ADA University

CSCI 6221: Advanced Software Paradigms

Investigation of Memory Allocation in Python

1. **Experimentation with Python code**

Experiment with below Python code resulted in different size for the different structures containing same 3 elements. Basically, size tuple object containing the elements is 48, while size of list is 80.

**Code**

tpl = (1, 2, 3)

print('Size of the tuple:', tpl.\_\_sizeof\_\_())

lst = [1, 2, 3]

print('Size of the list:', lst.\_\_sizeof\_\_())

**Output**

Size of the tuple: 48

Size of the list: 80

1. **Analysis**

The difference in memory usage between tuples and lists arises primarily from how the Python interpreter handles mutable versus immutable objects.

Tuples in Python are immutable, meaning their size and contents cannot change once defined. Due to this immutability, the Python interpreter allocates a smaller, fixed amount of memory for tuples. Since tuples are not expected to grow or change during runtime, this allocated memory remains constant, leading to more efficient memory usage.

In contrast, lists in Python are dynamic and mutable, allowing their size and contents to change at runtime. This dynamic nature creates extra overhead, therefore, the interpreter needs to allocate memory with additional space to accommodate potential growth. Lists require internal pointers and structures to support these modifications, leading to larger memory allocation than tuples.

1 of 2

1. **Memory Allocation:**

For a tuple with 3 elements (1, 2, 3), the size is 48 bytes. Interpreter allocates 48 bytes long memory space for the tuple for the reference to tuple object itself and to the structure that holds reference to its elements. Each element of the tuple is an integer object which has a small memory footprint in comparison to internal data structure that holds these elements.

For a list with 3 elements [1, 2, 3], the size is 80 bytes. This extra memory includes the list object, mutable internal structure and references to the 3 integer objects.

The exact memory size can vary depending on the Python interpreter implementation, but the pattern of tuples requiring less memory than lists remains consistent due to their fixed nature. The size for these objects also depends on factors such as platform architecture (32-bit vs. 64-bit) and the specific implementation of memory allocation in the interpreter.

2 of 2