**CIS-298 Intro to Python**

**With Professor Robert Mann**

**HW #5**

**Student: Demetrius Johnson**

**11 February 2023**

**Due: 17 February 2023 at 10pm**

# #Author: Demetrius Johnson

#CIS-298 Intro to Python with Professor Robert Mann

#HW5 - Lists, Dictionaries, and strings

#Remember to copy/paste your code for each question separately. Then, follow it with a snippet showing the output.

#1a-1c and 5.1 are to be skipped as we haven't talked about those yet.

#Due: Thu Feb 17, 2023 10:00pm

##----------------------------------------#

# #Question 1:

#Write a program which takes 2 digits, X,Y as input and generates a 2-dimensional array (list containing lists).

#The element value in the i-th row and j-th column of the array should be i\*j.

#Note: i=0,1.., X-1; j=0,1,¡­Y-1.

#Example

#Suppose the following inputs are given to the program:

#3,5

#Then, the output of the program should be:

#[[0, 0, 0, 0, 0], [0, 1, 2, 3, 4], [0, 2, 4, 6, 8]] :

#j 0 1 2 3 4

#[0, 0, 0, 0, 0] i = 0

#[0, 1, 2, 3, 4] i = 1

#[0, 2, 4, 6, 8] i = 2

i = int(input("enter a number for the number of rows in a 2D matrix: "))

j = int(input("enter a number for the number of columns in a 2D matrix: "))

my\_2d\_list = [list(range(j)) for \_ in range(i)]

for row, col in enumerate(my\_2d\_list): #this loop iterates through rows

for col\_val in col: #this loop iterates through columns

col[col\_val] = row \* col\_val

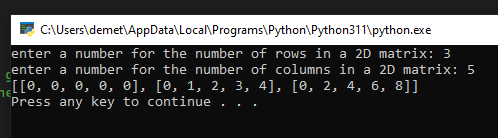
print(my\_2d\_list)

#[Winter 2023: skip 1a-1c]

#1a. modify your code to write the output to a file, once as standard output and again using binary mode. Is there any difference in the files?

#1b. modify your code to read both output files as standard input and as binary. Is there a difference in results from any of the four reads?

#1c. modify your code to write the output to a CSV file, each row on its own line.



A screenshot of a computer

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

A screenshot of a computer

Description automatically generated

##----------------------------------------#

# #Question 2:

#Write a program that accepts a comma separated sequence of words as input and prints the words in a comma-separated sequence after sorting them alphabetically.

#Suppose the following input is supplied to the program:

#without,hello,bag,world

#Then, the output should be:

#bag,hello,without,world

#print what #sort() returns from the string that #input() returns.

print(sorted(input("input a set of words (separated by ','); this program will sort each word alphabetically in ascending order: ").split(',')))

A screenshot of a computer

Description automatically generated

Text

Description automatically generated

# #Question 3:

#Write a program that accepts a sequence of whitespace separated words

#as input and prints the words

#after removing all duplicate words and sorting them alphanumerically.

#Suppose the following input is supplied to the program:

#hello world and practice makes perfect and hello world again

#Then, the output should be:

#again and hello makes perfect practice world

#print what #sorted() returns from the string that #input() returns. #split between ws in string; convert a list

my\_list = sorted(input("input a set of words; this program will sort each word alphabetically in ascending order, removing duplicates: ").split(' '))

print("this is your list of words:",my\_list)

for word in my\_list[:]: #my\_list[:] returns a deep copy of my\_list over the entire range of the list

if my\_list.count(word) > 1:

my\_list.remove(word)

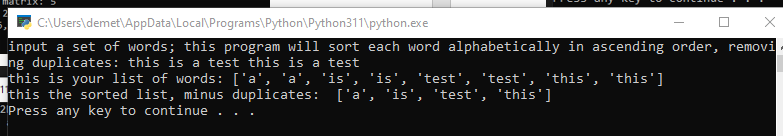
#we are iterating over the copy;

#thus changes to my\_list does

#not affect the returned my\_list[:] iteration object

#and all elements are checked (non skipped).

print("this the sorted list, minus duplicates: ", my\_list)



Graphical user interface, application

Description automatically generated

##----------------------------------------#

# #Question 4:

#Write a program that accepts a sentence and calculate the number of uppercase letters and lowercase letters.

