CSE401: Artificial Intelligence

Utkarsh Gupta
A2305217557
7CSE 8Y

July 18th, 2020

1 Overview of Python

This contains the solution of `List`, `Tuple`, `Set`, `Dictionary`, `Functions`, `Classes`, and `File Processing` questions as given in the lab assignment.

1.1 List

`List` are mutable ordered and indexed collections of objects. The items of a list are arbitrary Python objects. Lists are formed by placing a comma-separated list of expressions in square brackets.

Q1. Write a Python program to sum all the items in a list.

[1]:
```
numbers = list(map(int, input("Input the numbers: ").split()))

# way 1:
totalSum = 0
for i in numbers:
    totalSum += i

print(f"Total sum is: {totalSum}")

# way 2:
print()
print("Alternatively, we use the 'sum' function in Python:")
print(f"sum(numbers): {sum(numbers)}")
```

```
Input the numbers: 6 8 2 10 4
Total sum is: 30

Alternatively, we use the 'sum' function in Python:
sum(numbers): 30
```

Q2. Write a Python program to get the largest number from a list.

[2]:
```
numbers = list(map(int, input("Input the numbers: ").split()))

# way 1:
numbers.sort()
print(f"The largest number is: {numbers[-1]}")

# way 2:
print()
print("Alternatively, we can use the 'max' function in Python:")
print(f"max(numbers): {max(numbers)}")
```

```
Input the numbers: 6 8 2 10 4
The largest number is: 10

Alternatively, we can use the 'max' function in Python:
max(numbers): 10
```

Q3. Write a Python program to get the smallest number from a list.

[3]:
```
numbers = list(map(int, input("Input the numbers: ").split()))

# way 1:
numbers.sort()
print(f"The smallest number is: {numbers[0]}")

# way 2:
print()
print("Alternatively, we can use the 'min' function in Python:")
print(f"min(numbers): {min(numbers)}")
```

```
Input the numbers: 6 8 2 10 4
The smallest number is: 2

Alternatively, we can use the 'min' function in Python:
min(numbers): 2
```

Q4. Write a Python program to multiply all the items in a list.

[4]:
```
numbers = list(map(int, input("Input the numbers: ").split()))

# way 1:
product = 1
for i in numbers:
    product *= i
print(f"The product of all items: {product}")

# way 2:
import math
print()
print("Alternatively, we can use the 'prod' function from the 'math' library:")
print(f"math.prod(numbers): {math.prod(numbers)}")
```

```
Input the numbers: 6 8 2 10 4
The product of all items: 3840

Alternatively, we can use the 'prod' function from the 'math' library:
math.prod(numbers): 3840
```

1.2 Tuple

`Tuple` is a collection of objects which ordered and immutable. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Q1. Write a Python program to create a tuple.

[5]:
```
someTuple = (21, 12)
print(type(someTuple))
print(someTuple)
```

```
<class 'tuple'>
(21, 12)
```

Q2. Write a Python program to create a tuple with different data types.

[6]:
```
someTuple = (21, 'debian', 12)
print(someTuple)
```

```
(21, 'debian', 12)
```

Q3. Write a Python program to create a tuple with numbers and print one item.

[7]:
```
someTuple = tuple(map(int, input("Enter the items: ").split()))
itemNumber = int(input("Enter the item number you want to print: "))

print(f"Item number {itemNumber} is '{someTuple[itemNumber-1]}'.")
```

```
Enter the items: 6 8 2 10 4
Enter the item number you want to print: 3
Item number 3 is '2'.
```

Q4. Write a Python program to add an item in a tuple.

[8]:
```
someTuple = tuple(map(int, input("Enter the items: ").split()))
addItem = input("Enter the item to add: ")

# way 1:
newTuple = someTuple + (addItem,)
print(f"The new tuple is: {newTuple}")

# way 2:
print()
print("Alternatively, we can convert the tuple into list and add items.")
someList = list(someTuple)
someList.append(addItem)
newTuple = tuple(someList)
print(f"The new tuple is: {newTuple}")
```

```
Enter the items: 21 12 2019
Enter the item to add: debian
The new tuple is: (21, 12, 2019, 'debian')

Alternatively, we can convert the tuple into list and add items.
The new tuple is: (21, 12, 2019, 'debian')
```

Q5. Write a Python program to convert a tuple to a string.

[9]:
```
someTuple = tuple(map(str, input("Enter the characters: ").split()))
convertedString = ''.join(someTuple)
print(f"Converted string: '{convertedString}'.")
```

```
Enter the characters: d e b i a n
Converted string: 'debian'.
```

1.3 Set

A `Set` is an unordered collection of items. Every set element is unique (no duplicates) and must be immutable (cannot be changed). However, a set itself is mutable. We can add or remove items from it.

