UGE1197
Programming in Python Lab
AY: 2020-2021

Krithika Swaminathan
Date: 23/02/2021
S03018

Ex. No.: 9

1. Anagram Check

Aim

To check if two given words are anagrams or not.

Algorithm

Input: Any two words
Output: The conclusion of whether they are anagrams or not

Pre-condition: The words should contain only letters and should be in lower case.

Step 1: Start
Step 2: Define a function to check if two words are anagrams or not. The function accepts the two words as arguments.
Step 3: The function uses the sorted() function to sort the letters in each word lexicographically. If the sorted words are equal, then it returns that it is true that the two words are anagrams. If not, it returns that it is false.
Step 4: In the main function, two words are obtained as input from the user.
Step 5: The function is called with these parameters and the return value is printed.
Step 6: End

Program

```
# Python program to check if two given words are anagrams or not.

# Function definition to check if anagrams or not
def is_anagram(w1,w2):
```

Krithika Swaminathan
Date: 23/02/2021
S03018

Ex. No.: 9

```
  if sorted(w1)==sorted(w2):
    return True
  else:
    return False

# to get the two words as input from the user
w1=input("Enter word1: ")
w2=input("Enter word2: ")

print(is_anagram(w1,w2))
```

Output

Input:

```
listen silent
habit train
```

Output:

```
Enter word1: listen
Enter word2: silent
True
Enter word1: habit
Enter word2: train
False
```

Results / Inferences

Program for checking if two given words are anagrams or not is written and executed.

Krithika Swaminathan
Date: 23/02/2021
S03018

Ex. No.: 9

2. Chopping a list

Aim

To chop a list by removing the first and last elements from it.

Algorithm

Input: A list
Output: The chopped list

Step 1: Start
Step 2: Define a function to chop the list. The function accepts the list as the argument.
Step 3: The function removes the elements at the index values 0 and -1 of the list and returns None.
Step 4: In the main function, the list is obtained as input from the user.
Step 5: Call the function to chop the list.
Step 6: Print the list after the function call.
Step 7: End

Program

```
# Python program to chop a list by removing the first and last
elements.

# Function to chop the list
def chop(t):
  t.remove(t[0])
  t.remove(t[-1])
  return None

l=int(input("Enter size of list: "))
```

UGE1197
Programming in Python Lab
AY: 2020-2021

Krithika Swaminathan
Date: 23/02/2021
S03018

Ex. No.: 9

```
t=[int(input("Enter num: ")) for i in range(l)]
print("Chopping",t,"...")
chop(t) #Function call
print(t)
```

Output

Input:

[1,2,3,4]

Output:

```
Enter size of list: 4
Enter num: 1
Enter num: 2
Enter num: 3
Enter num: 4
Chopping [1, 2, 3, 4] ...
[2, 3]
```

Results / Inferences

Program for chopping a list is written and executed.

UGE1197
Programming in Python Lab
AY: 2020-2021

Krithika Swaminathan
Date: 23/02/2021
S03018

Ex. No.: 9

3. Mixed list

Aim

To extract the integer elements present in a given mixed list.

Algorithm

Input: A mixed list
Output: An integer-only list

Step 1: Start
Step 2: Read a list with elements of different data types.
Step 3: Check the datatype of each element in the list and if it is an integer, add the element to a new list using list comprehension.
Step 4: Print the list containing the integer elements.
Step 5: End

Program

```
# Python program to extract the integer elements present in a pre-defined mixed list.

L1 = [1, 'x', 4, 5.6, 'z', 9, 'a', 0, 4]
L = [L1[i] for i in range(len(L1)) if type(L1[i]) == int]
print("Integer list:", L)
```

Output

```
Input:

L1 = [1, 'x', 4, 5.6, 'z', 9, 'a', 0, 4]
```

UGE1197
Programming in Python Lab
AY: 2020-2021

Krithika Swaminathan
Date: 23/02/2021
S03018

Ex. No.: 9

Output:

```
Integer list: [1, 4, 9, 0, 4]
```

Results / Inferences

Program for extracting the integer elements from a mixed list is written and executed.

UGE1197
Programming in Python Lab
AY: 2020-2021

Krithika Swaminathan
Date: 23/02/2021
S03018

Ex. No.: 9

4. Extracting from a Nested List

Aim

To extract all elements from a nested list and store them in the list.