#Suppose the following input is supplied to the program:

#Hello world!

#Then, the output should be:

#UPPER CASE 1

#LOWER CASE 9

user\_string = input("Enter a string, program will output number of upper and lowercase letters:")

num\_upper = 0

num\_lower = 0

for character in user\_string:

if character.islower(): #track lower case letters

num\_lower += 1

elif character.isupper(): #track upper case letters

num\_upper += 1

print("UPPER CASE:", num\_upper)

print("lower case:", num\_lower)

Text

Description automatically generated

##----------------------------------------#

# #Question 5:

#

#Print only odd numbers from a list of integers

#[Winter 2023: skip 5.1]

for num in input("enter a list of integers; out will be only odd numbers from your list: ").split(sep = ' '):

if num.isalnum() and int(num) % 2 == 1: #first check if char is alphanumeric, then if it is cast as int and check if it is odd #.

print(num, end = ' ')

print()

Text

Description automatically generated

##----------------------------------------#

# #Question 6:

#Write a program to compute the frequency of the words from the input. Sort the words alphanumerically for printing.

#Suppose the following input is supplied to the program:

#New to Python or choosing between Python 2 and Python 3? Read Python 2 or Python 3.

#Then, the output should be:

#2:2

#3.:1

#3?:1

#New:1

#Python:5

#Read:1

#and:1

#between:1

#choosing:1

#or:2

#to:1

#I can recycle the code from question 3, since it results in a list of sorted words that remove duplicates:

#print what #sorted() returns from the string that #input() returns. #split between ws in string; convert a list

my\_list = sorted(input("input a set of words; this program will sort each word alphabetically in ascending order, removing duplicates: ").split(' '))

my\_list\_copy = my\_list[:] #my\_list[:] returns a deep copy of my\_list over the entire range of the list;

#we need this for iterating over all values + checking frequency later

print("this is your list of words:", my\_list)

for word in my\_list\_copy:

if my\_list.count(word) > 1:

my\_list.remove(word)

#we are iterating over my\_list\_copy;

#thus changes to my\_list does

#not affect the returned my\_list[:] iteration object

#and all elements from the original user input are checked (not skipped).

print("This the sorted list, minus duplicates: ", my\_list)

print("This is the frequency of words from the original list, and sorted: ")

for word in my\_list: #now my\_list is sorted and with no duplicates;

#so we can check each word only once and iterate over this already sorted list

print(my\_list\_copy.count(word), ': ', word, sep = '') # copy of my list contains all words, including duplicates; need to use it to check frequency

Text

Description automatically generated

A screenshot of a computer

Description automatically generated

# #Question 5.1:

#Use a list comprehension to square each odd number in a list. The list is input by a sequence of comma-separated numbers.

#Suppose the following input is supplied to the program:

#1,2,3,4,5,6,7,8,9

#Then, the output should be:

#1,9,25,49,81

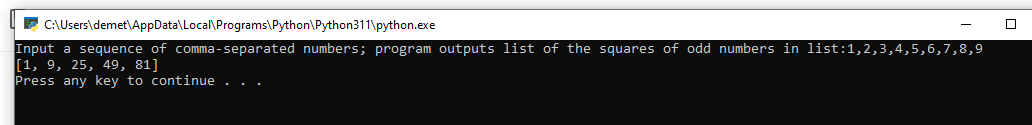
#first, get user input, split it by ',' delimeter into a list of alphanumeric strings

num\_list = input("Input a sequence of comma-separated numbers; program outputs list of the squares of odd numbers in list:").split(',')

#now print the list object returned from the list comprehension:

#cast string as int and square it and add it to list if it is an odd number (there is a remainder after dividing by 2)

print([int(num)\*\*2 for num in num\_list if (int(num) % 2 != 0)])



##----------------------------------------#

# #Question 7:

#Create a dictionary with telephone country prefixes and country name. Populate with multiple test examples.

tele\_country\_codes = dict(Turkey=90, United\_States=1, Zambia=260, Yemen=967) #initialize country code dictionary

print(tele\_country\_codes)

tele\_country\_codes['Ukraine'] = 380 #add an addtional country

print(tele\_country\_codes)

Text

Description automatically generated

##----------------------------------------#

# #Question 8:

#Write a program that contains multiple nested dictionaries.

