الجُمْهُورِيّةُ اَلْعَرَبِيَّةُ اَلسُّورِيَّة
وِزَارَةُ التَّعْلِيمِ الْعَالِيْ وَاَلْبَحث العِلْمِيْ
جَامِعَةُ تِشرِيْن
هَنْدَسَةُ الإِتّصَالَاتِ وَالإِلكترُونِيّاتْ
السَّنَة الدّرَاسِيّة الخَامسة لِلعَام 2024-1445

وظيفة
برمجة الشبكات

إعْدَادُ الطّالِب: **عيسى ثائر ملحم**
"3355"

إشْرَاف : الدكتور مهند عيسى

Name: Essa Thaer Mlhem , Number:3355 ,Submitted To GitHub:
https://github.com/Essamlhem3355

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 **}**

```
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}
```

1. القوائم L1 و L2:

- L1 تحتوي على قائمة من البروتوكولات مثل 'HTTP', 'HTTPS', 'FTP', 'DNS'.

- L2 تحتوي على قائمة من الأرقام المتعلقة بكل بروتوكول، على سبيل المثال، 80 لـ 'HTTP', 443 لـ 'HTTPS', وهكذا.

2. القاموس d:

- يتم استخدام تعبير التكرار (dictionary comprehension) لإنشاء القاموس d.

- يتم استخدام دالة zip() لدمج القوائم L1 و L2 معًا، حيث يتم إنشاء أزواج المفتاح والقيمة.

- بعد ذلك، يتم استخدام هذه الأزواج لإنشاء القاموس، حيث يكون كل عنصر في L1 هو المفتاح والعنصر المتوافق في L2 هو القيمة.

3. الطباعة:

- يتم طباعة القاموس d بعد إنشائه.

هذا الكود يوضح كيفية استخدام تعبير التكرار ودالة zip() لإنشاء قاموس بشكل سريع وفعال باستخدام قوائم متناسقة.

B-

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

```python
def factorial(n):
    res = 1
    for i in range(1, n + 1):
        res *= i
    return res

numb = int(input("Enter a number : "))
if numb < 0:
    print("Factorial is not defined.")
elif numb == 0:
    print("The factorial of 0 is 1")
else:
    res = factorial(numb)
    print(f"The factorial of {numb} is {res}")
```

Example:

```
Enter a number : 20
The factorial of 20 is 2432902008176640000
```

1. الدالة `factorial(n)`:

- تقوم بحساب عاملي العدد `n` باستخدام حلقة `for` تبدأ من 1 وتتواصل حتى `n` مع تحديث القيمة في كل دورة.
- تقوم بإعادة القيمة النهائية لعاملي العدد.

2. الجزء الرئيسي:

- يُطلب من المستخدم إدخال عدد صحيح باستخدام دالة `int(input())`.
- يتم التحقق مما إذا كان العدد أقل من صفر، حيث يتم طباعة رسالة تفيد بأن عاملي العدد غير معرف.
- إذا كان العدد يساوي صفر، يتم طباعة رسالة تفيد بأن عاملي العدد للرقم صفر هو واحد.
- إذا كان العدد أكبر من صفر، يتم استدعاء الدالة `factorial(numb)` لحساب عاملي العدد وثم يتم طباعة النتيجة

باختصار، هذا الكود يسمح للمستخدم بإدخال عدد صحيح ويقوم بحساب عاملي هذا العدد إذا كان العدد غير سالب، وإذا كان العدد سالبًا يتم طباعة رسالة تفيد بأن عاملي العدد غير معرف.

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.

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

```
Bio
```

1. القائمة `L`:

- تحتوي على عدد من العناصر مثل 'Network', 'Bio', 'Programming', 'Physics', 'Music'.

2. الحلقة `for`:

- يتم استخدام حلقة `for` للتكرار عبر كل عنصر في القائمة `L`.
- في كل تكرار، يتم التحقق مما إذا كان العنصر يبدأ بحرف "B" كبير أو صغير.

3. الشرط `if`:

- يتم استخدام الشرط `if` لفحص إذا كان العنصر يبدأ بحرف "B" كبير أو صغير.
- إذا كان العنصر يبدأ بهذه الحروف، يتم طباعته.

4. الإخراج:

- يتم عرض العناصر التي تبدأ بحرف "B" كبير أو صغير من القائمة `L`.

باختصار، هذا الكود يقوم بفحص كل عنصر في القائمة `L` ويقوم بطباعة العناصر التي تبدأ بحرف "B" كبير أو صغير. سيتم طباعة العناصر 'Bio' و 'Programming' لأنهم يبدأون بحرف "B" أو "b".

