

**Name:Yara Shahoud, Number:2122, Submitted To GitHub:** [**https://github.com/YaraShahoud/Network\_Programming\_Homework\_No1\_2024.git**](https://github.com/YaraShahoud/Network_Programming_Homework_No1_2024.git)

Question 1>>

Python Basics?

A-If you have two lists, L1=[‘HTTP’,’HTTPS’,’FTP’,’DNS’] L2=[80,443,21,53], convert it to generate this dictionary d={‘HTTP’:80,’HTTPS’:443,’FTP’:21,’DNS’:53 }

# initializing lists

L1 = ["HTTP", "HTTPS", "FTP", "DNS"]

L2 = [80, 443, 21, 53]

# Printing original keys-value lists

print("Original L1 list is : " + str(L1))

print("Original L2 list is : " + str(L2))

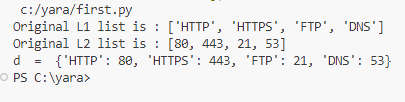
# using dictionary comprehension

# to convert lists to dictionary

res = {L1[i]: L2[i] for i in range(len(L1))}

# Printing resulting dictionary

print("d  =  " + str(res))



B- Write a Python program that calculates the factorial of a given number entered by user. #Define Factorial function that is 1 if number is 0 or 1 or n \* n-1 \* n-2 ... else

def factorial(n):

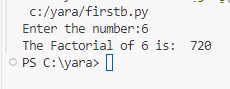
    # Recursive way to find factorial

    return 1 if (n==1 or n==0) else n \* factorial(n - 1)

number = int(input("Enter the number:"))  #ask user to input number

f = factorial(number)  #cal function to calculate factorial

print("The Factorial of", number, "is: ", f)  #print result to user



C- L=[‘Network’ , ’Bio’ , ’Programming’, ‘Physics’ , ‘Music’] In this exercise, you will implement a Python program that reads the items of the previous list and identifies the items that starts with ‘B’ letter, then print it on screen. Tips: using loop, ‘len ()’ , startswith() methods.

L = ["Network", "Bio", "Programming", "Physics", "Music"]

for i in range(len(L)): #loop list using len

    if L[i].startswith("B"): #check if element startswith B print it

        print(L[i]) #print element



D: Using Dictionary comprehension, Generate this dictionary d={0:1,1:2,2:3,3:4,4:5,5:6,6:7,7:8,8:9,9:10,10:11}

# creation using list comprehension

myDict = {x: x+1 for x in [0,1,2,3,4,5,6,7,8,9,10]}#make dict as x, x+1 looping from 0 to 10

print (myDict)#print resulting dictunary



Question 2>>

Convert from Binary to Decimal Write a Python program that converts a Binary number into its equivalent Decimal number. The program should start reading the binary number from the user. Then the decimal equivalent number must be calculated. Finally, the program must display the equivalent decimal number on the screen. Tips: solve input errors.

# binary to decimal

def binaryToDecimal(n):

    number = n;

    decimal\_value = 0;#inal decimal number

    # Initializing base

    # value to 1, i.e 2 ^ 0

    base = 1;#2^base for example bit zero has 2^0=1

    len1 = len(number);#get length of number

    for i in range(len1 - 1, -1, -1):#loop number from reight to left

        if (number[i] != '1') and (number[i] != '0'): #if digit is not 1 and not 0 print error and exit

            print("Wrong Binary number");

            exit();

        if (number[i] == '1'): #if digit is 1 multiply it by 2 ^ base

            decimal\_value += base;#add it to final decimal

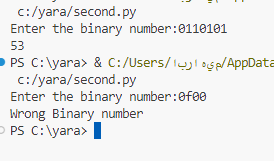
        base = base \* 2;#update base value

    return decimal\_value;#return final number

# Test Code

number = str(input("Enter the binary number:"))  #ask user to input binary number

print(binaryToDecimal(number));



Question 3>>

Working with Files” Quiz Program” Type python quiz program that takes a text or json or csv file as input for (20 (Questions, Answers)). It asks the questions and finally computes and prints user results and store user name and result in separate file csv or json file.

import json #import lib to deal with json files

import os

import time

import random #generate random numbers

helpLineUsed = False #is the user used help

def fiftyFifty(question\_ansDict): #choose random question

    print(question\_ansDict.keys()) #print answers

    keysList = list(question\_ansDict.keys())

    fiftyFiftyDict = {'q': question\_ansDict['q'],

                      'ca': question\_ansDict['ca'],

                      }

    print(question\_ansDict['ca'])

    correctAnswerKey = question\_ansDict['ca']

    fiftyFiftyDict[correctAnswerKey] = question\_ansDict[correctAnswerKey]

    keysList.pop(keysList.index('q'))

    keysList.pop(keysList.index('ca'))

    keysList.pop(keysList.index(correctAnswerKey))

    while len(keysList) > 1:

        randomIndex = random.randint(0, len(keysList) - 1)

        keysList.pop(randomIndex)

    else:

        fiftyFiftyDict[keysList[0]] = question\_ansDict[keysList[0]]

    return fiftyFiftyDict

def showOptionAnswer(question\_dict, isFiftyFifty):#show options that user can input

    print("\n {questionNo}) {question}".format(questionNo=i + 1, question=question\_dict['q']))

    optionBuilder = ""

    for key, value in sorted(question\_dict.items()):

        if key == 'ca' or key == 'q':

