Syrian Arab Republic

Lattakia - Tishreen University

Department of Communication and electrical engineering

5th , Network Programming : Homework No1

الجمهورية العربية السورية

اللاذقية -جامعــة تشريـــن

كلية الهندسة الكهربائية والميكانيكية

قسم هندسة الاتصالات والالكترونيات

السنة الخامسة: وظيفة1 برمجة شبكات

زهراء أنيس حبيب ٢٧٨٨

السؤال 1: أساسيات بايثون

A- تحويل القائمتين إلى قاموس

```
File  Edit  Format  Run  Options  Window  Help
L1 = ['HTTP', 'HTTPS', 'FTP', 'DNS']
L2 = [80, 443, 21, 53]

d = dict(zip(L1, L2))
print(d)
```

النتيجة ستكون:

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

B- برنامج لحساب المضروب (العاملة)

```
File  Edit  Format  Run  Options  Window  Help
def factorial(n):
  f=1
  if n == 0:
      return 1
  else:
    while n>0:
      f=f*n
      n-=1

    return f
while True:
  num = int(input("أدخل عددًا لحساب المضروب: "))
  print(f"مضروب {num} هو {factorial(num)}")
  s=input("Do you want to continue, y or n").lower()
  if s=="n":
    break
```

النتيجة ستكون:

```
أدخل عددًا لحساب المضروب: 1
1 هو 1 مضروب
Do you want to continue, y or ny
أدخل عددًا لحساب المضروب: 2
2 هو 2 مضروب
Do you want to continue, y or ny
أدخل عددًا لحساب المضروب: 3
6 هو 3 مضروب
Do you want to continue, y or ny
أدخل عددًا لحساب المضروب: 4
24 هو 4 مضروب
Do you want to continue, y or ny
أدخل عددًا لحساب المضروب: 5
120 هو 5 مضروب
Do you want to continue, y or ny
أدخل عددًا لحساب المضروب: 6
720 هو 6 مضروب
Do you want to continue, y or nn
>>>
```

C- برنامج لتحديد العناصر التي تبدأ بحرف 'B'

```
File  Edit  Format  Run  Options  Window  Help
L = ['Network', 'Bio', 'Programming', 'Physics', 'Music']

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

```

النتيجة ستكون:

```
Bio
>>>
```

D- توليد قاموس

توليد القاموس المطلوب باستخدام (Dictionary Comprehension)

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

النتيجة ستكون:

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

السؤال 2

لإنشاء برنامج يحول رقم ثنائي إلى ما يعادله من الأرقام العشرية، يجب القيام الخطوات التالية:

1. طلب إدخال الرقم الثنائي من المستخدم.
2. التحقق من صحة الإدخال للتأكد من أنه رقم ثنائي صحيح.
3. تحويل الرقم الثنائي إلى رقم عشري.
4. عرض النتيجة.

File Edit Format Run Options Window Help

```
def is_valid_binary(binary_str):
    """التحقق مما إذا كانت السلسلة المدخلة رقم ثنائي صالح."""
    for char in binary_str:
        if char not in ('0', '1'):
            return False
    return True

def binary_to_decimal(binary_str):
    """تحويل سلسلة ثنائية إلى رقم عشري."""
    decimal_number = 0
    binary_str = binary_str[::-1]  # عكس السلسلة لتسهيل الحساب
    for i in range(len(binary_str)):
        decimal_number += int(binary_str[i]) * (2 ** i)
    return decimal_number

def main():
    while True:
        binary_str = input("أدخل رقمًا ثنائيًا: ").strip()
        if is_valid_binary(binary_str):
            decimal_number = binary_to_decimal(binary_str)
            print(f"المكافئ العشري للرقم الثنائي {binary_str} هو {decimal_number
            break
        else:
            print("(1 و 0 يحتوي فقط على) خال غير صالح. يرجى إدخال رقم ثنائي صحيح

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

الشرح:

1. **التحقق من صلاحية الإدخال is_valid_binary**
   - هذا التابع يتحقق مما إذا كانت السلسلة المدخلة تحتوي فقط على الأحرف '0' و '1'.
2. **تحويل الثنائي إلى عشري binary_to_decimal**
   - يتم عكس السلسلة الثنائية لتسهيل عملية التحويل باستخدام الصيغ الرياضية.
   - يتم تحويل كل رقم ثنائي إلى ما يعادله عشرياً باستخدام الصيغة الرقم * 2^الموضع ويتم إضافته إلى المجموع الكلي.
3. **الحلقة الرئيسية للبرنامج:**
   - يقوم البرنامج بطلب إدخال المستخدم بشكل مستمر حتى يتم إدخال رقم ثنائي صالح.
   - عند استلام رقم ثنائي صالح يقوم البرنامج بتحويله إلى رقم عشري ويعرض النتيجة

النتيجة:

