الجمهورية العربية السورية
اللاذقية -جامعـــة تشريـــن
كلية الهندسة الكهربائية والميكانيكية
قسم هندسة الاتصالات والإلكترونيات
السنة الخامسة: وظيفة1 برمجة شبكات
الاسم: يوسف محمد العلي. الرقم الجامعي:٢١٥٤.

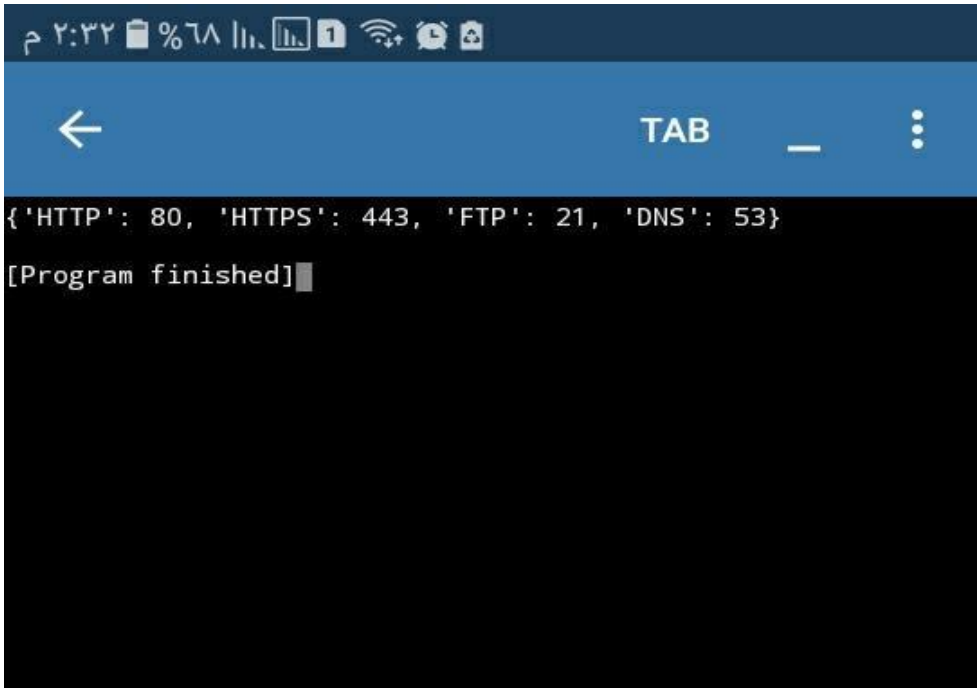
Submitted to GitHub:21/5/2024

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 }

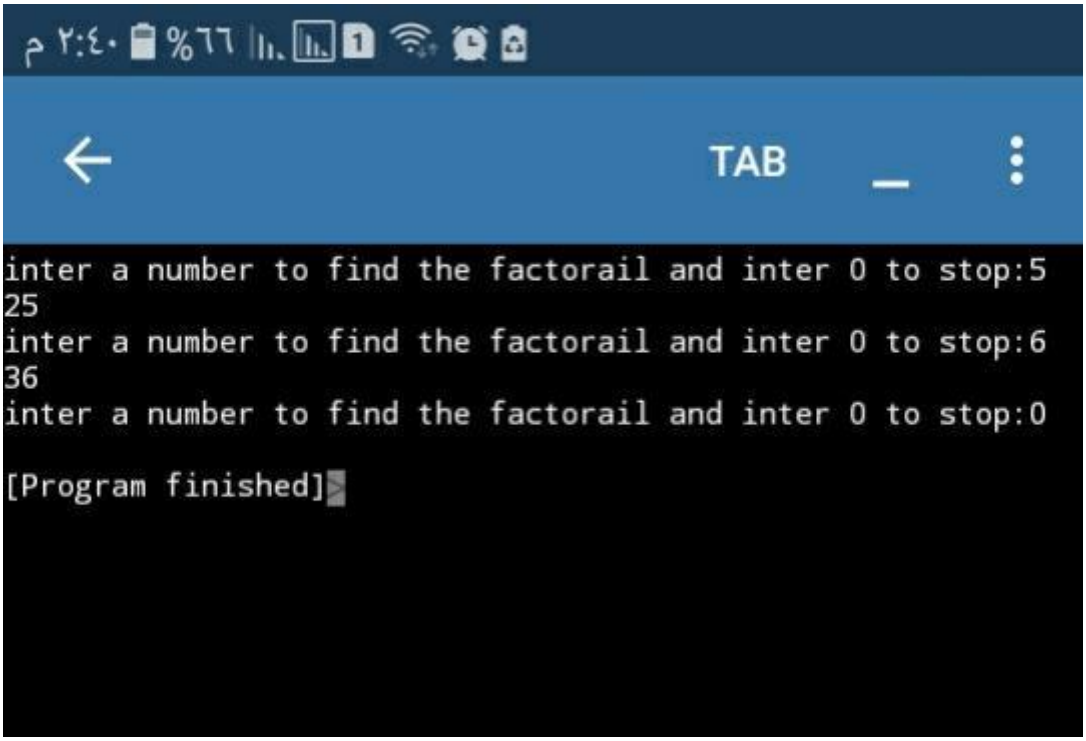
```
l1=['HTTP','HTTPS','FTP','DNS']
l2=[80,443,21,53]
d={l1[0]:l2[0],l1[1]:l2[1],l1[2]:l2[2],l1[3]:l2[3]}
print(d)
```



