Syrian Arab Republic

Lattakia - Tishreen University

Department of Communication and electrical engineering

5th , Network Programming : Homework No1

الجمهورية العربية السورية

اللاذقية -جامعـــة تشريـــن

كلية الهندسة الكهربائية والميكانيكية

قسم هندسة الاتصالات والالكترونيات

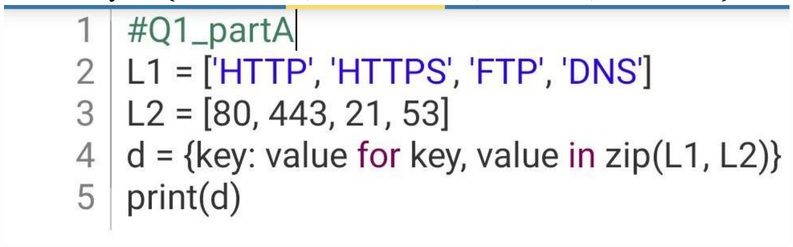
السنة الخامسة: وظيفة 1 برمجة شبكات

Name: yara al alouni , Number: 2263 , Submitted To GitHub:________

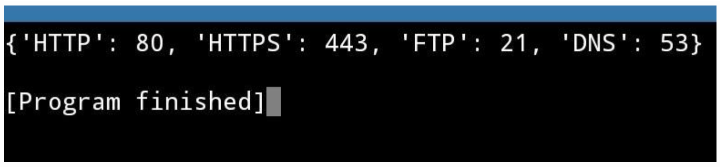
# First Network Programming Homework

**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 }

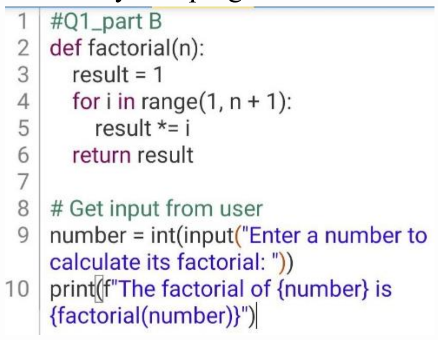


```python
#Q1_partA
L1 = ['HTTP', 'HTTPS', 'FTP', 'DNS']
L2 = [80, 443, 21, 53]
d = {key: value for key, value in zip(L1, L2)}
print(d)
```

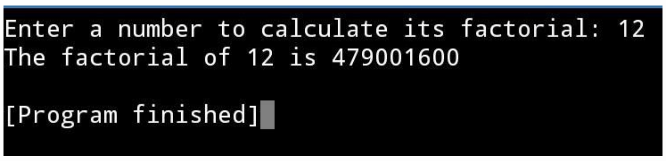


```
{'HTTP': 80, 'HTTPS': 443, 'FTP': 21, 'DNS': 53}

[Program finished]
```

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



```python
#Q1_part B
def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result

# Get input from user
number = int(input("Enter a number to calculate its factorial: "))
print(f"The factorial of {number} is {factorial(number)}")
```

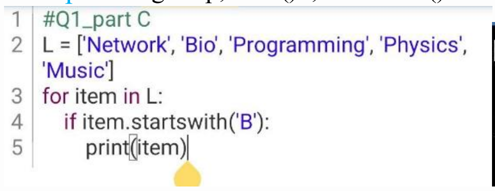


```
Enter a number to calculate its factorial: 12
The factorial of 12 is 479001600

[Program finished]
```

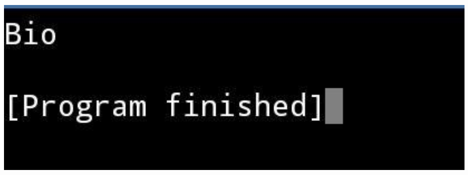
C- L=[‘Network’ , ’Bio’ , ’Programming’, ‘Physics’ , ‘Music’]

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.

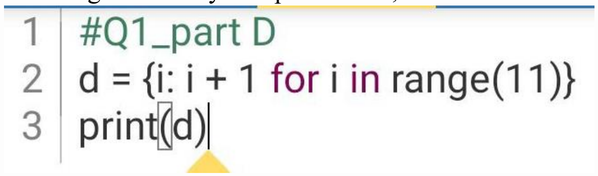


```python
#Q1_part C
L = ['Network', 'Bio', 'Programming', 'Physics', 'Music']
for item in L:
    if item.startswith('B'):
        print(item)
```

