



الْجُمْهُورِيَّةُ الْعَرَبِيَّةُ السُّورِيَّةُ  
وِزَارَةُ التَّعْلِيمِ الْعَالِي - جَامِعَةُ تَشْرِينَ  
كَلِيَّةُ الْهَنْدَسَةِ الْمِيكَانِيكِيَّةِ وَالْكَهْرَبَائِيَّةِ  
قِسْمُ هَنْدَسَةِ الْاِتِّصَالَاتِ وَالْاِلِكْتْرُونِيَّاتِ  
السَّنَةُ الدِّرَاسِيَّةُ الْخَامِسَةُ 2024-2023

## وِظِيْفَةُ بَرْمِجَةِ الشَّبَكَاتِ

إِعْدَادُ الطَّالِبِ:

خُضْرُ مَنْيَرِ عَوْدِهِ 2510

إِشْرَافُ الدَّكْتُورِ : مَهْنَدُ عِيْسَى

## 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 }

```
code1.py x
C: > Users > LCT > Desktop > kheder > code1.py > L1
1  L1 = ['HTTP', 'HTTPS', 'FTP', 'DNS']
2  L2 = [80, 443, 21, 53]
3  d = dict(zip(L1, L2))
4  print(d) # Output: {'HTTP': 80, 'HTTPS': 443, 'FTP': 21, 'DNS': 53}
5
```

هذا الكود يقوم بإنشاء قاموس في البايثون من خلال دمج قائمتين: واحدة للمفاتيح ('L1') والأخرى للقيم ('L2'). شرح الخطوات:

- 'L1 = ['HTTP', 'HTTPS', 'FTP', 'DNS']': تُعرف هذه القائمة بأنها قائمة من البروتوكولات.

- 'L2 = [80, 443, 21, 53]': تُعرف هذه القائمة بأنها قائمة من الأرقام التي تمثل منافذ TCP القياسية المستخدمة لكل بروتوكول من البروتوكولات في 'L1'.

- 'd = dict(zip(L1, L2))': هنا، يُستخدم الدالة 'zip()' لجمع القائمتين في أزواج كل مفتاح من 'L1' مقترن بقيمته المناظرة من 'L2'. بعد ذلك، يتم تحويل الناتج إلى قاموس باستخدام 'dict()', حيث تصبح العناصر في 'L1' المفاتيح والعناصر في 'L2' القيم المقابلة.

- `print(d)`: يقوم هذا السطر بطباعة القاموس `d` الذي تم إنشاؤه، حيث يتم عرض كل بروتوكول مع منفذه القياسي كزوج مفتاح-قيمة.

النتيجة النهائية التي يتم طباعتها ستكون:

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

## RESULTS

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
PS C:\Users\LCT> & C:/Users/LCT/AppData/Local/Programs/PowerShell/PowerShell.exe -Command { 'HTTP': 80, 'HTTPS': 443, 'FTP': 21, 'DNS': 53 }
PS C:\Users\LCT>

```

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

```
code2.py X
C:\> Users > LCT > Desktop > kheder > code2.py > ...
1 def factorial(n):
2     if n == 0 or n == 1:
3         return 1
4     else:
5         return n * factorial(n - 1)
6
7 number = int(input("Enter a number: "))
8 print(f"The factorial of {number} is {factorial(number)}")
9
```

## الخطوة 1: تعريف دالة العامل (factorial)

- يتم تعريف دالة تسمى factorial التي تأخذ عدد صحيح  $n$  كمُدخل.
- الدالة تحتوي على شرطين:
  - الشرط الأول: إذا كان  $n$  يساوي 0، فإن الدالة ترجع القيمة 1 (عامل 0 يساوي 1).
  - الشرط الثاني: إذا كان  $n$  أكبر من 0، فإن الدالة تقوم بعملية الضرب التالية:
    - $n$  مضروباً بدعوة جديدة للدالة factorial ولكن بقيمة  $n-1$  أي عامل العدد السابق.

## الخطوة 2: أخذ مدخل المستخدم

- يتم استخدام input لطلب من المستخدم إدخال رقم صحيح.
- يتم تحويل قيمة الإدخال إلى عدد صحيح باستخدام int.
- القيمة المدخلة يتم تخزينها في المتغير  $x$ .

## الخطوة 3: حساب عامل الرقم

- يتم استدعاء دالة factorial ويتم تمرير قيمة  $x$  كوسيلة لها.
- دالة factorial تقوم بحساب عامل  $x$  وتعيد النتيجة.

# RESULTS:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\LCT> & C:/Users/LCT/AppData/Local/Programs/Python/Python310/pytho
Enter a number: 7
The factorial of 7 is 5040
PS C:\Users\LCT> 
```