```
أدخل رقمًا ثنائيًا: 5
إدخال غير صالح. يرجى إدخال رقم ثنائي صحيح (يحتوي فقط على 0 و 1).
أدخل رقمًا ثنائيًا: 11
المكافئ العشري للرقم الثنائي 11 هو 3.
>

============================== RESTART: C:\h2\2.py ==============================
أدخل رقمًا ثنائيًا: ee
إدخال غير صالح. يرجى إدخال رقم ثنائي صحيح (يحتوي فقط على 0 و 1).
أدخل رقمًا ثنائيًا: 454
إدخال غير صالح. يرجى إدخال رقم ثنائي صحيح (يحتوي فقط على 0 و 1).
أدخل رقمًا ثنائيًا: 10001
المكافئ العشري للرقم الثنائي 10001 هو 17.
>
```

السؤال 3

برنامج قراءة أسئلة وأجوبة من ملف JSON، طرح الأسئلة على المستخدم، حساب النتيجة النهائية، وتخزين اسم المستخدم والنتيجة في ملف منفصل بصيغةJSON

الخطوات

1. **إنشاء ملف الأسئلة والأجوبة questions.json**
2. **قراءة الملف**

3. **طرح الأسئلة على المستخدم**
   - سنقوم بطرح الأسئلة على المستخدم وجمع الإجابات.
4. **حساب النتيجة النهائية**
   - سنقوم بحساب عدد الإجابات الصحيحة.
5. **تخزين النتائج:**
   - سنقوم بتخزين النتائج بملف results.json

الكود

File Edit Format Run Options Window Help

```
import json
import os

def read_questions_from_json(file_path):
    """اقرأ الأسئلة والأجوبة من ملف JSON."""
    with open(file_path, 'r', encoding='utf-8') as file:
        data = json.load(file)
        return [(item['question'], item['answer']) for item in data]

def ask_questions(questions):
    """اطرح الأسئلة على المستخدم وجمع الإجابات."""
    correct_answers = 0
    for question, correct_answer in questions:
        user_answer = input(f"{question}? ").strip()
        if user_answer == correct_answer:
            correct_answers += 1
    return correct_answers

def save_results_to_json(file_path, username, score, total_questions):
    """احفظ اسم المستخدم والنتيجة في ملف بصيغة JSON."""
    result = {
        "username": username,
        "score": score,
        "total_questions": total_questions
    }

    if not os.path.isfile(file_path):
        with open(file_path, 'w', encoding='utf-8') as file:
            json.dump([result], file, ensure_ascii=False, indent=4)
    else:
        try:
            with open(file_path, 'r+', encoding='utf-8') as file:
                try:
                    data = json.load(file)
                except json.JSONDecodeError:
                    data = []
                data.append(result)
                file.seek(0)
                json.dump(data, file, ensure_ascii=False, indent=4)
        except FileNotFoundError:
            with open(file_path, 'w', encoding='utf-8') as file:
                json.dump([result], file, ensure_ascii=False, indent=4)

def main():
    questions_file = 'questions.json'
    questions = read_questions_from_json(questions_file)

    username = input("أدخل اسم المستخدم: ").strip()
    correct_answers = ask_questions(questions)
    total_questions = len(questions)

    print(f"{username}, لقد أجبت بشكل صحيح على {correct_answers} من أصل {total_questions} سؤال.")