Algorithm

Input: A nested list
Output: A list having all the elements in the nested list

Step 1: Start
Step 2: Read a nested list.
Step 3: Using list comprehension, run through every element in the nested list using two loops and add each element to a new list.
Step 4: Copy the new list to the old list variable.
Step 5: Print the list.
Step 6: End

Program

```
# Python program to extract elements from a nested list to the list.
vals=[[1,2,3],[4,5,2],[3,2,6]]
l=[vals[j][i] for j in range(len(vals[0])) for i in range(len(vals))]
vals=l
print(vals)
```

Output

Input:

Krithika Swaminathan
Date: 23/02/2021
S03018

Ex. No.: 9

```
[[1,2,3],[4,5,2],[3,2,6]]
```

Output:

```
[1, 2, 3, 4, 5, 2, 3, 2, 6]
```

Results / Inferences

Program for extracting all elements from a nested list to the list is written and executed.

UGE1197
Programming in Python Lab
AY: 2020-2021

Krithika Swaminathan
Date: 23/02/2021
S03018

Ex. No.: 9

5. Matrix Addition and Subtraction

Aim

To perform addition and subtraction operations on two given matrices.

Algorithm

Input: Any two matrices
Output: The sum and the difference of the matrices

Pre-condition: The matrices should contain only numeric values.

Step 1: Start
Step 2: Define a function to calculate the sum of two matrices. The function accepts the two matrices as arguments.
Step 3: The function uses list comprehension to add the elements in the same position in both the matrices. It then returns the sum matrix.
Step 4: Define a function to calculate the difference of two matrices. The function accepts the two matrices as arguments.
Step 5: The function uses list comprehension to subtract the elements in the second matrix from the elements in the same position in the first matrix. It then returns the difference matrix.
Step 6: Read any number of rows and columns as input from the user. Then use list comprehension to read the matrix elements as input from the user for the two matrices.
Step 7: Call the functions to add and subtract the two matrices with these parameters and print the returned values.
Step 8: End

UGE1197
Programming in Python Lab
AY: 2020-2021

Krithika Swaminathan
Date: 23/02/2021
S03018

Ex. No.: 9

Program

```
# Python program to find the sum and difference of two matrices

def ADD(m1,m2):
  L=[[m1[i][j]+m2[i][j] for j in range(len(m1[0]))] for i in
range(len(m1))]
  return L

def SUB(m1,m2):
  L=[[m1[i][j]-m2[i][j] for j in range(len(m1[0]))] for i in
range(len(m1))]
  return L

r=int(input("Enter number of rows: "))
c=int(input("Enter number of columns: "))
print("Matrix 1")
a=[[int(input("Enter num: ")) for j in range(c)] for i in range(r)]
print("Matrix 2")
b=[[int(input("Enter num: ")) for j in range(c)] for i in range(r)]

# to print the sum and difference of the matrices
print("Sum:",ADD(a,b))
print("Difference:",SUB(a,b))
```

Output

```
Enter number of rows: 2
Enter number of columns: 2
Matrix 1
Enter num: 23
Enter num: 56
Enter num: 21
Enter num: 34
Matrix 2
```

UGE1197
Programming in Python Lab
AY: 2020-2021

Krithika Swaminathan
Date: 23/02/2021
S03018

Ex. No.: 9

```
Enter num: 12
Enter num: 32
Enter num: 41
Enter num: 76
Sum: [[35, 88], [62, 110]]
Difference: [[11, 24], [-20, -42]]
```

Results / Inferences

Program for adding and subtracting two matrices is written and executed.

UGE1197
Programming in Python Lab
AY: 2020-2021

Krithika Swaminathan
Date: 23/02/2021
S03018

Ex. No.: 9

6. Matrix Multiplication

Aim

To perform matrix multiplication on two given matrices.

Algorithm

Input: Any two matrices
Output: The product of the two matrices

Pre-condition: The matrices should contain only numeric values.

