**Background**

Complete the following exercises. Your answers must be submitted via Canvas. Answers may be typed in this document, handwritten and scanned, or completed in a separate code file. **REGARDLESS OF THE SUBMITTED FORMAT, PLEASE ONLY SUBMIT A SINGLE FILE, WHATEVER THE FORMAT**.

Please keep in mind what this assignment is testing (your ability to use the basic features of Python). As such, your solutions should not be importing libraries and should be implementing the underlying algorithm for each question.

Each question is worth 3 points. There are 27 points available. This assignment is out of 24 points.

**Questions**

1. Write a short Python function, is\_factor(a, b), that takes two integer values and returns True if *a* is a multiple of *b* (that is *b÷a* = *i* for some integer *i*), and False otherwise.

Def is\_factor(a,b):

If(b%a == 0):

Return True

Else:

Return False

1. What parameters should be sent to the range constructor to produce a range with values 11, 8, 5, 2, -1, -4, -7, -10, -13.

range(11,-14,-3)

1. How many lines will be printed by the following loop? Why does it stop where it does?

i = 0

while i < 10:

print(i)

i = i + 1

It prints out 10 lines because the while loop only goes until i>=10 and since I starts at 0 it can loop 10 times

1. Demonstrate how to use Python's **list comprehension syntax** to produce the list [1, 10, 100, 1000, 10000, 100000, 1000000].

list = [10\*\*x for x in range(7)]

1. Write a Python function that takes a sequence of characters and determines if there are any pairs of characters that **are not adjacent to each other** by returning True if there are any pairs, or False otherwise. Example: The string “abbot” would return False because the two b’s are adjacent to each other while the string “abba” would return True because, even though the two b’s are adjacent to each other, the two a’s are not.

def letter\_pairs(a):

b = list(a)

fm = False

for i in range(len(b)):

n=i+2

if (n>len(b)):

n = len(b)

for n in range(i+2, len(b), 1):

if(b[i]==b[n]):

fm = True

break

return fm

1. Demonstrate how to use Python's **list comprehension syntax** to produce the list ['Z','Y', 'X', ..., 'A'], but without having to type out all 26 characters literally and by hand. (HINT: What happens when you type chr(65) or chr(90) into the Python interpreter?)

list = [chr(90-x) for x in range(26)]

1. Write a short Python function, first\_and\_last(data), that takes a sequence of one or more strings, and returns the first and last string lexicographically, in the form of a tuple of length two. Do not use any built-in functions for your comparisons/determination of string ordering.

def first\_and\_last(data):

a = data.split()

first = ""

last = ""

comp = ""

for i in range(len(a)):

if(first==""):

firstStr = a[i]

first = list(a[i])

firstLen = len(a[i])

else:

comp = list(a[i])

compLen = len(a[i])

if (firstLen<compLen):

loopLen = firstLen

else:

loopLen = compLen

for j in range(loopLen):

if first[j] < comp[j]:

break

if first[j] > comp[j]:

first = comp

firstStr = a[i]

firstLen = len(first)

break

if(last==""):

lastStr = a[i]

last = list(a[i])

lastLen = len(a[i])

else:

comp = list(a[i])

compLen = len(a[i])

if (lastLen<compLen):

loopLen = lastLen

else:

loopLen = compLen

for j in range(loopLen):

if last[j] > comp[j]:

break

if last[j] < comp[j]:

last = comp

lastStr = a[i]

lastLen = len(last)

break

final = tuple((firstStr, lastStr))

return final

1. Write a Python function, leap\_year(year), that takes a year number and determines if the year is a leap year. The program returns True if it is, False otherwise. Leap years are divisible by 4 EXCEPT if the year number is divisible by 100 UNLESS the year is also divisible by 400. Example: 1980 is a leap year. 1985 is not a leap year. 1900 is not a leap year. 2000 is a leap year.

def leap\_year(year):

if (year%4==0):

if(year%100==0):

if(year%400==0):

return True

else:

return False

else:

return True

else:

return False

1. Will this loop run infinitely? Or will it stop? Why?

data = [0, 2, 4, 6, 8]

for i in range(len(data)):

data.append(data[i] + 1)

The loop will not run indefinitely because the for loop will only take the length of the data that is originally in the array and doesn’t count it whenever the length of data updates.