1. Python – Join Tuples if similar initial element
2. Python – Extract digits from Tuple list
3. Python – All pair combinations of 2 tuples
4. Python – Remove Tuples of Length K
5. Sort a list of tuples by second Item
6. Python program to Order Tuples using external List
7. Python – Flatten tuple of List to tuple
8. Python – Convert Nested Tuple to Custom Key Dictionary
9. Python Program for Binary Search (Recursive and Iterative)
10. Python Program for Linear Search
11. Python Program for Insertion Sort

Solve

Sure! Here’s the solution for each problem:

### 1. \*\*Python – Join Tuples if similar initial element\*\*

Join tuples in a list that have the same initial element.

```python

from itertools import groupby

# Sample list of tuples

tuples\_list = [(1, 4), (1, 5), (3, 2), (2, 1), (3, 7)]

# Group by the first element

grouped = groupby(sorted(tuples\_list, key=lambda x: x[0]), key=lambda x: x[0])

# Join tuples

result = [(key, \*[i[1] for i in group]) for key, group in grouped]

print(result)

```

### 2. \*\*Python – Extract digits from Tuple list\*\*

Extract digits from each element in a list of tuples.

```python

# Sample list of tuples

tuple\_list = [(12, 34), (56, 78), (90, 12)]

# Extract digits

result = [int(digit) for t in tuple\_list for num in t for digit in str(num)]

print(result)

```

### 3. \*\*Python – All pair combinations of 2 tuples\*\*

Find all pair combinations of elements from two tuples.

```python

from itertools import product

# Sample tuples

tuple1 = (1, 2)

tuple2 = (3, 4)

# All pair combinations

result = list(product(tuple1, tuple2))

print(result)

```

### 4. \*\*Python – Remove Tuples of Length K\*\*

Remove tuples from a list that have a specified length `K`.

```python

# Sample list of tuples

tuples\_list = [(1, 2), (3, 4, 5), (6, 7), (8,)]

# Length K

K = 2

# Remove tuples of length K

result = [t for t in tuples\_list if len(t) != K]

print(result)

```

### 5. \*\*Sort a list of tuples by second Item\*\*

Sort a list of tuples by the second element in each tuple.

```python

# Sample list of tuples

tuples\_list = [(1, 3), (4, 2), (5, 1)]

# Sort by second item

result = sorted(tuples\_list, key=lambda x: x[1])

print(result)

```

### 6. \*\*Python program to Order Tuples using external List\*\*

Order tuples based on an external list.

```python

# Sample list of tuples

tuples\_list = [(3, 'c'), (1, 'a'), (2, 'b')]

# External order list

order = [2, 3, 1]

# Order tuples

result = sorted(tuples\_list, key=lambda x: order.index(x[0]))

print(result)

```

### 7. \*\*Python – Flatten tuple of List to tuple\*\*

Flatten a tuple of lists into a single tuple.

```python

# Sample tuple of lists

tuple\_of\_lists = ([1, 2], [3, 4], [5, 6])

# Flatten to tuple

result = tuple(sum(tuple\_of\_lists, []))

print(result)

```

### 8. \*\*Python – Convert Nested Tuple to Custom Key Dictionary\*\*

Convert a nested tuple into a dictionary with custom keys.

```python

# Sample nested tuple

nested\_tuple = ((1, 'a'), (2, 'b'), (3, 'c'))

# Convert to dictionary with custom keys

result = {f'key\_{k}': v for k, v in nested\_tuple}

print(result)

```

### 9. \*\*Python Program for Binary Search (Recursive and Iterative)\*\*

Implement binary search in both recursive and iterative ways.

```python

# Iterative Binary Search

def binary\_search\_iterative(arr, x):

low, high = 0, len(arr) - 1

while low <= high:

mid = (low + high) // 2

if arr[mid] == x:

return mid

elif arr[mid] < x:

low = mid + 1

else:

high = mid - 1

return -1

# Recursive Binary Search

def binary\_search\_recursive(arr, low, high, x):

if high >= low:

mid = (low + high) // 2

if arr[mid] == x:

return mid

elif arr[mid] > x:

return binary\_search\_recursive(arr, low, mid - 1, x)

else:

return binary\_search\_recursive(arr, mid + 1, high, x)

else:

return -1

# Sample array

arr = [2, 3, 4, 10, 40]

x = 10

# Using iterative method

print(binary\_search\_iterative(arr, x))

# Using recursive method

print(binary\_search\_recursive(arr, 0, len(arr)-1, x))

```

### 10. \*\*Python Program for Linear Search\*\*

Implement a linear search algorithm.

```python

# Linear Search

def linear\_search(arr, x):

for i in range(len(arr)):

if arr[i] == x:

return i

return -1

# Sample array

arr = [2, 3, 4, 10, 40]

x = 10

# Perform linear search

print(linear\_search(arr, x))

```

### 11. \*\*Python Program for Insertion Sort\*\*

Implement an insertion sort algorithm.

```python

# Insertion Sort

def insertion\_sort(arr):

for i in range(1, len(arr)):

key = arr[i]

j = i - 1

while j >= 0 and key < arr[j]:

arr[j + 1] = arr[j]

j -= 1

arr[j + 1] = key

return arr

# Sample array

arr = [12, 11, 13, 5, 6]

# Perform insertion sort

print(insertion\_sort(arr))

```

These snippets should solve each of the requested problems. Let me know if you need further explanation or help!