B- Write a Python program that calculates the factorial of a given number entered by user.

```
while True:
N=int(input('inter a number to find the factorial and inter 0 to stop:'))
if N==0:
break
```

```
else:
print (N*N)
```

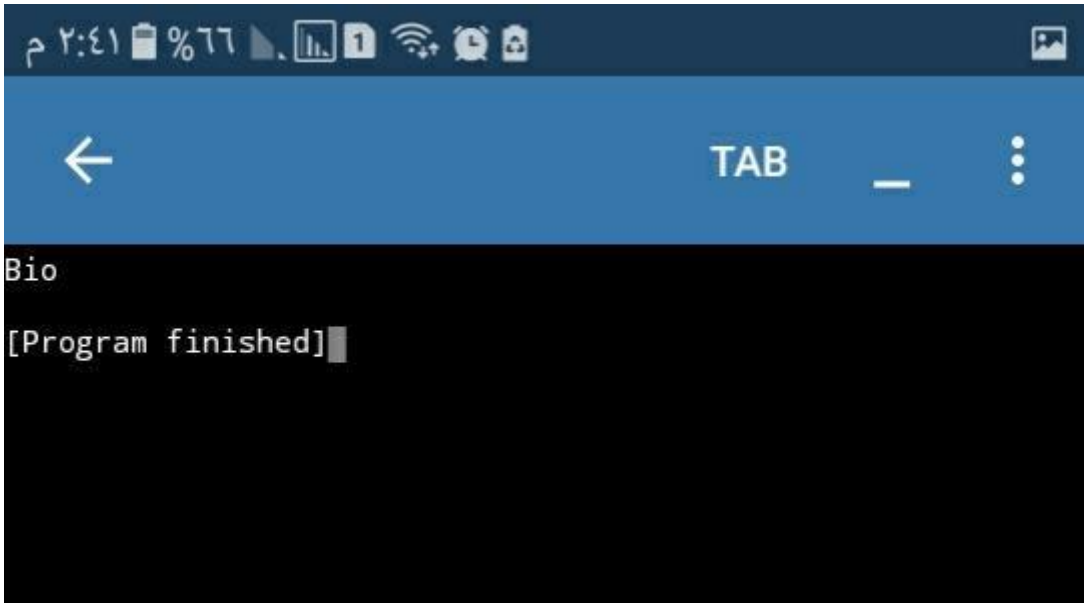


C- L=[‘Network’ , ’Bio’ , ’Programming’, ‘Physics’ , ‘Music’]

In this exercise, you will implement a Python program that reads the items of the previous list and identifies

