Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 7

Roll No : 418

Name : Bhavik A. Parmar

01) WAP to find sum of tuple elements.

Sample Input:

```
T = (10, 20, 30, 40)
```

Sample Output:

```
Sum of tuple elements = 100
```

In [3]:
```python
T = (10, 20, 30, 40, 50)
print("Sum OF Tuple Elements :", sum(T))

# T = (10, 20, 30, 40)
# total = 0
# for i in T:
#     total += i
# print("Sum of tuple elements =", total)
```

```
Sum OF Tuple Elements : 150
```

02) WAP to find Maximum and Minimum K elements in a given tuple.

Sample Input:

```
T = (7, 2, 9, 4, 1, 5)
K = 2
```

Sample Output:

```
Minimum 2 elements: (1, 2)
Maximum 2 elements: (9, 7)
```

In [9]:
```
# T = (7, 2, 9, 4, 1, 5)
# T1 = sorted(T)
# print(T1)

T = (7, 2, 9, 4, 1, 5)
K = int(input("Enter K :"))
sortT = sorted(T)

# max_k = tuple(sortT[-K:][::-1])
max = SortT[:K]
print(f"Maximum {K} Elements: {max}")

# min = SortT[-K:] # -> [7, 9]
min = SortT[-1:-K-1:-1]
print(f"Minimum {K} Elements: {min}")
```

```
Maximum 2 Elements: [1, 2]
Minimum 2 Elements: [9, 7]
```

03) WAP to find tuples which have all elements divisible by K from a list of tuples.

Sample Input:

```
tuple_list = [(2, 4, 6), (3, 6, 9), (4, 8, 12), (5, 10, 15)]
K = 2
```

Sample Output:

```
Tuples divisible by 2 : [(2, 4, 6), (4, 8, 12)]
```

In [10]:
```
tuple_list = [(2, 4, 6), (3, 6, 9), (4, 8, 12), (5, 10, 15)]
K = int(input("Enter K :"))
res = [t for t in tuple_list if all (num % K == 0 for num in t)]
print(f"Tuples Divisible By {K} : {res}")


# tuple_list = [(2, 4, 6), (3, 6, 9), (4, 8, 12), (5, 10, 15)]
# K = 2
# result = []
# for t in tuple_list:
#     if all(element % K == 0 for element in t):
#         result.append(t)
# print("Tuples divisible by", K, ":", result)
```

```
Tuples Divisible By 2 : [(2, 4, 6), (4, 8, 12)]
```

04) WAP to create a list of tuples from given list having number and its cube in each tuple.

Sample Input:

```
numbers = [1, 2, 3, 4, 5]
```

Sample Output:

```
List of tuples with number and its cube: [(1, 1), (2, 8), (3,
27), (4, 64), (5, 125)]
```

In [11]:
```
numbers = [1, 2, 3, 4, 5]
res = [(n, n**3) for n in numbers]
print("List OF Tuples with Number and its Cube :", res)


# numbers = [1, 2, 3, 4, 5]
# result = []
# for n in numbers:
#     result.append((n, n**3))
# print("List of tuples with number and its cube:", result)
```

```
List OF Tuples with Number and its Cube : [(1, 1), (2, 8), (3, 27), (4, 64), (5,
125)]
```

05) WAP to find tuples with all positive elements from the given list of tuples.

Sample Input:

```
tuple_list = [(1, 2, 3), (-1, 2, 3), (4, 5, 6), (0, 1, 2), (-3,
-4, -5)]
```

Sample Output:

```
Tuples with all positive elements: [(1, 2, 3), (4, 5, 6)]
```

In [12]:
```
tuple_list = [(1, 2, 3), (-1, 2, 3), (4, 5, 6), (0, 1, 2), (-3, -4, -5)]
res = [t for t in tuple_list if all (num > 0 for num in t)]
print("Tuples With All Positive Elements: ", res)


# tuple_list = [(1, 2, 3), (-1, 2, 3), (4, 5, 6), (0, 1, 2), (-3, -4, -5)]
# result = []
# for t in tuple_list:
#     if all(x > 0 for x in t):
#         result.append(t)
# print("Tuples with all positive elements:", result)
```

```
Tuples With All Positive Elements:  [(1, 2, 3), (4, 5, 6)]
```

06) WAP to remove tuples of length K.

