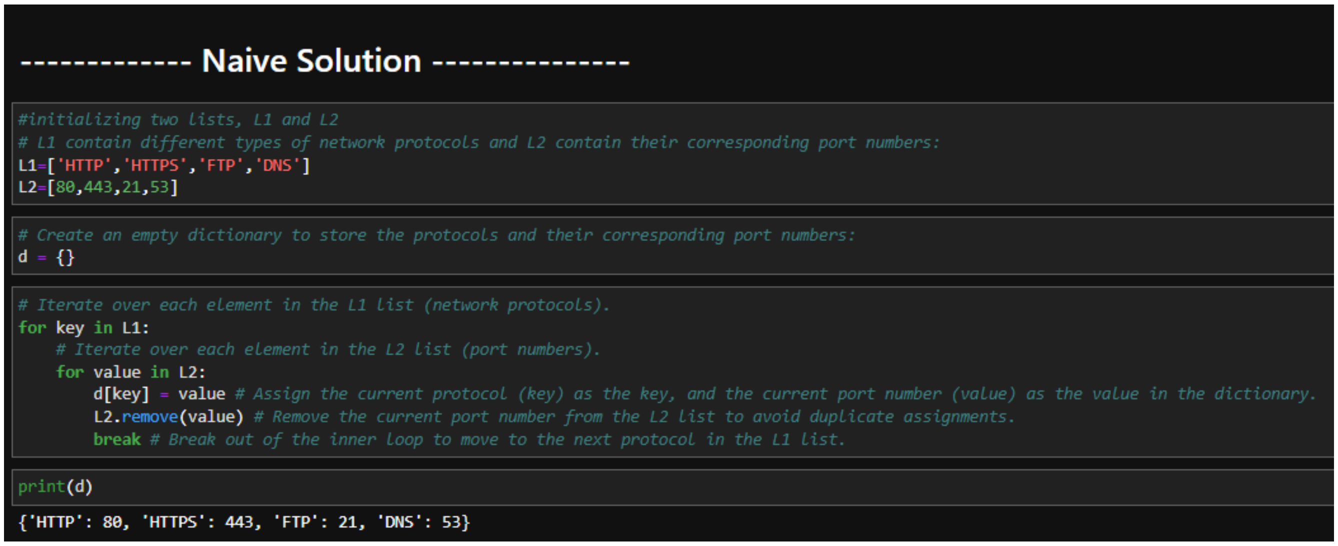
Problem 1: Python Basics?

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}**

Naive_Solution:



------------- Naive Solution ---------------

```python
#initializing two lists, L1 and L2
# L1 contain different types of network protocols and L2 contain their corresponding port numbers:
L1=['HTTP','HTTPS','FTP','DNS']
L2=[80,443,21,53]
```

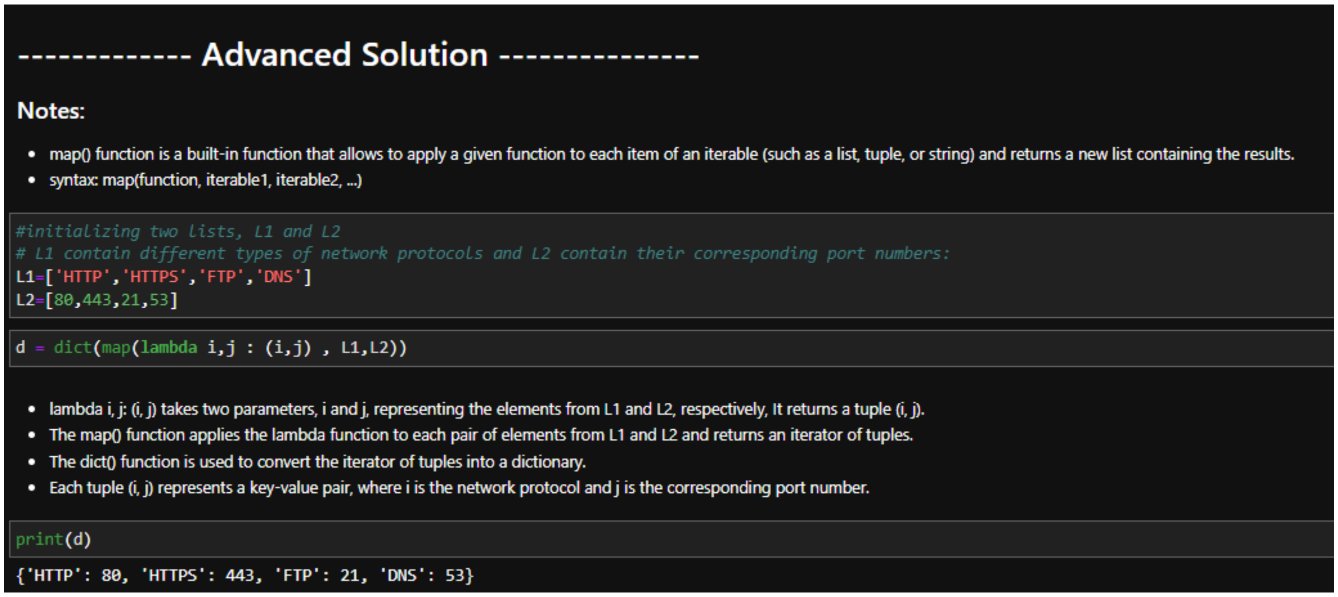
```python
# Create an empty dictionary to store the protocols and their corresponding port numbers:
d = {}
```

```python
# Iterate over each element in the L1 list (network protocols).
for key in L1:
    # Iterate over each element in the L2 list (port numbers).
    for value in L2:
        d[key] = value # Assign the current protocol (key) as the key, and the current port number (value) as the value in the dictionary.
        L2.remove(value) # Remove the current port number from the L2 list to avoid duplicate assignments.
        break # Break out of the inner loop to move to the next protocol in the L1 list.
```

```python
print(d)
```

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

Advanced Solution:



------------- Advanced Solution ---------------

Notes:

- map() function is a built-in function that allows to apply a given function to each item of an iterable (such as a list, tuple, or string) and returns a new list containing the results.
- syntax: map(function, iterable1, iterable2, ...)

```python
#initializing two lists, L1 and L2
# L1 contain different types of network protocols and L2 contain their corresponding port numbers:
L1=['HTTP','HTTPS','FTP','DNS']
L2=[80,443,21,53]
```

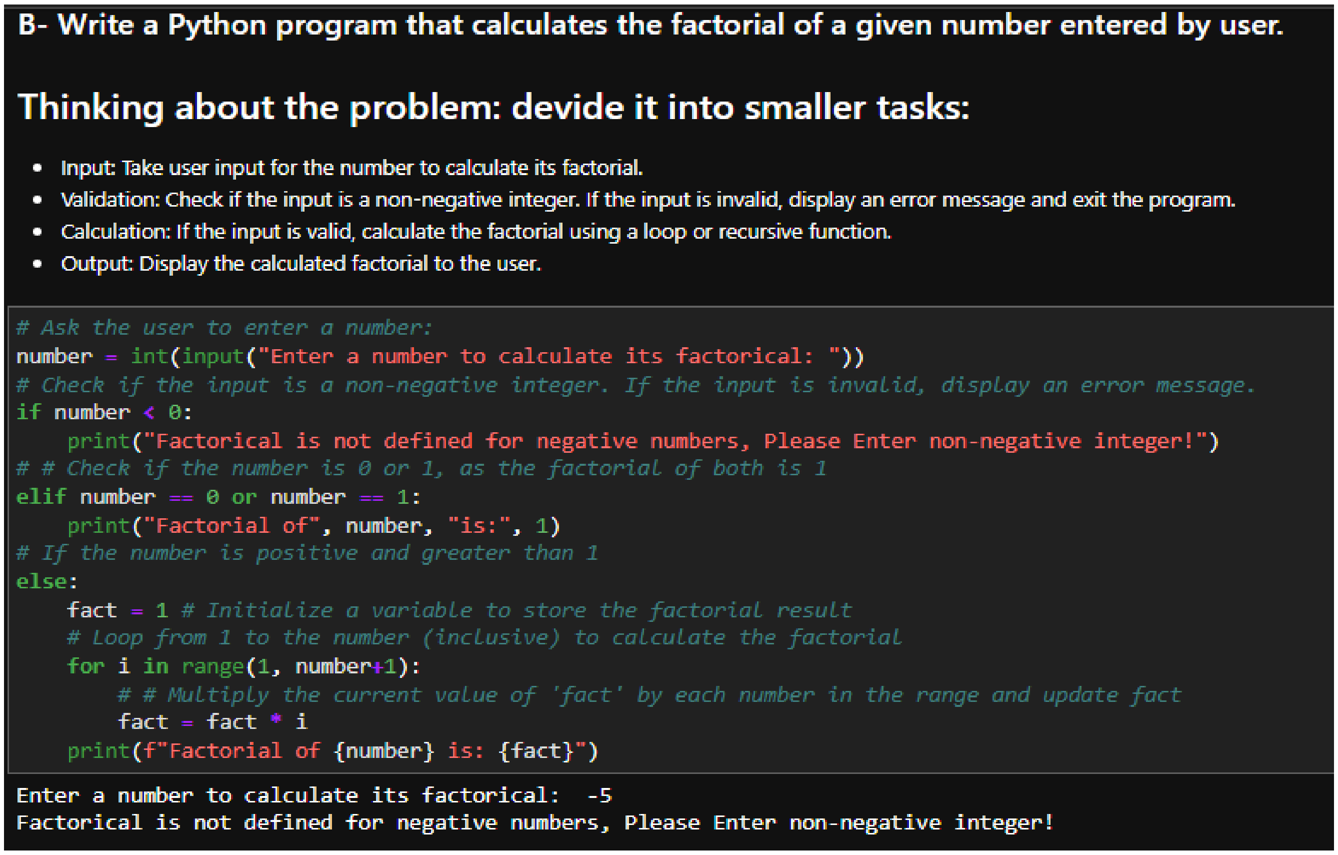
```python
d = dict(map(lambda i,j : (i,j) , L1,L2))
```

