

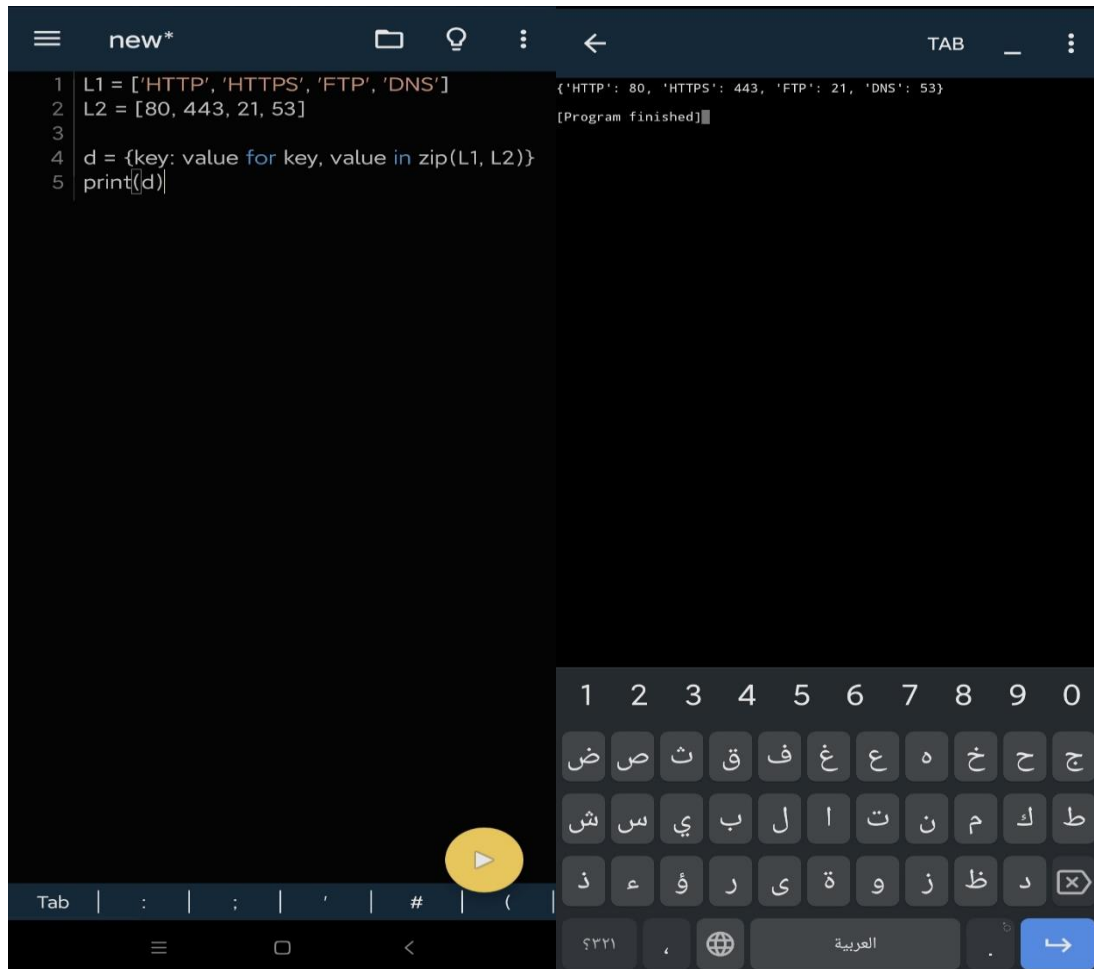
Name: Alaa Basel Ahmad , Number: 2845

### Question 1:

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 }.

#### Code

#### Output



The screenshot shows a code editor with a dark theme. The left pane displays the following Python code:

```
1 L1 = ['HTTP', 'HTTPS', 'FTP', 'DNS']
2 L2 = [80, 443, 21, 53]
3
4 d = {key: value for key, value in zip(L1, L2)}
5 print(d)
```

The right pane shows the output of the code:

```
{'HTTP': 80, 'HTTPS': 443, 'FTP': 21, 'DNS': 53}
[Program finished]
```

At the bottom of the editor, there is a virtual keyboard with Arabic characters and a play button icon.

