Name: Alex Barker
ID: U1512136

Assignment 2 Descriptive

A1.1.1

A1.1.1c1

A1.1.2

A1.1.3.1

IN: [x**2/16 - y**2/4 for x in range (0,5) for y in [4,8,12,16,20]]
OUT: [-4.0, -16.0, -36.0, -64.0, -100.0, -3.9375, -15.9375, -35.9375, -63.9375, -99.9375, -3.75, -15.75, -35.75, -63.75, -99.75, -3.4375, -15.4375, -35.4375, -63.4375, -99.4375, -3.0, -15.0, -35.0, -63.0, -99.0]

This works by using x-values from 0 to 5 in steps of 1 by default for the y-values [4,8,12,16,20]. The function's BIDMAS order is quite important, it squares both x and y-values then divides them by 16 and 4 respectively, it then subtracts the y**2/4 value from the x**2/16 value. It selects the order of values as follows. It will run through all the x=0 values against the y=4,8,12... then increase x by the step (not defined so default 1) so therefore, move onto x=1 for y-values of 4,8,12...

for loop equivalent:
List = []
for x in range (0,5):
   for y in [4,8,12,16,20]:
     List.append(x**2/16-y**2/4)
print(List)

A1.1.3.2

A1.1.3.3

The list comprehension given gets the numerical value from each entry which has the first part equal to 'John' for each key. It then sums these and returns the total for every mark that the entry 'John' has.

A1.1.3.4

The duplicates list comprehension uses the pop function to remove values from list 1 and add them to the entry of the created duplicates list. It iterates for each value in list1 checking if that value exists anywhere in list 2. The problem is if it removes a value in the list then increasing the iteration(i) value by 1. This leads to on the next iteration skipping the next value since they have all been shifted to the left and the i value is now not representative of the order which it should be checking.

This could be changed to:
duplicates = [list1.pop(list1.index(i)) for i in list1[::-1] if i in list2]
The "for i in list1[::-1]" reverses the order that the comprehension iterates over, therefore if a value is removed the i values still stand for decreasing tests.

A2.1

A2.2

A2.3

A3.1