3. Declare 4 variables and set it as inputs for the function, sample number, max and min of x.

#4. Declare a variable that divides the difference of the max of x and the min of x divided by the sample number.

#5. Create a while loop that states that the min of x is greater than or equal to the max of x.

#6. In the loop, append the 2 variables declared in step 2 and assign y for evaluation, then increment the min of x with the variable from step 4.

#7. Set the plot of the graph using pylab and the 2 variables declared in step 2.

#8. Create a for loop and use if statements in the for loop stating the inequalities of xs and ys using 0 as a comparison and

# use a print statement for each inequality.

#9. Make a pylab statement to show the graph.

#1.

import math

import pylab

#2.

xs = []

ys = []

#3.

fun\_str = input("Enter function with variable x:")

n = int(input("Enter a number of samples:"))

x = int(input("Enter xmin:"))

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

#4.

dx = (xmax - x)/n

#5.

while x <= xmax:

#6.

xs.append(x)

y = eval(fun\_str)

ys.append(y)

x += dx

#7.

pylab.plot(xs, ys, "rx-")

#8.

for i in range(n):

if xs[i] >= 0 and ys[i] >= 0:

print('+{:.4f}{:4s}{:.4f}'.format(xs[i],' ',ys[i]))

elif xs[i] >= 0 and ys[i] <= 0:

print('+{:.4f}{:4s}{:.4f}'.format(xs[i],' ',ys[i]))

elif xs[i] <= 0 and ys[i] <= 0:

print('+{:.4f}{:4s}{:.4f}'.format(xs[i],' ',ys[i]))

#9.

pylab.show()

