الجمهورية العربية السورية

جامعة تشرين

كلية الهندسة الميكانيكية والكهربائية

قسم هندسة الاتصالات والإلكترونيات

السنة الخامسة

وظيفة python

إعداد الطالب :

نور هديوه

إشراف :

د. مهند عيسى

العام الدراسي : 2023 - 2024

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 }

```
L1 = ['HTTP', 'HTTPS', 'FTP', 'DNS']

L2 = [80, 443, 21, 53]

d = dict(zip(L1, L2))

print(d)
```

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

```
def factorial(n):


  if n < 0:

    return

  elif n == 0:

    return 1

  else:

    result = 1

    for i in range(1, n + 1):

      result *= i

    return result


# Get the number from the user

num = int(input("Enter a non-negative integer: "))


# Calculate the factorial

factorial_result = factorial(num)


# Print the result

if isinstance(factorial_result, str):

  print(factorial_result)

else:

  print(f"The factorial of {num} is {factorial_result}")
```

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.

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


# Iterate through the list
for item in L:
  # Check if the item starts with 'B' (case-sensitive)
  if item.startswith('B'):
    print(item)
```

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 = {i: i + 1 for i in range(11)}
print(d)
```

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.

```
def binary_to_decimal(binary_str):


  # Check if the input string only contains 0s and 1s
  if not all(char in '01' for char in binary_str):
    return None


  decimal_value = 0
  power = 0
  for digit in binary_str[::-1]:
    # Convert digit to integer (handles potential '0' or '1' input)
    digit_int = int(digit)
```

```
    decimal_value += digit_int * (2 ** power)
    power += 1
  return decimal_value

# Get binary input from the user
while True:
  binary_str = input("Enter a binary number: ")
  decimal_equivalent = binary_to_decimal(binary_str)

  if decimal_equivalent is None:
    print("Invalid binary input. Please enter a string containing only 0s and 1s.")
  else:
    print(f"The decimal equivalent of {binary_str} is {decimal_equivalent}.")
    break
```