**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.

```
code3.py X
C: > Users > LCT > Desktop > kheder > code3.py > L
1  L = ['Network', 'Bio', 'Programming', 'Physics', 'Music']
2
3  for item in L:
4      if item.startswith('B'):
5          print(item) # Output: Bio
6
7
```

يُستخدم هذا الكود لطباعة الكلمات من قائمة L التي تبدأ بالحرف "B".

#### 1. إنشاء القائمة "list":

- يتم إنشاء قائمة تسمى L تحتوي على العناصر التالية:

['Network', 'Bio', 'Programming', 'Physics', 'Music']

#### 2. حلقة for:

- يتم استخدام حلقة for لتكرار كل عنصر في القائمة L.
- في كل تكرار، يتم تعيين العنصر الحالي للقائمة إلى المتغير item.

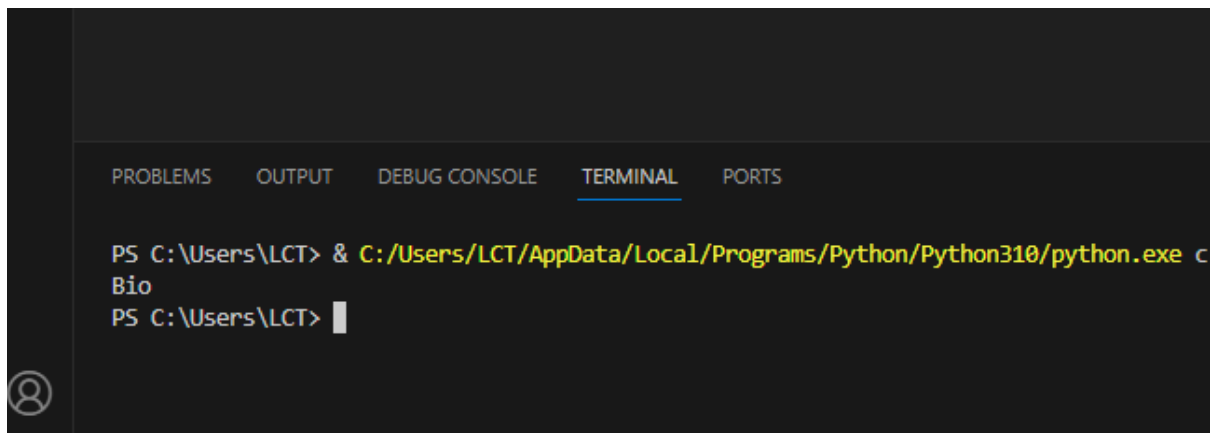
#### 3. شرط if:

- داخل حلقة for، يتم استخدام شرط if للتحقق مما إذا كان العنصر item يبدأ بالحرف "B".

○ يتم استخدام دالة startswith() لتحديد ما إذا كان العنصر يبدأ بحرف معين.

○ إذا كان الشرط صحيحًا (أي أن العنصر يبدأ بـ "B")، يتم طباعة العنصر على الشاشة.