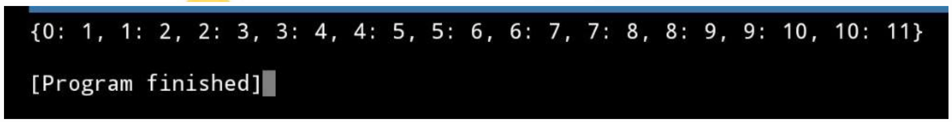


```
Bio

[Program finished]
```

**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}



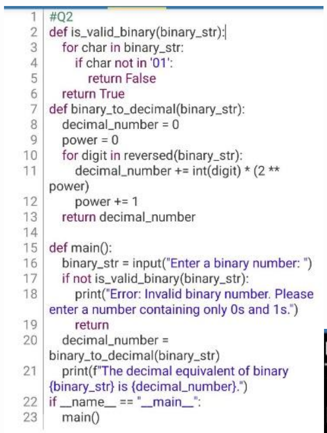
```python
#Q1_part D
d = {i: i + 1 for i in range(11)}
print(d)
```



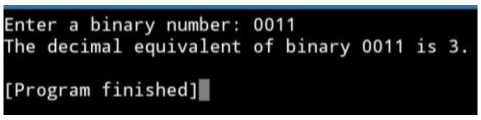
```
{0: 1, 1: 2, 2: 3, 3: 4, 4: 5, 5: 6, 6: 7, 7: 8, 8: 9, 9: 10, 10: 11}

[Program finished]
```

**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.



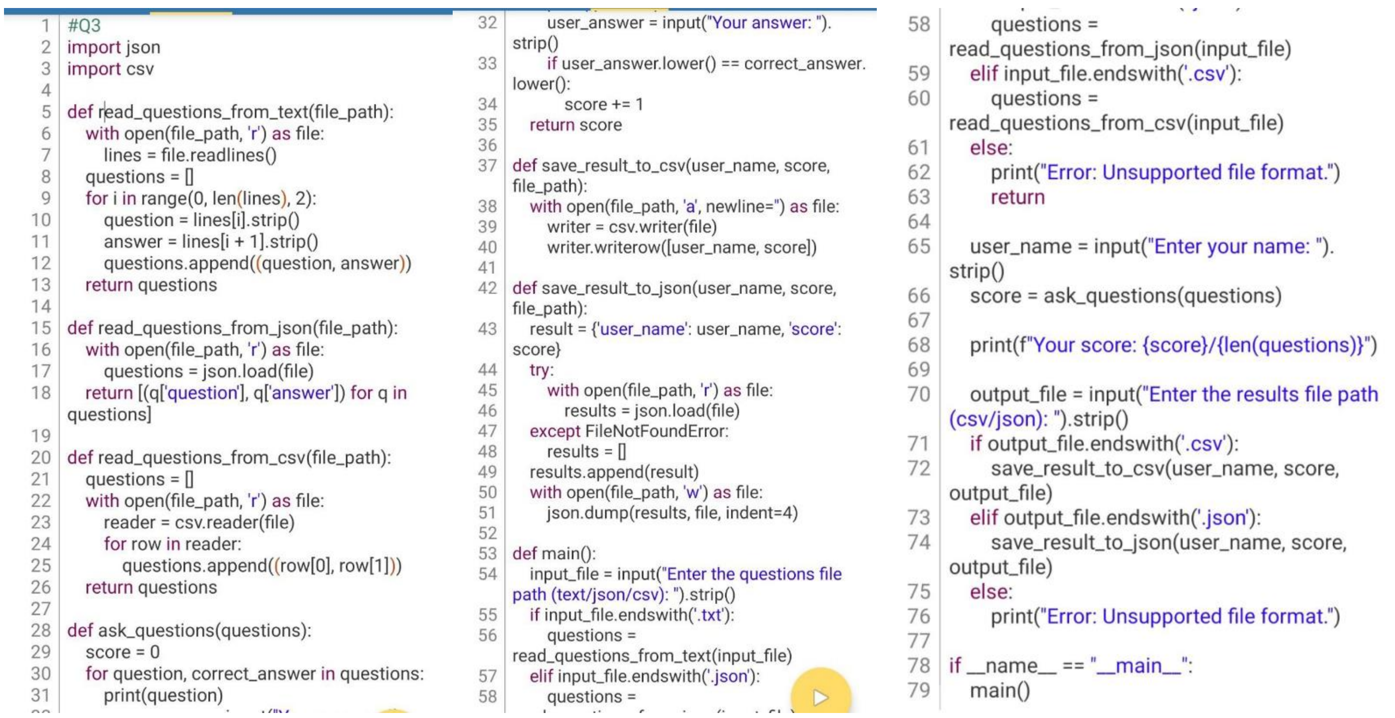
```python
#Q2
def is_valid_binary(binary_str):
    for char in binary_str:
        if char not in '01':
            return False
    return True
def binary_to_decimal(binary_str):
    decimal_number = 0
    power = 0
    for digit in reversed(binary_str):
        decimal_number += int(digit) * (2 ** power)
        power += 1
    return decimal_number

def main():
    binary_str = input("Enter a binary number: ")
    if not is_valid_binary(binary_str):
        print("Error: Invalid binary number. Please enter a number containing only 0s and 1s.")
        return
    decimal_number = binary_to_decimal(binary_str)
    print(f"The decimal equivalent of binary {binary_str} is {decimal_number}.")
if __name__ == "__main__":
    main()
```



```
Enter a binary number: 0011
The decimal equivalent of binary 0011 is 3.

[Program finished]
```