#The first dictionary has key of city name. The inner dictionary contains sport as key.

#The next inner dictionary contains team name, #wins, #losses.

#Display for every team, its win percentage (wins / (wins+losses)).

city\_dictionary\_sports = {

#Cities:

"Clevland":{

#Sports: Teams: W, L

"Basketball":{"Cavaliers":[38,23]},

"Football":{"Browns":[7,10]}

},

"Denver":{

#Sports: Teams: W, L

"Basketball":{"Nuggets":[41,18]},

"Football":{"Broncos":[5,12]}

},

"Los Angeles":{

#Sports: Teams: W, L

"Basketball":{"Lakers":[27,32],

"Clippers":[33,28]},

"Football":{"Chargers":[10,7],

"Rams":[5,12]}

}

}

track\_teams\_win\_loss\_avg = {} #need this dictionary for tracking average win and loss percentage of all cities

#use these nested loops to output all cities, associated teams, sports, and stats, including finding and storing w/w+l percentage and avg:

for city in city\_dictionary\_sports: #iterate over city dictionary keys

sum\_w\_l\_percentage = 0 #track sum of w\_l percentage of the city

num\_teams = 0 #track number of teams iterated over for the given city

print("City:",city)

for sport in city\_dictionary\_sports[city]: #iterate over sports keys

print("\tSport:",sport)

for team in city\_dictionary\_sports[city][sport]: #iterate over team keys

print("\t\tTeam:",team)

win\_loss = city\_dictionary\_sports[city][sport][team] #get w,l list stored in value returned by team key

print("\t\t\tWins, Losses:",win\_loss)

win\_loss\_perc = float(win\_loss[0]) / ( float(win\_loss[0]) + float(win\_loss[1]) ) #calculate w/w+l

city\_dictionary\_sports[city][sport][team] += [round(win\_loss\_perc,2)] #append w\_l% to [w,l] list value returned by team key

print("\t\t\tWin% --> W/(W+L):", city\_dictionary\_sports[city][sport][team][2])

sum\_w\_l\_percentage += city\_dictionary\_sports[city][sport][team][2] # track sum

num\_teams += 1 #track number of teams included in the sum

#now calculate city avg and store it in a new dictionary so we can sort it later and find team with best avg win percentage:

track\_teams\_win\_loss\_avg[city] = float(sum\_w\_l\_percentage) / float(num\_teams)

print("Avg W\_L% for the city:", track\_teams\_win\_loss\_avg[city], '\n')

#Find the city that has the highest combined average winning percentage from all of its teams

#(add winning percentage of each team in city and divide by number of teams).

#sort the (key,value) tuples returned by .items() in descending order, based on the second element in the tuple (the value)

#then print first element in that new sorted tuple list that was returned, since it will be the city with highest win percentage.

#note: setting key to sort with key = lambda is simply creating a function with no label,

#passing tuple\_element to it, and returning tuple\_element[1] to key.

#So each tuple is sorted based on what the lambda function passes to key,

#a list of keys with associated elements # is created and sorted in background based on keys

#then, sorted will create a new list by taking element#s from (key,element#) pairs that have been sorted, in that order.

#for example: original element order is 0,1,2,3...after sort, it may be 1,2,4,3...

#then newlist = list[1] + list[2] + list[4] + list[3] ...

print("Here is the city with the highest avg winning percentage:")

print(sorted(track\_teams\_win\_loss\_avg.items(), key = lambda tuple\_element: tuple\_element[1], reverse=True)[0]) #reverse=true = descending order

Text

Description automatically generated

##----------------------------------------#

#

# #Question 9:

#Write a program which accepts a sequence of comma separated 4 digit binary numbers as its input and then check whether they are divisible by 5 or not.

#The numbers that are divisible by 5 are to be printed in a comma separated sequence.

#Example:

#0100,0011,1010,1001

#Then the output should be:

#1010

#example for converting binary to decimal, starting with lowest digit first:

#1010 --> need to reverse it --> 0101, and then powers go from 0, 1, 2, 3...

