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

Name:ناصر أكثم حسن , Number 2474

First Network Programming Homework

Question 1:

A:

```
L1 = ['HTTP', 'HTTPS', 'FTP', 'DNS']
L2 = [80, 443, 21, 53]
d = {}
for i in range(len(L1)):
    d[L1[i]] = L2[i]
print(d)
```

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

B:

```
def fa(a):
    f = 1
    while a>0:
        f *= a
        a-=1
    return f
while True:
    num = int(input("Enter number "))
    if num==-2:
        break
    print("The factorial is ",fa(num))
```

```
Enter number 1
The factorial is  1
Enter number 2
The factorial is  2
Enter number 3
The factorial is  6
Enter number 4
The factorial is  24
Enter number 5
The factorial is  120
Enter number -2
```

C:

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

```
Bio
```

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}
```

Question 2:

```
def b_to_d(x):
    # تحقق من أن الإدخال يحتوي فقط على الأرقام 0 و 1
    for i in x:
        if i not in "10":
            return "ادخال خاطئ"

    d= 0
    b_reversed = x[::-1]
    index = 0

    while index < len(b_reversed):
        d += int(b_reversed[index]) * (2 ** index)
        index += 1

    return d
while True:
    b = input("Enter binary number or a to stop : ")
    if b == "a":
        break
    print(b to d(b))
```

قمنا بكتابة برنامج لتحويل الأرقام الثنائية إلى عشرية، حيث يحتوي هذا البرنامج على تابعb_to_d الذي يقوم بعملية التحويل، بالإضافة إلى حلقة رئيسية تطلب من المستخدم إدخال رقم ثنائي للاستمرار أو حرف "a" للتوقف.

1 شرح الكود

1. **التحقق من صحة الإدخال:**
   في تابع b_to_d، قمنا أولاً بالتحقق من أن الإدخال يحتوي فقط على الأرقام "0" و "1". إذا وجدنا أي حرف آخر، نرجع رسالة "ادخال خاطئ."

2. **تحويل الرقم الثنائي إلى عشري:**
إذا كان الإدخال صحيحًا، نقوم بعملية التحويل. أولًا، عكسنا ترتيب الأرقام الثنائية لتسهيل عملية الحساب باستخدام الأسس. ثم استخدمنا حلقة whileلحساب القيمة العشرية، حيث نضرب كل رقم في 2 مرفوع للأساس المناسب له (أي موقعه في السلسلة العكسية) ونضيف النتيجة إلى المتغير .d

3. **الحلقة الرئيسية:**
في الحلقة الرئيسية، طلبنا من المستخدم إدخال رقم ثنائي أو حرف "a" للتوقف. إذا أدخل المستخدم"a" ، نخرج من الحلقة. إذا أدخل رقمًا ثنائيًا، نستخدم تابع b_to_dلتحويله إلى عشري ونعرض النتيجة. أما إذا كان الإدخال غير صحيح، نعرض رسالة "ادخال خاطئ."

2 الخرج

عند تشغيل البرنامج، قمنا بإدخال القيم التالية:

```
Enter binary number or a to stop : 111
7
Enter binary number or a to stop : 222
ادخال خاطئ
Enter binary number or a to stop : 101
5
Enter binary number or a to stop : a
```

Question 3:

```python
# قراءة الأسئلة والأجوبة من الملف النصي
def read_questions(filename):
    questions = []
    with open(filename, 'r') as file:
        for line in file:
            question, answer = line.strip().split('=')
            questions.append((question.strip(), int(answer.strip())))
    return questions

# حساب النتيجة
def calculate_score(questions):
    score = 0
    for question, correct_answer in questions:
        user_answer = int(input(f"{question} = "))
        if user_answer == correct_answer:
            score += 1
    return score

# حفظ النتيجة في ملف CSV
def save_score(filename, username, score):
    with open(filename, 'a') as file:
        file.write(f"{username},{score}\n")

# البرنامج الرئيسي
def main():
    username = input("Enter your name: ")
    questions = read_questions('questions.txt')
    score = calculate_score(questions)
    print(f"{username}, your score is: {score}/20")
    save_score('results.csv', username, score)

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

1. **قراءة الأسئلة والأجوبة من الملف النصي:**
   قمنا بكتابة تابع read_questionsالذي يأخذ اسم الملف كمدخل، ويقرأ الأسئلة والأجوبة منه. كل سطر في الملف يجب أن يكون على الشكل سؤال = جواب .ثم نقوم بإضافة كل سؤال وجوابه كزوج (tuple) إلى قائمة questionsونرجعها.

