

Question 1 A:

اولا نقوم بتعريف متغيرين من النوع LIST ثم نقوم بتعريف dictionary فارغة وبعدها باستخدام الحلقة for والتابع zip يتم إضافة عنصر من القائمة الأولى كمفتاح وإضافة عنصر من القائمة الثانية كقيمة للمفتاح وذلك بالترتيب وبعد الانتهاء من الحلقة يتم طباعة القاموس الناتج

```
filebrowser-extension > Questino 1_A.py > ...
1 L1=["HTTP","HTTPS","FTP","DNS"]
2 L2=[80,443,21,53]
3 D={}
4 for i in range(len(L1)):
5     D[L1[i]]=L2[i]
6 print(D)
```

PROBLEMS 100 OUTPUT DEBUG CONSOLE TERMINAL PORTS

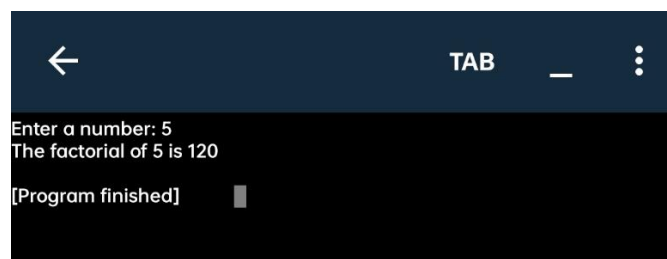
```
PS C:\Users\Windows.10\.jupyter\lab\user-settings\@jupyterlab\filebrowser-extension>
ab/user-settings/@jupyterlab/filebrowser-extension/Questino 1_A.py"
{'HTTP': 80, 'HTTPS': 443, 'FTP': 21, 'DNS': 53}
PS C:\Users\Windows.10\.jupyter\lab\user-settings\@jupyterlab\filebrowser-extension>
```

Question 1 B:

تعريف تابع factorial يأخذ متغير من النوع int مضروب الصفر هو واحد لذلك اذا ادخل المستخدم 0 يقوم التابع بإعادة القيمة 1 اذا ادخل المستخدم رقم موجب يتم الدخول في الحلقة التنازلية والتي تبدأ بالرقم المدخل مطروحا منه واحد وتنتهي بالرقم صفر وتقوم بإعادة قيمة مضروب العدد والا يقوم المستخدم بإدخال عدد سالب فتظهر له رسالة خطأ وانه لا يوجد مضروب للعدد المدخل

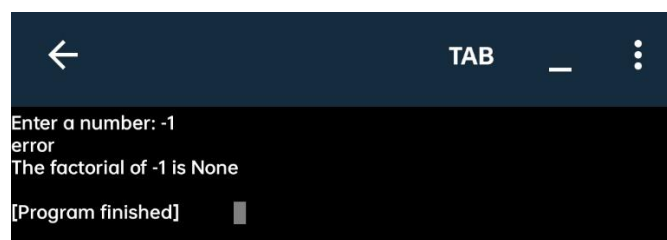
PYTHON

```
1 def factorial(n):
2
3     if n == 0:
4         return 1
5     elif n > 0:
6         for x in range(n-1, 0, -1):
7             n = n*(x)
8         return n
9     else:
10        print('error')
11
12    # Get the number from the user
13    num = int(input("Enter a number: "))
14
15    # Calculate the factorial
16    result = factorial(num)
17
18    # Print the result
19    print('The factorial of', num, 'is',
20          result)
```



A screenshot of a terminal window with a dark background. The top bar is dark blue with a back arrow, the text 'TAB', and a menu icon. The terminal text shows the program's execution for input 5: 'Enter a number: 5', 'The factorial of 5 is 120', and '[Program finished]'.

```
← TAB — ⋮
Enter a number: 5
The factorial of 5 is 120
[Program finished]
```



A screenshot of a terminal window with a dark background. The top bar is dark blue with a back arrow, the text 'TAB', and a menu icon. The terminal text shows the program's execution for input -1: 'Enter a number: -1', 'error', 'The factorial of -1 is None', and '[Program finished]'.

```
← TAB — ⋮
Enter a number: -1
error
The factorial of -1 is None
[Program finished]
```

