Joshua Pollock

CS 126: 2-159

**Problem 5-5**

Step 1:

This problem is asking to find out if a given set of numbers has the same ending digit or not. All the info needed is given and I will be assuming the user will only input numbers. A constraint is that if the numbers end in the same digit True is returned and if they do not have the same ending False is returned

Step 2:

For this I plan to use simple string manipulation using the [:] commands. This is the easiest solution I think will work for this problem. In my code I plane to use the string conversion command ( str()), if statements, else statements, and return commands.

Step 3:

def sameending(n1,n2,n3):

n1=str(n1)

n2=str(n2)

n3=str(n3)

if n1[-1]==n2[-1]:

if n1[-1]==n3[-1]:

return True

else:

return False

else:

return False

Step 4:

This solution seems to be the easiest I can make it. If I wanted to dumb it down even more I could possibly do def sameending(str(n1),str(n2),str(n3)):. I feel this would make the code harder to follow if needed to be changed. This program was a bit difficult to create until I decided to change the inputted numbers into strings.

**Problem 5-6:**

Step 1:

This problem is asking to find the second highest number of a set of 3. This problem doesn’t have many constraints but it must return the second highest value. An assumption will be that the user will only enter in numbers.

Step 2:

This problem is actually really simple. First you may think to use if statements but python has a built in feature for lists. All I need to do is .sort() on a created list.

Step 3:

def secondhighest(n1,n2,n3):

lis=[]

lis.append(n1)

lis.append(n2)

lis.append(n3)

lis.sort()

return lis[1]

Step 4:

This solution I find is the easiest way to solve the problem at hand. Instead of using complicated if statements, this simply creates a list and returns the second highest value with the built in .sort() function. A way to refactor this problem would be finding out how to integrate n1 n2 and n3 in the list without needing the use of the .append() command. This would help to simplify the code even further.

**Problem 5-10:**

Step 1:

This problem is asking me to create a program that can convert a time into the number of seconds that have passed since midnight. A constraint is that the time must be given in seconds. I will be assuming the user will input the time in the proper formatting as given by the problem.

Step 2:

For this I will just use simple algebraic commands and string manipulation for my code. This will hopefully provide the easiest solution. What I am thinking is to do something along the lines of string[0]\*60\*60+string….. and so on.

Step 3:

def timesincemidnight(str):

ans=(int(str[0])\*60\*60)+(int(str[2:4])\*60)+int(str[5:7])

return ans

Step 4:

This solution is really simple. I tried to simplify the math by doing int(str[0])\*360 but that gave out an incorrect solution. I switched it to int(str[0])\*60\*60 and it worked flawlessly. When the code was first ran I did not change the string to a integer, this made the program give a huge string as the return value. This code also depends on the user inputting the time in the format of ‘h:mm:ss’. Without the string in this format the program will return incorrectly.