- lambda i, j: (i, j) takes two parameters, i and j, representing the elements from L1 and L2, respectively, It returns a tuple (i, j).
- The map() function applies the lambda function to each pair of elements from L1 and L2 and returns an iterator of tuples.
- The dict() function is used to convert the iterator of tuples into a dictionary.
- Each tuple (i, j) represents a key-value pair, where i is the network protocol and j is the corresponding port number.

```python
print(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.



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

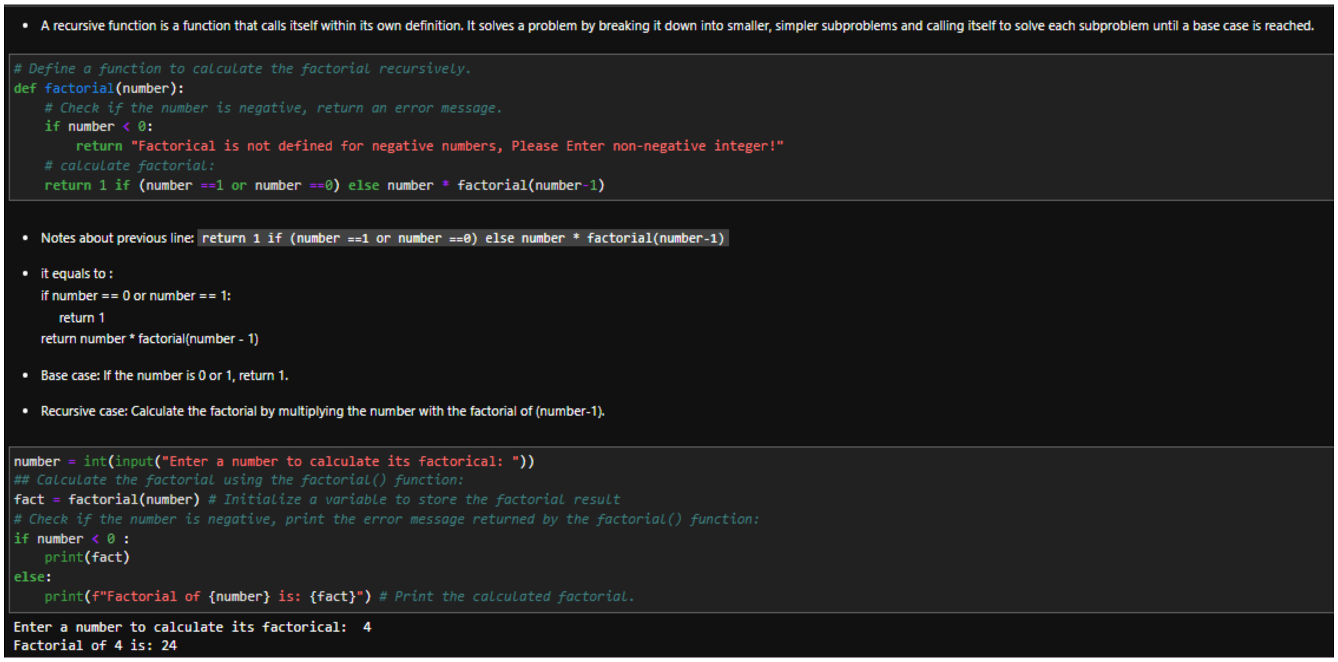
Thinking about the problem: devide it into smaller tasks:

- Input: Take user input for the number to calculate its factorial.
- Validation: Check if the input is a non-negative integer. If the input is invalid, display an error message and exit the program.
- Calculation: If the input is valid, calculate the factorial using a loop or recursive function.
- Output: Display the calculated factorial to the user.

```python
# Ask the user to enter a number:
number = int(input("Enter a number to calculate its factorical: "))
# Check if the input is a non-negative integer. If the input is invalid, display an error message.
if number < 0:
    print("Factorical is not defined for negative numbers, Please Enter non-negative integer!")
# # Check if the number is 0 or 1, as the factorial of both is 1
elif number == 0 or number == 1:
    print("Factorial of", number, "is:", 1)
# If the number is positive and greater than 1
else:
    fact = 1 # Initialize a variable to store the factorial result
    # Loop from 1 to the number (inclusive) to calculate the factorial
    for i in range(1, number+1):
        # # Multiply the current value of 'fact' by each number in the range and update fact
        fact = fact * i
    print(f"Factorial of {number} is: {fact}")
```

```
Enter a number to calculate its factorical:  -5
Factorical is not defined for negative numbers, Please Enter non-negative integer!
```

Advanced Solution:



- A recursive function is a function that calls itself within its own definition. It solves a problem by breaking it down into smaller, simpler subproblems and calling itself to solve each subproblem until a base case is reached.

```python
# Define a function to calculate the factorial recursively.
def factorial(number):
    # Check if the number is negative, return an error message.
    if number < 0:
        return "Factorical is not defined for negative numbers, Please Enter non-negative integer!"
    # calculate factorial:
    return 1 if (number ==1 or number ==0) else number * factorial(number-1)
```

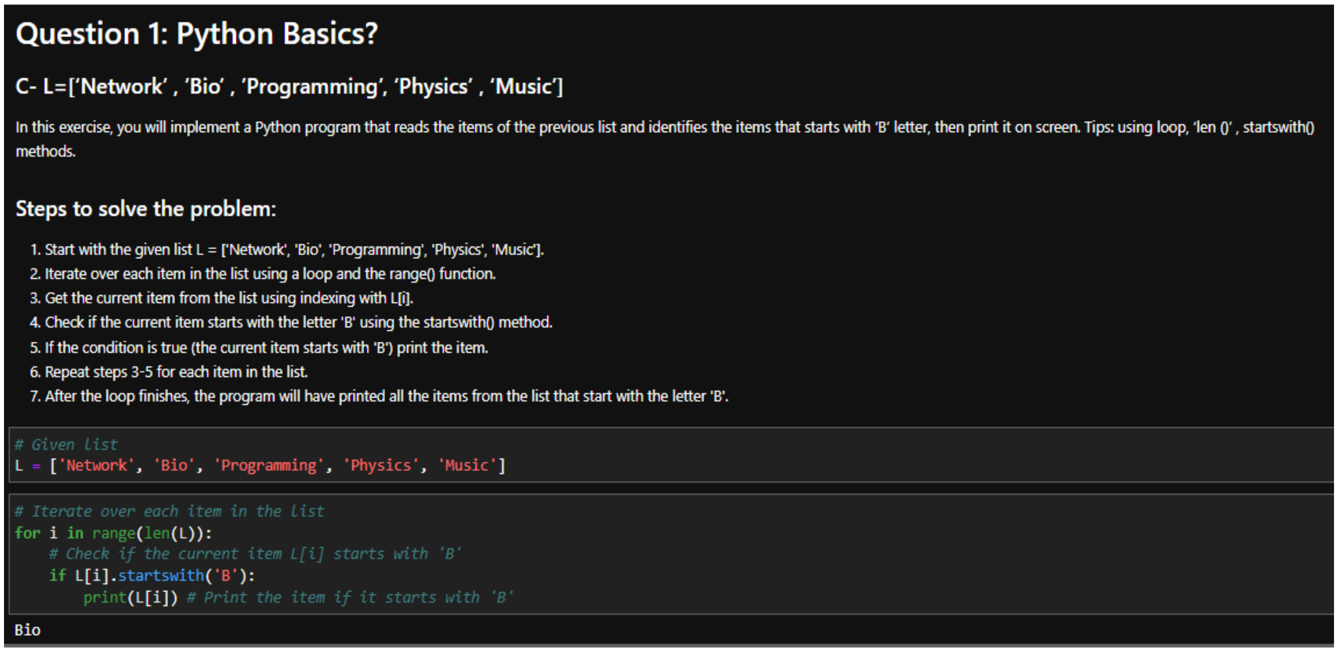
- Notes about previous line: `return 1 if (number ==1 or number ==0) else number * factorial(number-1)`
- it equals to :
  if number == 0 or number == 1:
      return 1
  return number * factorial(number - 1)
- Base case: If the number is 0 or 1, return 1.
- Recursive case: Calculate the factorial by multiplying the number with the factorial of (number-1).