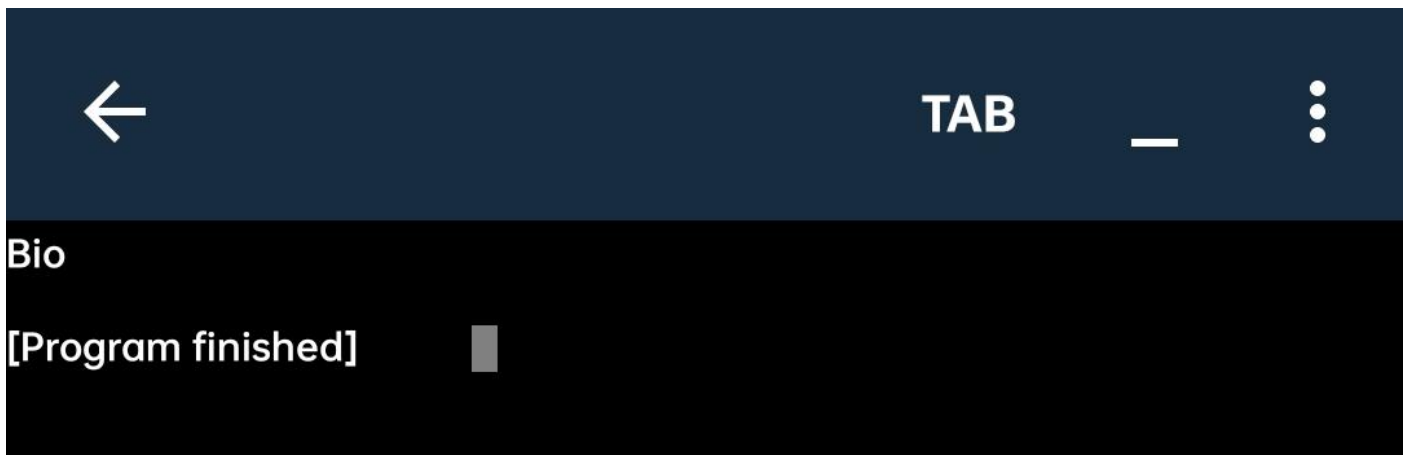
Question 1 C:

تعريف متحول L من النوع List

يتم الدخول في الحلقة ضمن مجال عدد عناصر القائمة

والتحقق من كل عنصر في القائمة وإذا كان العنصر يبدأ بالمحرف B يتم طباعة ذلك العنصر

```
PYTHON
1 L=['Network','Bio','Programming',
    , 'Physics','Music']
2 for x in range(len(L)):
3     if L[x].startswith('B'):
4         print(L[x])
5
```



Question 1 D:

تعريف متغير d من النوع dictionary يحوي على حلقة بمجال يبدأ بال ٠ وينتهي بال ١٠ ويتم وضع قيمة المتغير الخاص بالحلقة (x) كمفتاح للقاموس ووضع قيمة المتغير (x) مضاف له ١ كقيمة لذلك المفتاح

وهكذا حتى نصل الى نهاية الحلقة حيث يكون المفتاح ١٠ وقيمه ١١

وبعدها يتم الخروج من الحلقة وطباعة القاموس الناتج

PYTHON

```
1 d={x:x+1 for x in range (11)}  
2 print(d)  
3
```



TAB



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

[Program finished]

Question 2:

الطلب من المستخدم بإدخال رقم ثنائي

يتم تخزين الادخال ضمن متغير y كسلسلة حرفية string

ثم نقوم بتعريف متغير sum ليحفظ قيمة التحويل ونعطيه قيمة بدائية 0

ونقوم بتعريف متغير اخر z والذي سيكون فهرس ل y ونعطيه القيمة 1- وبعدها نقوم بإنقاصها ضمن الحلقة وذلك للتحقق من الرقم المدخل من اليمين الى اليسار

واخيرا نعرف المتغير e من النوع string ونتركه فارغ وذلك لكي لا يقوم الكود بطباعة رسالة العدد العشري اذا تم الخروج من الحلقة ب تعليمة break ويحفظ ضمن الحلقة رسالة خطأ يتم اظهارها للمستخدم اذا ادخل عدد ليس ثنائي