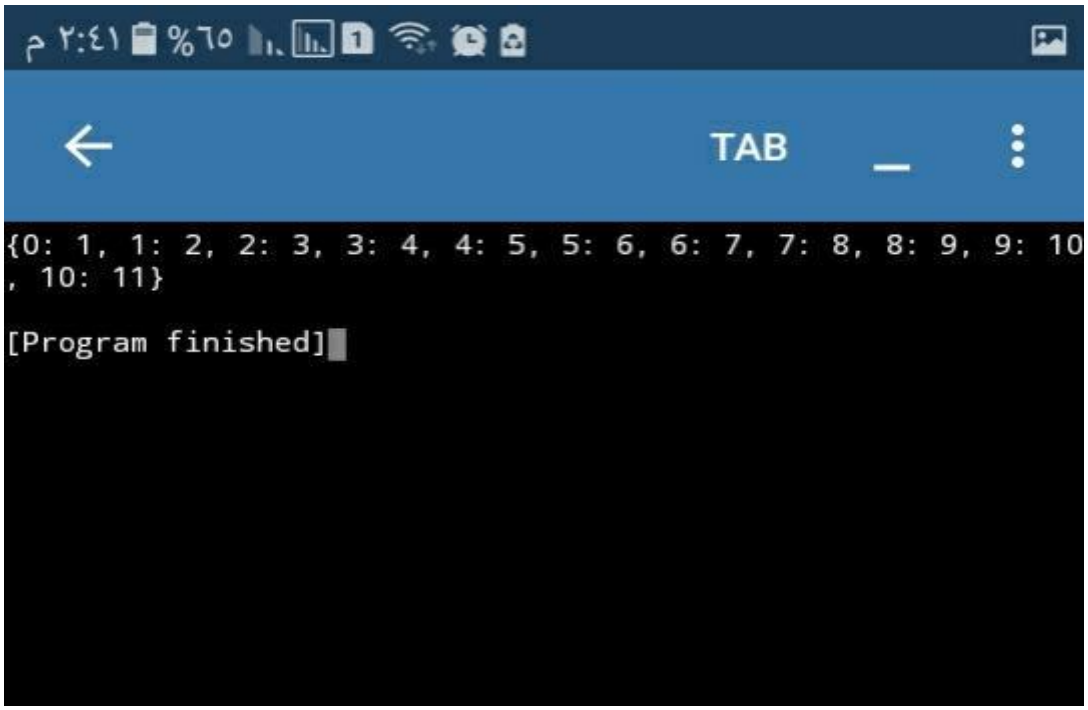
the items that starts with ‘B’ letter, then print it on screen

```
l = ['network', 'Bio', 'Programming', 'Physics', 'Music']
for list in l:
if list.startswith('B'):
print(list)
```



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}

```
l1=[0,1,2,3,4,5,6,7,8,9,10]
l2=[1,2,3,4,5,6,7,8,9,10,11]
d={l1[0]:l2[0],l1[1]:l2[1],l1[2]:l2[2],l1[3]:l2[3],l1[4]:l2[4],l1[5]:l2[5],l1[6]:l2[6],l1[7]:l2[
7],l1[8]:l2[8],l1[9]:l2[9],l1[10]:l2[10]}
print(d)
```



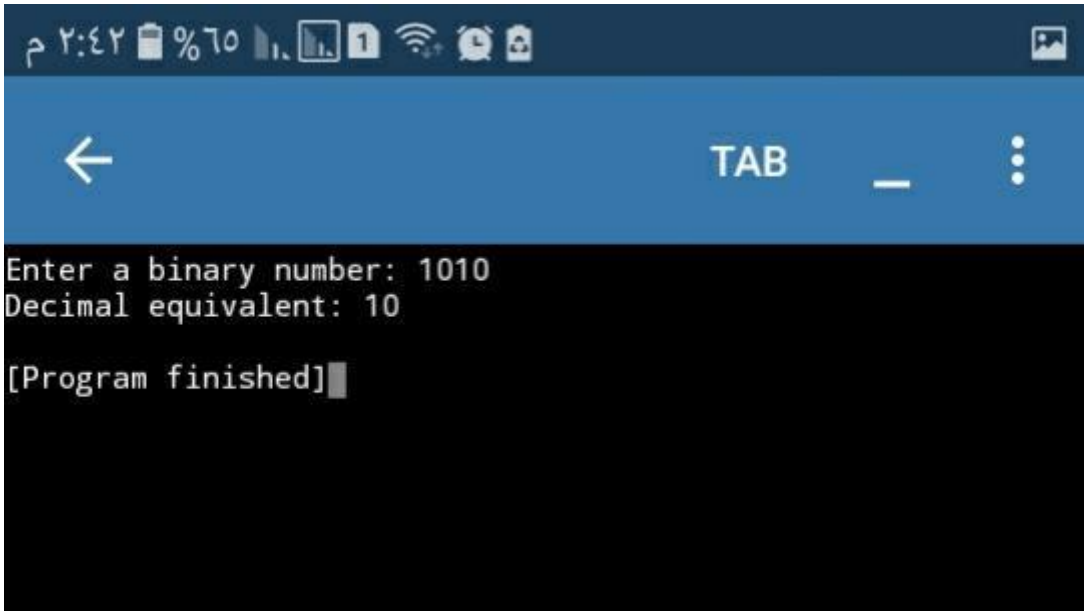
Question 2: Convert from Binary to Decimal