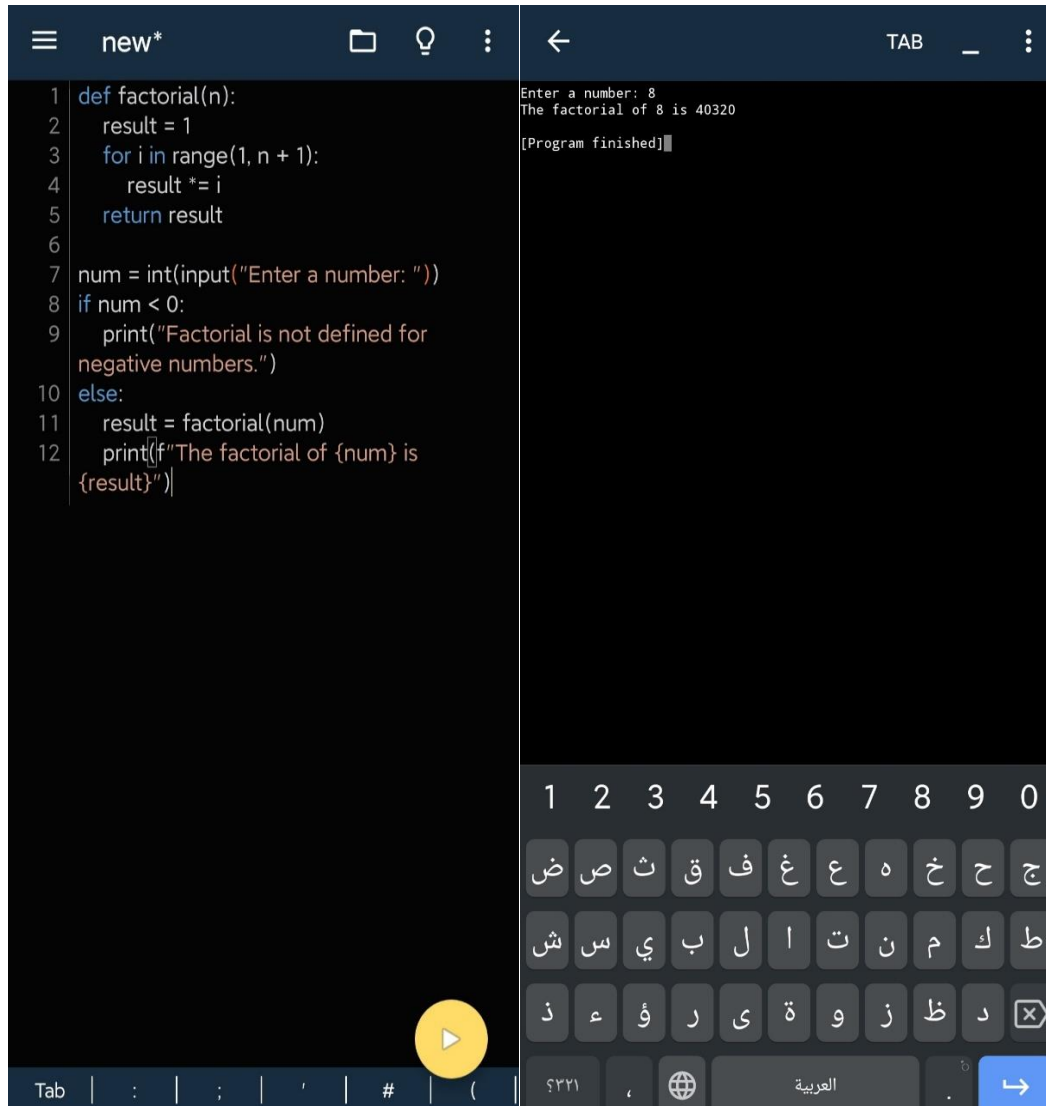
Zip() : تدمج قائمتين معاً

أنشأت حلقة تمر على عناصر القاموس الذي يحتوي القائمتين معاً ثم قمت بطباعته.