والا تطبع قيمة sum التي تحتوي على العدد العشري الموافق

PYTHON

```
1 y=input('Enter a binary number : ')
2 sum=0
3 z=-1
4 e=' '
5 for x in range(len(y)):
6     if y[z]=='0':
7
8         z=z-1
9         pass
10
11     elif y[z]=='1':
12
13         z=z-1
14         sum=sum+2**x
15
16     else:
17
18         e='error'
19         print(e)
20         break
21 if e!='error':
22     print('The decimal number is
23         : ',sum)
```

← TAB _ ⋮

Enter a binary number : 1110
The decimal number is : 14

[Program finished] █

← TAB _ ⋮

Enter a binary number : 1122
error

[Program finished] █

Question 3:

اولا : نقوم باستيراد المكتبة json لاننا نريد القراءة من ملف json وايضا الكتابة في ملف json اخر

ثانيا : نطلب من المستخدم ادخال اسمه

وبعدها نقوم بقراءة ملف json الذي اعددناه مسبقا ووضعنا فيه قائمة تحتوي عنصرين

العنصر الاول هو قائمة تحتوي على الاسئلة والعنصر الثاني هو قائمة ايضا تحتوي على الاجوبة

ثم نقوم بتعريف متغير sum ونسند له القيمة 0 والذي سنقوم بزيادته بمقدار 1 عند كل جواب صحيح يدخله المستخدم

واخيرا نعرف المتغير z الذي سيتيح لنا الوصول للقائمة الثانية وبعدها كل دخول في الحلقة نزيده بمقدار 1 وذلك لمقارنة كل جواب يدخله المستخدم بالجواب الموجود في القائمة الثانية بالترتيب على كل سؤال مقابل من القائمة الاولى

واخيرا يتم طباعة اسم المستخدم والنتيجة التي حصل عليها وبعدها يتم كتابتهم في ملف json اخر ك dictionary ويكون المفتاح هو اسم المستخدم وقيمه هي العلامة التي حصل عليها

C: > Users > Windows.10 > Desktop > yazan1 > Q3.py > ...

```
1 import json
2 name=input('Enter your Name : ')
3 f=open('C:\\Q3.json ', 'r')
4 l=json.load(f)
5 f.close()
6 sum=0
7 z=0
8 for Question in l[0]:
9     print(Question)
10    Answr=input('ANSWR : ')
11    if Answr==l[1][z]:
12        sum=sum+1
13    z=z+1
14 print(name,':',sum,'/21')
15 L=dict()
16 L[name]=sum
17 f=open('C:\\Q3Result ', 'w')
18 json.dump(L,f)
19 f.close()
20
21
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
ANSWR : 19
What is the result of 5+15 ?
ANSWR : 20
What is the result of 5+16 ?
ANSWR : 21
What is the result of 5+17 ?
ANSWR : 0
What is the result of 5+18 ?
ANSWR : 0
yazan : 19 /21
PS C:\Users\Windows.10\jupyter\lab\user-settings\@jupyterlab\filebrowser-extension>
```

Question 4:

C: > Users > Windows.10 > Desktop > yazan1 > Q4.py > BankAccount

```
1 class BankAccount:
2     def __init__(self, account_number, account_holder, balance=0.0):
3         self.account_number = account_number
4         self.account_holder = account_holder
5         self.balance=balance
6
7     def deposit(self, amount):
8         self.balance += amount
9         print('deposited $',amount,'into account',self.account_number,'and balance now is:$',self.balance)
10
11    def withdraw(self, amount):
12        if amount <= self.balance:
13            self.balance -= amount
14            print('Withdrew $',amount,'from account',self.account_number,'and balance now is :$',self.balance)
15        else:
16            print("'sorry' you don't have enough money")
17
18    def get_balance(self):
19        return self.balance
20
21    bank_account = BankAccount('2706', 'Yazan Qerata')
22
23    bank_account.deposit(1000)
24
25    bank_account.withdraw(500)
26
27    print('balance now is :$', bank_account.get_balance())
28
29    class SavingsAccount(BankAccount):
30        def __init__(self,account_number,account_holder,balance,interest_rate):
31            super().__init__(account_number, account_holder,balance)
32            self.interest_rate = interest_rate
33
34        def apply_interest(self):
35            interest = self.balance * self.interest_rate
36            self.balance += interest
37            print('Applied',self.interest_rate * 100,'% interest and balance now is :',self.balance)
38
39        def __str__(self):
40            return('Account Number: ' + str(self.account_number) + ' Account Holder: ' + str(self.account_holder)
41                + ' Balance: ' + str(self.balance) + ' Interest Rate: ' + str(self.interest_rate * 100) + '%')
42
43    savings_account = SavingsAccount('2706','Yazan Qerata',500,0.05)
44    savings_account.apply_interest()
45    print(savings_account)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python + - [] [] ... ^ X