# RESULTS:

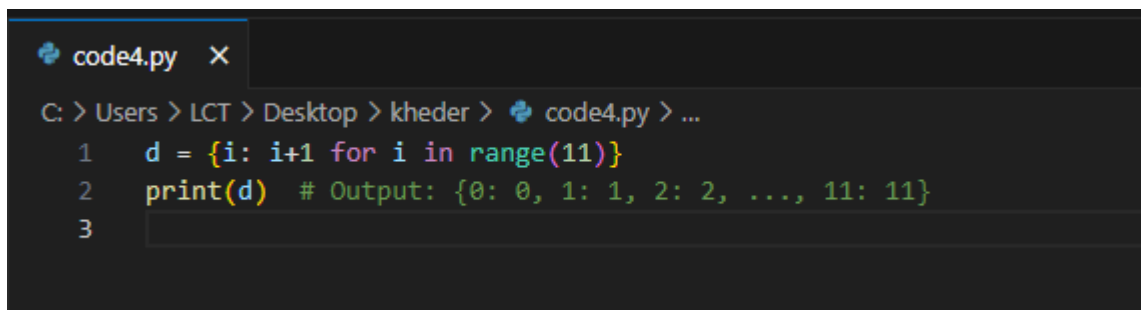


```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\LCT> & C:/Users/LCT/AppData/Local/Programs/Python/Python310/python.exe c
Bio
PS C:\Users\LCT> |
```

**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}`



```
code4.py X
C: > Users > LCT > Desktop > kheder > code4.py > ...
1 d = {i: i+1 for i in range(11)}
2 print(d) # Output: {0: 0, 1: 1, 2: 2, ..., 11: 11}
3
```

يُستخدم هذا الكود لإنشاء قاموس (dictionary) يحتوي على مفاتيح (keys) وقيم (values) محددة.

1. إنشاء القاموس: "dictionary"

- يتم إنشاء متغير يسمى d.
- يتم تعيين قيمة d باستخدام دقة قاموس (dictionary comprehension).
  - دقة القاموس هي طريقة مختصرة لإنشاء قواميس في لغة بايثون.

◦ تتكون دقة القاموس من جزئين:

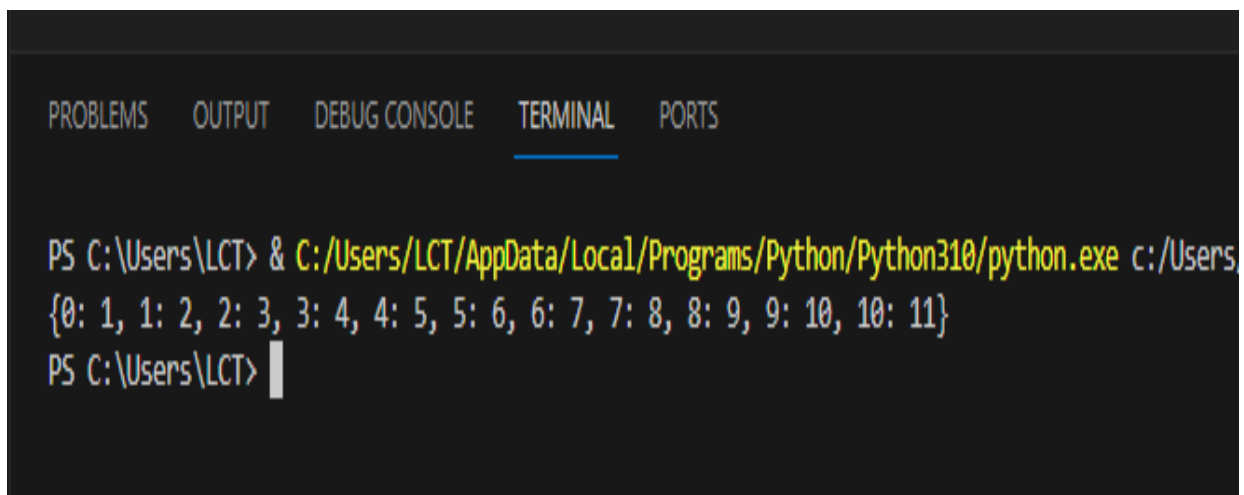
الجزء الأول  $i: i+1$

- هذا الجزء يحدد كيفية إنشاء مفاتيح وقيم القاموس.
- أ هو متغير يتكرر عبر الأرقام من 0 إلى 10 (باستثناء 11).
- $i+1$  هي القيمة التي ستُربط بكل مفتاح  $i$ .