```python
number = int(input("Enter a number to calculate its factorical: "))
## Calculate the factorial using the factorial() function:
fact = factorial(number) # Initialize a variable to store the factorial result
# Check if the number is negative, print the error message returned by the factorial() function:
if number < 0 :
    print(fact)
else:
    print(f"Factorial of {number} is: {fact}") # Print the calculated factorial.
```

```
Enter a number to calculate its factorical:  4
Factorial of 4 is: 24
```

....

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.



Question 1: Python Basics?

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.

Steps to solve the problem:

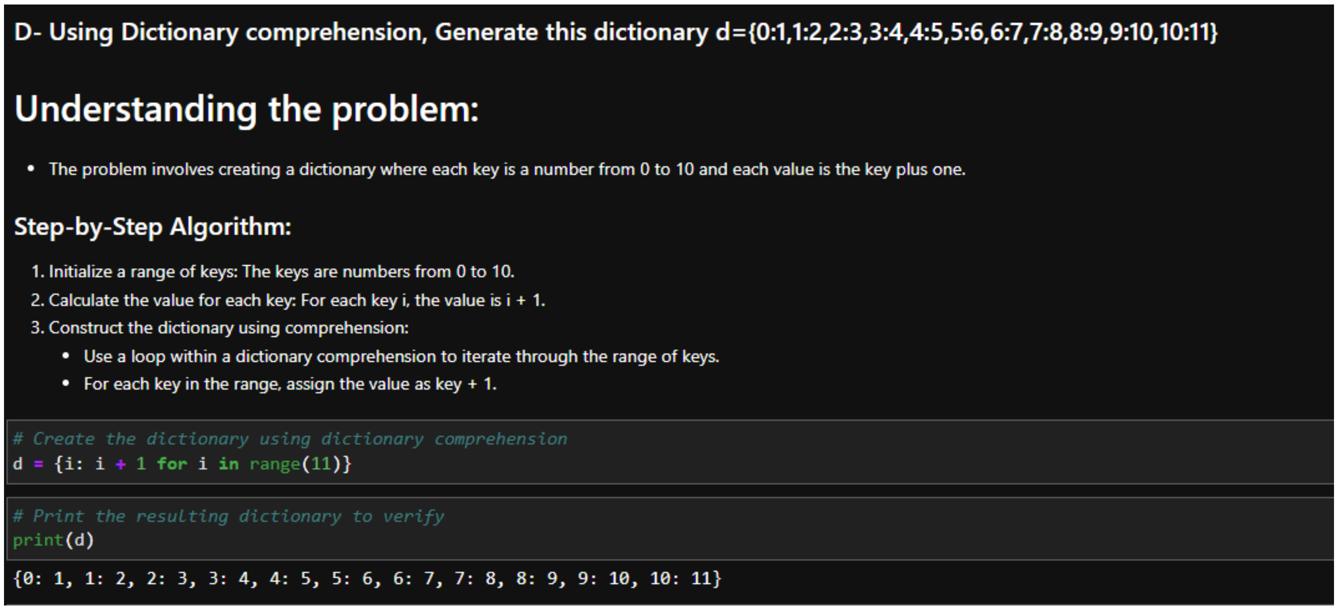
1. Start with the given list L = ['Network', 'Bio', 'Programming', 'Physics', 'Music'].
2. Iterate over each item in the list using a loop and the range() function.
3. Get the current item from the list using indexing with L[i].
4. Check if the current item starts with the letter 'B' using the startswith() method.
5. If the condition is true (the current item starts with 'B') print the item.
6. Repeat steps 3-5 for each item in the list.
7. After the loop finishes, the program will have printed all the items from the list that start with the letter 'B'.

```
# Given list
L = ['Network', 'Bio', 'Programming', 'Physics', 'Music']
```

```
# Iterate over each item in the list
for i in range(len(L)):
    # Check if the current item L[i] starts with 'B'
    if L[i].startswith('B'):
        print(L[i]) # Print the item if it starts with 'B'
```

```
Bio
```

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- 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}

Understanding the problem:

- The problem involves creating a dictionary where each key is a number from 0 to 10 and each value is the key plus one.

Step-by-Step Algorithm:

1. Initialize a range of keys: The keys are numbers from 0 to 10.
2. Calculate the value for each key: For each key i, the value is i + 1.
3. Construct the dictionary using comprehension:
   - Use a loop within a dictionary comprehension to iterate through the range of keys.
   - For each key in the range, assign the value as key + 1.

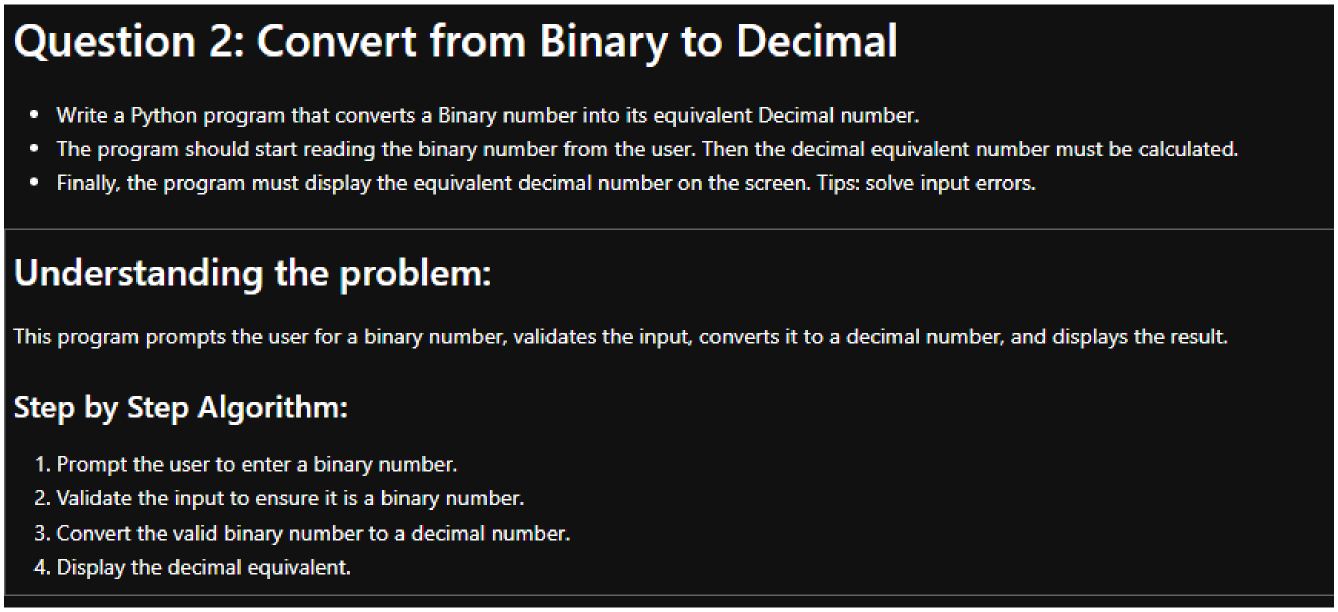
```
# Create the dictionary using dictionary comprehension
d = {i: i + 1 for i in range(11)}
```

