الجمهورية العربية السورية

وزارة التعليم العالي- جامعة تشرين

هندسة الاتصالات والالكترونيات

السنة الدراسية الخامسة– الفصل الثاني

وظيفة في برمجة الشبكات

إعداد الطالب:

حنين نضال معلا ٢٨٤٤

إشراف: الدكتور مهند عيسى

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 **}**

Answer:

```
d={}
l1=['HTTP','HTTPS','FTP','DNS']
l2=[80,433,20,53]
for i,j in zip(l1,l2):
    d[i]=j
print(d)
```

```
{'HTTP': 80, 'HTTPS': 433, 'FTP': 20, 'DNS': 53}
```

Explanation:

This Python code creates a lookup table that associates protocol names with their corresponding port numbers. Here's a simplified explanation:

1. **Setting up an empty container:**
   - `d = {}` creates a dictionary named `d` to store the protocol names and port numbers.
2. **Preparing the data:**
   - Two lists are created:
     - `l1 = ['HTTP', 'HTTPS', 'FTP', 'DNS']` contains a list of protocol names (like HTTP, HTTPS, etc.).
     - `l2 = [80, 443, 20, 53]` contains the corresponding port numbers for each protocol in the same order
3. **Building the lookup table:**
   - The `for` loop iterates through both lists simultaneously using `zip`. In each iteration:
     - `i` takes on a protocol name from `l1`.
     - `j` takes on the corresponding port number from `l2`.
   - Inside the loop, `d[i] = j` adds an entry to the dictionary `d`. The protocol name (`i`) becomes the key, and the port number (`j`) becomes the value. This builds the association between protocols and their ports.

.

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

Answer:

```
num = int (input("enter a number: "))
for i in range (0,11):
    x=i*num
    print (num,  'x', i, "=", x)
```

```
Python Console
enter a number: >? 3
3 x 0 = 0
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30
```

Explanation:

This Python code generates a multiplication table for a number entered by the user.

1. **User Input:**
    - `num = int(input("enter a number: "))` prompts the user to enter a number.
2. **Looping for Multiplication:**
    - `for i in range(0, 11):` starts a `for` loop that iterates 10 times The variable `i` takes on a value from 0 to 9 in each iteration.
3. **Calculating the Product:**
    - `x = i * num` inside the loop calculates the product of `i` and the number entered by the user (`num`). This gives you the result of multiplying `num` by different numbers from 0 to 9.
4. **Printing the Table:**

- `print(num, 'x', i, "=", x)` inside the loop prints the multiplication table in a formatted way. It displays the user's number (`num`), "x" (multiplication symbol), the current loop counter (`i`), "=", and the calculated product (`x`).

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.

Answer:

```
L=['Network', 'Bio', 'Programming', 'physics', 'Music']
i=0
for i in range(len(L)):
    if L[i].startswith("B"):
        print(L[i])
```

```
Bio
```

Explanation:

This Python code filters and prints words from a list that start with the letter "B". Here's a step-by-step explanation:

1. **Looping through the List:**
   - `i = 0` sets a variable `i` to 0, which will be used as a counter for iterating through the list.
   - `for i in range(len(L)):` starts a `for` loop. `range(len(L))` generates a sequence of numbers from 0 to the length of the list `L` -1
   - In each iteration, the value of `i` increases by 1, effectively going through each element's index in the list `L`.
2. **Checking for Words Starting with "B"**
   - `if L[i].startswith("B"):` checks if the word at the current index (`i`) in the list `L` starts with the letter "B" using the `startswith` method.
   - If the condition is `True` (the word starts with "B"), the code inside the `if` block executes.
3. **Printing Matching Words:**

- `print(L[i])` inside the `if` block prints the word at the current index (`i`) from the list `L`. Since only words starting with "B" are checked within the `if` block, this will print only those words.