الجزء الثاني (11) `for i in range(11):`

- هذا الجزء يحدد كيفية تكرار المتغير  $i$
- يتم استخدام دالة `range(11)` لجعل  $i$  يتخذ كل قيمة من 0 إلى 10.

## RESULTS:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\LCT> & C:/Users/LCT/AppData/Local/Programs/Python/Python310/python.exe c:/Users
{0: 1, 1: 2, 2: 3, 3: 4, 4: 5, 5: 6, 6: 7, 7: 8, 8: 9, 9: 10, 10: 11}
PS C:\Users\LCT> |
```

## 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.

```
> Users > LCT > Desktop > kheder > Q2.py > binary_to_decimal_alt
1 def binary_to_decimal_alt(binary_str):
2     decimal_value = 0
3     for i, digit in enumerate(reversed(binary_str)):
4         if digit not in '01':
5             raise ValueError("Invalid binary number: Input contains non-binary digits.")
6
7         # Convert binary digit to integer
8         digit_int = int(digit)
9
10        # Calculate decimal contribution using bitwise operations
11        decimal_value += digit_int << i
12
13    return decimal_value
14
15 if __name__ == "__main__":
16     while True:
17         try:
18             binary_str = input("Enter a binary number (or 'kheder' to quit): ")
19
20             if binary_str.lower() == 'q':
21                 break
```

### • دالة binary\_to\_decimal\_alt :

- تأخذ سلسلة ثنائية (binary\_str) كمدخل.
- تُنشئ متغيراً القيمة\_العشرية لتخزين القيمة العشرية المكافئة.
- تستخدم دورة for مع enumerate للتكرار من خلال السلسلة الثنائية بترتيب عكسي (من اليمين إلى اليسار).
- داخل الدورة:



- يتم التحقق من صحة كل رقم للتأكد من أنه إما "0" أو "1".
  - يتم تحويل الرقم الثنائي الحالي إلى عدد صحيح باستخدام `int()`.
  - يتم حساب المساهمة العشرية للرقم الحالي باستخدام الرقم\_كعدد\_صحيح
- << i.

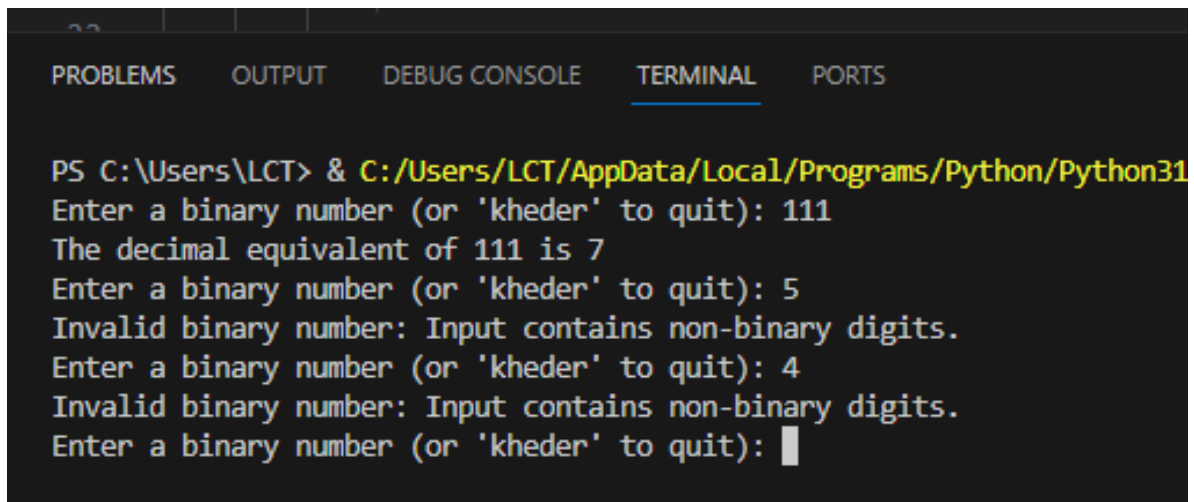
- << يُمثل عملية الإزاحة الثنائية لليسا.
- أن يمثل القوة الحالية لـ 2 (تبدأ من 0 وتزداد مع كل تكرار).
- يتم إضافة المساهمة العشرية إلى القيمة\_العشرية.
- تُعيد الدالة القيمة\_العشرية المحسوبة.

الجزء الرئيسي: (if \_\_name\_\_ == "\_\_main\_\_":)

- يستخدم دورة `while` لتكرار عملية التحويل حتى يدخل المستخدم "q" للخروج.
- داخل الدورة:
- يُطلب من المستخدم إدخال رقم ثنائي (أو "q" للخروج)
- يتم التحقق من صحة إدخال المستخدم للتأكد من أنه سلسلة ثنائية صالحة.
- يتم استدعاء دالة `binary_to_decimal_alt` لتحويل الرقم الثنائي إلى عشري.
- يتم عرض المكافئ العشري للمستخدم.

- يتم التعامل مع أي أخطاء `ValueError` (مثل أرقام غير ثنائية في الإدخال) وعرض رسالة خطأ مناسبة

# RESULTS:




```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\LCT> & C:/Users/LCT/AppData/Local/Programs/Python/Python31
Enter a binary number (or 'kheder' to quit): 111
The decimal equivalent of 111 is 7
Enter a binary number (or 'kheder' to quit): 5
Invalid binary number: Input contains non-binary digits.
Enter a binary number (or 'kheder' to quit): 4
Invalid binary number: Input contains non-binary digits.
Enter a binary number (or 'kheder' to quit): █
```

## 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.



```
Q3.py X
C: > Users > LCT > Desktop > kheder > Q3.py > ...
1 import json
2 f=open('E:\kheder.json','r')
3 file = json.load(f)
4 f.close()
5 name=input("enter name:")
6 number=input("enter number:")
7 result=0
8 for i in file:
9     ans = int (input (i))
10    if ans == file[i]:
11        result = result + 1
12 a=open('E:\ khederaudi.json','w')
13 json.dump ({"name" : name , "number" : number , "result" : result},a)
```

هذا الكود يقوم بفتح ملف JSON، قراءة البيانات من الملف، ثم يطلب من المستخدم إدخال اسم ورقم. بعد ذلك، يقوم بطرح أسئلة للمستخدم استنادًا إلى البيانات الموجودة في الملف JSON ويحسب عدد الإجابات الصحيحة. أخيرًا، يقوم بكتابة اسم المستخدم ورقمه وعدد الإجابات الصحيحة في ملف JSON جديد.

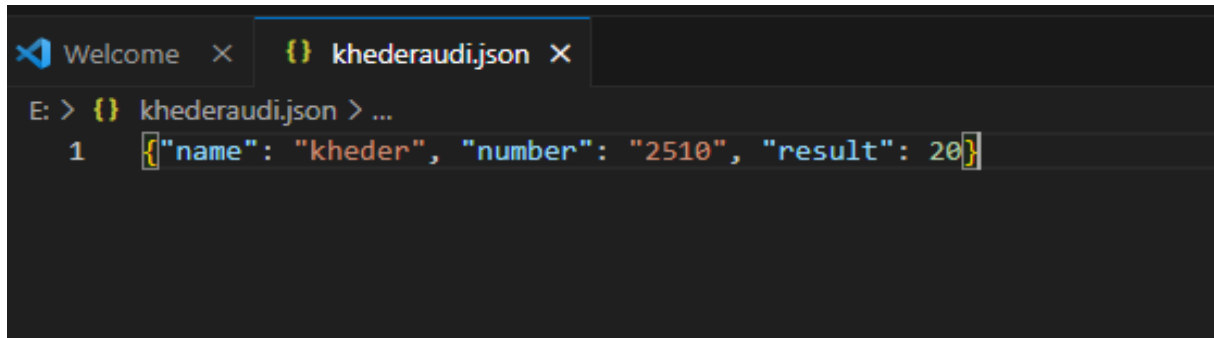
## RESULTS:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\LCT> & C:/Users/LCT/AppData/Local/Programs/Python/Python310/python.exe c:/Users/LCT/Desktop/kheder
enter name:kheder
enter number:2510
1*2=2
2*2=4
3*2=6
4*2=8
5*2=10
6*2=16
7*2=14
8*2=16
9*2=18
10*2=20
1*3=3
2*3=6
3*3=9
4*3=12
5*3=15
6*3=18
7*3=21
8*3=24
9*3=27
10*3=30
```

