**Question 1:**

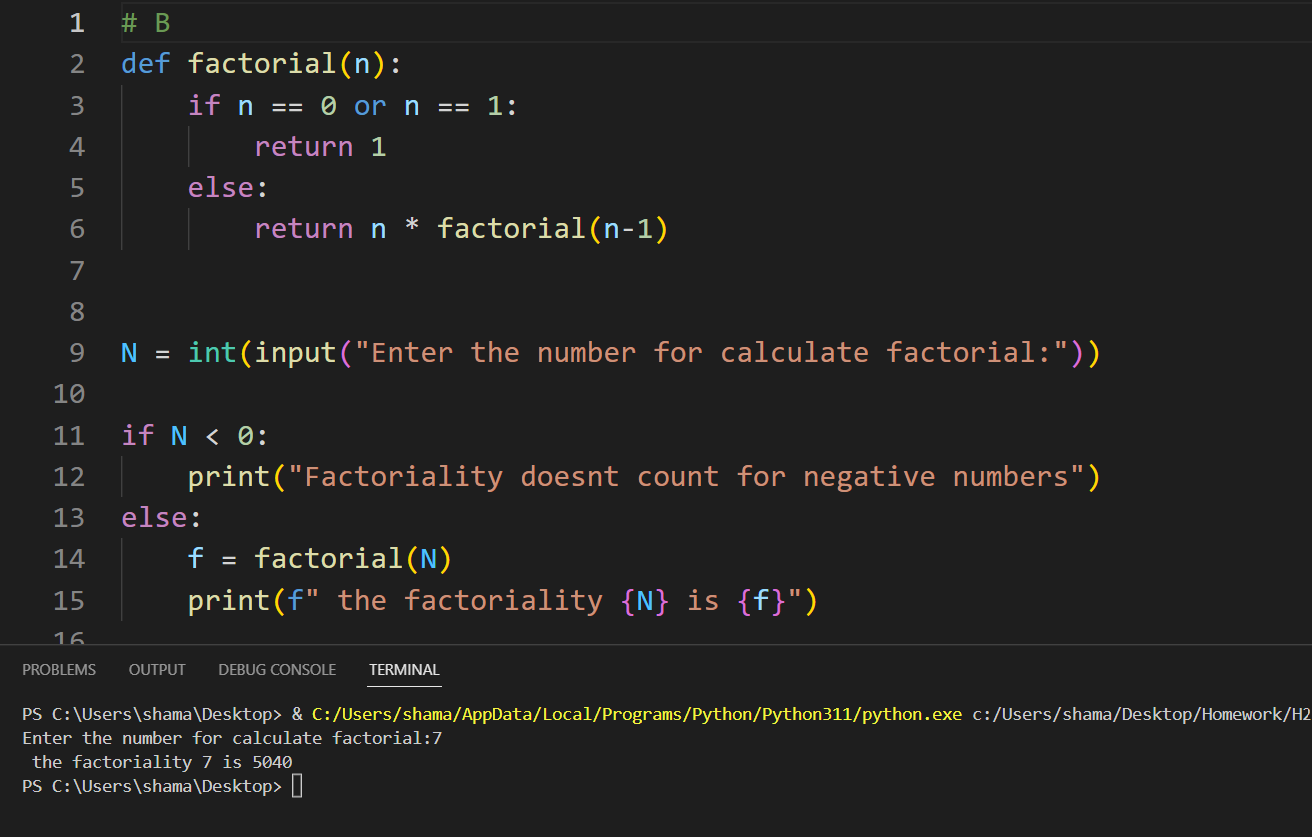
A-If you have two lists,

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 }

****

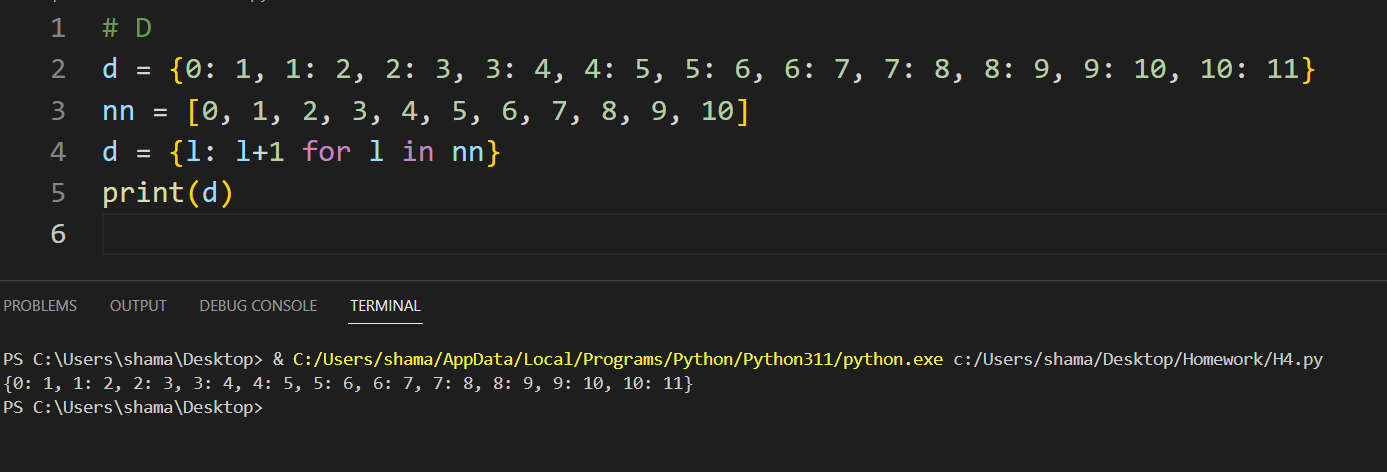
B- Write a Python program that calculates the factorial of a given number entered by user.

****

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.

****

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}

****

**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.

**A screenshot of a computer program

Description automatically generated**

**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

**import** json

**def** load\_quiz(quiz\_file):

**with** open(quiz\_file, 'r', encoding='utf-8') **as** file:

**return** json.load(file)

**def** save\_results(user\_name, score, results\_file):

results = {'name': user\_name, 'score': score}

**with** open(results\_file, 'w') **as** file:

json.dump([results], file) # Save results as a list of dictionaries

**def** run\_quiz(questions):

score = 0

**for** q **in** questions:

**print**(q['question'])

answer = input("Answer: ")

**if** answer.lower() == q['answer'].lower():

score += 1

**print**("Correct!")

**else**:

**print**("Incorrect!")

**print**()

**return** score

**def** main():

quiz\_file = "C:\\Users\\shama\\Desktop\\Homework\\quiz.json"

results\_file = "C:\\Users\\shama\\Desktop\\Homework\\results.json"

user\_name = input("What's your name? ")

question = load\_quiz(quiz\_file)

**if** len(question) != 20:

**print**("Error: The quiz must contain 20 questions.")

**return**

score = run\_quiz(question)

