**Discussion 1**

When choosing a password for online accounts, there are typically certain requirements for the strength of the password. Develop a Python program for testing if a string satisfies some appropriate criteria for a strong password. It’s up to you to define the requirements.

*Hint: string functions and methods, e.g., isupper(), islower(), isdigit(), len(),*

#Design a program taking in a stringhi

#At least 1 upper, lower, digit char

#Password at least 8 characters to be strong

#Function requesting user to input strong password

def enter\_pass():

print(

"Your password must be strong to be accepted. It must contain at least:\n\

- 1 Upper Case Character\n\

- 1 Lower Case Character\n\

- 1 Digit\n\

- 8 Characters Long")

print("Enter '####' to exit.")

print()

mypass = input("Enter a password: ")

if mypass == "####":

print()

print("Program exited.")

quit()

return mypass

mypass = enter\_pass()

#Strong password fulfils all 4 criterias.

strongPass = 4

strength = 0

while strength != 4:

strength = 0

#Check for upper case character

for char in mypass:

if char.isupper():

strength += 1

break

if strength < 1:

print()

print("Your password does not contain an UPPER CASE character, try again!")

print()

mypass = enter\_pass()

continue

#Check for lower case character

for char in mypass:

if char.islower():

strength += 1

break

if strength < 2:

print()

print("Your password does not contain a LOWER CASE character, try again!")

print()

mypass = enter\_pass()

continue

#Check for a digit

for char in mypass:

if char.isdigit():

strength += 1

break

if strength < 3:

print()

print("Your password does not contain a DIGIT, try again!")

print()

mypass = enter\_pass()

continue

#Check for password length

if len(mypass) >= 8:

strength += 1

if strength < 4:

print()

print("Your password does not contain AT LEAST 8 CHARACTERS, try again!")

print()

mypass = enter\_pass()

continue

print()

print("Your entered password is strong! Nice!")

**Discussion 2**

Consider a system for storing anonymous grades of each lab class. Define a data structure, which can identify individuals in each lab group by an ID number 1-40 (inclusive). To identify the person in the entire class you would also need the group name, e,g., ‘FE2’. Each corresponding person should have a number between 1-100 (inclusive) to define grade.

*Hint: Select suitable one/ones from the built-in data structures provided by Python (strings, lists, tuples, dictionaries)*

#Create a system for storing anonymous grades for each lab class

#Each lab class will have different lab groups, i.e. FE2

#Each individual in each lab group has an ID of x, where 1 <= x <= 40

#Each individual should also have a grade of y assigned to them, where 1 <= y <= 100

#Function for storing data

def key\_in(labClass):

    individual = input("Enter the details using the format below.\nLab Group, ID(1-40), Grade(1-100): ")

    mylist = [i.strip() for i in individual.split(",")]

    group, ID, grade = mylist[0].upper(), mylist[1], mylist[2]

    labClass[group] = {}

    labClass[group][ID] = grade

    print("This individual's grade has been stored.\n")

#Function for calling data

def check(labClass):

    details = input("Enter the details using the format below.\nLab Group, ID(1-40): ")

    mydetails = [j.strip() for j in details.split(",")]

    group, ID = mydetails[0].upper(), mydetails[1]

    print(f"\nThis individual's grade is {labClass[group][ID]}.\n")

#Start of looping program

labClass = {}

print("Lab Class Grading Dictionary v1")

while True:

    #Asking if user wishes to store or check an individual's grade in the dictionary

    print("Enter 'exit' to exit from the program.")

    storeOrCheck = input("Do you wish to STORE or CHECK an individual's grade?\n")

    if storeOrCheck.upper() == "EXIT":

        quit()

    elif storeOrCheck.upper() == "STORE":

        key\_in(labClass)

    elif storeOrCheck.upper() == "CHECK":

        check(labClass)

**Discussion 3**

Given two lists of grades (list of integers) from two classes, write a Python program that will check which class has the highest average score and the highest maximum score.

*Hint: common functions and methods of Python lists, e.g., min(), max(), len(), etc.*

#Take 2 lists of integer grades from two classes

#Returns the highest avg score

#Returns the highest maximum score

#Sample input

#Class 1: [90,10,50]

#Class 2: [80,50,30]

print("Key in the class grade using the format below.")

print("Grade 1, Grade 2, Grade 3...")

class1 = list(map(int, input("Enter First Class' grades: ").split(",")))

class2 = list(map(int, input("Enter Second Class' grades: ").split(",")))

print()

class1\_avg = sum(class1)/len(class1)

class2\_avg = sum(class2)/len(class2)

if class1\_avg % 1 == 0:

    class1\_avg = int(class1\_avg)

else:

    class1\_avg = round(class1\_avg, 1)

if class2\_avg % 1 == 0:

    class2\_avg = int(class2\_avg)

else:

    class2\_avg = round(class2\_avg, 1)

if class1\_avg > class2\_avg:

    print(f"Class 1 has a higher average score of {class1\_avg}.")

elif class1\_avg < class2\_avg:

    print(f"Class 2 has a higher average score of {class2\_avg}.")

else:

    print(f"Both classes have the same average score of {class1\_avg}.")

class1\_max, class2\_max = max(class1), max(class2)

if class1\_max > class2\_max:

    print(f"Class 1 has the highest maximum score of {class1\_max}.")

elif class1\_max < class2\_max:

    print(f"Class 2 has the highest maximum score of {class2\_max}.")

else:

    print(f"Both classes have the highest maximum score of {class1\_max}.")

**Discussion 4**

Write a Python program, in the fewest number of lines possible, which creates a list of all the square numbers: x2 (where 1<=x<=100) that are divisible by 3.

*Hint: Python list comprehension*

print([x\*\*2 for x in range(1,101) if x\*\*2 % 3 == 0])