```
PS C:\Users\Windows.10\.jupyter\lab\user-settings\@jupyterlab\filebrowser-extension> & C:/Python312/python.exe c:/Users/Windows.10/Desktop/yazan1/Q4.py
deposited $ 1000 into account 2706 and balance now is:$ 1000.0
Withdrew $ 500 from account 2706 and balance now is :$ 500.0
balance now is :$ 500.0
Applied 5.0 % interest and balance now is : 525.0
Account Number: 2706 Account Holder: Yazan Qerata Balance: 525.0 Interest Rate: 5.0%
PS C:\Users\Windows.10\.jupyter\lab\user-settings\@jupyterlab\filebrowser-extension>
```

١. تعريف كلاس BankAccount

له ثلاثة عناصر: account_number (رقم الحساب كسلسلة محرفية string) و
account_holder (اسم صاحب الحساب كسلسلة محرفية string) و
balance (الرصيد بقيمة بدائية 0.0).

وله الطرق التالية:

• __init__

وهذه الطريقة هي افتراضية، حيث تأخذ رقم الحساب واسم صاحب الحساب كمدخلات وتقوم بتعيينهما

• deposit(amount)

تقوم بإضافة المبلغ المحدد إلى رصيد الحساب وتطبع رسالة بأن المبلغ تم إيداعه بقيمة الرصيد الآن

• withdraw(amount)

تقوم بخصم المبلغ المحدد من رصيد الحساب إذا كان الرصيد كافٍ، وتطبع رسالة بأن المبلغ تم سحبه
وقيمة الرصيد الآن. وإذا لم يكن الرصيد كافٍ، تطبع رسالة بعدم امتلاك مال كافٍ.

• get_balance()

تقوم بإرجاع الرصيد الحالي للحساب.

٢. وبعدها ننشئ غرض من BankAccount باسم bank_account

مع رقم الحساب ٢٧٠٦ واسم صاحب الحساب Yazan Qerata.

٣. إيداع مبلغ ١٠٠٠ دولار في الحساب وسحب مبلغ ٥٠٠ دولار من الحساب.

٤. تعريف كلاس ابن باسم SavingsAccount يرث من BankAccount ويضيف عنصر إضافي وهو interest_rate (سعر الفائدة).

وله الطرق التالية :

• __init__

تأخذ رقم الحساب واسم صاحب الحساب وسعر الفائدة والرصيد كمدخلات، وتقوم بتعيينها باستخدام
__init__ من الكلاس الاب BankAccount عن طريق تعليمة super .

• apply_interest ()

تقوم بحساب المبلغ المستحق من الفائدة على أساس سعر الفائدة والرصيد الحالي، وتضيف هذا المبلغ إلى الرصيد وتطبع رسالة بأن الفائدة تم تطبيقها وقيمة الرصيد الآن .

• __str__

هذه الطريقة تقوم بإرجاع تمثيل نصي للكلاس SavingsAccount بتضمين رقم الحساب واسم صاحب الحساب والرصيد وسعر الفائدة.

٥. إنشاء غرض باسم savings_account من SavingsAccount مع رقم الحساب ٢٧٠٦ واسم صاحب الحساب Yazan Qerata وسعر فائدة 0.05 (٥%).

وأخيرا تطبيق الفائدة على الحساب باستخدام () apply_interest.

وطباعة معلومات الحساب