```
# Print the resulting dictionary to verify
print(d)
```

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

Problem 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.
Tips: solve input errors



Question 2: Convert from Binary to Decimal

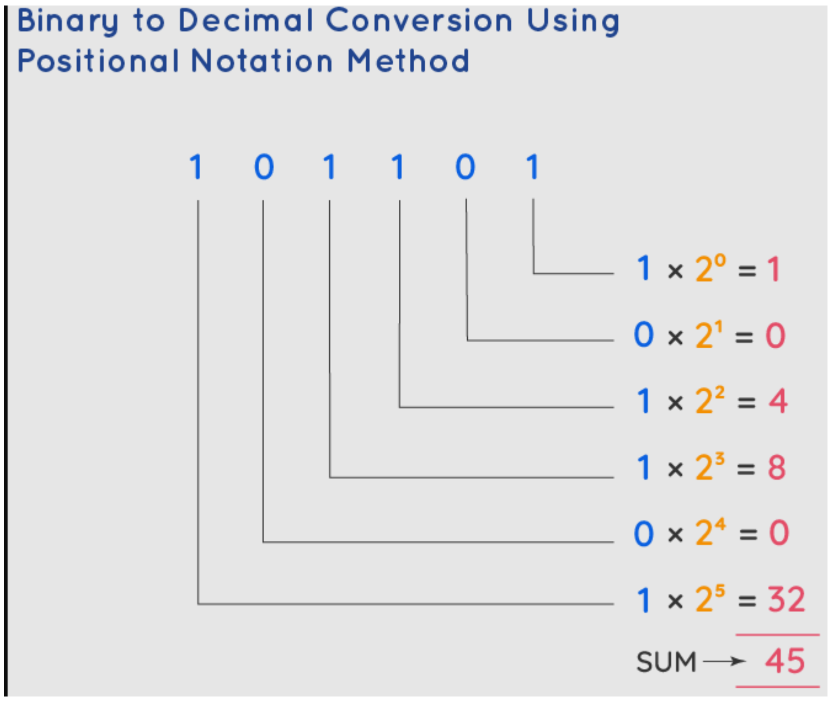
- Write a Python program that converts a Binary number into its equivalent Decimal number.
- The program should start reading the binary number from the user. Then the decimal equivalent number must be calculated.
- Finally, the program must display the equivalent decimal number on the screen. Tips: solve input errors.

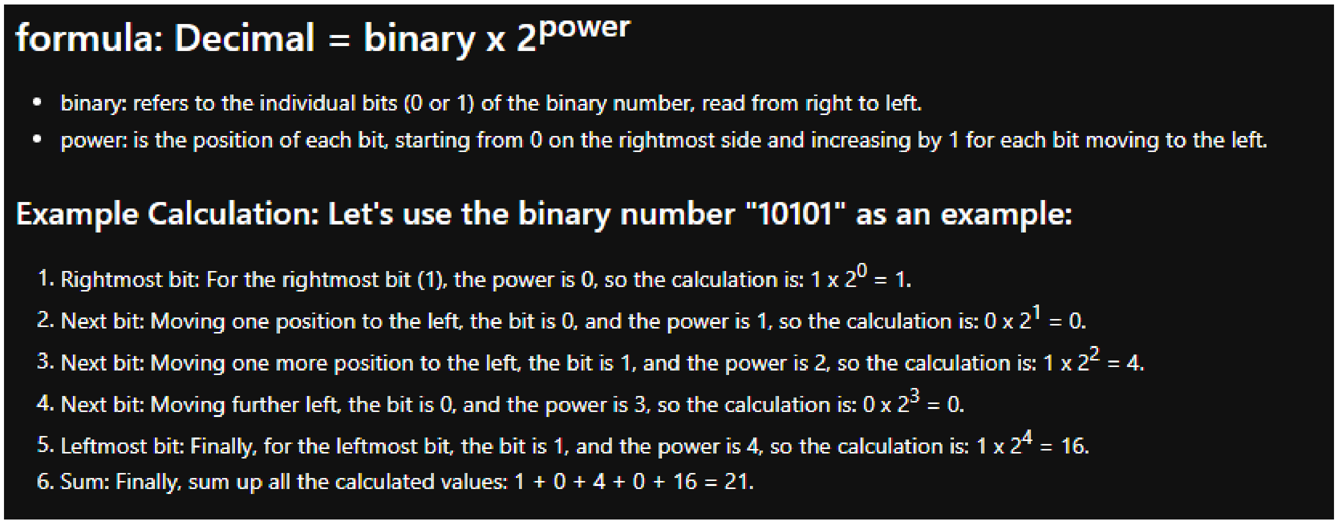
Understanding the problem:

This program prompts the user for a binary number, validates the input, converts it to a decimal number, and displays the result.

Step by Step Algorithm:

1. Prompt the user to enter a binary number.
2. Validate the input to ensure it is a binary number.
3. Convert the valid binary number to a decimal number.
4. Display the decimal equivalent.



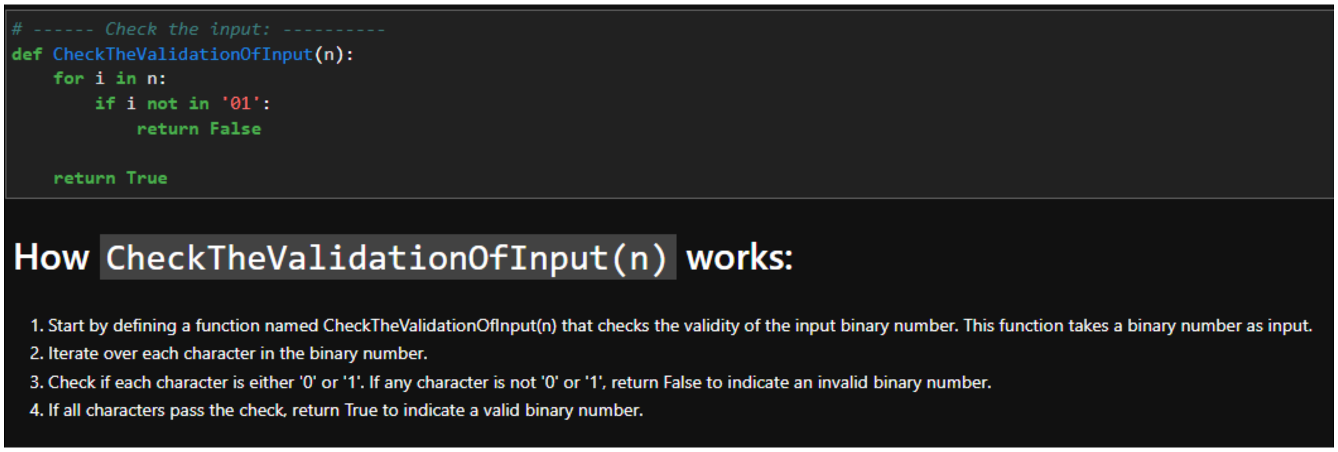


formula: Decimal = binary x 2^{power}

- binary: refers to the individual bits (0 or 1) of the binary number, read from right to left.
- power: is the position of each bit, starting from 0 on the rightmost side and increasing by 1 for each bit moving to the left.

Example Calculation: Let's use the binary number "10101" as an example:

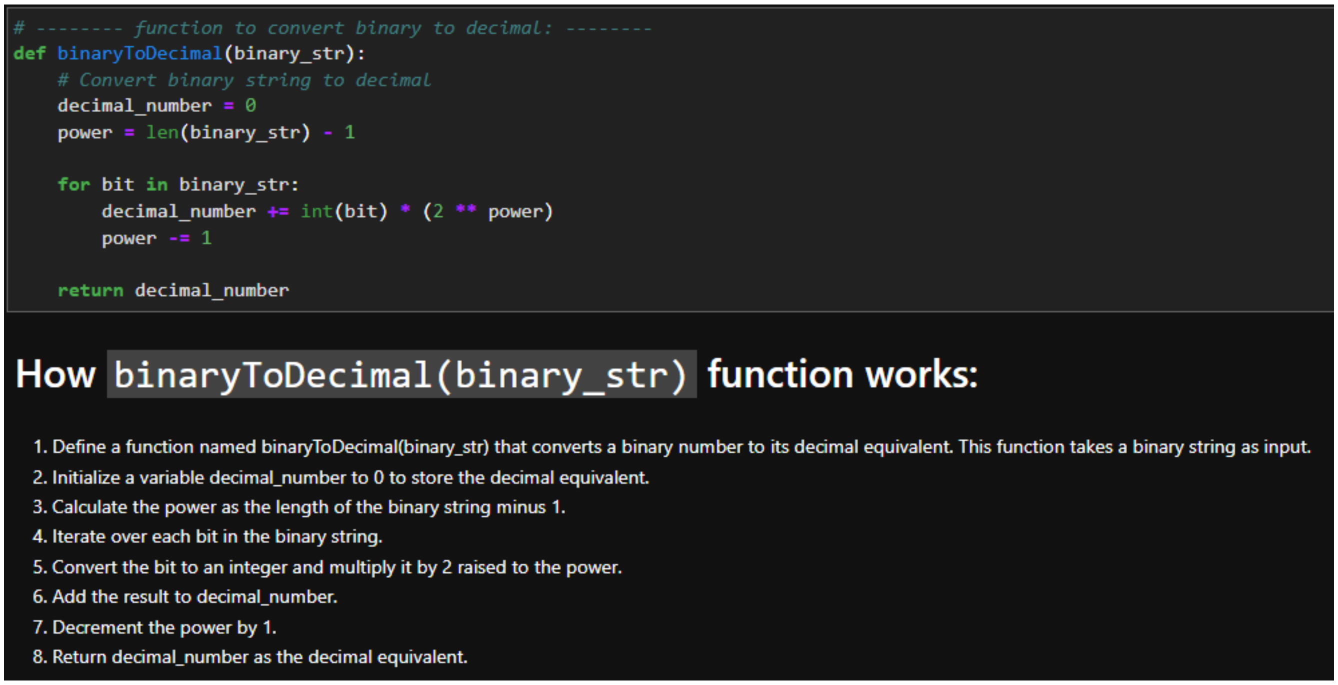
1. Rightmost bit: For the rightmost bit (1), the power is 0, so the calculation is: 1 x 2^0 = 1.
2. Next bit: Moving one position to the left, the bit is 0, and the power is 1, so the calculation is: 0 x 2^1 = 0.
3. Next bit: Moving one more position to the left, the bit is 1, and the power is 2, so the calculation is: 1 x 2^2 = 4.
4. Next bit: Moving further left, the bit is 0, and the power is 3, so the calculation is: 0 x 2^3 = 0.
5. Leftmost bit: Finally, for the leftmost bit, the bit is 1, and the power is 4, so the calculation is: 1 x 2^4 = 16.
6. Sum: Finally, sum up all the calculated values: 1 + 0 + 4 + 0 + 16 = 21.