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}

Answer:

```
d={a:a+1 for a in range(0,11)}
print(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:
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.

Answer:

```
def is_binary(num):
    try:
        int(num, 2)
        return True
    except ValueError:
        return False
def main():
    bi = input("Enter a binary number: ")
    if is_binary(bi):
        D = 0
        for i in range(len(bi)):
            if bi[i] == '1':
                D = D + 2**(len(bi)-i-1)
        print("The decimal of the number is",D)
    else:
        print("The enterd number is invalid format,please enter a binary number")

main()
```

```
Enter a binary number: 101010
The decimal of the number is 42
```

Explenation:

The given code defines two functions:

1. `is_binary(num)`: This function checks if a given number (`num`) is a binary number. It tries to convert the number to a base-2 integer (`int(num, 2)`) and returns `True` if the conversion is successful (meaning the number only contains 0s and 1s). If there's an error during the conversion (due to invalid characters or non-binary format), it returns `False`.
2. `main()`: This function prompts the user to enter a binary number using `input("Enter a binary number: ")`. Then, it calls the `is_binary` function to check if the entered value is a valid binary number.
    - If it is binary (`is_binary(bi) returns True`), the code converts the binary number to its decimal equivalent. It iterates through each digit of the binary number (`bi[i]`) and adds the corresponding power of 2 to a variable `D` if the digit is '1'. Finally, it prints the decimal value of the binary number.
    - If it is not binary (`is_binary(bi) returns False`), the code prints an error message indicating that the entered value is not a valid binary number.

Question 3:

Working with Files" Quiz Program"

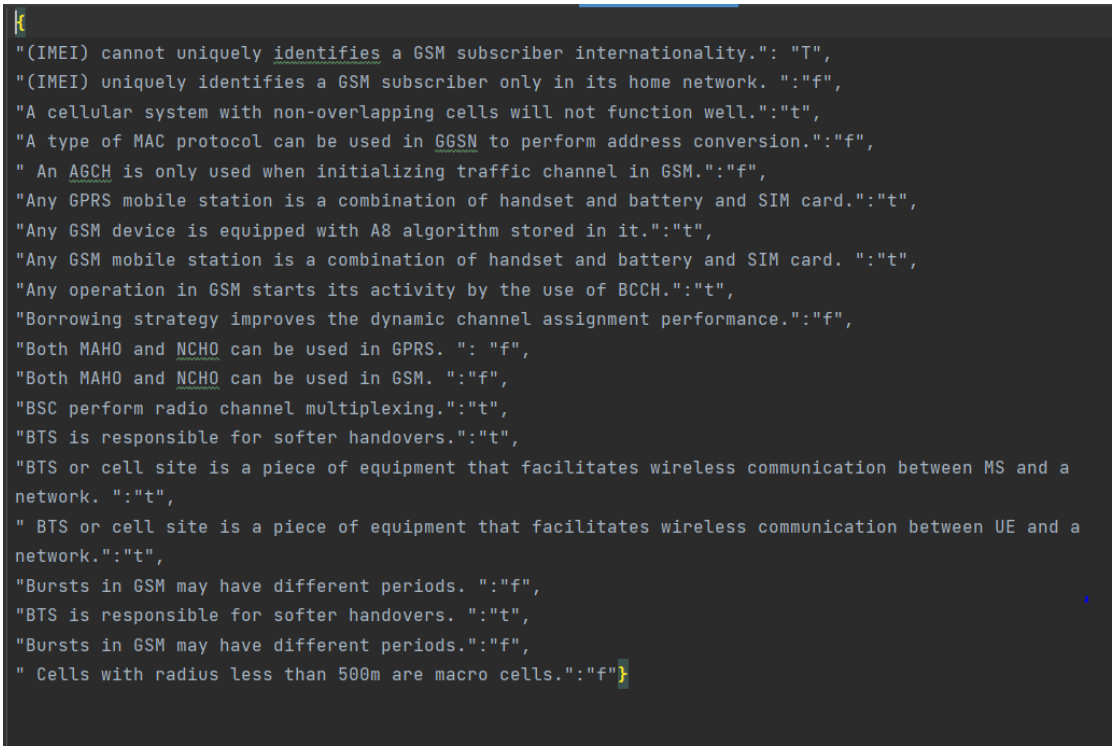
Type python quiz program that takes a text or json or csv file as input for (20 (Questions, Answers)). It asks the
questions and finally computes and prints user results and store user name and result in separate file csv or json file.

Answer:

```
import json
questions = {}

scores = 0

number=1

f = open("questions.txt",'r')
questions = json.load(f)
f.close()

print("python quiz programm")
print("Enter t for True or f for False")
name = input("Enter your full name: ")

for ques in questions.keys():

    print("Question",number,": ", ques)
    ans = input("The answer is ")

    if ans.upper() == questions[ques].upper():
        scores = scores + 1
        print("Correct ")
    else:
        print ("Wrong")
    number = number + 1

result={name:scores}
m = open("score.txt",'w')
result = json.dump(result,m)
m.close()
```

```
python quiz programm
Enter t for True or f for False
Enter your full name: haneen nidal moalla
Question 1 :  10.0.0.5 is a private ip address.
The answer is t
Correct
Question 2 :  153.16.2.8 is a private ip address.
The answer is t
Wrong
Question 3 :  ARP refers to Address Resolution Protocol.
The answer is f
Wrong
Question 4 :  TCP is a network layer protocol.
The answer is f
Correct
Question 5 :  IPv4 is a 128-bit address.
The answer is t
Wrong
Question 6 :  IPv6 is a 128-bit address.
The answer is f
Wrong
Question 7 :  SDN refers to Software Defined Network.
The answer is t
Correct
Question 8 :  UDP is a Transport Layer protocol.
The answer is f
Wrong
Question 9 :  224.0.0.9 is a multicast address.
The answer is t
Correct
Question 10 :  192.168.1.1 is a class A address.
```

Questiones:



```
{
"(IMEI) cannot uniquely identifies a GSM subscriber internationality.": "T",
"(IMEI) uniquely identifies a GSM subscriber only in its home network. ":"f",
"A cellular system with non-overlapping cells will not function well.":"t",
"A type of MAC protocol can be used in GGSN to perform address conversion.":"f",
" An AGCH is only used when initializing traffic channel in GSM.":"f",
"Any GPRS mobile station is a combination of handset and battery and SIM card.":"t",
"Any GSM device is equipped with A8 algorithm stored in it.":"t",
"Any GSM mobile station is a combination of handset and battery and SIM card. ":"t",
"Any operation in GSM starts its activity by the use of BCCH.":"t",
"Borrowing strategy improves the dynamic channel assignment performance.":"f",
"Both MAHO and NCHO can be used in GPRS. ": "f",
"Both MAHO and NCHO can be used in GSM. ":"f",
"BSC perform radio channel multiplexing.":"t",
"BTS is responsible for softer handovers.":"t",
"BTS or cell site is a piece of equipment that facilitates wireless communication between MS and a
network. ":"t",
" BTS or cell site is a piece of equipment that facilitates wireless communication between UE and a
network.":"t",
"Bursts in GSM may have different periods. ":"f",
"BTS is responsible for softer handovers. ":"t",
"Bursts in GSM may have different periods.":"f",
" Cells with radius less than 500m are macro cells.":"f"}
```

Explenation:

1. **Import and Load Questions:**
    - It starts by importing the `json` library, which is used to work with JSON formatted data.
    - It then defines a dictionary named `questions` to store the quiz questions and answers.
    - The code opens a file named "questions.txt" and loads its content into the `questions` dictionary.
2. **Welcome and User Input:**
    - It prints a message "Python Quiz Program".
    - It asks the user to enter their full name and stores it in the variable `name`.
3. **Quiz Loop:**
    - The code iterates through each question stored in the `questions` dictionary.
        - For each question:
            - It prints the question number and the question itself.
            - It prompts the user for their answer and stores it in the variable `ans`.

- It checks if the user's answer (converted to uppercase) matches the correct answer (also converted to uppercase) stored in the dictionary.
    - If the answer is correct, it increases the score by 1 and prints "Correct".
    - If the answer is wrong, it prints "Wrong".
- It increments the question number (`number`).

4. **Save Results:**
    - After the quiz, it creates a dictionary named `result` that stores the user's name and score.
    - It opens a file named "score.txt" in write mode (`'w'`) and uses the `json.dump` function to save the `result` dictionary in JSON format to the file.

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.

Answer:

```python
class BankAccount:
    def __init__(self, account_number, account_holder):
        self.account_number = account_number
        self.account_holder = account_holder
        self.balance = 0.0

    def deposit(self, amount):
        self.balance += amount
        return self.balance

    def withdraw(self, amount):
        if amount <= self.balance:
            self.balance -= amount
            return self.balance
        else:
            return "Insufficient funds"

    def get_balance(self):
        return self.balance

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_amount = self.balance * self.interest_rate
        self.balance += interest_amount
        print(f"Interest rate: {self.interest_rate}")
        print(f"Current balance: {self.balance}")
```

```python
def main():
  id = BankAccount("2790230", "souzzan alhalabi")
  print("Bank Account:")
  print(f"Initial balance: {id.get_balance()}")
  id.deposit(1000)
  print(f"Balance after deposit: {id.get_balance()}")
  id.withdraw(500)
  print(f"Balance after withdrawal: {id.get_balance()}")

  ids = SavingsAccount("2790230", "souzzan alhalabi", 0.03)
  print("\nSavings Account:")
  print(f"Initial balance: {ids.get_balance()}")
  ids.deposit(2000)
  print(f"Balance after deposit: {ids.get_balance()}")
  ids.apply_interest()

main()
```

```
Bank Account:
Initial balance: 0.0
Balance after deposit: 1000.0
Balance after withdrawal: 500.0

Savings Account:
Initial balance: 0.0
Balance after deposit: 2000.0
Interest rate: 0.03
Current balance: 2060.0
```

Explenation:

BankAccount Class:

- This class serves as a blueprint for creating regular bank accounts.
    - **`__init__(self, account_number, account_holder):`** This is the constructor method that gets called whenever you create a new `BankAccount` object. It takes two arguments: `account_number` and `account_holder` . It initializes three attributes for the object: `account_number`, `account_holder`, and `balance` (which is set to 0.0 by default).
    - **`deposit(self, amount):`** This method allows depositing money into the account. It takes an `amount` as input and adds it to the current balance. It then returns the updated balance.
    - **`withdraw(self, amount):`** This method allows withdrawing money from the account. It takes an `amount` as input. It checks if there are sufficient funds (`amount <= self.balance`). If yes, it subtracts the amount from the balance and returns the updated balance. Otherwise, it returns a message indicating "Insufficient funds".
    - **`get_balance(self):`** This method simply returns the current balance of the account.

SavingsAccount Class:

- This class inherits from the `BankAccount` class, meaning it gets all the functionalities of a bank account. It adds an additional feature specific to savings accounts: earning interest.

- **`__init__(self, account_number, account_holder, interest_rate):`** Similar to the base class constructor, this one takes the same arguments for account details and adds an extra argument `interest_rate`. It uses `super().__init__(account_number, account_holder)` to call the base class constructor and then initializes the `interest_rate` attribute.
- **`apply_interest(self):`** This method calculates the interest earned based on the current balance and the account's interest rate. It adds the interest amount to the balance and prints information about the applied interest and the updated balance.

main() Function:

- This function demonstrates how to create and use the `BankAccount` and `SavingsAccount` classes.
  - It creates a `BankAccount` object named `id` and showcases depositing, withdrawing, and checking balance functionalities.
  - Then, it creates a `SavingsAccount` object named `ids`, deposits money, and applies the interest, demonstrating its additional feature.