جامعة تشرين
كلية الهمك
قسم هندسة الإتصالات والإلكترونيات
السنة الخامسة
وظيفة 1 برمجة شبكات

First Network Programming Homework

إعداد الطالبة
زينب سمير قيراطه 2851

Question 1

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 }

problem solving:

main.py

```python
protocols = ["HTTP", "HTTPS", "FTP", "DNS"]
ports = [80, 443, 21, 53]
# Create dictionary by zipping the two lists together
network_dict = dict(zip(protocols, ports))
print(network_dict)
```

Ln: 5, Col: 20

Run | Share | Command Line Arguments

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


** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

Question 1_B:

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

problem solving:

main.py

```python
def calculate_factorial(n):
    if n == 0:
        return 1
    else:
        return n * calculate_factorial(n - 1)
# Input: Prompt the user to enter a non-negative integer
number = int(input("Enter a non-negative integer: "))
# Calculate the factorial of the input number
result = calculate_factorial(number)
# Output: Display the result
print(f"The factorial of {number} is {result}")
```

Ln: 11, Col: 48

Run Share Command Line Arguments

```
Enter a non-negative integer:
2
The factorial of 2 is 2


** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

Question 1_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.

problem solving:

main.py

```python
# List of subjects
subjects = ["Network", "Bio", "Programming", "Physics", "Music"]
# Iterate over each item in the list
for subject in subjects:
    # Check if the current item starts with the letter 'B'
    if subject.startswith("B"):
        # Print the item if it starts with 'B'
        print(subject)
```

Ln: 8, Col: 23

Run Share Command Line Arguments

```
Bio


** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

Question 1_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}

problem solving:

main.py

```python
# Create a dictionary using a dictionary comprehension
# The keys are integers from 0 to 10, and the values are the keys incremented by
dictionary = {i: i + 1 for i in range(11)}
# Print the created dictionary
print(dictionary)
```

Ln: 5, Col: 18

Run Share Command Line Arguments

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


** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

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.

problem solving:

main.py

```python
def bin_to_dec(input):
    try:
        return int(input, 2)
    except ValueError:
        return None
# Read binary number input from user
input = input("Enter a binary number: ")
# Check for valid binary input
if not all(char in '01' for char in input):
    print("Invalid input.")
else:
    # Convert to decimal and display the result
    dec_num = bin_to_dec(input)
    if dec_num is not None:
        print(f" {input} is {dec_num}")
    else:
        print("Invalid binary number.")
```

Ln: 5, Col: 20

Run | Share | Command Line Arguments

```
Enter a binary number:
010
 010 is 2


** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

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.

problem solving:

main.py questions.json

```python
import json

# Load questions from JSON file
with open("questions.json", "r") as file:
    questions = json.load(file)

# Get user name
user_name = input("Enter your name: ")
# Initialize score
score = 0
# Ask each question
for q in questions:
    answer = int(input(f"{q['question']} = "))
    if answer == q["answer"]:
        score += 1
# Print user score
print(f"{user_name}, your score is {score}/{len(questions)}")
# Load existing results or initialize an empty list
try:
    with open("results.json", "r") as file:
        results = json.load(file)
except FileNotFoundError:
    results = []
# Append new result
results.append({"name": user_name, "score": score})
# Save results back to JSON file
with open("results.json", "w") as file:
    json.dump(results, file, indent=4)
```

Ln: 10, Col: 10

Run Share Command Line Arguments

```
Enter your name:
Zainab
1*1 =
1
1*2 =
2
1*3 =
4
```

```
Enter your name:
Zainab
1*1 =
1
1*2 =
2
1*3 =
4
1*4 =
5
1*5 =
6
1*6 =
7
1*7 =
8
1*8 =
9
1*9 =
1
1*10 =
1
2*1 =
1
2*2 =
2
2*3 =
4
2*4 =
1
2*5 =
2
2*6 =
1
2*7 =
1
2*8 =
1
2*9 =
2
2*10 =
20
Zainab, your score is 3/20
```

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.

problem solving:

The solution is on the next page.

main.py

```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
        print(f"Deposited ${amount}. Current balance: ${self.balance}")
    def withdraw(self, amount):
        if amount <= self.balance:
            self.balance -= amount
            print(f"Withdrew ${amount}. Current balance: ${self.balance}")
        else:
            print("Insufficient balance")
    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):
        self.balance += self.balance * (self.interest_rate / 100)
        print(f"Applied interest. Current balance: ${self.balance}")
    def __str__(self):
        return f"Balance: ${self.balance}, Interest rate: {self.interest_rate}%"
# Create an instance of BankAccount
account = BankAccount("123456789", "Zainab Samer")
# Perform a deposit of $1000
account.deposit(1000)
# Perform a withdrawal of $500
account.withdraw(500)
# Print the final balance
print(f"Final balance: ${account.get_balance()}")
# Create an instance of SavingsAccount
savings_account = SavingsAccount("987654321", "Zainab Samer", 5)
# Perform a deposit of $1000
savings_account.deposit(1000)
# Apply interest
savings_account.apply_interest()
# Print the balance and interest rate
print(savings_account)
```

Ln: 41, Col: 23

Run | Share | Command Line Arguments

```
Deposited $1000. Current balance: $1000.0
Withdrew $500. Current balance: $500.0
Final balance: $500.0
Deposited $1000. Current balance: $1000.0
Applied interest. Current balance: $1050.0
Balance: $1050.0, Interest rate: 5%


** Process exited - Return Code: 0 **
Press Enter to exit terminal
```