## Data in JSON:

```
Q3.py kheder.json X
E: > {} kheder.json > ...
1 [{"1*2=": 2, "2*2=": 4, "3*2=": 6, "4*2=": 8, "5*2=":10,"6*2=":12,"7*2=":14,"8*2=":16,"9*2=":18,"10*2=":20,
2 "1*3=": 3, "2*3=": 6, "3*3=": 9, "4*3=":12, "5*3=":15,"6*3=":18,"7*3=":21,"8*3=":24,"9*3=":27,"10*3=":30}]
```

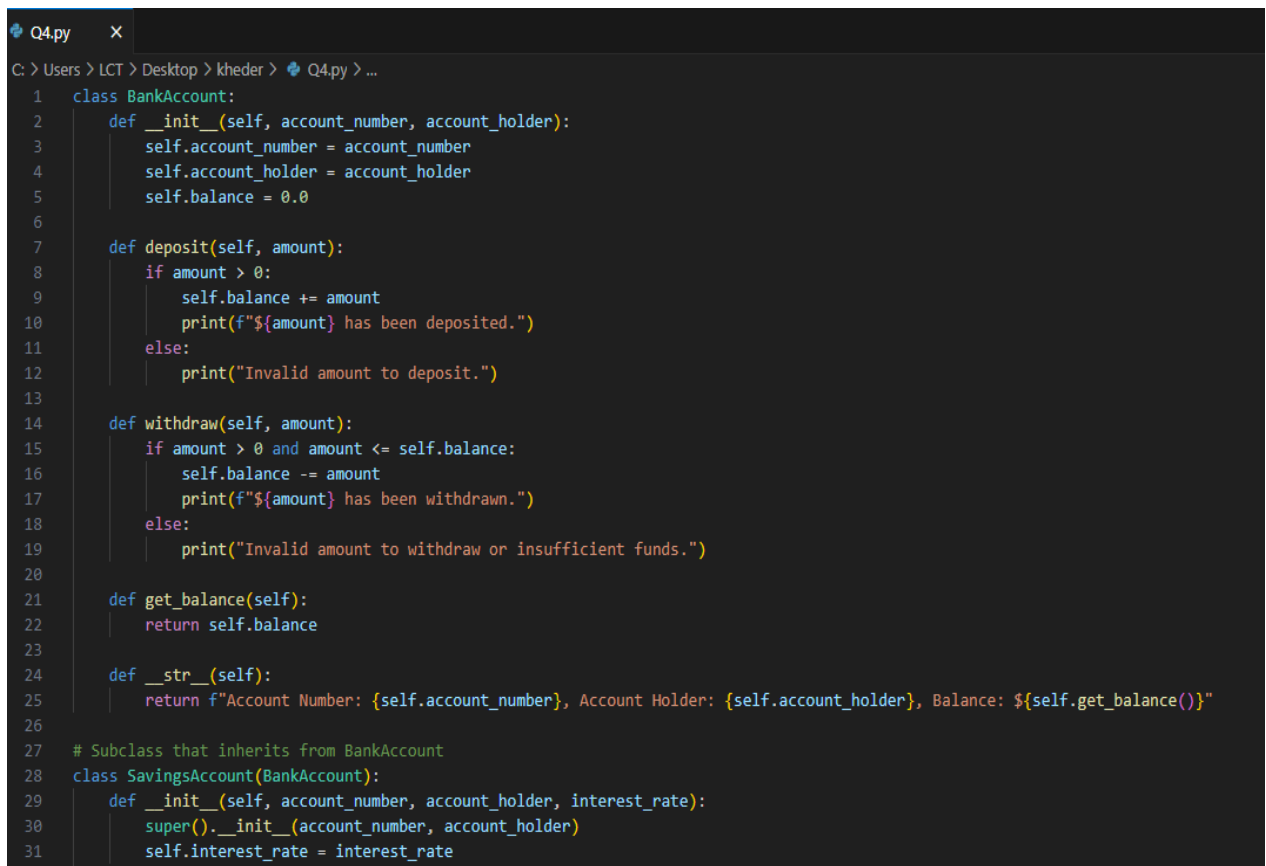
## RESULTS IN JSON:



```
Welcome x khederaudi.json x
E: > {} khederaudi.json > ...
1 [{"name": "kheder", "number": "2510", "result": 20}]
```