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.

```
[
  {"question": "What is the capital of France?", "answer": "Paris"},
  {"question": "What is 2 + 2?", "answer": "4"},
  {"question": "What is the color of the sky?", "answer": "Blue"},
  {"question": "What is the largest planet in our solar system?", "answer": "Jupiter"},
  {"question": "What is the boiling point of water?", "answer": "100"},
  {"question": "What is the currency of the United States?", "answer": "Dollar"},
  {"question": "Who wrote 'To Kill a Mockingbird'?", "answer": "Harper Lee"},
  {"question": "What is the chemical symbol for gold?", "answer": "Au"},
  {"question": "What is the capital of Japan?", "answer": "Tokyo"},
  {"question": "What is the largest mammal?", "answer": "Blue Whale"},
  {"question": "What is the smallest prime number?", "answer": "2"},
  {"question": "What is the main ingredient in guacamole?", "answer": "Avocado"},
  {"question": "What is the hardest natural substance on Earth?", "answer": "Diamond"},
```

```
    {"question": "What is the tallest mountain in the world?", "answer": "Mount Everest"},
    {"question": "Who painted the Mona Lisa?", "answer": "Leonardo da Vinci"},
    {"question": "What is the capital of Canada?", "answer": "Ottawa"},
    {"question": "What is the main gas found in the air we breathe?", "answer": "Nitrogen"},
    {"question": "Who is known as the Father of Computers?", "answer": "Charles Babbage"},
    {"question": "What is the square root of 64?", "answer": "8"},
    {"question": "What is the longest river in the world?", "answer": "Nile"}
]



import json
import csv

def load_quiz_data(filename):
  """
  Loads quiz data (questions and answers) from a file.

  Args:
    filename: The path to the quiz data file (text, JSON, or CSV).

  Returns:
    A list of dictionaries, where each dictionary represents a question-answer pair.
  """
  with open(filename, 'r') as f:
    if filename.endswith('.json'):
      data = json.load(f)
    elif filename.endswith('.csv'):
      reader = csv.reader(f)
      data = list(reader)
    else:
      # Handle text file format (assuming each line is a question-answer pair separated by a
colon)
      data = [line.strip().split(':') for line in f.readlines()]
  return data

def ask_question(question):
  """
  Asks the user a question and returns their answer.
  """
  answer = input(question + " ")
  return answer.strip()

def grade_quiz(questions, answers):
  """
  Grades the quiz and calculates the score.
```

```
  Args:
    questions: A list of questions.
    answers: A list of user answers.

  Returns:
    The user's score (number of correct answers).
  """
  score = 0
  for i, (question, answer) in enumerate(zip(questions, answers)):
   if answer.lower() == question['answer'].lower():
    score += 1
  return score

def save_results(username, score, filename):
  """
  Saves the user's name and score to a file (CSV or JSON).

  Args:
    username: The user's name.
    score: The user's quiz score.
    filename: The path to the results file (CSV or JSON).
  """
  data = {'username': username, 'score': score}
  with open(filename, 'a') as f:
   if filename.endswith('.json'):
    json.dump(data, f, indent=2)
   else:
    writer = csv.writer(f)
    writer.writerow([username, score])

def main():
  # Get quiz data filename
  filename = input("Enter the quiz data file (text, JSON, or CSV): ")

  # Load quiz data
  quiz_data = load_quiz_data(filename)

  # Get user name
  username = input("Enter your name: ")

  # Initialize empty lists for questions and answers
  questions = []
  answers = []

  # Ask questions and store answers
  for question in quiz_data:
   if isinstance(question, dict):
    questions.append(question['question'])
   else:
    questions.append(question[0])
   answer = ask_question(question)
   answers.append(answer)
```

```
 # Calculate score
 score = grade_quiz(questions, answers)

 # Display results
 print(f"Hi {username}, your score is {score} out of {len(questions)}.")

 # Save results (optional)
 result_filename = input("Enter a filename to save your results (optional, CSV or JSON): ")
 if result_filename:
  save_results(username, score, result_filename)

if __name__ == "__main__":
 main()
```

Question 4: Object-Oriented Programming - Bank ClassDefine 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):

  self.account_number = account_number

  self.account_holder = account_holder

  self.balance = 0.0  # Initialize balance to 0.0


 def deposit(self, amount):


  self.balance += amount

  print(f"Deposited ${amount:.2f}. New balance: ${self.balance:.2f}")


 def withdraw(self, amount):
```

```
    if amount > self.balance:
        print(f"Insufficient funds. Available balance: ${self.balance:.2f}")
    else:
        self.balance -= amount
        print(f"Withdrew ${amount:.2f}. New balance: ${self.balance:.2f}")

    def get_balance(self):

        return self.balance

    def __str__(self):

        return f"Account Number: {self.account_number}\nAccount Holder:
{self.account_holder}\nBalance: ${self.balance:.2f}"

class SavingsAccount(BankAccount):


    def __init__(self, account_number, account_holder, interest_rate):

        super().__init__(account_number, account_holder)
        self.interest_rate = interest_rate

    def apply_interest(self):

        interest = self.balance * self.interest_rate
        self.balance += interest
        print(f"Interest applied: ${interest:.2f}. New balance: ${self.balance:.2f}")

    def __str__(self):
```

```
    return super().__str__() + f"\nInterest Rate: {self.interest_rate:.2%}"  # Display interest rate as a percentage

# Create a BankAccount instance
my_account = BankAccount("12345678", "John Doe")

# Deposit $1000
my_account.deposit(1000.00)

# Withdraw $500
my_account.withdraw(500.00)

# Print current balance (should be $500.00)
print(my_account)

# Create a SavingsAccount instance with 2% interest
savings = SavingsAccount("87654321", "Jane Smith", 0.02)

# Deposit $1500
savings.deposit(1500.00)

# Apply interest
savings.apply_interest()

# Print current balance and interest rate (should be $1530.00 and 2.00%)
print(savings)
```