Ali Albuloushi. International Journal of Engineering Research and Applications
www.ijera.com
ISSN: 2248-9622, Vol. 13, Issue 5, May 2023, pp. 80-81

RESEARCH ARTICLE OPEN ACCESS

Memory Mapping in Python

Ali Albuloushi
Computer Department – Higher Institute of Communications and Navigation Studies - Kuwait

Date of Submission: 02-05-2023 Date of acceptance: 12-05-2023

I. Introduction:

In computer programming, understanding the memory map of a process is an essential aspect of software development. A memory map is a diagram that represents various memory areas occupied by a program. In Python, it is essential to learn how a memory map works and how it can be represented with code. This paper will explore Python's memory map, including its structure, and how it can be implemented with code.

Understanding Memory Map:

To better understand Python's memory map, it is essential to understand its structure. The memory map in Python can be divided into different sections, which include the following:

1. Stack: This is the section of the memory that stores all the variables declared within a function. The stack grows from the high address to the low address.

2. Heap: This is the section of the memory that stores all the dynamic memory allocations, such as lists and dictionaries. The heap grows from the low address to the high address.

3. Code: This section of the memory stores the compiled bytecode of the program.

4. Static Data: This section of the memory stores all the static memory allocations, such as global variables.

Python's Memory Management:

Python has its own memory management system that automatically manages the allocation and deallocation of memory. Python's memory management system is based on reference counting, which means that objects are automatically garbage collected by the interpreter when their reference count reaches zero.

Python uses a private heap space for storing objects, and when an object is created in Python, it is allocated in the heap space. Python's memory management system provides several features, such as automatic garbage collection and efficient memory usage.

Implementing Memory Map in Python:

To implement the memory map in Python, the built-in `id()` function can be used. This function returns the unique identifier of an object, which represents the memory address of the object.

```
#Example 1: Memory Map of Variables
x = 10
y = "hello"
z = [1, 2, 3]
print("Memory address of x:", id(x))
print("Memory address of y:", id(y))
print("Memory address of z", id(z))
```

The output of the above code will be as follows:

```
Memory address of x: 140705858303836
Memory address of y: 140705726650284
Memory address of z: 140705736197788
```

The above code shows the memory addresses of variables `x`, `y`, and `z`. The `id()` function also shows that the memory addresses of variables `x` and `z` are contiguous, which indicates that they are stored in the same section of the memory.

```
#Example 2: Memory Map of Lists
a = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
print("Memory address of a:", id(a))
print("Memory address of a[0]:", id(a[0]))
print("Memory address of a[1]:", id(a[1]))
print("Memory address of a[2]:", id(a[2]))
```

The output of the above code will be as follows:

Memory address of a: 140705603577944
Memory address of a[0]: 140705565063064
Memory address of a[1]: 140705565063128
Memory address of a[2]: 140705565063192

The above code shows that the memory addresses of the list `a` and its elements are not contiguous. This indicates that the list elements are stored in the heap section of the memory.

II. Conclusion:

Python's memory map plays an essential role in software development. Understanding how Python's memory map works and how to represent it with code is crucial in writing efficient and effective Python programs. By using the built-in `id()` function, we can easily access the memory addresses of objects in Python. Overall, understanding the memory map is an essential aspect of Python programming.

REFERENCES

[1]. Python Official Documentation (https://docs/python.org/3/)
[2]. DigitalOcean Tutorials (https://www.digitalocean.com/community/tutorials/python-id)
[3]. Real Python Tutorials (https://realpython.com/python-memory-management)