Republic Arab Syrian

University Tishreen

Department of Communication and electrical engineering

الجمهورية العربية السورية

اللاذقية -جامعـــة تشريـــن

كلية الهندسة الكهربائية والميكانيكية

قسم هندسة الاتصالات والاكترونيات السنة الخامية

5 th , Network Programming : Homework No1

وظيفة1 برمجة شبكات

الاسم: سحر الطبق الرقم الجامعي:2275

السؤال الأول :

A –

```
In [1]: L1 = ['HTTP', 'HTTPS', 'FTP', 'DNS']
        L2 = [80, 443, 21, 53]
        d = {}
        for key, value in zip(L1, L2):
            d[key] = value
        print(d)

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

هنا نقوم بالتعامل مع قامتين **"list"** الأولى نجوي محارف وسوف تكون ال **key** والأخرى عبارة عن قيم **value** يتم جمع كل قيمة ومقابلتها من المفاتيح من خلال المرور بالحلقة وكل مرة ننفذ التعليمة **d[key] = value** ونطبع بالنهاية **D**

B – حساب العاملي :

```
In [2]: number = int(input("أدخل عددًا لحساب عامليه: "))
        factorial = 1
        for i in range(1, number + 1):
            factorial *= i
        print(f"عاملي العدد {number} هو: {factorial}")

        أدخل عددًا لحساب عامليه: 10
        عاملي العدد 10 هو: 3628800
```

نطلب من المستخدم ادخال القيمة المراد حساب العاملي لها ونعطي قيمة بدائية للعاملي واحد **"1"** بعدها يتم تنفيذ العملية الحسابية **factorial *= i** بعدد مرات القيمة التي ادخلناها ونطبع قيمة العاملي

C - **إيجاد المحرف "B" ضمن القائمة "list" :**

```
In [3]: L = ['Network', 'Bio', 'Programming', 'Physics', 'Music']
        B_items = [item for item in L if item.startswith('B')]
        print("العناصر التي تبدأ بحرف 'B':", B_items)
```

```
العناصر التي تبدأ بحرف 'B': ['Bio']
```

لدينا القائمة **"L"** يتم المرور على كل **item** من القائمة وفحص اذا تبدا بالحرف **"B" startswith** من خلال التعليمة التالية

items = [item for item in L if item.startswith('B')]

– D

```
In [5]: 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}
```

تتم انشاء القاموس بحيث يمر على ال**"key"** و القيمة **"value"** نفس العداد في الحلقة **for** كما نرى في التعليمة و بعدها تتم طباعة النتيجة

السؤال الثاني :

التحويل من ثنائي الى عشري

In [1]:
```
def binary_to_decimal(binary_str):
    # This function converts a binary string to a decimal number.
    decimal_number = 0
    length = len(binary_str)

    for i in range(length):
        # Convert each character to an integer (0 or 1)
        bit = int(binary_str[i])
        # Calculate its value in decimal and add it to the total
        decimal_number += bit * (2 ** (length - 1 - i))

    return decimal_number

def is_binary_string(s):
    # This function checks if a given string is a valid binary number
    for char in s:
        if char not in '01':
            return False
    return True

def main():
    while True:
        binary_str = input("Enter a binary number: ")

        if is_binary_string(binary_str):
            decimal_number = binary_to_decimal(binary_str)
            print(f"The decimal equivalent of binary {binary_str} is {decimal_number}.")
            break
        else:
            print("Invalid input. Please enter a valid binary number (containing only 0 and 1).")

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

```
Enter a binary number: 1011
The decimal equivalent of binary 1011 is 11.
```

الدالة binary_to_decimal **يتم إعطائها التسلسل للرقم الثنائي المطلوب تحويله أولا يتم حساب طول السلسلة بعدها يقوم بتحويل هذه السلسلة الى بتات رقمية من خلال التعليمة** bit = int(binary_str[i]) **بعدها من خلال التعليمة التالية يتم حسابه للرقم العشري بالطريقة التالية** decimal_number += bit * (2 ** (length - 1 - i)) **ويتم التكرار من اجل كل بت**

الدالة is_binary_string **تكشف اذا كانت السلسلة هي رقم ثنائي او لا والبرنامج الرئيسي يطلب من المستخدم ادخال القيمة الثنائية لتحويلها يتم اختبار اذا القيمة المدخلة ثنائية اذا نعم يتم تحويلها اذا لا يتم طباعة العبارة التالية** Invalid input. Please enter a valid binary number (containing only 0 and 1).