            continue

        optionBuilder += key + "/"

        print(" <{optionNo}> {option}".format(optionNo=key, option=value))

    if isFiftyFifty:

        answer = input("\n Enter the option ({}) ".format(optionBuilder))

    else:

        answer = input("\n Enter the option (a/b/c/d) or h for helpline or q to quit: ")

    return answer

print(" ~~~~~~ WELCOME TO QUIZ APP ~~~~~") #print welcome screen

print("\n ### RULES OF THE GAME ###")

print(" 1. Choose one of the correct option from each question (a/b/c/d)")

print(" 2. h for help")

print(" 3. q to quit")

print("\n ### BEST OF LUCK ###\n")

with open("qa.json", "r", encoding='utf-8') as qa: #read json file to get data

    questionSet = qa.read()

    questionsList = json.loads(questionSet)#load questions

    rightAnswer = 0

    i = 0

    while i < len(questionsList): #while we have questions

        question\_dict = questionsList[i]#read question

        answer = showOptionAnswer(question\_dict, False)#show answers

        i += 1

        if answer == 'h':#if user ask for help

            os.system("clear")

            if helpLineUsed:

                print("<<< Help Line Already Used >>>")#inshure help not used before

                i -= 1

            else:

                helpLineUsed = True#mark help as used

                question\_dict = fiftyFifty(question\_dict)

                answer = showOptionAnswer(question\_dict, True)#print answer

        if answer == 'q':#if user quits

            print("\nGame Over")

            print("You made {} right. Your score is {} ".format(rightAnswer, str(rightAnswer \* 10)))

            exit()

        if question\_dict['ca'] == answer:#if answer is wright

            print("You predicted right answer")

            rightAnswer += 1

        else:

            if not helpLineUsed:

                print("\nYour answer is wrong")

                print("\nCorrect answer is {correctAnswer}".format(correctAnswer=question\_dict['ca']))

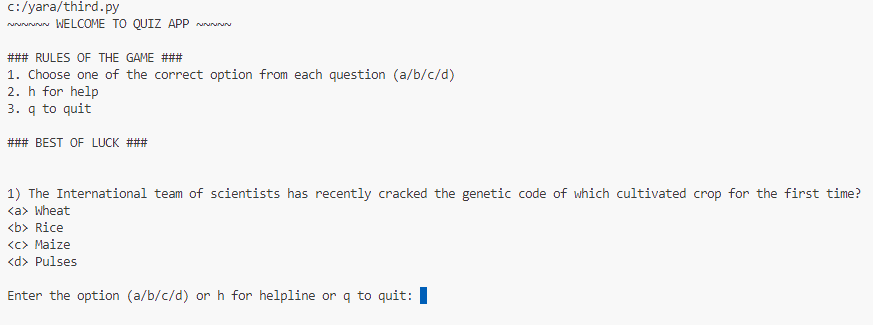
        time.sleep(1)

        os.system("clear")

    else:

        print("\n!!!Game Over")

        print("You made {} right. Your score is {} ".format(rightAnswer, str(rightAnswer \* 10)))#print results



Question 4>>

Object-Oriented Programming - Bank Class Define a class BankAccount with the following attributes and methods: Attributes: account\_number (string), account\_holder (string), balance (float, initialized to 0.0) Methods:deposit(amount), withdraw(amount) , get\_balance() - Create an instance of BankAccount, - Perform a deposit of $1000, - Perform a withdrawal of $500. - Print the current balance after each operation. - Define a subclass SavingsAccount that inherits from BankAccount and adds interest\_rate Attribute and apply\_interest() method that Applies interest to the balance based on the interest rate. And Override print() method to print the current balance and rate. - Create an instance of SavingsAccount , and call apply\_interest() and print() functions.

class BankAccount: #bank class

    def \_\_init\_\_(self, account\_holder, account\_number, balance=0.0):#constructer balance is default vale = 0.0

        self.account\_holder = account\_holder#assign values to members

        self.account\_number = account\_number

        self.balance = balance

    def deposit(self, amount):#deposit method

        self.balance += amount#add it to balance

        print(f"{self.account\_holder} Deposited {amount} $")#print depositing event

    def withdraw(self, amount):#withdraw method

        if self.balance >= amount:#check user have enough balance

            self.balance -= amount#discount it from user balance

            print(f"{self.account\_holder} Withdrew {amount} $")#print event to user

        else:

            print("You don't have enough funds to withdraw.")#print insufficient balance error

    def get\_balance(self):# print method

         print(f" Current balance is: {self.balance} $.")

test = BankAccount("yara", "1234")#create object

test.deposit(1000)

test.withdraw(500)

test.get\_balance()#ptint balance

class SavingsAccount(BankAccount):#inheritance

    def \_\_init\_\_(self, account\_holder, account\_number, interest\_rate, balance=0.0):#constructor

        super().\_\_init\_\_(account\_holder, account\_number, balance)#call parent constructor

        self.interest\_rate = interest\_rate#assign value

    def apply\_interest(self):#apply interest method

         interest = self.balance \* self.interest\_rate

         self.deposit(interest)

    def get\_balance(self):#override print method

         print(f" Current balance is: {self.balance} $ and current rate is {self.interest\_rate} .")

saving = SavingsAccount("yara", "123456789", 20) #test class

saving.deposit(1000)

saving.apply\_interest()

saving.get\_balance()