```python
# ------ Check the input: ----------
def CheckTheValidationOfInput(n):
    for i in n:
        if i not in '01':
            return False

    return True
```

How `CheckTheValidationOfInput(n)` works:

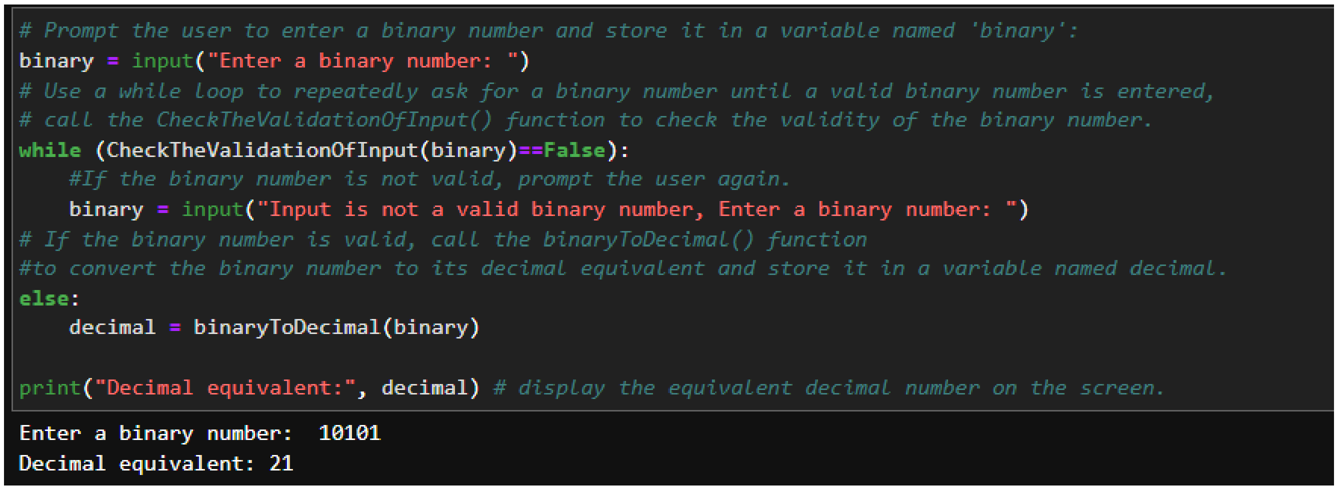
1. Start by defining a function named CheckTheValidationOfInput(n) that checks the validity of the input binary number. This function takes a binary number as input.
2. Iterate over each character in the binary number.
3. Check if each character is either '0' or '1'. If any character is not '0' or '1', return False to indicate an invalid binary number.
4. If all characters pass the check, return True to indicate a valid binary number.



```python
# -------- function to convert binary to decimal: --------
def binaryToDecimal(binary_str):
    # Convert binary string to decimal
    decimal_number = 0
    power = len(binary_str) - 1

    for bit in binary_str:
        decimal_number += int(bit) * (2 ** power)
        power -= 1

    return decimal_number
```

How `binaryToDecimal(binary_str)` function works:

1. Define a function named binaryToDecimal(binary_str) that converts a binary number to its decimal equivalent. This function takes a binary string as input.
2. Initialize a variable decimal_number to 0 to store the decimal equivalent.
3. Calculate the power as the length of the binary string minus 1.
4. Iterate over each bit in the binary string.
5. Convert the bit to an integer and multiply it by 2 raised to the power.
6. Add the result to decimal_number.
7. Decrement the power by 1.
8. Return decimal_number as the decimal equivalent.



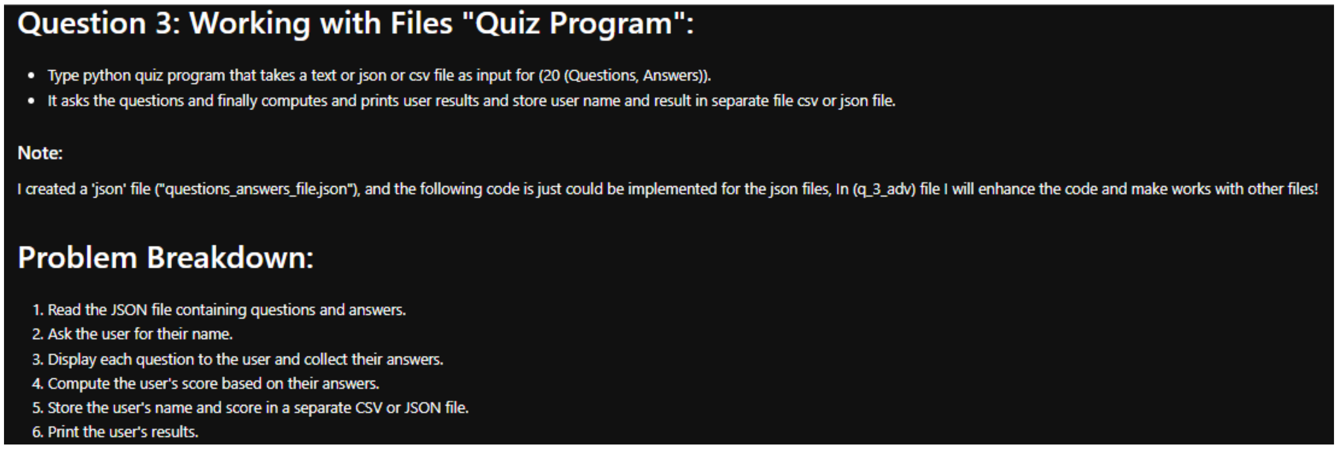
```python
# Prompt the user to enter a binary number and store it in a variable named 'binary':
binary = input("Enter a binary number: ")
# Use a while loop to repeatedly ask for a binary number until a valid binary number is entered,
# call the CheckTheValidationOfInput() function to check the validity of the binary number.
while (CheckTheValidationOfInput(binary)==False):
    #If the binary number is not valid, prompt the user again.
    binary = input("Input is not a valid binary number, Enter a binary number: ")
# If the binary number is valid, call the binaryToDecimal() function
#to convert the binary number to its decimal equivalent and store it in a variable named decimal.
else:
    decimal = binaryToDecimal(binary)

print("Decimal equivalent:", decimal) # display the equivalent decimal number on the screen.
```