## 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.



```
Q4.py x
C: > Users > LCT > Desktop > kheder > Q4.py > ...
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         if amount > 0:
9             self.balance += amount
10            print(f"${amount} has been deposited.")
11        else:
12            print("Invalid amount to deposit.")
13
14    def withdraw(self, amount):
15        if amount > 0 and amount <= self.balance:
16            self.balance -= amount
17            print(f"${amount} has been withdrawn.")
18        else:
19            print("Invalid amount to withdraw or insufficient funds.")
20
21    def get_balance(self):
22        return self.balance
23
24    def __str__(self):
25        return f"Account Number: {self.account_number}, Account Holder: {self.account_holder}, Balance: ${self.get_balance()}"
26
27    # Subclass that inherits from BankAccount
28    class SavingsAccount(BankAccount):
29        def __init__(self, account_number, account_holder, interest_rate):
30            super().__init__(account_number, account_holder)
31            self.interest_rate = interest_rate
```

```

Q4.py X
C: > Users > LCT > Desktop > kheder > Q4.py > ...
27 # Subclass that inherits from BankAccount
28 class SavingsAccount(BankAccount):
29     def __init__(self, account_number, account_holder, interest_rate):
30         super().__init__(account_number, account_holder)
31         self.interest_rate = interest_rate
32
33     def apply_interest(self):
34         if self.interest_rate > 0 and self.balance > 0:
35             interest_amount = self.balance * (self.interest_rate / 100)
36             self.deposit(interest_amount)
37             print(f"Interest applied at {self.interest_rate}%. New balance is ${self.balance}.")
38         else:
39             print("Interest rate must be positive and balance must not be negative.")
40
41     def __str__(self):
42         return super().__str__() + f", Interest Rate: {self.interest_rate}%"
43
44 # Main Program Execution
45
46 # Creating an instance of BankAccount
47 bank_account = BankAccount("1234511", "kheder")
48 bank_account.deposit(1000)
49 bank_account.withdraw(500)
50 print(bank_account) # Using the __str__ method to print current balance
51
52 # Creating an instance of SavingsAccount
53 savings_account = SavingsAccount("6789011", "khedeer audi", interest_rate=5)
54 savings_account.deposit(1000)
55 savings_account.apply_interest()
56 print(savings_account) # Using the __str__ method to print current balance and rate
57

```

هذا الكود يوضح إنشاء فئتين (Classes) في لغة بايثون لتمثيل حسابات بنكية:

## 1- الفئة الأساسية حساب بنكي BankAccount:

- تُستخدم هذه الفئة لإنشاء حساب بنكي عام.
- دالة البناء: `__init__`
  - تأخذ هذه الدالة ثلاثة متحولات كمدخلات:
    - `account_number` رقم الحساب
    - `account_holder` اسم صاحب الحساب
    - `balance` الرصيد - اختياري، الافتراضي
  - تقوم الدالة بتعريف وتعيين قيم المتغيرات التالية داخل الكائن (object):
    - `self.account_number`: يمثل رقم الحساب

▪ self.account\_holder يمثل اسم صاحب الحساب

▪ self.balance يمثل رصيد الحساب

#### • دالة إيداع deposit:

- تأخذ هذه الدالة مبلغ الإيداع amount كمدخل.
- تضيف قيمة amount إلى رصيد الحساب self.balance.
- تطبع رسالة توضيحية على الشاشة تشير بقيمة الإيداع والمبلغ الجديد للرصيد.

#### • دالة سحب withdraw :

- تأخذ هذه الدالة مبلغ السحب amount كمدخل.
- تتحقق من كفاية الرصيد لسحب المبلغ المطلوب.
- إذا كان الرصيد غير كافٍ، تطبع رسالة تفيد بذلك.
- إذا كان الرصيد كافيًا، يتم خصم قيمة السحب amount من الرصيد self.balance.
- تطبع رسالة توضيحية على الشاشة تشير بقيمة السحب والمبلغ الجديد للرصيد.

#### • دالة الحصول على الرصيد get\_balance:

- لا تأخذ هذه الدالة أي قيم كمدخل.
- ترجع قيمة رصيد الحساب المخزن في المتغير self.balance.

### 2- الفئة المشتقة حساب توفير:

• ترث هذه الفئة خصائص الفئة الأساسية BankAccount.

#### • دالة البناء \_\_init\_\_

- تستدعي دالة البناء للصف الأساسي self.\_\_init\_\_ super() لتعيين القيم الأساسية (رقم الحساب، اسم صاحب الحساب، الرصيد).
- بالإضافة إلى ذلك، تعرف المتغير self.interest\_rate لتخزين نسبة الفائدة على الحساب.

### • دالة تطبيق الفائدة `apply_interest` :

- تحسب قيمة الفائدة عن طريق ضرب رصيد الحساب `self.balance` بنسبة الفائدة `self.interest_rate`.
- تستدعي دالة `deposit` لإضافة قيمة الفائدة المحسوبة إلى رصيد الحساب.
- تطبع رسالة توضيحية على الشاشة تشير بنسبة الفائدة المطبقة والمبلغ الجديد للرصيد.

### 3. استخدام الفئات:

يتم إنشاء كائن من الفئة الفرعية `SavingsAccount` وذلك بتعريف متغير `account`.

- يتم تمرير المعلومات اللازمة عند إنشاء الكائن:
  - `'12345'` : رقم الحساب
  - `'kheder'` : اسم صاحب الحساب
  - `0.05 (5%)` : نسبة الفائدة
- يتم استدعاء دالة `deposit` لإيداع مبلغ 1000 في الحساب.
- يتم استدعاء دالة `withdraw` لسحب مبلغ 500 من الحساب.
- يتم استدعاء دالة `apply_interest` لحساب وتطبيق الفائدة على رصيد الحساب.

```
Problems (Ctrl+Shift+M) - total 0 Problems
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\LCT> & C:/Users/LCT/AppData/Local/Programs/Python/Python310/python.exe c:/Users/LCT/Desktop/khede
$1000 has been deposited.
$500 has been withdrawn.
Account Number: 1234511, Account Holder: kheder, Balance: $500.0
$1000 has been deposited.
$50.0 has been deposited.
Interest applied at 5%. New balance is $1050.0.
Account Number: 6789011, Account Holder: khedeer audi, Balance: $1050.0, Interest Rate: 5%
PS C:\Users\LCT> 
```

The end .....