    results_file = 'results.json'
    save_results_to_json(results_file, username, correct_answers, total_questions)

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

يتكون الكود من أربعة توابع رئيسية. التابع الأول، read_questions_from_json، يقرأ الأسئلة والأجوبة من ملف JSON ويعيد قائمة من الأزواج (السؤال والإجابة الصحيحة). التابع الثاني، ask_questions، يطرح الأسئلة على المستخدم ويجمع الإجابات الصحيحة، حيث يتم مقارنة إجابة المستخدم بالإجابة الصحيحة وزيادة العداد إذا كانت الإجابة صحيحة. التابع الثالث، save_results_to_json، يحفظ اسم المستخدم ونتيجته في ملف .JSON إذا كان الملف موجوداً بالفعل، فإن التابع يقرأ محتوياته، يضيف النتيجة الجديدة، ثم يعيد كتابة الملف. إذا كان الملف غير موجود، يقوم بإنشائه وكتابة النتيجة الجديدة مباشرة. التابع الأخير، main، هو التابع الرئيسي الذي ينسق بين التوابع الأخرى: يقرأ الأسئلة، يطلب من المستخدم إدخال اسمه، يطرح الأسئلة، ثم يعرض النتيجة ويحفظها في ملف JSON يُنفذ التابع main عند تشغيل الملف.

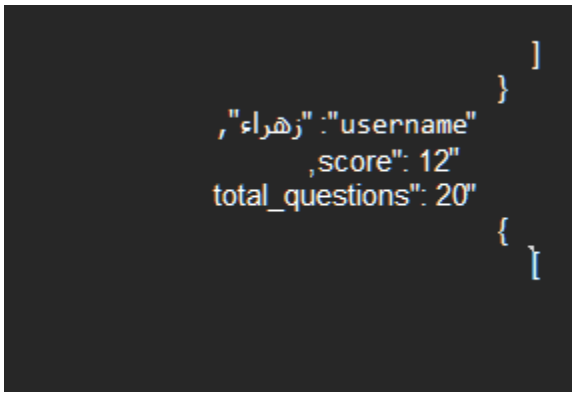
questions.json


```
]
,{"answer": "8" ,"3 + 5 ما هو" :"question"}
,{"answer": "5" ,"2 - 7 ما هو" :"question"}
,{"answer": "18" ,"3 * 6 ما هو" :"question"}
,{"answer": "3" ,"4 / 12 ما هو" :"question"}
,{"answer": "10" ,"1 + 9 ما هو" :"question"}
,{"answer": "4" ,"4 - 8 ما هو" :"question"}
,{"answer": "14" ,"2 * 7 ما هو" :"question"}
,{"answer": "8" ,"2 / 16 ما هو" :"question"}
,{"answer": "15" ,"5 + 10 ما هو" :"question"}
,{"answer": "8" ,"3 - 11 ما هو" :"question"}
,{"answer": "16" ,"4 * 4 ما هو" :"question"}
,{"answer": "4" ,"5 / 20 ما هو" :"question"}
,{"answer": "20" ,"6 + 14 ما هو" :"question"}
,{"answer": "11" ,"7 - 18 ما هو" :"question"}
,{"answer": "15" ,"5 * 3 ما هو" :"question"}
,{"answer": "5" ,"5 / 25 ما هو" :"question"}
,{"answer": "20" ,"7 + 13 ما هو" :"question"}
,{"answer": "10" ,"9 - 19 ما هو" :"question"}
,{"answer": "16" ,"8 * 2 ما هو" :"question"}
{"answer": "4" ,"6 / 24 ما هو" :"question"}
[
```