Q1. Write a Python program to create a set.

[10]:
```
someSet = {21, 12}
print(type(someSet))
print(someSet)
```

```
<class 'set'>
{12, 21}
```

Q2. Write a Python program to add member(s) in a set.

[11]:
```
someSet = {21, 12}
member = input("Enter the member element: ")

someSet.add(member)
print(f"The updated set is {someSet}.")
```

```
Enter the member element: debian
The updated set is {'debian', 12, 21}.
```

Q3. Write a Python program to remove item(s) from set.

[12]:
```
someSet = {'debian', 21, 12}
print(f"The set is {someSet}.")
itemNumber = int(input("Enter the number you want to remove: "))

someSet.remove(itemNumber)
print()
print(f"The updated set is {someSet}.")
```

```
The set is {'debian', 12, 21}.
Enter the number you want to remove: 21

The updated set is {'debian', 12}.
```

Q4. Write a Python program to remove an item from a set if it is present in the set.

[13]:
```
someSet = {21, 12, 2019, 2020}
print(f"The set is {someSet}.")
itemNumber = int(input("Enter the number you want to remove: "))

if itemNumber in someSet:
    someSet.remove(itemNumber)
else:
    print(f"{itemNumber} is not present in the set!")

print()
print(f"The updated set is {someSet}.")
```

```
The set is {2019, 12, 21, 2020}.
Enter the number you want to remove: 2020

The updated set is {2019, 12, 21}.
```

1.4 Dictionary

A `Dictionary` is a collection which is unordered, changeable and indexed. In Python dictionaries are written with curly brackets, and they have keys and values.

Q1. Write a Python program to create a dictionary.

[14]:
```
someDict = {"Year": 2020, "Month": "July"}
print(type(someDict))
print(someDict)
```

```
<class 'dict'>
{'Year': 2020, 'Month': 'July'}
```

Q2. Write a Python script to add a key to a dictionary.

[15]:
```
someDict = {"Year": 2020, "Month": "July"}
print(f"The dictionary right now is: {someDict}.")
inputDict = input("Enter {key} and {value} separated by a space to add in the␣
 ↪dictionary: ")

someDict.update(dict(x.split() for x in inputDict.splitlines()))

print()
print(f"The updated dictionary is: {someDict}.")
```

```
The dictionary right now is: {'Year': 2020, 'Month': 'July'}.
Enter {key} and {value} separated by a space to add in the dictionary: Date 18

The updated dictionary is: {'Year': 2020, 'Month': 'July', 'Date': '18'}.
```

Q3. Write a Python script to concatenate following dictionaries to create a new one.

[16]:
```
someDict = {'Name': 'Utkarsh Gupta'}
print(f"The dictionary right now is: {someDict}.")
key, value = input("Enter key and value separated by a space: ").split()

newDict = dict()
newDict[key] = value
someDict.update(newDict)

print()
print(f"The concatenated dictionary is: {someDict}.")
```

```
The dictionary right now is: {'Name': 'Utkarsh Gupta'}.
Enter key and value separated by a space: Enrollment A2305217557

The concatenated dictionary is: {'Name': 'Utkarsh Gupta', 'Enrollment':
'A2305217557'}.
```

Q4. Write a Python script to check whether a given key already exists in a dictionary.

[17]:
```
someDict = {"Name": "Utkarsh Gupta", "Enrollment": "A2305217557"}
key = input("Enter the key to check: ")

if key in someDict.keys():
    print(f"The key '{key}' is present.")
else:
    print(f"The key '{key}' is NOT present.")
```

```
Enter the key to check: Name
The key 'Name' is present.
```

1.5 Functions

A `Function` is a block of code which only runs when it is called. You can pass data, known as parameters, into a function. A function can return data as a result.

Q1. Write a Python function that takes a list and returns a new list with unique elements of the first list.

[18]:
```
def uniqueList(someList):
    newList = []
    for i in someList:
        if i in newList:
            continue
        else:
            newList.append(i)

    print(f"The original list is: {someList}.")
    print(f"The list of unique elements is: {newList}.")

someList = list(map(int, input("Input the list: ").split(",")))
print()
uniqueList(someList)
```

```
Input the list: 1,2,3,4,5,4,3,2,1,2,3,4,5

The original list is: [1, 2, 3, 4, 5, 4, 3, 2, 1, 2, 3, 4, 5].
The list of unique elements is: [1, 2, 3, 4, 5].
```

1.6 Mini Projects

Project 1: Make a simple mathematical calculator which can perform addition, subtraction, multiplication and division.