```
Enter a binary number:  10101
Decimal equivalent: 21
```

Problem 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.

Basic Solution:



Question 3: Working with Files "Quiz Program":

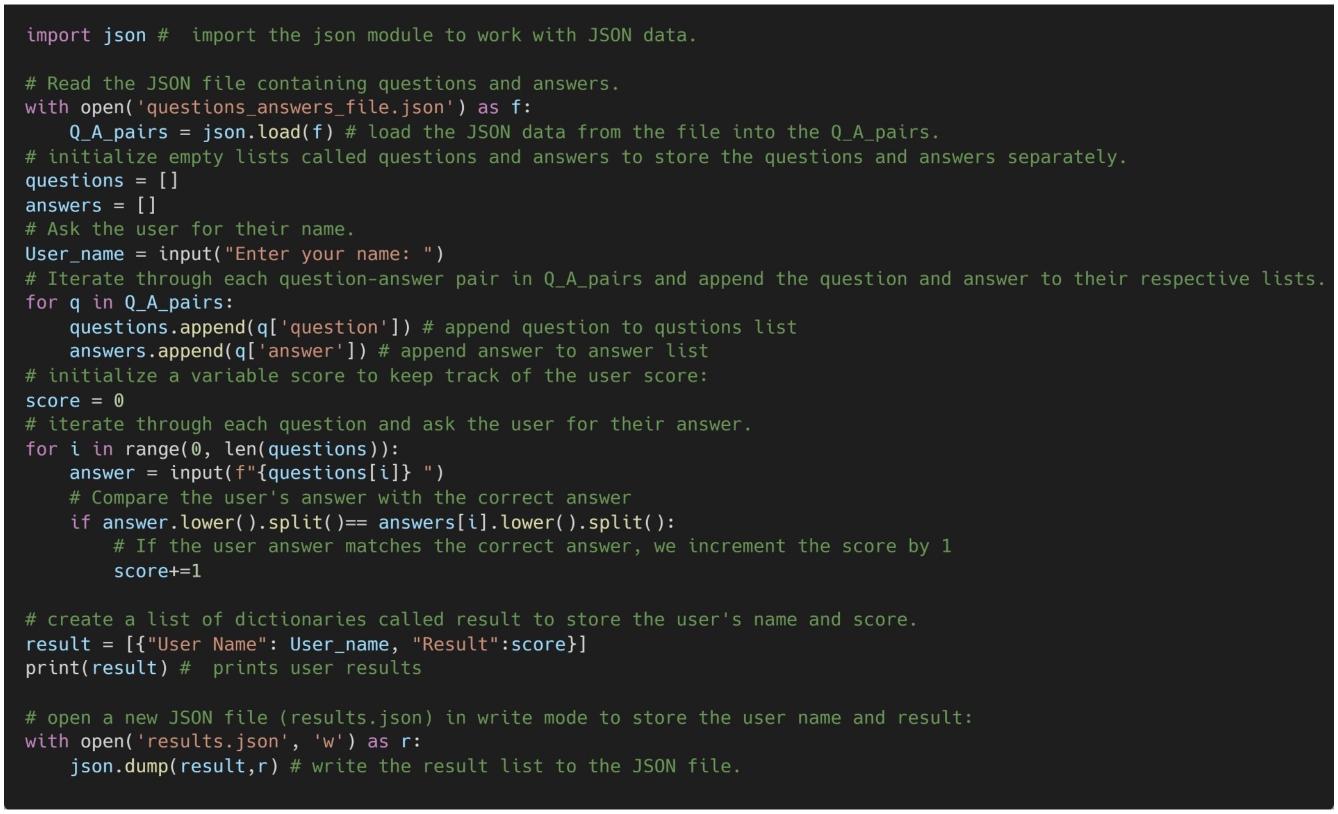
- Type python quiz program that takes a text or json or csv file as input for (20 (Questions, Answers)).
- It asks the questions and finally computes and prints user results and store user name and result in separate file csv or json file.

Note:

I created a 'json' file ("questions_answers_file.json"), and the following code is just could be implemented for the json files, In (q_3_adv) file I will enhance the code and make works with other files!

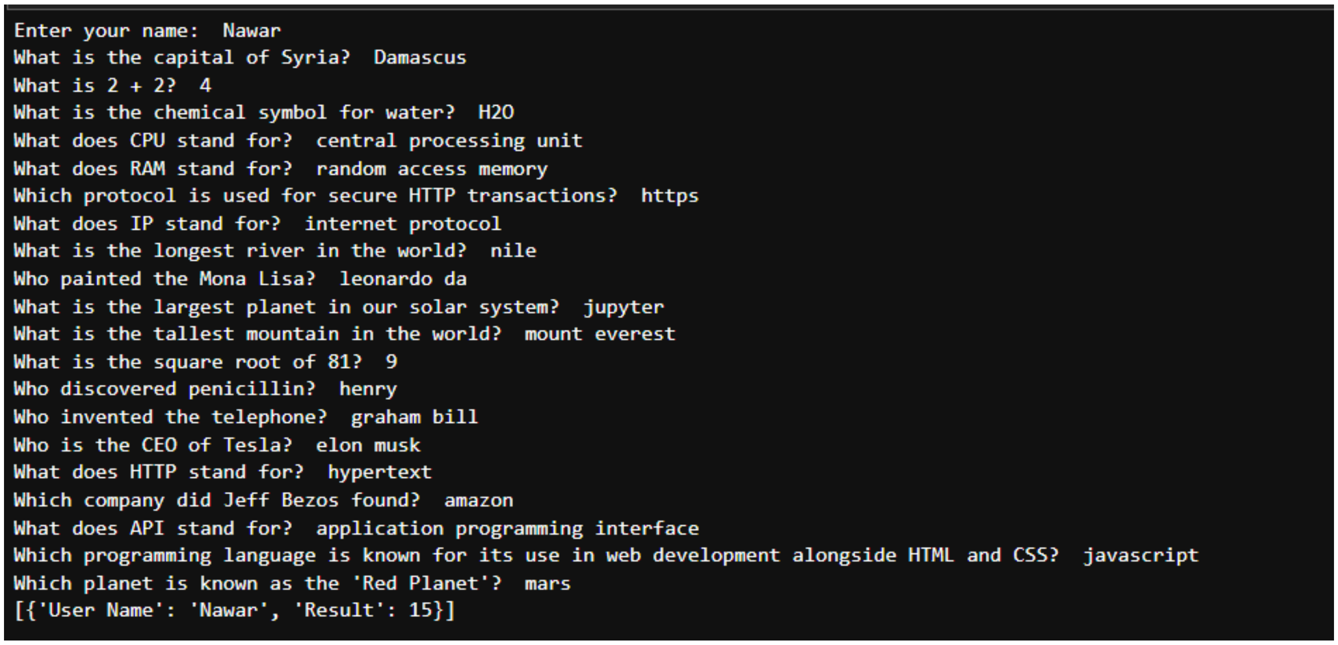
Problem Breakdown:

1. Read the JSON file containing questions and answers.
2. Ask the user for their name.
3. Display each question to the user and collect their answers.
4. Compute the user's score based on their answers.
5. Store the user's name and score in a separate CSV or JSON file.
6. Print the user's results.



```python
import json #  import the json module to work with JSON data.

# Read the JSON file containing questions and answers.
with open('questions_answers_file.json') as f:
    Q_A_pairs = json.load(f) # load the JSON data from the file into the Q_A_pairs.
# initialize empty lists called questions and answers to store the questions and answers separately.
questions = []
answers = []
# Ask the user for their name.
User_name = input("Enter your name: ")
# Iterate through each question-answer pair in Q_A_pairs and append the question and answer to their respective lists.
for q in Q_A_pairs:
    questions.append(q['question']) # append question to qustions list
    answers.append(q['answer']) # append answer to answer list
# initialize a variable score to keep track of the user score:
score = 0
# iterate through each question and ask the user for their answer.
for i in range(0, len(questions)):
    answer = input(f"{questions[i]} ")
    # Compare the user's answer with the correct answer
    if answer.lower().split()== answers[i].lower().split():
        # If the user answer matches the correct answer, we increment the score by 1
        score+=1

# create a list of dictionaries called result to store the user's name and score.
result = [{"User Name": User_name, "Result":score}]
print(result) #  prints user results

# open a new JSON file (results.json) in write mode to store the user name and result:
with open('results.json', 'w') as r:
    json.dump(result,r) # write the result list to the JSON file.
```

Output:



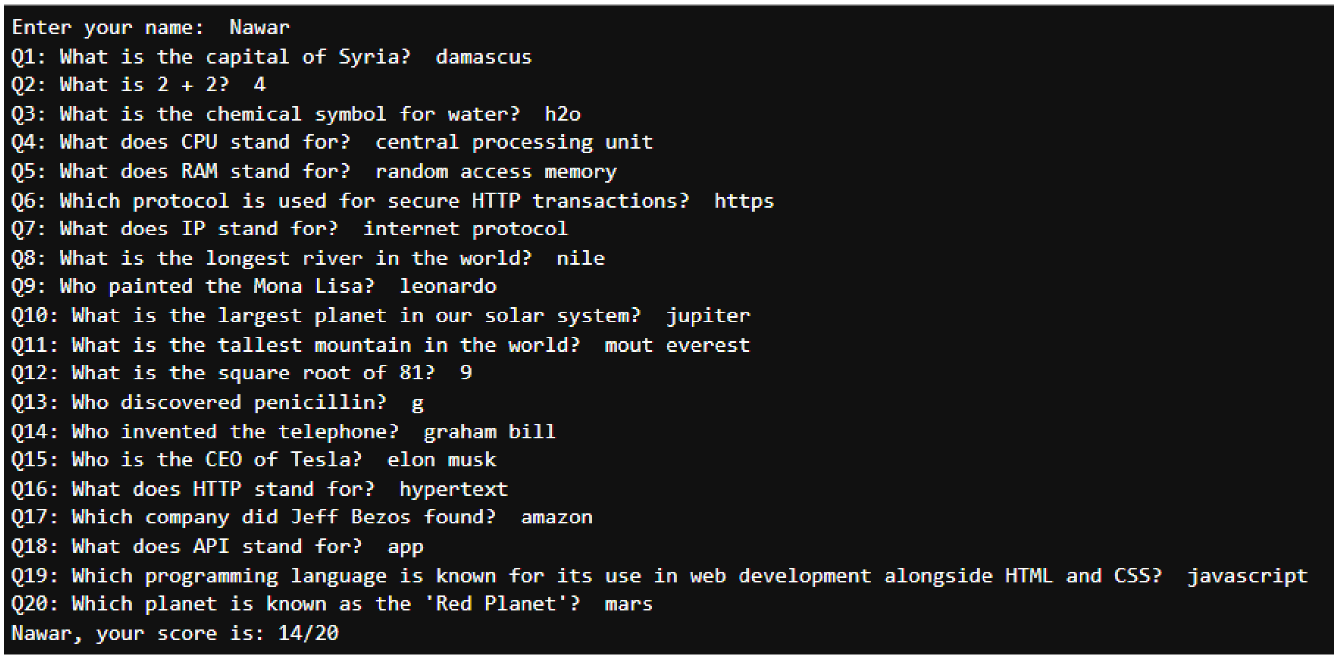
```
Enter your name:  Nawar
What is the capital of Syria?  Damascus
What is 2 + 2?  4
What is the chemical symbol for water?  H2O
What does CPU stand for?  central processing unit
What does RAM stand for?  random access memory
Which protocol is used for secure HTTP transactions?  https
What does IP stand for?  internet protocol
What is the longest river in the world?  nile
Who painted the Mona Lisa?  leonardo da
What is the largest planet in our solar system?  jupyter
What is the tallest mountain in the world?  mount everest
What is the square root of 81?  9
Who discovered penicillin?  henry
Who invented the telephone?  graham bill
Who is the CEO of Tesla?  elon musk
What does HTTP stand for?  hypertext
Which company did Jeff Bezos found?  amazon
What does API stand for?  application programming interface
Which programming language is known for its use in web development alongside HTML and CSS?  javascript
Which planet is known as the 'Red Planet'?  mars
[{'User Name': 'Nawar', 'Result': 15}]
```

Enhanced Solution:

في الحل التالي سنأخذ بعين الاعتبار أن يكون ملف الأسئلة والأجوبة إما JSON أو CSV أو text وسنعتمد على استخدام التوابع ليكون الكود أوضح و أكثر نظافة، حيث سيكون هنالك تابع لقراءة ملف الاسئلة والأجوبة والتعامل معه حسب صيغته، وتابع لحساب نتيجة المستخدم، وتابع لتخزين اسم المستخدمه ونتيجته في ملف منفصل.