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}

```
d = {k: v for k, v in zip(range(11), range(1, 12))}
print(d)
```

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

1. `range(11)`:

- تنشئ تسلسل من الأرقام من 0 إلى 10، أي بمجموع 11 عنصر.

2. `range(1, 12)`:

- تنشئ تسلسل آخر من الأرقام من 1 إلى 11، أي بمجموع 11 عنصر.

3. `zip(range(11), range(1, 12))`:

- يقوم بدمج التسلسلين السابقين معًا لإنشاء أزواج من القيم، حيث يتم تشكيل الأزواج كالتالي: (0, 1), (1, 2), ..., (10, 11).

4. التعبير `{k: v for k, v in zip(range(11), range(1, 12))}`:

- يستخدم تعبير التكرار (dictionary comprehension) لإنشاء قاموس.

- في كل تكرار، يتم استخدام القيمة من التسلسل الأول كمفتاح والقيمة من التسلسل الثاني كقيمة في القاموس.

5. `print(d)`:

- يتم طباعة القاموس `d` الذي تم إنشاؤه.

هذا الكود يوضح كيفية استخدام دالة zip() وتعبير التكرار لإنشاء قاموس بسرعة باستخدام تسلسلات من الأرقام.

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.

```python
def binary_to_decimal(binary):
    try:
        d = int(binary, 2)
        return d
    except ValueError:
        return None
while True:
    binary_number = input("Enter a binary number: ")

    if all(char in '01' for char in binary_number):
        break
    else:
        print("Invalid input.")

d_result = binary_to_decimal(binary_number)

if d_result is not None:
    print(f"The decimal equivalent of {binary_number} is: {d_result}")
else:
    print("Invalid binary number.")
```

```
Enter a binary number: 39
Invalid input.
Enter a binary number: 00110011
The decimal equivalent of 00110011 is: 51
```

1. الدالة `binary_to_decimal(binary)`:

- تقوم بتحويل العدد الثنائي الذي يُمر إليها كمعامل إلى عدد عشري.

- تستخدم دالة `int()` لتحويل العدد الثنائي إلى عدد عشري باستخدام القاعدة 2.

- تقوم بإرجاع القيمة العشرية المحسوبة، وإذا حدث خطأ (مثل إدخال عدد غير صالح) تقوم بإرجاع `None`.

2. الحلقة `while True`:

- تطلب من المستخدم إدخال عدد ثنائي.

- تتحقق مما إذا كانت جميع الأحرف في الإدخال تتكون من '0' أو '1'، وإذا كان الشرط صحيحًا فإنها تنهي الحلقة.

3. تحويل العدد الثنائي إلى عدد عشري:

- يتم استدعاء الدالة `binary_to_decimal(binary_number)` لتحويل العدد الثنائي إلى عدد عشري.

- إذا تم التحويل بنجاح، يتم طباعة القيمة العشرية المحسوبة.

- إذا كان هناك خطأ في التحويل (مثل إدخال غير صالح)، يتم طباعة رسالة تفيد بأن العدد الثنائي غير صالح.

باختصار، هذا الكود يسمح للمستخدم بإدخال عدد ثنائي صالح، ثم يقوم بتحويله إلى عدد عشري ويعرض القيمة العشرية المحسوبة. في حالة إدخال عدد ثنائي غير صالح، سيتم عرض رسالة بأن العدد غير صالح.

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
import json
questions = { }
scores = 0
number=1
f = open("questions.txt",'r')
questions = json.load(f)
f.close()
print("python quiz programm")
print("Enter t for True or f for False")
name = input("Enter your full name: ")
for ques in questions.keys():
    print("Question",number,": ", ques)
    ans = input("The answer is ")
    if ans.upper() == questions[ques].upper():
        scores = scores + 1
        print("Correct ")
    else:
        print ("Wrong")
    number = number + 1
result={name:scores}
m = open("score.txt",'w')
result = json.dump(result,m)
m.close()
```

questions - Notepad

File Edit Format View Help

```
{
"130.130.130.130 is a class C address. ":"f",
"ARP refers to Address Resolution Protocol. ":"t",
"TCP is a network layer protocol. ":"f",
"10.0.0.5 is a private ip address. ":"t",
"IPv6 is a 128-bit address. ":"t",
"SDN refers to Software Defined Network. ":"t",
"UDP is a Transport Layer protocol. ":"t",
"224.0.0.9 is a multicast address. ":"t",
"IPv4 is a 128-bit address. ":"f",
"Python is a programming language. ": "t",
"IPv4 is a 32-bit address. ":"t",
"MAC is address is 6 byte address. ":"t",
"153.16.2.8 is a private ip address. ":"f",
"IP is a network Layer protocol. ":"t",
"OSPF is a Routing Protocol. ":"t",
"ICMP refers to Internet Control Message Protocol. ":"t",
"192.168.1.1 is a class A address. ":"f",
"bridge is a layer 3 device. ":"f"}
```

score - Notepad

File Edit Format View Help