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

### Code

### Output



```
1 def factorial(n):
2     result = 1
3     for i in range(1, n + 1):
4         result *= i
5     return result
6
7 num = int(input("Enter a number: "))
8 if num < 0:
9     print("Factorial is not defined for
10    negative numbers.")
11 else:
12     result = factorial(num)
13     print(f"The factorial of {num} is
14     {result}")
```

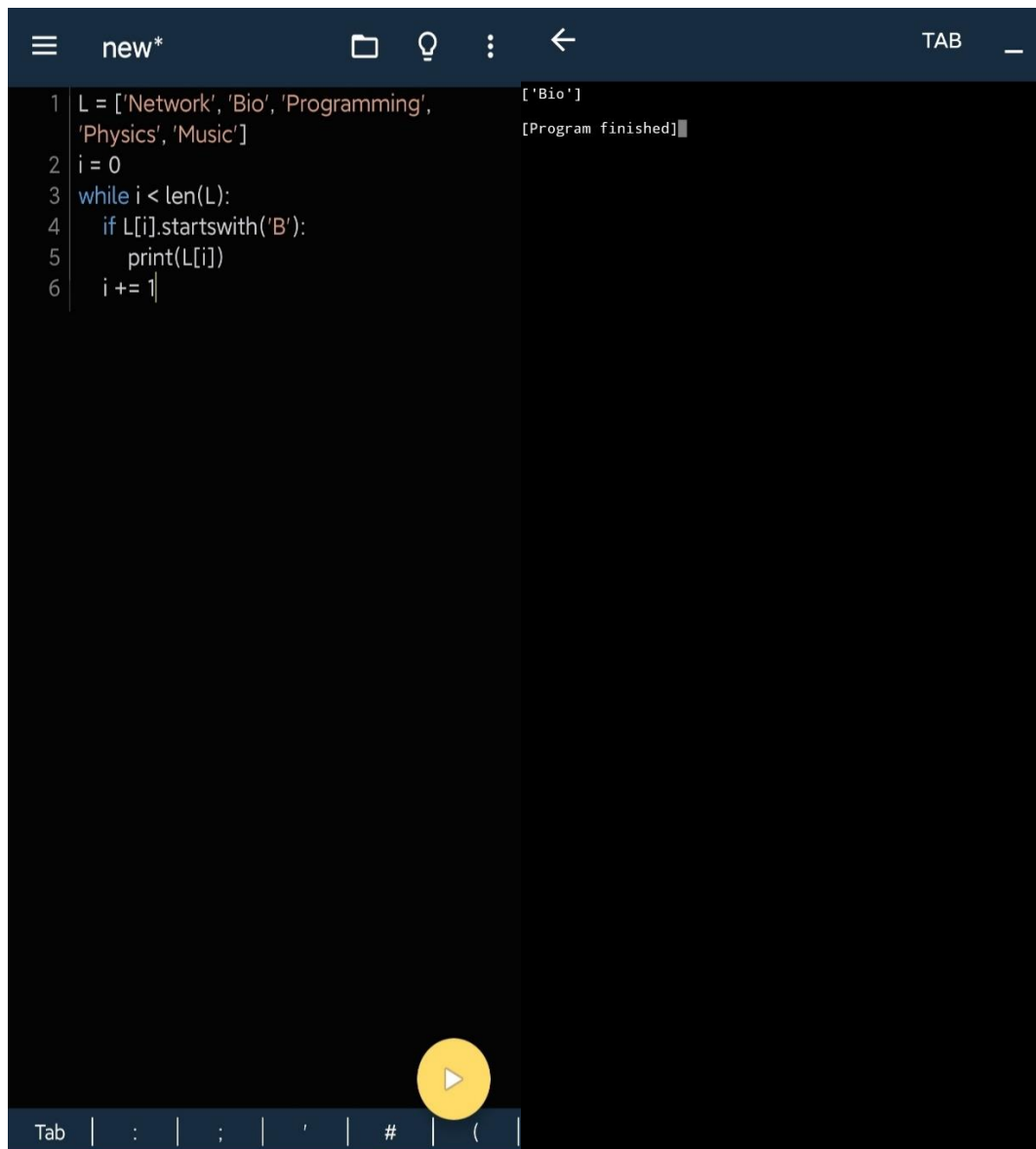
```
Enter a number: 8
The factorial of 8 is 40320
[Program finished]
```

