الجمهورية العربية السورية
وزارة التعليم العالي – جامعة تشرين
هندسة الاتصالات والإلكترونيات
السنة الدراسية الخامسة – الفصل الثاني

First Network Programming Homework

إعداد الطالبة :
آلاء عبد الرحمن قاسم

إشراف الدكتور :
مهند عيسى

Question 1: Python Basic

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 }

The answer is :

You can achieve this by using the `zip()` function to combine the two lists into a dictionary. Here's how you can do it :

```
python
L1 = ['HTTP', 'HTTPS', 'FTP', 'DNS']
L2 = [80, 443, 21, 53]

d = dict(zip(L1, L2))

print(d)
```

When you run this code, you will get the desired dictionary 'd' :

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

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

The answer is :

Sure! Below is a Python program that calculates the factorial a given number entered by the user :

```
python
```

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)


try:
    num = int(input("Enter a number to calculate its factorial: "))
    if num < 0:
        print("Factorial is not defined for negative numbers.")
    else:
        result = factorial(num)
        print(f"The factorial of {num} is: {result}")
except ValueError:
    print("Please enter a valid integer.")
```

In this program, the `factorial()` function recursively calculates the factorial of a number. The user is prompted to enter a number, and the program calculates the factorial of that number. It also includes error handling to handle cases where the user enters invalid input .

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 .

The answer is :

Certainly! You can achieve this by iterating over the list and using the `startswith()` method to identify items that start with the letter 'B'. Here's a Python program that does that :

```
python
L = ['Network', 'Bio', 'Programming', 'Physics', 'Music']


for item in L:
if item.startswith('B'):
print(item)
```

In this program :

We iterate over each item in the list 'L'

We use the `startswith()` method to check if each item starts with the letter 'B'

If an item starts with 'B', we print it on the screen

When you run this program with the given list, it will identify the items that start with the letter 'B' and print them on the screen .

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}

The answer is :

You can generate the dictionary `d` using dictionary comprehension in Python. Here's how you can create the dictionary with the specified key-value pairs :

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

```
print(d)
```

When you run this code, you will get the dictionary ` d ` as follows :

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

This dictionary comprehension creates a dictionary where the keys range from 0 to 10, and the values are one more than the corresponding key .

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 error .

The answer is :

Certainly! Below is a Python program that converts a binary number entered by the user into its equivalent decimal number. It includes error handling to handle input errors .

```
python

def binary_to_decimal(binary):

   decimal = 0

   for digit in binary:
       if digit not in ['0', '1']:
        return None
       decimal = decimal * 2 + int(digit)
   return decimal


try:
   binary_number = input("Enter a binary number: ")
   decimal_number = binary_to_decimal(binary_number)


   if decimal_number is not None:
       print(f"The decimal equivalent of {binary_number} is: {decimal_number}")
   else:
       print("Invalid binary number. Please enter a binary number containing only 0s and 1s.")
except ValueError:
   print("Invalid input. Please enter a valid binary number.")
```

In this program :

-The `binary_to_decimal()` function converts a binary number to its decimal equivalent

-The program prompts the user to enter a binary number

-It then calls the `binary_to_decimal()` function to convert the binary number to decimal

-If the input binary number contains any digit other than '0' or '1', the program handles this as an error

-The program displays the equivalent decimal number if the conversion is successful, or an error message if the input is invalid .

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 .

The answer is :

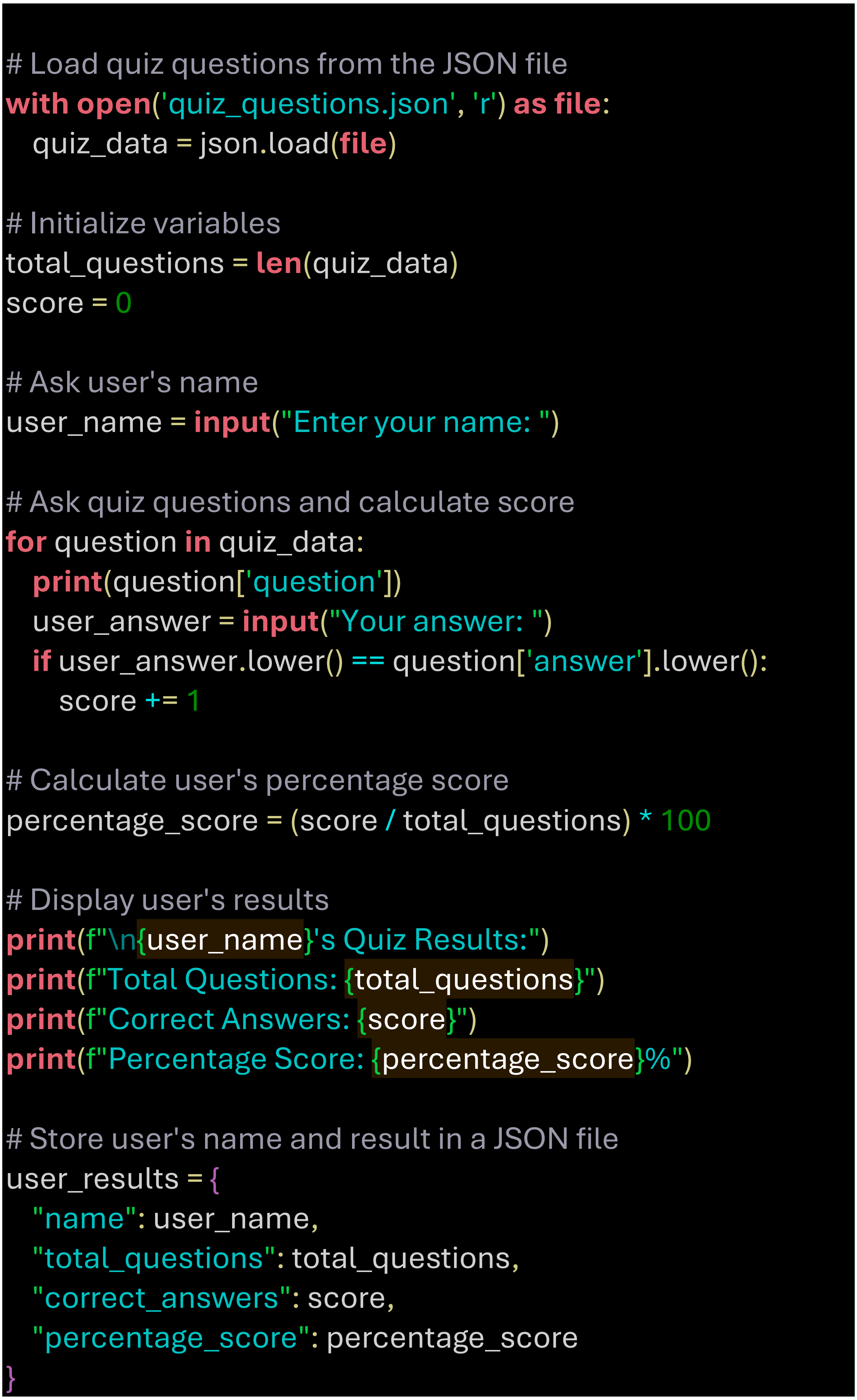
Sure! Here's a Python quiz program that reads quiz questions from a JSON file, asks the user to answer them, computes and prints the user's results, and stores the user's name and result in a separate JSON file:

Here's the step-by-step breakdown of the program:

1. Create a JSON file named "quiz_questions.json" containing 20 questions and their correct answers.

2. Create a Python program that reads the questions from the JSON file, asks the user to answer them, computes the user's score, and stores the user's name and result in a JSON file named "user_results.json".