النتيجة:

```
---------------------------------- RESTART: C:\n2\3.py ---
أدخل اسم المستخدم: زهراء
3 + 5 هو ما? 8
2 - 7 هو ما? 5
3 * 6 هو ما? 18
4 / 12 هو ما? 3
1 + 9 هو ما? 10
4 - 8 هو ما? 4
2 * 7 هو ما? 14
2 / 16 هو ما? 8
5 + 10 هو ما? 15
3 - 11 هو ما? 8
4 * 4 هو ما? 16
5 / 20 هو ما? 4
6 + 14 هو ما? 0
7 - 18 هو ما? 0
5 * 3 هو ما? 0
5 / 25 هو ما? 0
7 + 13 هو ما? 0
9 - 19 هو ما? 0
8 * 2 هو ما? 0
6 / 24 هو ما? 0
زهراء, لقد أجبت بشكل صحيح على 12 من أصل 20 سؤال.
>>
```

results.json



السؤال 4: كلاس حساب بنكي

```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):
        self.balance += amount
        print(f"Deposited: ${amount:.2f}")
        print(f"Current Balance: ${self.balance:.2f}")

    def withdraw(self, amount):
        if amount > self.balance:
            print("Insufficient balance!")
        else:
            self.balance -= amount
            print(f"Withdrew: ${amount:.2f}")
            print(f"Current Balance: ${self.balance:.2f}")

    def get_balance(self):
        return self.balance

    def __str__(self):
        return f"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)
        self.interest_rate = interest_rate

    def apply_interest(self):
        interest = self.balance * (self.interest_rate / 100)
        self.balance += interest
        print(f"Interest Applied: ${interest:.2f}")
        print(f"New Balance: ${self.balance:.2f}")

    def __str__(self):
        return f"Account Holder: {self.account_holder}, Balance: ${self.balance:.2f}, Interest Rate: {self.interest_rate}%"

# إنشاء كائن من فئة BankAccount
account = BankAccount("2788", "زهراء")

# إجراء عملية إيداع بمبلغ 1000 دولار
account.deposit(1000)

# إجراء عملية سحب بمبلغ 500 دولار
account.withdraw(500)

print(account)

# إنشاء كائن من فئة SavingsAccount
savings = SavingsAccount("2789", "زهراء حبيب", 35)

# إجراء عملية إيداع بمبلغ 1000 دولار
savings.deposit(1000)

# تطبيق الفائدة وطباعة الرصيد الحالي والمعدل
savings.apply_interest()
print(savings)
```

نقوم بتعريف كلاس BankAccount الذي يمثل حساباً بنكياً، ثم نقوم بإنشاء كائن من هذا الكلاس وإجراء عمليات إيداع وسحب. يحتوي الكلاس على الخصائص account_number رقم الحساب، account_holder(اسم صاحب الحساب)، و balance (الرصيد) الذي يتم تهيئته بصفر. يتضمن الكلاس أيضاً التوابع التالية deposit(amount) :لإيداع مبلغ في الحساب، withdraw(amount)لسحب مبلغ من الحساب مع التحقق من كفاية الرصيد، get_balance()للحصول على الرصيد الحالي، و __str__() لطباعة معلومات الحساب بشكل منسق. بعد ذلك، نقوم بإنشاء كائن منBankAccount ، إجراء عمليات إيداع وسحب، وطباعة الرصيد بعد كل عملية. ثم نقوم بتعريف الكلاس الابن SavingsAccount الذي يرث من BankAccount ويضيف خاصية interest_rate (معدل الفائدة) وتابع apply_interest()الذي يطبق الفائدة على الرصيد. تم إعادة تعريف التابع __str__() في الكلاس الابن لطباعة معلومات الحساب مع معدل الفائدة. في النهاية، نقوم بإنشاء كائن منSavingsAccount ، إجراء عملية إيداع، تطبيق الفائدة، وطباعة الرصيد مع المعدل..

النتيجة:

```
Deposited: $1000.00
Current Balance: $1000.00
Withdrew: $500.00
Current Balance: $500.00
Account Holder: زهراء, Balance: $500.00
Deposited: $1000.00
Current Balance: $1000.00
Interest Applied: $350.00
New Balance: $1350.00
Account Holder: زهراء حبيب, Balance: $1350.00, Interest Rate: 35%
>>>
```