في البداية يتم تعريف التابع `factorial(n)` ذو البارامتر الوحيد `n` وتعريف متغير `result=1` وتعيين القيمة الابتدائية 1 لأن العاملي تبدأ بالضرب بواحد، ثم أنشأت حلقة `for` لتكرار الأعداد من 1 إلى `n` ضمناً، في كل تكرار للحلقة يتم ضرب القيمة الحالية ل `result` في العدد الحالي للتسلسل `i` ، بعد انتهاء الحلقة يقوم التابع بإرجاع قيمة `result` النهائية والتي تمثل قيمة العاملي للعدد المدخل. يطلب من المستخدم إدخال عدد ويقوم بتحويله إلى عدد صحيح باستخدام `int` ، يتحقق الكود من العدد الذي تم إدخاله: إذا كان سالباً يطبع رسالة بأن العاملي غير معرفة للأعداد السالبة وإلا فيتم حساب العاملي للعدد عن طريق التابع `factorial` ويطبع النتيجة.

**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. Tips: using loop, 'len ()' , startswith() methods.

### Code

### Output



The screenshot shows a code editor with a dark theme. The left pane contains the following Python code:

```
1 L = ['Network', 'Bio', 'Programming',  
2   'Physics', 'Music']  
3 i = 0  
4 while i < len(L):  
5     if L[i].startswith('B'):  
6         print(L[i])  
7     i += 1
```

The right pane shows the output of the program:

```
['Bio']  
[Program finished]
```

At the bottom of the editor, there is a yellow play button icon and a status bar with symbols for Tab, colon, semicolon, apostrophe, hash, and parentheses.