السؤال الثالث :

```
In [9]: import json
        import csv
        import os

        def load_quiz(file_path):
            if file_path.endswith('.json'):
                with open(file_path, 'r') as file:
                    return json.load(file)
            elif file_path.endswith('.csv'):
                questions = []
                with open(file_path, 'r') as file:
                    reader = csv.reader(file)
                    for row in reader:
                        if len(row) == 2:
                            questions.append({'question': row[0], 'answer': row[1]})
                return questions
            elif file_path.endswith('.txt'):
                questions = []
                with open(file_path, 'r') as file:
                    lines = file.readlines()
                    for i in range(0, len(lines), 2):
                        questions.append({'question': lines[i].strip(), 'answer': lines[i+1].strip()})
                return questions
            else:
                raise ValueError("Unsupported file format")

        def take_quiz(questions):
            score = 0
            for question in questions:
                print(question['question'])
                answer = input("Enter answer: ")
                if answer.lower() == question['answer'].lower():
                    score += 1
            return score

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

الدالة الأولى **load_quiz** تقوم بتحميل الملف الذي يحوي الاسئلة بغض النظر عن الملف المراد القراءة منه كما هو موضح اما الدالة الثانية **take_quiz** تقوم بعملية حساب النتيجة النهائية للاختبار

بالإضافة للدالة الثالثة التي تقوم بتسجيل النتائج في ملف **json**

```python
# Load questions from a file
questions_file = input("Enter the path to the questions file (text, json, csv): ").strip()
questions = load_quiz(questions_file)

# Conduct the quiz
user_score = take_quiz(questions)

# Get user name and save results
user_name = input("Enter your name: ")
results_file = 'result.csv'
file_exists = os.path.isfile(results_file)

with open(results_file, "a", newline='') as file:
    writer = csv.writer(file)
    if not file_exists:
        writer.writerow(['username', 'score'])
    writer.writerow([user_name, user_score])

print("Result is:", user_score)
```

```
Enter the path to the questions file (text, json, csv): quiz.json
what's your name
 answer is: sahar
3=
 answer is: 3
1-1=
 answer is: 0
0=
 answer is: 0
3-3
 answer is: 0
0+0=
 answer is: 0
2-2=
```

هنا يطلب من المستخدم ادخال نوع الملف بعدها يتم البدء بالاختبار وفي النهاية يقوم بحفظ النتيجة بملف **json**

النتائج التي ظهرت بالشكل التالي

```
3-3
 answer is: 0
0+0=
 answer is: 0
2-2=
 answer is: 0
1+5=
 answer is: 6
10*10=
 answer is: 100
1/1=
 answer is: 1
1-1=
 answer is: 0
9-8=
 answer is: 1
1+1=
 answer is: 2
7*7=
 answer is: 40
1-1=
 answer is: 1
2*2=
 answer is: 4
3==5
 answer is: 4
4*2=
 answer is: 8
1*1=
 answer is: 1
is it good
 answer is: yes
Enter your name: sahar
Result is: 17
```

```
[
    {
        "username": "sahar",
        "score": 17
    }
]
```

السؤال الرابع :

class BankAccount:

```
In [4]: 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):
                if amount > 0:
                    self.balance += amount
                    print(f"Deposited ${amount:.2f}. New balance: ${self.balance:.2f}")

            def withdraw(self, amount):
                if amount > 0 and amount <= self.balance:
                    self.balance -= amount
                    print(f"Withdrew ${amount:.2f}. New balance: ${self.balance:.2f}")

            def get_balance(self):
                return self.balance

        class SavingsAccount(BankAccount):
            def __init__(self, account_number, account_holder, interest_rate=0.05):
                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"Interest applied: ${interest:.2f}. New balance: ${self.balance:.2f}")
```

صنف الحساب البنكي BankAccount تم تعريف الدوال اللازمة لعملية السحب والايداع من الحساب وأيضا على معرفة الرصيد الحالي للحساب

الصنف الثاني SavingsAccount وهو الصنف الابن من صنف حساب البنك هو حساب بنكي عادي ولكنه توفيري أي يحوي نفس المعلومات ولكن هنا يتم تطبيق الفائدة ويحوي على نفس عمليات السحب والايداع

التطبيق العملي

```python
# Create a BankAccount instance
bank_account = BankAccount("12345678", "sahar")

# Deposit $1000
bank_account.deposit(1000)

# Withdraw $500
bank_account.withdraw(500)

# Print the current balance
print(f"Current balance: ${bank_account.get_balance():.2f}")

# Create a SavingsAccount instance
savings_account = SavingsAccount("87654321", "sahar")  # 5% interest rate

# Apply interest
savings_account.apply_interest()

# Print the account details
print(savings_account)
```

```
Deposited $1000.00. New balance: $1000.00
Withdrew $500.00. New balance: $500.00
Current balance: $500.00
Interest applied: $0.00. New balance: $0.00
<__main__.SavingsAccount object at 0x000001E565765520>
```