Sample Input:

```
tuple_list = [(1, 2), (3, 4, 5), (6,), (7, 8, 9), (10, 11)]
K = 2
```

Sample Output:

```
List after removing tuples of length 2 : [(3, 4, 5), (6,), (7,
8, 9)]
```

In [13]:
```
tuple_list = [(1, 2), (3, 4, 5), (6,), (7, 8, 9), (10, 11)]
K = int(input("Enter K :"))
res = [t for t in tuple_list if len(t) != K]
print(f"List After Removing Tuples of Length {K} : {res}")


# tuple_list = [(1, 2), (3, 4, 5), (6,), (7, 8, 9), (10, 11)]
# K = 2
# result = []
# for t in tuple_list:
#     if len(t) != K:
#         result.append(t)
# print("List after removing tuples of length", K, ":", result)
```

```
List After Removing Tuples of Length 2 : [(3, 4, 5), (6,), (7, 8, 9)]
```

07) WAP to remove all occurrences of a given element from a tuple.

Sample Input:

```
T = (3, 5, 3, 7, 3, 9)
x = 3
```

Sample Output:

```
(5, 7, 9)
```

In [16]:
```
T = (3, 5, 3, 7, 3, 9)
X = int(input("Enter X :"))
res = [t for t in T if t != X]
print(f"Remove {X} All Occurrences : {res}")


# T = (3, 5, 3, 7, 3, 9)
# x = 3
# result = []
# for i in T:
#     if i != x:
#         result.append(i)
# result = tuple(result)
# print(result)
```

```
Remove 3 All Occurrences : [5, 7, 9]
```

08) WAP to remove duplicates from tuple.

Sample Input:

```
T = (3, 5, 3, 7, 3, 9)
```

Sample Output:

```
(3, 5, 7, 9)
```

In [17]:
```
T = (3, 5, 3, 7, 3, 9)
res = tuple(set(T))
res = sorted(res)
print("Remove Duplicates From Tuple : ", res)


# T = (3, 5, 3, 7, 3, 9)
# result = []
# for i in T:
#     if i not in result:
#         result.append(i)
# result = tuple(result)
# print(result)
```

```
Remove Duplicates From Tuple :  [3, 5, 7, 9]
```

09) WAP to multiply adjacent elements of a tuple and print that resultant tuple.

Sample Input:

```
t = (2, 3, 4, 5)
```

Sample Output:

```
Resultant tuple after multiplying adjacent elements: (6, 12, 20)
```

In [138…
```
T = (2, 3, 4, 5)
res = ()
for i in range(len(T)-1):
    res += (T[i]*T[i+1],)
print("Resultant tuple after multiplying adjacent elements: ", res)


# t = (2, 3, 4, 5)
# result = []
# for i in range(len(t) - 1):
#     result.append(t[i] * t[i + 1])
# result = tuple(result)
# print("Resultant tuple after multiplying adjacent elements:", result)
```

```
Resultant tuple after multiplying adjacent elements:  (6, 12, 20)
```

10) WAP to test if the given tuple is distinct or not.

Sample Input:

```
tuple1 = (1, 2, 3, 4)
tuple2 = (1, 2, 2, 3)
```

Sample Output:

```
(1, 2, 3, 4) is distinct? -> True
(1, 2, 2, 3) is distinct? -> False
```

In [18]:
```
tuple1 = (1, 2, 3, 4)
tuple2 = (1, 2, 2, 3)
res1 = len(tuple1) == len(set(tuple1))
res2 = len(tuple2) == len(set(tuple2))
print("Tuple1 is Distinct :", res1)
print("Tuple2 is Distinct :", res2)
```

```
Tuple1 is Distinct : True
Tuple2 is Distinct : False
```

11) WAP to rotate the elements of a tuple to the right by `k` positions.

Sample Input:

```
T = (10, 20, 30, 40, 50)
k = 2
```

Sample Output:

```
(40, 50, 10, 20, 30)
```

In [21]:
```
T = (10, 20, 30, 40, 50)
k = 2
# Adjust k if it is greater than length of tuple
k = k % len(T)

newT = T[-K:] + T[:K+1]
print("Original Tuple :", T)
print(f"Rotate Element By {K} Tuple : {newT}")
```

```
Original Tuple : (10, 20, 30, 40, 50)
Rotate Element By 2 Tuple : (40, 50, 10, 20, 30)
```

12) WAP to merge two tuples of equal length alternately.