```
python
import json
```



```
# Load quiz questions from the JSON file
with open('quiz_questions.json', 'r') as file:
    quiz_data = json.load(file)

# Initialize variables
total_questions = len(quiz_data)
score = 0

# Ask user's name
user_name = input("Enter your name: ")

# Ask quiz questions and calculate score
for question in quiz_data:
    print(question['question'])
    user_answer = input("Your answer: ")
    if user_answer.lower() == question['answer'].lower():
        score += 1

# Calculate user's percentage score
percentage_score = (score / total_questions) * 100

# Display user's results
print(f"\n{user_name}'s Quiz Results:")
print(f"Total Questions: {total_questions}")
print(f"Correct Answers: {score}")
print(f"Percentage Score: {percentage_score}%")

# Store user's name and result in a JSON file
user_results = {
    "name": user_name,
    "total_questions": total_questions,
    "correct_answers": score,
    "percentage_score": percentage_score
}
```

```
with open('user_results.json', 'w') as file:
    json.dump(user_results, file, indent=4)

print("User results have been saved in user_results.json file.")
```

In this program :

- The quiz questions are stored in a JSON file named "quiz_questions.json".

- The program reads the questions from the JSON file, asks the user to answer them, and calculates the user's score.

- After the quiz is completed, the user's name, total questions, correct answers, and percentage score are displayed.

- The user's name and results are stored in a JSON file named "user_results.json".

Make sure to create the necessary JSON files with the questions and answers before running this program.

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 .

The answer is :

Below is the implementation of the BankAccount class and its subclass SavingsAccount as per the requirements provided :

```
python

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
        return self.balance

    def withdraw(self, amount):
        if self.balance >= amount:
            self.balance -= amount
            return self.balance
        else:
            return "Insufficient funds"

    def get_balance(self):
        return self.balance

class SavingsAccount(BankAccount):
    def __init__(self, account_number, account_holder, interest_rate, balance=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
        return interest_amount

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

# Create an instance of BankAccount
bank_acc = BankAccount("123456", "John Doe")
print("Bank Account Details:")
print(f"Initial Balance: ${bank_acc.get_balance()}")
bank_acc.deposit(1000)
print(f"Balance after depositing $1000: ${bank_acc.get_balance()}")
bank_acc.withdraw(500)
print(f"Balance after withdrawing $500: ${bank_acc.get_balance()}")

# Create an instance of SavingsAccount
```

```
savings_acc = SavingsAccount("789012", "Jane Smith", 2.5)

print("\nSavings Account Details:")

print(f"Initial Balance: ${savings_acc.get_balance()}")

savings_acc.deposit(1500)

savings_acc.apply_interest()

print("Balance after applying interest:")

savings_acc.print()
```

In this implementatin :

-The `BankAccount` class represents a basic bank account with attributes `account_number`, `account_holder`, and `balance`, along with methods for depositing, withdrawing, and getting the balance .

-The `SavingsAccount` subclass inherits from `BankAccount` and adds an `interest_rate` attribute and an `apply_interest()` method to apply interest to the balance based on the interest rate. It also overrides the `print()` method to display account details .

-Instances of both `BankAccount` and `SavingsAccount` classes are created with operations performed as requested .

The solution is over