Step 1: Start
Step 2: Define a function to calculate the product of two matrices. The function accepts the two matrices as arguments.
Step 3: The function uses list comprehension to sum the products of the elements in the same position for each row in the first matrix mapped to each row in the zipped second matrix respectively. It then returns the product matrix.
Step 4: Read any number of rows and columns as input from the user. Then use list comprehension to read the matrix elements as input from the user for the two matrices.
Step 5: Call the function to multiply the matrices with these parameters and print the returned value.
Step 6: End

Program

```
# Python program to perform matrix multiplication.

def MUL(m1,m2):
  n=list(zip(*m2))
```

Krithika Swaminathan
Date: 23/02/2021
S03018

Ex. No.: 9

```
  res=[[sum(a*b for a,b in zip(m1row,m2col)) for m2col in n] for
m1row in m1]
  return res

r=int(input("Enter number of rows: "))
c=int(input("Enter number of columns: "))
print("Matrix 1")
x=[[int(input("Enter num: ")) for j in range(c)] for i in range(r)]
print(x)
print("Matrix 2")
y=[[int(input("Enter num: ")) for j in range(c)] for i in range(r)]
print(y)
print("Product:")
print(MUL(x,y))
```

Output

```
Enter number of rows: 2
Enter number of columns: 2
Matrix 1
Enter num: 1
Enter num: 2
Enter num: 3
Enter num: 4
[[1, 2], [3, 4]]
Matrix 2
Enter num: 1
Enter num: 2
Enter num: 3
Enter num: 4
[[1, 2], [3, 4]]
Product:
[[7, 10], [15, 22]]
```

UGE1197
Programming in Python Lab
AY: 2020-2021

Krithika Swaminathan
Date: 23/02/2021
S03018

Ex. No.: 9

Results / Inferences

Program for multiplying two matrices is written and executed.

Krithika Swaminathan
Date: 23/02/2021
S03018

Ex. No.: 9

7. Tuple manipulation

Aim

To store every number present at an odd index value of a tuple to a new tuple.

Algorithm

Input: A tuple
Output: The tuple containing the elements from the odd index positions

Step 1: Start
Step 2: Create an empty tuple.
Step 3: For every element in the tuple, check if the index value of the element is odd by checking if it is not divisible by 2. If yes, then add the element to the new tuple.
Step 4: Print the new tuple.
Step 5: End

Program

```
# Python program to store numbers present in the odd indices of a tuple to a new tuple.

T=(1,3,2,4,6,5)
t=()
for i in range(len(T)):
  if i%2==1:
    t+=(T[i],)
print(t)
```

UGE1197
Programming in Python Lab
AY: 2020-2021

Krithika Swaminathan
Date: 23/02/2021
S03018

Ex. No.: 9

Output

Input:

(1,3,2,4,6,5)

Output:

(3, 4, 5)

Results / Inferences

Program for storing numbers at odd indices of a tuple to a new tuple is written and executed.

UGE1197
Programming in Python Lab
AY: 2020-2021

Krithika Swaminathan
Date: 23/02/2021
S03018

Ex. No.: 9

8. Grades

Aim

To assign a grade to each student from a list of their marks.

Algorithm

Input: The mark-list
Output: The corresponding grades of each student

Step 1: Start
Step 2: Read the given mark-list.
Step 3: Store the zipped mark-list in a variable.
Step 4: Clear the mark-list.
Step 5: Access the marks by going to the second row in the zipped mark-list. For every mark, assign a grade based on the given conditions.

- if Marks >=90 then grade A
- if Marks >=80 & <90 then grade B
- if Marks >65 & < 80 then grade C
- if Marks > =40 & <=65 then grade D

Step 6: Add every student no., mark and grade as a tuple to the empty mark-list.
Step 7: Print the mark-list.
Step 8: End

Program

```
# Python program to assign a grade to each student based on their marks.

mark_list=[(120,55),(121,94),(122,73),(123,88),(124,62)]
```

Krithika Swaminathan
Date: 23/02/2021
S03018

Ex. No.: 9

```
m=list(zip(*mark_list))
mark_list.clear()
for i in range(len(m[1])):
  marks=m[1][i]
  if marks>=90:
    g="Grade A"
  elif marks>=80 and marks<90:
    g="Grade B"
  elif marks>=65 and marks<80:
    g="Grade C"
  elif marks>=40 and marks<60:
    g="Grade D"
  mark_list.append((m[0][i],marks,g))
print(mark_list)
```

Output

Input:

```
mark_list=[(120,55),(121,94),(122,73),(123,88),(124,62)]
```

Output:

```
[(120, 55, 'Grade D'), (121, 94, 'Grade A'), (122, 73,
'Grade C'), (123, 88, 'Grade B'), (124, 62, 'Grade B')]
```

Results / Inferences

Program for assigning grades is written and executed.