```python
import json # import the json module to work with JSON data.
import csv # import the csv module to work with csv data.
#Defint function read_questions_answers() with a single parameter file_path.
#'file_path' represents the path to the file containing the questions and answers.
# initialize empty lists called questions and answers to store the questions and answers separately.
def read_questions_answers(file_path):
    questions = []
    answers = []

    # checks if the file_path ends with .json, indicating that it is a JSON file:
    if file_path.endswith('.json'):
        with open(file_path, 'r') as file:
            data = json.load(file) # load contents of the JSON file into the 'data' variable, which becomes a list of
dictionaries.
            # iterates over each item (dictionary) in 'data':
            for item in data:
                questions.append(item['question']) # appends the value of the 'question' key to the questions list
                answers.append(item['answer']) # the value of the 'answer' key to the answers list.

    # If the 'file_path' does not end with .json but ends with .csv, indicating that it is a CSV file:
    elif file_path.endswith('.csv'):
        with open(file_path, 'r') as file: # Opens the file using open() in read mode and assigns the file object to
file.
            reader = csv.reader(file) # Create a 'reader' object using csv.reader() to read the CSV file.
            for row in reader: # Iterates over each row in the 'reader', where each row is a list of values.
                questions.append(row[0]) # Appends the first value (row[0]) to the questions list
                answers.append(row[1]) # Appends the second value (row[1]) to the answers list.

    #If the file_path does not end with .json or .csv, the function assumes it is a text file.
    else:
        with open(file_path, 'r') as file: # Opens the file in read mode using open() and assigns the file object to
'file'.
            for line in file: # Iterates over each line in the file using a for loop.
                # For each line, remove leading and trailing whitespace using strip().
                # and split the line using split(',') to separate the question and answer, assuming they are comma-
separated.
                # The question is assigned to question, and the answer is assigned to answer.
                question, answer = line.strip().split(',')
                # Appends question to the questions list and answer to the answers list.
                questions.append(question) # The question is assigned to questions
                answers.append(answer) # The answer is assigned to answers

    return questions, answers # returns the questions and answers lists as a tuple.

#Define function calculate_score() two parameters: 'questions' and 'answers', representing the lists of questions and
corresponding answers.
def calculate_score(questions, answers):
    score = 0 # initialize a variable 'score' to 0 to keep track of the user's score.
    for i in range(len(questions)): # Iterate through each question
        # prompt the user with the question number (i+1), followed by the actual question from the questions list:
        answer = input(f"Q{i+1}: {questions[i]} ") # take the user's answer as input and assigns it to 'answr'
        # Check if the answer, after removing leading and trailing whitespace using strip()
        # and converting it to lowercase using lower(), matches the corresponding answer from the answers list:
        if answer.strip().lower() == answers[i].strip().lower():
            score += 1 # # If the user answer matches the correct answer, we increment the score by 1
    return score #Returns the calculated score.


#Store user name and result in separate  json file:

# Define store_user_name_and_result function with two parameters: 'User_name', and 'score'.
def store_user_name_and_result(User_name, score):
    # # create a list of dictionaries called result to store the user's name and score.
    result = [{"User Name": User_name, "Result":score}]

    # open a new JSON file (results.json) in write mode to store the user name and result:
    with open('results.json', 'w') as r:
        json.dump(result,r) # write the result list to the JSON file.


# Read questions and answers from the file:
Q_A_file = 'questions_answers_file.csv'
questions, answers = read_questions_answers(Q_A_file)

#Prompt the user for their name:
User_name = input("Enter your name: ")

#Ask the questions and calculate the score:
score = calculate_score(questions, answers)

# Print the user's score
print(f"{User_name}, your score is: {score}/20")

# Save the user's name and score to a file
result_file = 'results.json'
store_user_name_and_result(User_name, score)
```

Output:

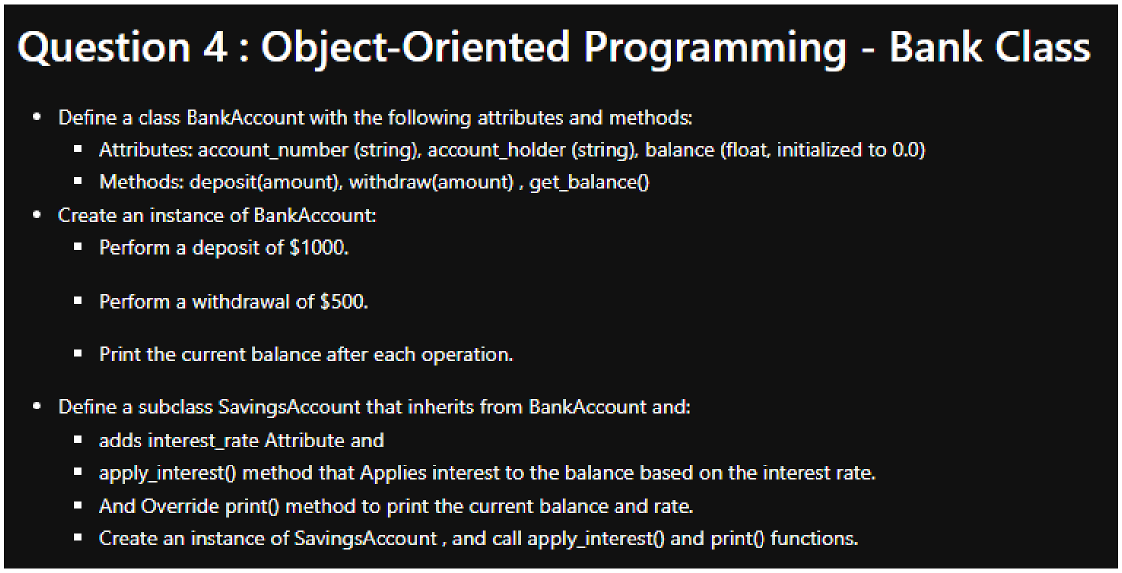


