Tuples

1. Create an empty tuple:

empty\_tuple = ()

1. Create a tuple with a single element:

single\_element\_tuple = (1,) # The comma is mandatory

1. Access elements in a tuple:

my\_tuple = (1, 2, 3)

first\_element = my\_tuple[0] # Accessing the first element

1. Check the length of a tuple:

length = len(my\_tuple)

1. Check if an element exists in a tuple:

|  |
| --- |
| my\_tuple = (1, 2, 3)  if 2 in my\_tuple:  print("it exixts")  output:  it exists |

1. Concatenate two tuples:

|  |
| --- |
| t1 = (1, 2)  t2 = (3, 4)  concatenated\_tuple = t1 + t2  print(concatenated\_tuple)  output:  (1,2,3,4) |

1. Convert a list into a tuple:

|  |
| --- |
| my\_list = [1, 2, 3]  my\_tuple = tuple(my\_list) |

1. Access a slice of a tuple (e.g., the first three elements):

slice\_of\_tuple = my\_tuple[:3]

1. Unpack values from a tuple into variables:

a, b, c = (1, 2, 3) #here a=1, b=2 and c=3

1. Reverse a tuple:

reversed\_tuple = my\_tuple[::-1]

1. Iterate over a tuple:

for element in my\_tuple:

print(element)

1. Convert a tuple into a string:

tuple\_as\_string = ''.join(map(str, my\_tuple))

1. Find the index of an element in a tuple:

index\_of\_element = my\_tuple.index(2)

1. Count occurrences of a specific element in a tuple:

count\_of\_element = my\_tuple.count(2)

1. Use tuples as dictionary keys:

|  |
| --- |
| my\_dict = {('key1', 'key2'): 'value'}  value = my\_dict[('key1', 'key2')] |

1. Sort a list of tuples based on the second element:

|  |
| --- |
| list\_of\_tuples = [(1, 3), (2, 1), (4, 2)]  sorted\_list = sorted(list\_of\_tuples, key=lambda x: x[1]) # This is a simple, anonymous (lambda) that takes a tuple x and returns its second element (x[1]).  print(sorted\_list)  output:  [(2, 1), (4, 2), (1, 3)] |

1. Merge multiple tuples into one:

|  |
| --- |
| t1 = (1, 2)  t2 = (3, 4)  t3 = (5, 6)  merged\_tuple = t1 + t2 + t3 |

1. Nest tuples within another tuple and access nested elements:

|  |
| --- |
| nested\_tuple = (1, (2, 3), 4)  nested\_element = nested\_tuple[1][1] # Accessing the second element of the nested tuple |

1. Swap values in two variables using a tuple:

|  |
| --- |
| a, b = 1, 2  a, b = b, a # Swapping |

1. Check if a tuple contains other tuples as elements:

|  |
| --- |
| my\_tuple = (1, (2, 3), "hello", (4, 5))  # Check if the tuple contains other tuples  contains\_tuple = any(isinstance(item, tuple) for item in my\_tuple)  print(contains\_tuple)  Output:  True |

1. Find the length of the longest tuple in a list of tuples:

|  |
| --- |
| list\_of\_tuples = [(1, 2), (1, 2, 3), (1,)]  max\_length = max(len(t) for t in list\_of\_tuples) |

1. Perform element-wise addition of two tuples of the same length:

|  |
| --- |
| t1 = (1, 2, 3)  t2 = (4, 5, 6)  added\_tuple = tuple(a + b for a, b in zip(t1, t2)) |

1. Handle a tuple of unknown size when unpacking into variables:

a, b, \*rest = (1, 2, 3, 4) # a = 1, b = 2, rest = [3, 4]

1. Implementing a tuple as an immutable list and handle operations like appending: Since tuples are immutable, we can't append directly. We can make it by creating a new tuple:

|  |
| --- |
| original\_tuple = (1, 2)  new\_tuple = original\_tuple + (3,) |

1. Use a generator to create a tuple:

|  |
| --- |
| generator = (i for i in range(3)) # This creates a generator  generated\_tuple = tuple(generator) |