**Question 3**: Working with Files" Quiz Program"

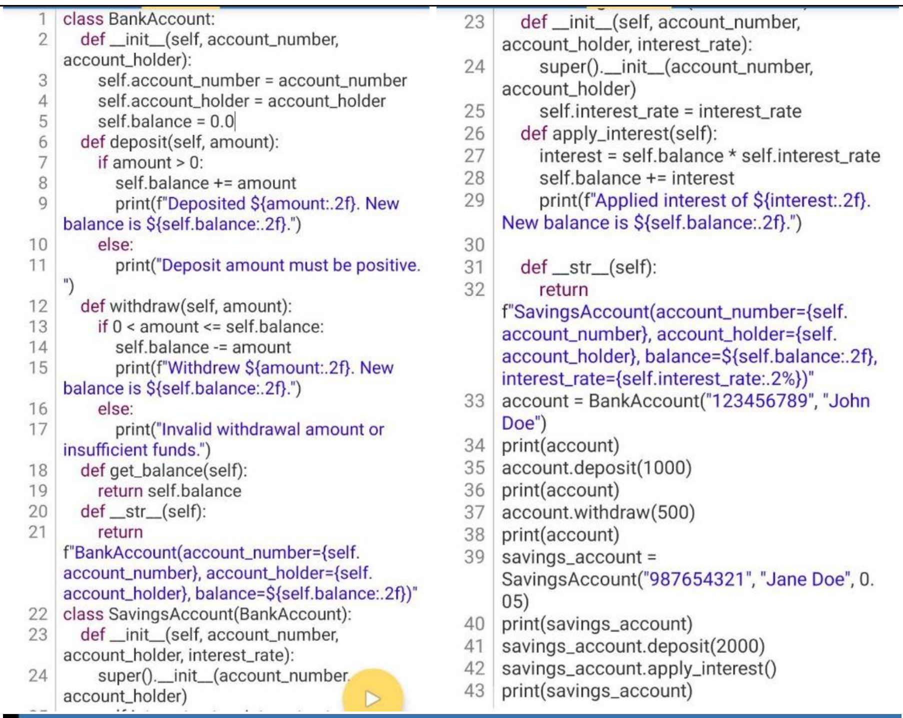
Type python quiz program that takes a text or json or csv file as input for (20 (Questions, Answers)). It asks the questions and finally computes and prints user results and store user name and result in separate file csv or json file.



```python
#Q3
import json
import csv

def read_questions_from_text(file_path):
    with open(file_path, 'r') as file:
        lines = file.readlines()
    questions = []
    for i in range(0, len(lines), 2):
        question = lines[i].strip()
        answer = lines[i + 1].strip()
        questions.append((question, answer))
    return questions

def read_questions_from_json(file_path):
    with open(file_path, 'r') as file:
        questions = json.load(file)
    return [(q['question'], q['answer']) for q in questions]

def read_questions_from_csv(file_path):
    questions = []
    with open(file_path, 'r') as file:
        reader = csv.reader(file)
        for row in reader:
            questions.append((row[0], row[1]))
    return questions

def ask_questions(questions):
    score = 0
    for question, correct_answer in questions:
        print(question)
        user_answer = input("Your answer: ").strip()
        if user_answer.lower() == correct_answer.lower():
            score += 1
    return score

def save_result_to_csv(user_name, score, file_path):
    with open(file_path, 'a', newline='') as file:
        writer = csv.writer(file)
        writer.writerow([user_name, score])

def save_result_to_json(user_name, score, file_path):
    result = {'user_name': user_name, 'score': score}
    try:
        with open(file_path, 'r') as file:
            results = json.load(file)
    except FileNotFoundError:
        results = []
    results.append(result)
    with open(file_path, 'w') as file:
        json.dump(results, file, indent=4)

def main():
    input_file = input("Enter the questions file path (text/json/csv): ").strip()
    if input_file.endswith('.txt'):
        questions = read_questions_from_text(input_file)
    elif input_file.endswith('.json'):
        questions = read_questions_from_json(input_file)
    elif input_file.endswith('.csv'):
        questions = read_questions_from_csv(input_file)
    else:
        print("Error: Unsupported file format.")
        return

    user_name = input("Enter your name: ").strip()
    score = ask_questions(questions)

    print(f"Your score: {score}/{len(questions)}")

    output_file = input("Enter the results file path (csv/json): ").strip()
    if output_file.endswith('.csv'):
        save_result_to_csv(user_name, score, output_file)
    elif output_file.endswith('.json'):
        save_result_to_json(user_name, score, output_file)
    else:
        print("Error: Unsupported file format.")

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

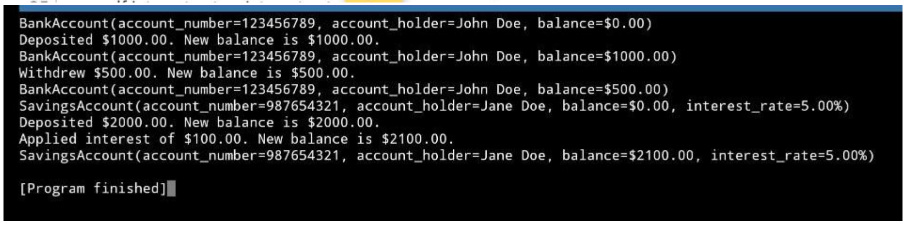
**Question 4**: Object-Oriented Programming - Bank Class
Define a class BankAccount with the following attributes and methods:
Attributes: account_number (string), account_holder (string), balance (float, initialized to 0.0)
Methods:deposit(amount), withdraw(amount) , get_balance()

- 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.