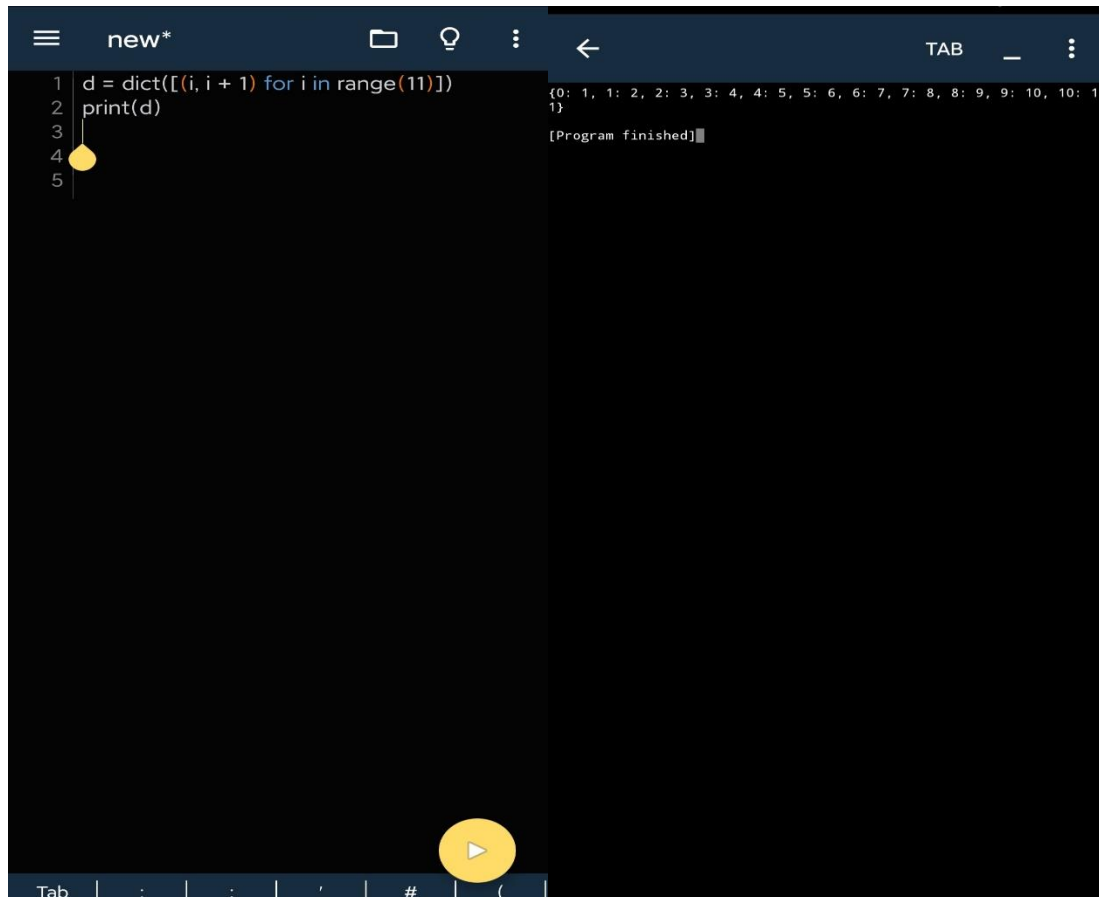
قمت بتعريف قائمة تحتوي على مجموعة من العناصر، ثم عرفت متغير  $i=0$  كقيمة ابتدائية  
لحلقة while التي تختبر قيمة  $i$  بحيث تكون أصغر تماماً من طول القائمة فإذا كان العنصر ذو  
الفهرس  $i$  يبدأ بحرف B فتقوم بطباعته ثم تزيد قيمة  $i$  بمقدار واحد مع كل مرة.