save\_results(user\_name, score, results\_file)

**print**(f"{user\_name}, your final score is {score} out of {len(question)}.")

**if** \_\_name\_\_ == "\_\_main\_\_":

main()

**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

**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):

**if** amount > 0:

self.balance += amount

**print**(f"Deposited: ${amount}")

self.get\_balance()

**def** withdraw(self, amount):

**if** amount > 0 **and** amount <= self.balance:

self.balance -= amount

**print**(f" Withdrawal done : ${amount}")

self.get\_balance()

**else**:

**print**("The amount required for withdrawal exceeds the current balance")

**def** get\_balance(self):

**print**(f"The amount: ${self.balance}")

b\_a = BankAccount("58225", "Rayyan")

b\_a.deposit(7000)

b\_a.withdraw(1500)

**class** SavingsAccount(BankAccount):

**def** \_\_init\_\_(self, account\_number, account\_holder, interest\_rate, balance=0.0):

super().\_\_init\_\_(account\_number, account\_holder, balance)

self.interest\_rate = interest\_rate

**def** apply\_interest(self):

self.balance += self.balance \* self.interest\_rate

**print**(f" Interest is applied at a rate: {self.interest\_rate \* 100}%")

self.get\_balance()

**def** get\_balance(self):

super().get\_balance()

**print**(f"interest rate : {self.interest\_rate \* 100}%")

savings\_account = SavingsAccount("952287", "Sham Alkadour", 0.05)

savings\_account.deposit(8000)

savings\_account.apply\_interest()

savings\_account.get\_balance()

**A screenshot of a computer

Description automatically generated**