```
{"Essa M1hem": 19}
```

1. استيراد المكتبة وقراءة الأسئلة:

- يتم استيراد مكتبة json.

- يتم فتح ملف "questions.txt" لقراءة الأسئلة الموجودة فيه كملف JSON وتخزينها في متغير questions.

- يتم إغلاق الملف بعد القراءة.

2. طباعة رسالة الترحيب وطلب اسم المستخدم:

- يتم طباعة رسالة ترحيبية للمستخدم.

- يتم طلب اسم المستخدم وتخزينه في المتغير name.

3. عرض الأسئلة وجمع النقاط:

- يتم عرض كل سؤال من الأسئلة مع رقم السؤال.

- يتم طلب الإجابة عن كل سؤال وتخزينها في المتغير ans.

- إذا كانت الإجابة صحيحة (بالنظر للقيمة في المتغير questions)، يتم زيادة عدد النقاط (scores) بمقدار واحد وطباعة "Correct".

- إذا كانت الإجابة غير صحيحة، يتم طباعة "Wrong".

4. حساب النتيجة وحفظها:

- يتم إنشاء قاموس يحتوي على اسم المستخدم وعدد النقاط (scores).

- يتم فتح ملف "score.txt" للكتابة وحفظ النتيجة فيه كملف JSON.

- يتم إغلاق الملف بعد الكتابة.

باختصار، هذا البرنامج يسمح للمستخدم بإجابة على مجموعة من الأسئلة حول Python ويقوم بتقييم الإجابات وحساب النتيجة النهائية، ثم يقوم بحفظ نتيجة المستخدم في ملف "score.txt" كملف JSON.

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, 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 amount <= self.balance:
            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, balance=0.0, interest_rate=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
        self.balance += interest_amount
        return interest_amount

    def print_details(self):
        print(f"Current balance: {self.balance}, Interest rate: {self.interest_rate}")

bank_account = BankAccount("998877", "Basel")
print("Initial Balance:", bank_account.get_balance())
bank_account.deposit(1000)
print("Balance after deposit: $", bank_account.get_balance())
bank_account.withdraw(500)
print("Balance after withdrawal: $", bank_account.get_balance())

savings_account= SavingsAccount("23232", "Samer", 5000, 0.05)
print("\nInitial Savings Account:")
savings_account.print_details()

savings_account.apply_interest()
print("\nBalance after applying interest:")
savings_account.print_details()
```

```
Initial Balance: 0.0
Balance after deposit: $ 1000.0
Balance after withdrawal: $ 500.0

Initial Savings Account:
Current balance: 5000, Interest rate: 0.05

Balance after applying interest:
Current balance: 5250.0, Interest rate: 0.05
```

1. **صندوق البنك (Bank Account)**

- يتم تعريف الصندوق البنكي كفئة تسمى `BankAccount`.

- يتم تعريف المتغيرات ``account_number`، `account_holder``، و `balance` في الدالة `__init__` كمتغيرات خاصة للصندوق.

- تحتوي الصندوق على دوال لإيداع الأموال (`deposit`)، سحب الأموال (`withdraw`)، والحصول على الرصيد الحالي (`get_balance`).

2. **حساب التوفير (Savings Account)**

- يتم تعريف حساب التوفير كفئة تستمد من فئة `BankAccount` باستخدام ` class `SavingsAccount(BankAccount).

- يتم تعريف المتغير الإضافي `interest_rate` في الدالة `__init__` لحساب التوفير.

- تحتوي حساب التوفير على دالة لحساب الفائدة (`apply_interest`) وطباعة تفاصيل الحساب (`print_details`) بالإضافة إلى الدوال الموروثة من الصندوق البنكي.

3. **استخدام الكود**

- يتم إنشاء صندوق بنكي جديد وعرض الرصيد الأولي والرصيد بعد الإيداع والسحب.

- يتم إنشاء حساب توفير جديد وعرض التفاصيل الأولية للحساب.

- يتم تطبيق الفائدة على حساب التوفير وعرض التفاصيل بعد تطبيق الفائدة.

باختصار، هذا الكود يوضح كيفية إنشاء صندوق بنكي وحساب توفير باستخدام التوجيه في Python، ويوفر وظائف لإيداع الأموال، سحب الأموال، حساب الفائدة، وعرض تفاصيل الحساب.