```python
class BankAccount:
  def __init__(self, account_number, account_holder):
    self.account_number = account_number
    self.account_holder = account_holder
    self.balance = 0.0
  def deposit(self, amount):
    if amount > 0:
      self.balance += amount
      print(f"Deposited ${amount:.2f}. New balance is ${self.balance:.2f}.")
    else:
      print("Deposit amount must be positive.")
  def withdraw(self, amount):
    if 0 < amount <= self.balance:
      self.balance -= amount
      print(f"Withdrew ${amount:.2f}. New balance is ${self.balance:.2f}.")
    else:
      print("Invalid withdrawal amount or insufficient funds.")
  def get_balance(self):
    return self.balance
  def __str__(self):
    return f"BankAccount(account_number={self.account_number}, account_holder={self.account_holder}, balance=${self.balance:.2f})"
class SavingsAccount(BankAccount):
  def __init__(self, account_number, account_holder, interest_rate):
    super().__init__(account_number, account_holder)
```

```python
  def __init__(self, account_number, account_holder, interest_rate):
    super().__init__(account_number, account_holder)
    self.interest_rate = interest_rate
  def apply_interest(self):
    interest = self.balance * self.interest_rate
    self.balance += interest
    print(f"Applied interest of ${interest:.2f}. New balance is ${self.balance:.2f}.")

  def __str__(self):
    return f"SavingsAccount(account_number={self.account_number}, account_holder={self.account_holder}, balance=${self.balance:.2f}, interest_rate={self.interest_rate:.2%})"
account = BankAccount("123456789", "John Doe")
print(account)
account.deposit(1000)
print(account)
account.withdraw(500)
print(account)
savings_account = SavingsAccount("987654321", "Jane Doe", 0.05)
print(savings_account)
savings_account.deposit(2000)
savings_account.apply_interest()
print(savings_account)
```



```
BankAccount(account_number=123456789, account_holder=John Doe, balance=$0.00)
Deposited $1000.00. New balance is $1000.00.
BankAccount(account_number=123456789, account_holder=John Doe, balance=$1000.00)
Withdrew $500.00. New balance is $500.00.
BankAccount(account_number=123456789, account_holder=John Doe, balance=$500.00)
SavingsAccount(account_number=987654321, account_holder=Jane Doe, balance=$0.00, interest_rate=5.00%)
Deposited $2000.00. New balance is $2000.00.
Applied interest of $100.00. New balance is $2100.00.
SavingsAccount(account_number=987654321, account_holder=Jane Doe, balance=$2100.00, interest_rate=5.00%)

[Program finished]
```

**Notes "! important"**

- Homework is accepted as **well explained** Pdf & "Nicely Formatted Code" "You can do all job in one notebook then print as pdf or "copy and paste" on word document "use" then convert into pdf with extra info " -You have to show: Question number >>Question itself>> your answer code with explanations > your Result "you can use this doc as template"

-You Have to Show code execution as Screenshots from your laptop or phone".

-Apply your full name and number, Homework number to pdf.

-Similar Solutions will rejected and not accepted.

- The Homework is accepted until the date of "27/5/2024", if after >> mark=mark- (current_date -27/5/2024)*0.3

- upload your code to your GitHub Account, "PDF + Code"