#0\*2^0 + 1\*2^1 + 0\*2^2 +1\*2^3 = 0 + 2 + 0 + 8 = 10

binary\_four\_digits\_list = input("Enter a sequence of comma-separated 4-digit binary numbers: ").split(',')

decimal\_vals\_list = []

print("Here are your binary numbers:",binary\_four\_digits\_list)

for binary\_num in binary\_four\_digits\_list: #iterate through the list to convert every 4-digit binary number

decimal\_equivalent = 0

for power,digit in enumerate(reversed(binary\_num)): #1010 --> need to reverse it --> 0101

# and then powers go from 0, 1, 2, 3 (this is conveniently our powers of 2 that we need for each binary digit)

decimal\_equivalent += (int(digit) \* 2\*\*power)

decimal\_vals\_list += [decimal\_equivalent] #append converted decimal value to decimal value list

print("Decimal Equivalents of your list:",decimal\_vals\_list)

#now check which values are divisble by 5 and output those values which are:

divisble\_binary\_digits\_list = []

for element,decimal\_val in enumerate(decimal\_vals\_list):

if decimal\_val % 5 == 0: #remainder == 0 means value is divisble by 5

divisble\_binary\_digits\_list += [binary\_four\_digits\_list[element]] #add element from binary list

print("Here is the list of your binary values that are divisble by 5:", divisble\_binary\_digits\_list)

Text

Description automatically generated

##----------------------------------------#

#

# #Question 10:

#Write a program, which will find all such numbers between 1000 and 3000 (both included) such that each digit of the number is an even number.

#The numbers obtained should be printed in a comma-separated sequence on a single line.

for number in range(1000, 3001, 2): # range [1000, 3001] and only return a number every 2 values (step = 2)

print(number, ', ', end = '')

A computer screen capture

Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated with medium confidence

##----------------------------------------#

# #Question 11:

#Write a program that accepts a sentence and calculate the number of letters and digits.

#Suppose the following input is supplied to the program:

#hello world! 123

#Then, the output should be:

#LETTERS 10

#DIGITS 3

#I will use my code from hw4\_question10 since it is essentially the same

def calc\_Dig\_Alph(sample = "Python 3.2"):

Letters = 0

Digits = 0

for i in sample:

if i.isalpha():

Letters += 1

if i.isdigit():

Digits += 1

print("Test String:", sample)

print("Letters:",Letters)

print("Digits:", Digits)

calc\_Dig\_Alph()

sample = input("Enter any string/digits, and the number of digits and letters will be calculated:")

calc\_Dig\_Alph(sample)

Text

Description automatically generated

##----------------------------------------#

# #Question 12:

#A website requires the users to input username and password to register.

#Write a program to check the validity of password input by users.

#Following are the criteria for checking the password:

#1. At least 1 letter between [a-z]

#2. At least 1 number between [0-9]

#3. At least 1 letter between [A-Z]

#4. At least 1 character from [$#@]

#5. Minimum length: 6

#6. Maximum length: 12

#Your program should accept a sequence of comma separated passwords and will check them according to the above criteria.

#Passwords that match the criteria are to be printed, each separated by a comma.

#Example

#If the following passwords are given as input to the program:

#ABd1234@1,a F1#,2w3E\*,2We3345

#Then, the output of the program should be:

#ABd1234@1

user\_passwords = input("Enter a list of comma-separated passwords; if strngeth criteria met, password will be printed: ").split(',')

for password in user\_passwords: #check every password

#reset flag criteria values

check\_has\_alpha = False

check\_has\_upper = False

check\_has\_digit = False

check\_has\_special = False

if (len(password) >= 6) and (len(password) <= 12): # check criterias 5 and 6, if true for both, proceed.

for character in password: #check every letter of every password, check criteria 1

if character.isalpha():

check\_has\_alpha = True

break

for character in password: #check every letter of every password, check criteria 2

if character.isalnum():

check\_has\_digit = True

break

for character in password: #check every letter of every password, check criteria 3

if character.isupper():

check\_has\_upper = True

break

for character in password: #check every letter of every password, check criteria 4

if character == "$" or character == "#" or character == "@" :

check\_has\_special = True

break

if(check\_has\_alpha and check\_has\_upper and check\_has\_digit and check\_has\_special):

print(password)