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

```
binary_number = input("Enter a binary number: ")
decimal_number = int(binary_number, 2)
print("Decimal equivalent:", decimal_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 csv

```
def load_questions_from_csv(file_name):
    questions = []
    with open(file_name, 'r', newline='', encoding='utf-8') as file:
        reader = csv.reader(file)
        for row in reader:
            question = {
                "question": row[0],
                "options": row[1:5],
                "answer": row[5]
            }
```

```
            questions.append(question)
    return questions

def quiz(questions):
    score = 0
    for i, q in enumerate(questions, 1):
        print(f"Question {i}: {q['question']}")
        for j, option in enumerate(q['options'], 1):
            print(f"{j}. {option}")
        user_answer = input("Your answer: ")
        if user_answer.lower() == q['answer'].lower():
            score += 1
    return score

def main():
    file_name = 'questions.csv'
    questions = load_questions_from_csv(file_name)
    user_name = input("Enter your name: ")
    user_score = quiz(questions)
    print(f"Dear {user_name}, your score is {user_score} out of {len(questions)}")

    with open('user_scores.csv', 'a', newline='', encoding='utf-8') as file:
        writer = csv.writer(file)
        writer.writerow([user_name, user_score])

if __name__ == "__main__":
    main()
```

#In this example, the csv library is used to read the CSV file containing the questions and answers. The quiz() function then asks the questions and calculates the user's score. Finally, the user name and its result are stored in a separate CSV file, and when we run the code, it will create a CSV file that creates questions and answers to .them, and we can open it via Excel or via Google Sheet

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:
    def __init__(self, account_number, account_holder, balance=0.0):
        self.account_number = account_number
        self.account_holder = account_holder
        self.balance = balance

    def deposit(self, amount):
        self.balance += amount
        print(f"Deposited ${amount}. Current balance: ${self.balance}")

    def withdraw(self, amount):
        if amount <= self.balance:
            self.balance -= amount
            print(f"Withdrew ${amount}. Current balance: ${self.balance}")
        else:
            print("Insufficient funds.")

    def get_balance(self):
        return self.balance

class SavingsAccount(BankAccount):
    def __init__(self, account_number, account_holder, balance=0.0,
interest_rate=0.0):
        super().__init__(account_number, account_holder, balance)
        self.interest_rate = interest_rate

    def apply_interest(self):
        interest_amount = self.balance * (self.interest_rate / 100)
        self.balance += interest_amount
        print(f"Interest applied: ${interest_amount}. Current balance: ${self.balance}")

    def print(self):
        print(f"Account Number: {self.account_number}")
        print(f"Account Holder: {self.account_holder}")
        print(f"Balance: ${self.balance}")
        print(f"Interest Rate: {self.interest_rate}%")

# Create an instance of BankAccount
bank_account = BankAccount("123456789", "John Doe")
bank_account.deposit(1000)
bank_account.withdraw(500)
```

```
print(f"Current balance: ${bank_account.get_balance()}")

# Create an instance of SavingsAccount
savings_account = SavingsAccount("547329186", "Yousef.Alali", interest_rate=2.5)
savings_account.deposit(2000)
savings_account.apply_interest()
savings_account.print()
```