**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}`.

### Code

### Output



```
1 d = dict([(i, i + 1) for i in range(11)])
2 print(d)
3
4
5
```

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

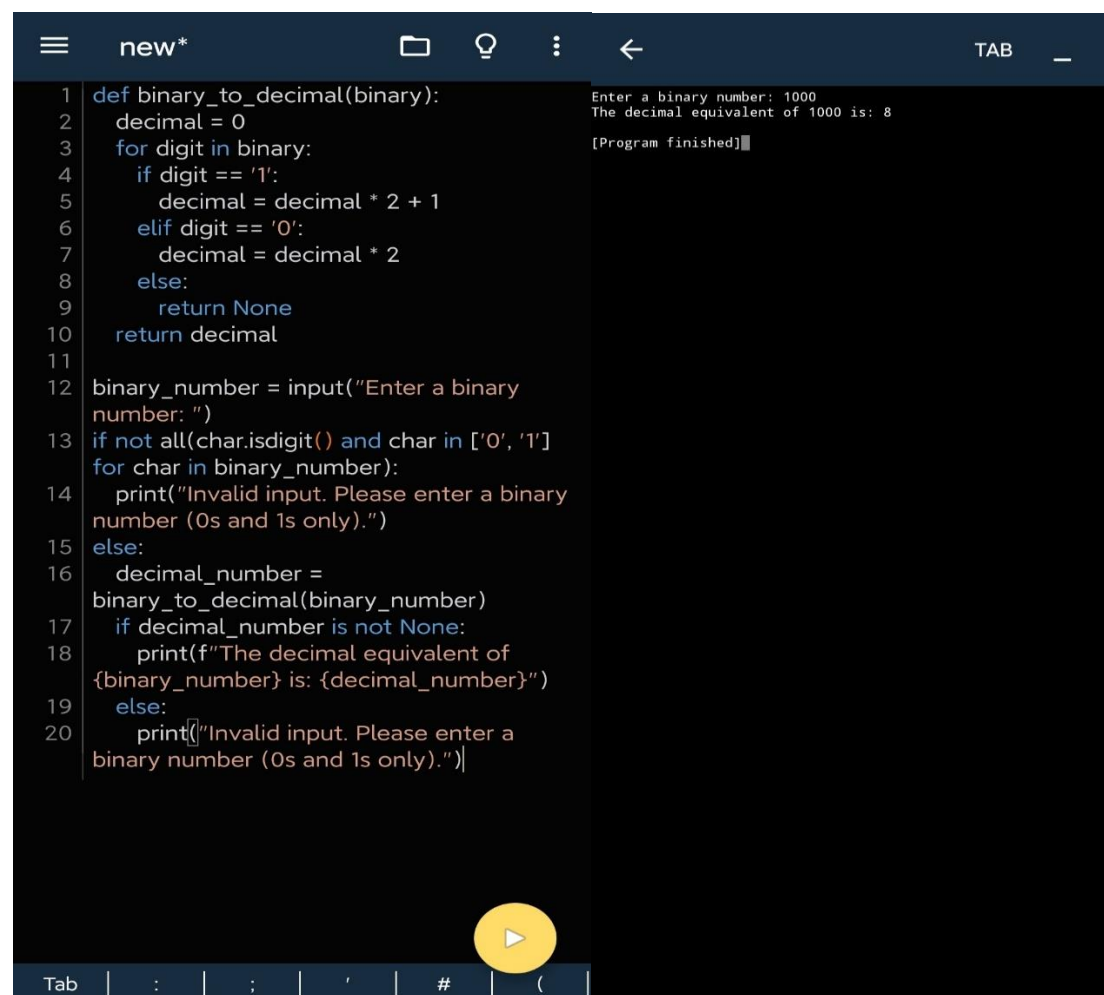
استخدمت comprehension لإنشاء قائمة من الأزواج حيث كل زوج يتكون من عددين: العدد الأول هو المفتاح  $i$  والعدد الثاني هو القيمة  $i+1$  ، الحلقة for المرتبطة بالمفتاح  $i$  تتكرر ضمن المجال من 0 إلى 10 ضمناً حيث أن التعليمة `dict()` تحول القائمة التي تم إنشاؤها إلى قاموس حيث العدد الأول هو المفتاح والثاني هو القيمة، وتقوم بطباعة هذا القاموس.

## Question 2:

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. Tips: solve input errors.

### Code

### Output



```
1 def binary_to_decimal(binary):
2     decimal = 0
3     for digit in binary:
4         if digit == '1':
5             decimal = decimal * 2 + 1
6         elif digit == '0':
7             decimal = decimal * 2
8         else:
9             return None
10    return decimal
11
12    binary_number = input("Enter a binary
13    number: ")
14    if not all(char.isdigit() and char in ['0', '1']
15    for char in binary_number):
16        print("Invalid input. Please enter a binary
17        number (0s and 1s only).")
18    else:
19        decimal_number =
20        binary_to_decimal(binary_number)
21        if decimal_number is not None:
22            print(f"The decimal equivalent of
23            {binary_number} is: {decimal_number}")
24        else:
25            print("Invalid input. Please enter a
26            binary number (0s and 1s only).")
```

Enter a binary number: 1000  
The decimal equivalent of 1000 is: 8  
[Program finished]

تعريف التابع (binary\_to\_decimal (binary) الذي يأخذ سلسلة من الأرقام الثنائية (أصفر و واحدات) ويحولها إلى القيمة العشرية المقابلة لها، داخل هذا التابع يتم تعريف متغير decimal=0

واعطائه قيمة ابتدائية مثلاً 0 والذي سيحتوي على القيمة العشرية النهائية، حيث تتكرر حلقة for عبر كل رقم في السلسلة الثنائية، يتم التحقق من الرقم الثنائي: فإذا كان الرقم هو 1 يتم ضرب القيمة العشرية الحالية في 2 وإضافة 1 ، أما إذا كان الرقم هو 0 يتم فقط ضرب القيمة الحالية في 2 وإذا لم يكن الرقم 0 أو 1 يُرجع التابع none للدلالة على وجود خطأ في الإدخال، بعد انتهاء الحلقة يُعيد التابع القيمة العشرية المحسوبة.