[19]:
```
def add(x, y):
    return x + y

def subtract(x, y):
    return x - y

def multiply(x, y):
    return x * y

def divide(x, y):
    return x / y

print("Select operation:")
print("1. Add")
print("2. Subtract")
print("3. Multiply")
print("4. Divide")
print()

while True:
    choice = input("Enter choice (1/2/3/4): ")

    if choice in ('1', '2', '3', '4'):
        number1 = float(input("Enter first number: "))
        number2 = float(input("Enter second number: "))
        print()

        if choice == '1':
            print(f"{number1} + {number2} = {add(number1, number2)}")

        elif choice == '2':
            print(f"{number1} - {number2} = {subtract(number1, number2)}")

        elif choice == '3':
            print(f"{number1} * {number2} = {multiply(number1, number2)}")

        elif choice == '4':
            print(f"{number1} / {number2} = {divide(number1, number2)}")

        break
    else:
        print("Invalid input.")
```

```
Select operation:
1. Add
2. Subtract
3. Multiply
4. Divide

Enter choice (1/2/3/4): 3
Enter first number: 2
Enter second number: 3.5

2.0 * 3.5 = 7.0
```

Project 2: Make a rock-paper-scissors game where it is the player vs the computer. The computer's answer will be randomly generated, while the program will ask the user for their input. This project will better your understanding of while loops and if statements.

[20]:
```python
import random

print("GAME RULES:")
print("Rock  vs  Paper    :  Paper wins!")
print("Paper vs  Scissor  :  Scissor wins!")
print("Rock  vs  Scissor  :  Rock wins!")
print("-----------------------------------")

while(True):
    print("\nSelect:\n 1. Rock\n 2. Paper\n 3. Scissor\n 4. Stop\n")
    choice = int(input("Please enter your choice: "))

    options = {1: "Rock", 2: "Paper", 3: "Scissor", 4: "Stop"}

    if (choice == 4):
        print("\nGame exited, thank you for playing!")
        break

    if(choice not in options):
        print()
        print("Incorrect choice, exiting...")

    print(f"Your choice: {options[choice]}")

    system = random.randint(1,3)
    print(f"System's choice: {options[system]}")

    if (choice == system):
        print("\nIt's a tie!")
```

```
    elif((choice == 1 and system == 2) or (choice == 2 and system ==1 )):
        print("\nPaper wins!")

    elif((choice == 1 and system == 3) or (choice == 3 and system == 1)):
        print("\nRock wins!")

    else:
        print("\nScissor wins!")

    print("-------------------------------------")
```

```
GAME RULES:
Rock  vs  Paper    :  Paper wins!
Paper vs  Scissor  :  Scissor wins!
Rock  vs  Scissor  :  Rock wins!
-------------------------------------

Select:
 1. Rock
 2. Paper
 3. Scissor
 4. Stop

Please enter your choice: 3
Your choice: Scissor
System's choice: Rock

Rock wins!
-------------------------------------

Select:
 1. Rock
 2. Paper
 3. Scissor
 4. Stop

Please enter your choice: 2
Your choice: Paper
System's choice: Paper

It's a tie!
-------------------------------------

Select:
 1. Rock
 2. Paper
```

```
 3. Scissor
 4. Stop

Please enter your choice: 4

Game exited, thank you for playing!
```

1.7 Classes

A `Class` is a code template for creating objects. Objects have member variables and have behaviour associated with them. In python a class is created by the keyword class . An object is created using the constructor of the class. This object will then be called the instance of the class.

Q1. Write a Python class to find two elements from a list that have sum equal to zero.

[21]:
```
class SumZero:
    def main(self, someList):
        for i in someList:
            for j in someList:
                if (i + j == 0):
                    print(f"({i}, {j})")
                else:
                    continue

someList = list(map(int, input("Input the list: ").split(",")))
print()
print("The possible sets are: ")
SumZero().main(someList)
```

```
Input the list: -1,2,3,4,1,-3,5

The possible sets are:
(-1, 1)
(3, -3)
(1, -1)
(-3, 3)
```

1.8 File Processing

Q1. Write a Python program to count the frequency of words in the file.

[22]:
```
# Creating a file with some dummy content.
f = open("ai.txt", "w+")
f.write("This is test file with some dummy content\n")
f.write("Dummy content is the best for testing\n")
f.write("Testing is important\n")
f.close()

# Performing the frequency search.
dummyText = open("ai.txt", "r")
someDict = dict()

for line in dummyText:
    line = line.strip()
    line = line.lower()
    words = line.split(" ")

    for word in words:
        if word in someDict:
            someDict[word] = someDict[word] + 1
        else:
            someDict[word] = 1


print("The frequency of words are:\n")
for key in list(someDict.keys()):
    print(f"{key}: {someDict[key]}")
```

```
The frequency of words are:

this: 1
is: 3
test: 1
file: 1
with: 1
some: 1
dummy: 2
content: 2
the: 1
best: 1
for: 1
testing: 2
important: 1
```