2. **حساب النتيجة:**
   تابع calculate_scoreيقوم بعرض كل سؤال للمستخدم، ويطلب منه إدخال الجواب. إذا كان جواب المستخدم صحيحاً، نزيد النتيجة بواحد. في النهاية نرجع القيمة النهائية للنتيجة.

3. **حفظ النتيجة في ملف CSV**
   تابع save_scoreيأخذ اسم الملف، واسم المستخدم، والنتيجة، ويقوم بكتابة هذه المعلومات في الملف، بحيث يتم إضافة النتيجة الجديدة إلى نهاية الملف.

4. **البرنامج الرئيسي:**
   في البرنامج الرئيسي، طلبنا من المستخدم إدخال اسمه، ثم قرأنا الأسئلة من الملف questions.txt. بعد ذلك، حسبنا النتيجة باستخدام تابع calculate_score،

وعرضنا النتيجة للمستخدم على شكل "اسم المستخدم :your score is, النتيجة/20".
أخيراً، حفظنا النتيجة في ملف results.csv باستخدام تابع save_score.

الخرج:

أجاب المستخدم "ناصر" على الأسئلة، وكانت جميع إجاباته خاطئة باستثناء السؤال الأول، مما جعله يحصل على نتيجة 1 من 20. في النهاية تم عرض النتيجة بالشكل "ناصر, your score is: 1/20" وتم حفظ النتيجة في ملف results.csv.

```
Enter your name: ناصر
5 + 3 = 8
12 - 4 = 0
6 * 7 = 0
15 / 3 = 0
9 + 10 = 0
20 - 5 = 0
8 * 2 = 0
18 / 6 = 0
7 + 7 = 0
14 - 7 = 0
5 * 5 = 0
25 / 5 = 0
11 + 2 = 0
17 - 9 = 0
3 * 3 = 0
27 / 3 = 0
4 + 4 = 0
13 - 6 = 0
2 * 8 = 0
24 / 4 = 0
ناصر, your score is: 1/20
```

	A	B	C
1	ناصر	1	
2			
3			

Question 4:

هذا الكود يعرف كلاستين: BankAccount و SavingsAccount. BankAccount هي كلاس أساسية تمثل حساب بنكي عادي، وتحتوي على خصائص مثل رقم الحساب والاسم وكذلك طرق للإيداع والسحب والحصول على الرصيد.

SavingsAccount هي كلاس مشتق من BankAccount والذي يمثل حساب توفير. هذا الكلاس له خاصية إضافية وهي معدل الفائدة، وطريقة apply_interest() التي تضيف الفائدة المستحقة إلى الرصيد. تم استخدام طريقة super().__init__() لاستدعاء التابع الباني الخاص بكلاس الأب BankAccount.

تم إنشاء كائن من نوع BankAccount باستخدام رقم الحساب والاسم، وإجراء عمليات الإيداع والسحب. بعد ذلك، تم إنشاء كائن من نوع SavingsAccount باستخدام رقم الحساب والاسم ومعدل الفائدة، وإجراء عملية الإيداع وتطبيق الفائدة. تم عرض الرصيد والمعلومات الأخرى باستخدام طرق get_balance() و __str__().

```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

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

    def get_balance(self):
        return self.balance

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):
        self.balance += self.balance * self.interest_rate

    def __str__(self):
        s=f"Current balance: ${self.balance:.2f}, Interest rate: {self.interest_rate * 100:.2f}%"
        return s

# إنشاء حساب بنك
account = BankAccount("2474", "Nasser")
account.deposit(1000)
print(f"Balance after deposit: ${account.get_balance():.2f}")
account.withdraw(500)
print(f"Balance after withdrawal: ${account.get_balance():.2f}")

# إنشاء حساب توفير
savings = SavingsAccount("5555", "Nasser", 0.05)
savings.deposit(1000)
savings.apply_interest()
print(savings)
```

```
Balance after deposit: $1000.00
Balance after withdrawal: $500.00
Current balance: $1050.00, Interest rate: 5.00%
```