بعد ذلك يُطلب من المستخدم إدخال الرقم الثنائي ويتم التحقق من صحة الإدخال للتأكد من أنه يحتوي على أرقام 0 و 1 فقط.

إذا كان الإدخال يحتوي على أرقام غير 0 أو 1 يتم طباعة رسالة خطأ أما إذا كان الإدخال صحيح فيستدعى التابع لتحويل الرقم الثنائي إلى عشري، إذا كانت القيمة المرجعة ليست none يتم طباعة القيمة العشرية المقابلة للرقم الثنائي، أما إذا كانت القيمة المرجعة هي none أي أن الإدخال غير صحيح ويتم طباعة رسالة خطأ.

### Question 3:

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.

#### Code

#### Output

```
1 import json
2
3 def
load_questions_from_json_file(file_path):
4     with open(file_path, 'r') as file:
5         questions = json.load(file)
6         return questions
7
8 # The administer_quiz function remains the
same as in the text file example
9
10 questions_file = "questions.json"
11 user_name = input("Enter your name: ")
12 quiz_questions =
load_questions_from_json_file(questions_fil
e)
13 user_score =
administer_quiz(quiz_questions)
14
15 print(f"Dear {user_name}, your score is:
{user_score}/20")
16
17 with open("user_results.json", "a") as
result_file:
18     result_file.write(json.dumps({user_name:
user_score}) + "\n")
```

```
Enter your name: Alaa
Traceback (most recent call last):
  File "/data/user/0/ru.iiec.pydroid3/files/accomp_files/iiec_run/i
iec_run.py", line 31, in <module>
    start(fakepyfile,mainpyfile)
  File "/data/user/0/ru.iiec.pydroid3/files/accomp_files/iiec_run/i
iec_run.py", line 30, in start
    exec(open(mainpyfile).read(), __main__.__dict__)
  File "<string>", line 12, in <module>
  File "<string>", line 4, in load_questions_from_json_file
FileNotFoundError: [Errno 2] No such file or directory: 'questions.
json'

[Program finished]
```

هذا الكود يقوم بتحميل أسئلة اختبار من ملف JSON يدير الاختبار ويحفظ نتائج المستخدم في ملف JSON آخر حيث يقوم ب:

(١) استيراد مكتبة JSON :

Import json: هذا السطر يقوم باستيراد مكتبة json والتي تستخدم للتعامل مع بيانات Json في بايثون.

(٢) تعريف التابع:

Load\_questions\_from\_json\_file: هذا التابع يقوم بتحميل الأسئلة من ملف JSON\_file\_path : المسار إلى ملف JSON الذي يحتوي على الأسئلة.  
Withopen(file\_path, "r") as file: تستخدم لفتح الملف للقراءة.  
question=json.Load(file): يقوم بتحميل البيانات من الملف وتحويلها لتتنسيق json إلى قاموس Python .  
return question: يُعيد الأسئلة المحملة.

(٣) التعامل مع ملف الأسئلة:

questions\_file=" questions.json": يُعرف مسار ملف Json الذي يحتوي على الأسئلة.

quiz\_questions= load\_questions\_from\_json\_file(question-file): يتم تحميل الأسئلة من الملف وتخزينها في المتغير quiz\_questions.

(٤) أخذ اسم المستخدم وتشغيل الاختبار:

User\_name= input("Enter Your name: ") : يُطلب من المستخدم إدخال اسمه.  
Administer\_quiz(quiz\_questions): يفترض أن هناك تابع يسمى Administer\_quiz وتقوم بإدارة الاختبار وتُرجع النتيجة.

(٥) طباعة نتيجة المستخدم:

يتم طباعة نتيجة المستخدم مع رسالة تُبين النتيجة التي حصل عليها من أصل 20 .