Text

Description automatically generated

##----------------------------------------#

#

# #Question 13:

#Practice with the basic string methods:

#Isalpha(), islower(), isupper(), isdigit(),

print('!','!'.isalpha(), "#not alpha")

print('a','a'.isalpha(), "#is alpha")

print('A','A'.isalpha(), "#is alpha")

print('a','a'.islower(), "#is lowercase")

print('A','A'.islower(), "#not lowercase")

print('a','a'.isupper(), "#not uppercase")

print('A','A'.isupper(), "#is uppercase")

print('hello!','hello!'.isdigit(), "#not integer digit")

print('234'.isdigit(), "#is integer digit")

print('234.4'.isdigit(), "#not integer digit")

#startswith(), endswith(),

print("\"string\" starts with \"str\"", "string".startswith('str'))

print("\"string\" ends with \"ing\"", "string".endswith('ing'))

#lower(), upper(),

print("\"string\" ends with \"ing\"", "string".endswith('ing'))

#title(), lstrip(), rstrip(), strip(),

print("This is the title where all the first letters of words are uppercase".title(), '\n')

print(" This is an example of the lstrip() function on strings")

print(" This is an example of the lstrip() function on strings".lstrip()) #remove leading ws char by default

print("^^^^^^^^^This is an example of the lstrip() function on strings")

print("^^^^^^^^^This is an example of the lstrip() function on strings".lstrip('^'), '\n') #remove leading '^' chars

#ljust(), rjust(), center()

print("this is an example of ljust() function of strings".ljust(100, '.'))

print("this is an example of rjust() function of strings".rjust(100, '.'))

print("this is an example of center() function of strings".center(100, '.'))

Text

Description automatically generated

#Get the ordinal value of '%'. Get the character value of 94.

print(ord('%'), chr(94))

#

#Slice string "delicious" to get first, last, second, second to last characters.

delicious\_string = "delicious"

#first #last #second #second to last

print(delicious\_string[0],delicious\_string[len(delicious\_string) - 1],delicious\_string[1],delicious\_string[len(delicious\_string) - 2])

#Get substrings beginning to position 5, position 7 to end, positions 3-5.

print(delicious\_string[5:],delicious\_string[7:],delicious\_string[3:6] ) #note --> with range [a:b], b is not inclusive, so had to do [3:6] to get range [3-5]

#What does print[-1:] do? What does print[:-1] do?

print(delicious\_string[-1:],delicious\_string[:-1])

#[-1:] means print from last character to last character [-1,end of string], where element -1 == last character of list

#[:-1] means print from first character up until but not including last character == -1

#Print a line of 36 '^'.

print('^' \* 36)

#Print "Hello World!" 5 times.

print("Hello Wolrd!" \* 5)

#Use ''' to create a multi-line string and print its length.

print('''\nthis is a multi-line

string that I am writing out\n''')

#Phone\_number = '1-800-555-1212'

#Use .find and .replace to change it to 1 (800) 555.1212

Phone\_number = '1-800-555-1212'

#each .replace() function returns a new object; thus I iteratively operate on returned objects by calling .replace() for each returned object

Phone\_number = Phone\_number.replace('-', ' (', 1).replace('-', ') ', 1).replace('-', '.', 1)

print(Phone\_number, '\n')

#Quote="These are the times that try men's souls".

print("\"These are the times that try men's souls\"")

#Print each word on its own line.

print("\"These are the times that try men's souls\"".replace(' ', '\n',-1)) #-1 means replace all instances

#Print the last 4 words.

#string object calls split() function which returns another object which is a list of the words,

#then we index into it starting at 4th to last character (denoted by -4) to the end of the list,

#thus we print that final returned object of the last 4 words of the list.

#So in total, there are 3 temporary objects involved that we never stored but only printed the final object:

print("\"These are the times that try men's souls\"".split(' ')[-4:])

#Date = "02/14/2019" separate it into variables month, day, year and print as year/month/day

Date = "02/14/2019"

Month = Date.split('/')[0]

Day = Date.split('/')[1]

Year = Date.split('/')[2]

print(Year, Month, Day, sep='/')

Text

Description automatically generated