```
Enter your name:  Nawar
Q1: What is the capital of Syria?  damascus
Q2: What is 2 + 2?  4
Q3: What is the chemical symbol for water?  h2o
Q4: What does CPU stand for?  central processing unit
Q5: What does RAM stand for?  random access memory
Q6: Which protocol is used for secure HTTP transactions?  https
Q7: What does IP stand for?  internet protocol
Q8: What is the longest river in the world?  nile
Q9: Who painted the Mona Lisa?  leonardo
Q10: What is the largest planet in our solar system?  jupiter
Q11: What is the tallest mountain in the world?  mout everest
Q12: What is the square root of 81?  9
Q13: Who discovered penicillin?  g
Q14: Who invented the telephone?  graham bill
Q15: Who is the CEO of Tesla?  elon musk
Q16: What does HTTP stand for?  hypertext
Q17: Which company did Jeff Bezos found?  amazon
Q18: What does API stand for?  app
Q19: Which programming language is known for its use in web development alongside HTML and CSS?  javascript
Q20: Which planet is known as the 'Red Planet'?  mars
Nawar, your score is: 14/20
```

<u>Question 4</u>: 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



Question 4 : Object-Oriented Programming - Bank Class

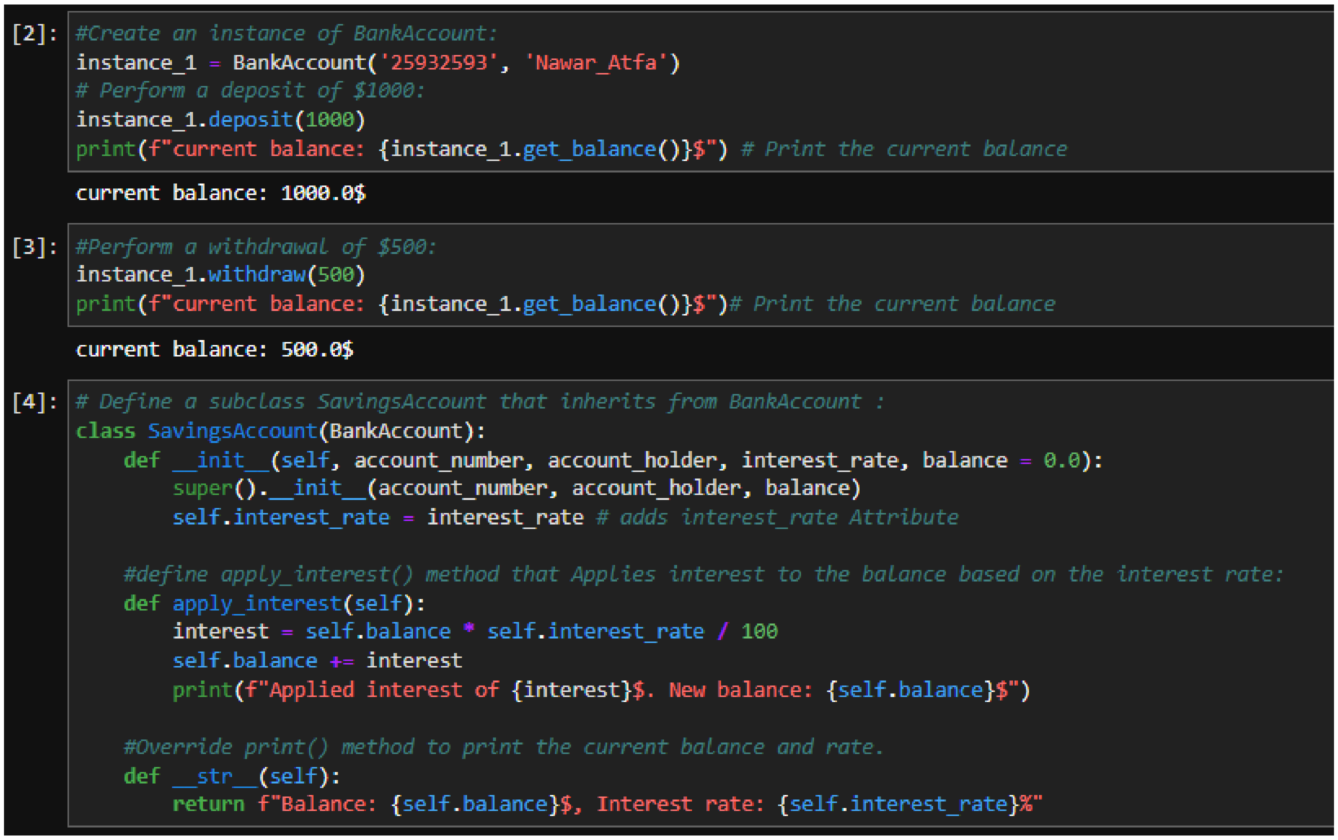
- Define a class BankAccount with the following attributes and methods:
  - Attributes: account_number (string), account_holder (string), balance (float, initialized to 0.0)
  - Methods: deposit(amount), withdraw(amount) , get_balance()
- Create an instance of BankAccount:
  - Perform a deposit of $1000.
  - 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.

Notes: Definition of some terms for a better understanding of the program

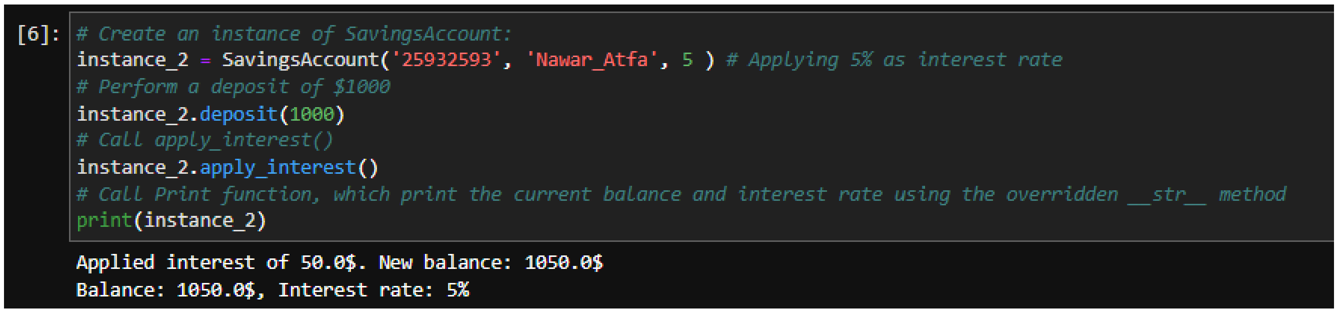
- Deposit: A deposit is the act of adding funds (money) to a bank account. In the code When we call deposit(amount), it updates the balance attribute of the BankAccount instance by adding the specified amount to the current balance.
- Withdrawal: A withdrawal is the act of taking funds out of a bank account. In the code, When we call withdraw(amount), it checks if the balance is sufficient to cover the withdrawal. If so, it subtracts the amount from the balance. If not, it prints an error message indicating insufficient funds.
- Balance: The balance is the amount of money currently available in a bank account. In the code, The balance attribute keeps track of the current amount of money in the account. It is initialized to 0.0 and is modified by the deposit and withdraw methods. You can get the current balance using the get_balance method or by accessing the balance attribute directly.
- Interest Rate: The interest rate is the percentage at which interest is calculated on the balance of a savings account. In the code, the interest_rate attribute in the SavingsAccount class is an attribute that determines how much interest is applied to the balance, and it used in the apply_interest() method to calculate the interest. The method multiplies the balance by the interest_rate (divided by 100 to convert from percentage) to determine the interest amount, which is then added to the balance.

[1]:

```python
# Define a class BankAccount:
class BankAccount:
    # Attributes:
    def __init__(self, account_number, account_holder, balance = 0.0):
        self.account_number = account_number
        self.account_holder = account_holder
        self.balance = balance
    # Methods:
    def deposit(self,amount):
        self.balance += amount

    def withdraw(self, amount):
        if self.balance >= amount:
            self.balance -= amount
        else:
            print("you don't have enough money on your balance!")
    def get_balance(self):
        return self.balance
```



```
[2]: #Create an instance of BankAccount:
     instance_1 = BankAccount('25932593', 'Nawar_Atfa')
     # Perform a deposit of $1000:
     instance_1.deposit(1000)
     print(f"current balance: {instance_1.get_balance()}$") # Print the current balance

     current balance: 1000.0$

[3]: #Perform a withdrawal of $500:
     instance_1.withdraw(500)
     print(f"current balance: {instance_1.get_balance()}$")# Print the current balance

     current balance: 500.0$

[4]: # Define a subclass SavingsAccount that inherits from BankAccount :
     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 # adds interest_rate Attribute

         #define apply_interest() method that Applies interest to the balance based on the interest rate:
         def apply_interest(self):
             interest = self.balance * self.interest_rate / 100
             self.balance += interest
             print(f"Applied interest of {interest}$. New balance: {self.balance}$")

         #Override print() method to print the current balance and rate.
         def __str__(self):
             return f"Balance: {self.balance}$, Interest rate: {self.interest_rate}%"
```



```
[6]: # Create an instance of SavingsAccount:
     instance_2 = SavingsAccount('25932593', 'Nawar_Atfa', 5 ) # Applying 5% as interest rate
     # Perform a deposit of $1000
     instance_2.deposit(1000)
     # Call apply_interest()
     instance_2.apply_interest()
     # Call Print function, which print the current balance and interest rate using the overridden __str__ method
     print(instance_2)

     Applied interest of 50.0$. New balance: 1050.0$
     Balance: 1050.0$, Interest rate: 5%
```