(٦) حفظ نتائج المستخدم في ملف JSON :

With open("user\_result.json", "a") as result\_file: يتم فتح ملف يُسمى user\_result.json لتسجيل نتائج الاختبار.

Result\_file.write(json.dumps({user\_name: user\_score}) + "n"): يتم كتابة نتيجة المستخدم في الملف بتنسيق json حيث يتم تحويل القاموس الذي يحتوي



على اسم المستخدم والنتيجة إلى سلسلة باستخدام json.dumps ويضاف إلى نهاية السطر حرف الانتقال ("n") لفصل النتائج عن بعضها بسطر جديد.

#### Question 4:

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.

```
new* TAB _ :
1 class BankAccount:
2     def __init__(self, account_number,
account_holder):
3         self.account_number =
account_number
4         self.account_holder = account_holder
5         self.balance = 0.0
6
7     def deposit(self, amount):
8         self.balance += amount
9
10    def withdraw(self, amount):
11        if self.balance >= amount:
12            self.balance -= amount
13
14    def get_balance(self):
15        return self.balance
16
17    # إنشاء مثيل من BankAccount
18    account1 = BankAccount("123456", "Alice")
19    account1.deposit(1000)
20    print(f"الرصيد الحالي بعد الإيداع:
${account1.get_balance()}")
21    account1.withdraw(500)
22    print(f"الرصيد الحالي بعد السحب:
${account1.get_balance()}")
23
24    class SavingsAccount(BankAccount):
25        def __init__(self, account_number,
account_holder, interest_rate):
```

الرصيد الحالي بعد الإيداع: \$1000.00  
الرصيد الحالي بعد السحب: \$500.00  
الرصيد الحالي: \$2100.00، سعر الفائدة: 0.05

[Program finished]

```
26        super().__init__(account_number,
account_holder)
27        self.interest_rate = interest_rate
28
29        def apply_interest(self):
30            self.balance += self.balance * self.
interest_rate
31
32        def __str__(self):
33            return f"الرصيد الحالي:
${self.balance}، سعر الفائدة: {self.interest_rate}"
34
35        # إنشاء مثيل من SavingsAccount
36        savings_account =
SavingsAccount("789012", "Bob", 0.05)
37        savings_account.deposit(2000)
38        savings_account.apply_interest()
39        print(savings_account)
```



Tab | : | : | ' | # | ( |

يتم تعريف class BankAccount والذي يمثل حساب بنكي، يتم في هذا الصنف إنشاء تابع باني يحتوي على رقم الحساب account\_number واسم صاحب الحساب account\_holder ويبدأ الرصيد بالقيمة 0.0 .

التابع desposit يسمح بإيداع مبلغ معين amount في الحساب وذلك بإضافة المبلغ إلى الرصيد الحالي.

التابع withdraw يسمح بسحب مبلغ معين amount من الحساب إذا كان الرصيد كافياً.

التابع get\_balance يُعيد الرصيد الحالي للحساب.

ثم إنشاء ابن من الأب BankAccount يسمى Account1 تتم عليه عمليات الإيداع والسحب وطباعة الرصيد بعد كل عملية.

تم تعريف class SavingsAccount ترث من BankAccount وتمثل حساب توفير، ننشئ تابع باني لهذا الصنف والتي تُستدعى عند إنشاء كائن من SavingsAccount تقوم بتعيين رقم الحساب واسم صاحب الحساب من خلال استدعاء التابع الباني الأصلي super().\_\_init\_\_ بالإضافة إلى تعيين سعر الفائدة interest\_rate .

التابع apply\_interest يضيف الفائدة إلى الرصيد الحالي للحساب وفقاً لسعر الفائدة.

التابع \_\_str\_\_ يحول كائن إلى سلسلة نصية وهنا يُرجع الرصيد الحالي وسعر الفائدة.

بعد ذلك يتم إنشاء كائن savings\_account من SavingsAccount ويتم إيداع مبلغ في الحساب وتطبيق الفائدة، وأخيراً يتم طباعة معلومات الحساب باستخدام التابع str الذي تم تعريفه.