Sample Input:

```
T1 = (1, 3, 5)
T2 = (2, 4, 6)
```

Sample Output:

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

In [22]:
```
T1 = (1, 3, 5)
T2 = (2, 4, 6)
T3 = T1 + T2
res = tuple(sorted(T3))
print("Merge Two Tuples :", res)

# T1 = (1, 3, 5)
# T2 = (2, 4, 6)
# result = tuple(x for pair in zip(T1, T2) for x in pair)
# print(result)
```

```
Merge Two Tuples : (1, 2, 3, 4, 5, 6)
```

13) WAP to create a tuple of squares of elements that are even and greater than 10.

Sample Input:

```
T = (4, 12, 7, 18, 10)
```

Sample Output:

```
(144, 324)
```

In [24]:
```
T = (4, 12, 7, 18, 10)
res = []
for i in T:
    if i > 10 and i % 2 == 0:
       res.append(i*i)
res = tuple(res)
print("Tuple OF Squares of Elements : ", res)

# T = (4, 12, 7, 18, 10)
# result = tuple(x**2 for x in T if x > 10 and x % 2 == 0)
# print(result)
```

```
Tuple OF Squares of Elements :  (144, 324)
```

14) WAP to create (index, value) pairs for prime numbers.

Sample Input:

```
T = (4, 7, 9, 11, 15)
```

Sample Output:

```
((1, 7), (3, 11))
```

In [25]:
```
# T = (4, 7, 9, 11, 15)
# res = tuple(
#     (i, T[i]) for i in range(len(T))
#     if T[i] > 1 and all(T[i] % j != 0 for j in range(2, int(T[i]**0.5) + 1))
# )
# print(res)
```

```python
T = (4, 7, 9, 11, 15)

result = []

for i in range(len(T)):
    num = T[i]
    if num > 1:
        isPrime = True
        for j in range(2, int(num**0.5) + 1):
            if num % j == 0:
                isPrime = False
                break
        if isPrime:
            result.append((i, num))

result = tuple(result)
print("(index, value) pairs for prime numbers :",  result)


T = (4, 7, 9, 11, 15)

# With Function
# def is_prime(n):
#     if n <= 1:
#         return False
#     for i in range(2, int(n**0.5) + 1):
#         if n % i == 0:
#             return False
#     return True
# result = tuple((i, T[i]) for i in range(len(T)) if is_prime(T[i]))
# print(result)
```

```
(index, value) pairs for prime numbers : ((1, 7), (3, 11))
```

15) WAP to split a tuple into even-index and odd-index tuples.

Sample Input:

```
T = ('a', 'b', 'c', 'd', 'e')
```

Sample Input:

```
even = ('a', 'c', 'e')
odd = ('b', 'd')
```

In [26]:
```python
T = ('a', 'b', 'c', 'd', 'e')
even = ()
odd = ()
for i in range(len(T)):
    if i % 2 == 0:
        even += tuple(T[i],)
    else:
        odd += tuple(T[i],)
print("Even-index Tuple : ", even)
print("ODd-index Tuple : ", odd)

# T = ('a', 'b', 'c', 'd', 'e')
# even = T[0::2]   # elements at even indexes
# odd = T[1::2]    # elements at odd indexes
# print("even =", even)
# print("odd =", odd)
```

```
Even-index Tuple :  ('a', 'c', 'e')
ODd-index Tuple :  ('b', 'd')
```

16) WAP to compute column-wise sum from a tuple of lists.

Sample Input:

```
T = ([1, 2, 3], [4, 5, 6], [7, 8, 9])
```

Sample Output:

```
12 15 18
```

In [28]:

```
T = ([1, 2, 3], [4, 5, 6], [7, 8, 9])
result = [sum(idx) for idx in zip(*T)]
print("Column-wise sums:", *result)

# T = ([1, 2, 3], [4, 5, 6], [7, 8, 9])
# result = []
# for col in zip(*T):
#     result.append(sum(col))
# for val in result:
#     print(val, end=" ")
```

```
Column-wise sums: 12 15 18
```

In [ ]: