Introduction to mutable and immutable in Python

Author: Srishti Sawla

LinkedIn URL: www.linkedin.com/in/srishtisawla

In Python, everything is an object. An object has its own internal state. Some objects allow you to change their internal state and others don't.

An object whose internal state can be changed is called a mutable object, while an object whose internal state cannot be changed is called an immutable object.

The following are examples of immutable objects:

- Numbers (int, float, bool,)
- Strings
- Tuples
- Frozen sets

And the following are examples of mutable objects:

- Lists
- Sets
- Dictionaries

User-defined classes can be mutable or immutable, depending on whether their internal state can be changed or not.

Python immutable example

When you declare a variable and assign it's an integer, Python creates a new integer object and sets the variable to reference that object:

```
counter = 100

print(id(counter))
```

Output:

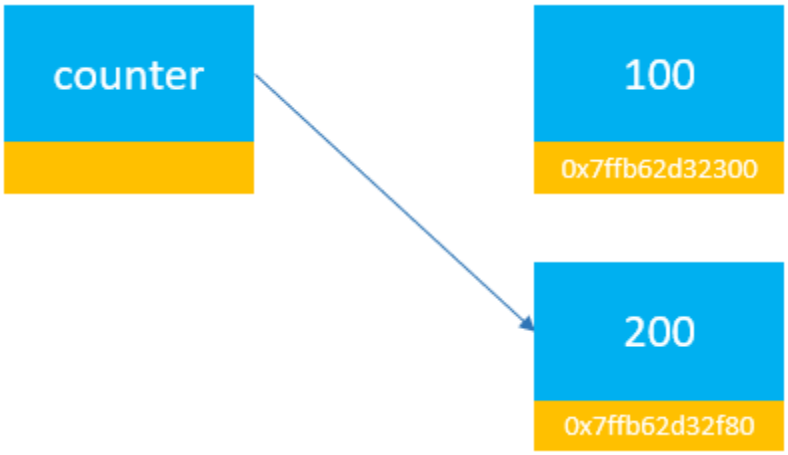
```
140717671523072
```

In the memory, you have a variable called counter that references an integer object located at the 0x7ffb62d32300 address:

If you change the value of counter to 200

Counter = 200

Python creates a new integer object with the value 200 and reassigns the counter variable so that it references the new object like this:



Python mutable example

The following defines a list of numbers and displays the memory address of the list:

ratings = [1, 2, 3]
print(hex(id(ratings)))
Output: 0x1840f97a340

Behind the scenes, Python creates a new list object and sets the ranks variable to reference the list:

When you add a number to the list like this:

ratings.append(4)

Python directly changes the value of the list object:

And Python doesn't create a new object like the previous immutable example.

Python mutable and immutable example

It's important to understand that immutable objects are not something frozen or constant. Let's look at an example.

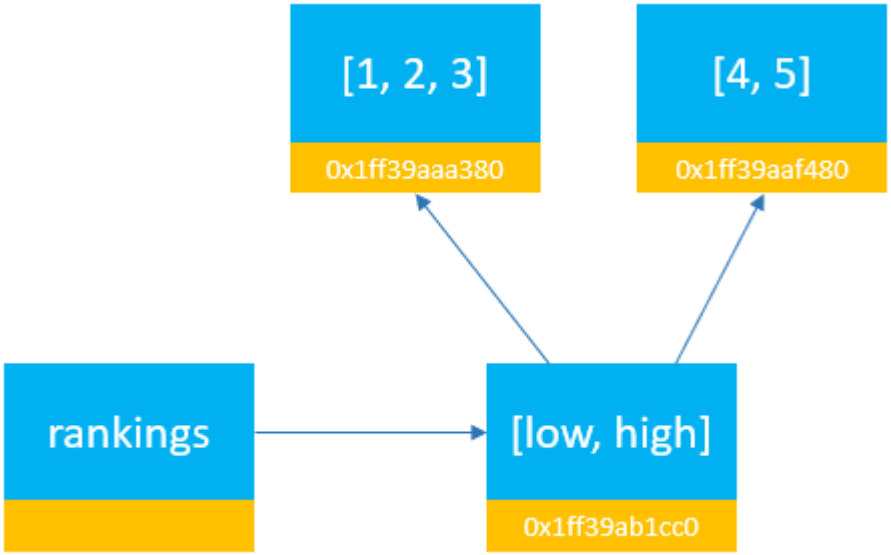
The following defines a tuple whose elements are the two lists:

low = [1, 2, 3]

high = [4, 5]

rankings = (low, high)

Since the rankings is a tuple, it's immutable. So you cannot add a new element to it or remove an element from it.



However, the rankings tuple contains two lists that are mutable objects. Therefore, you can add a new element to the high list without any issue:

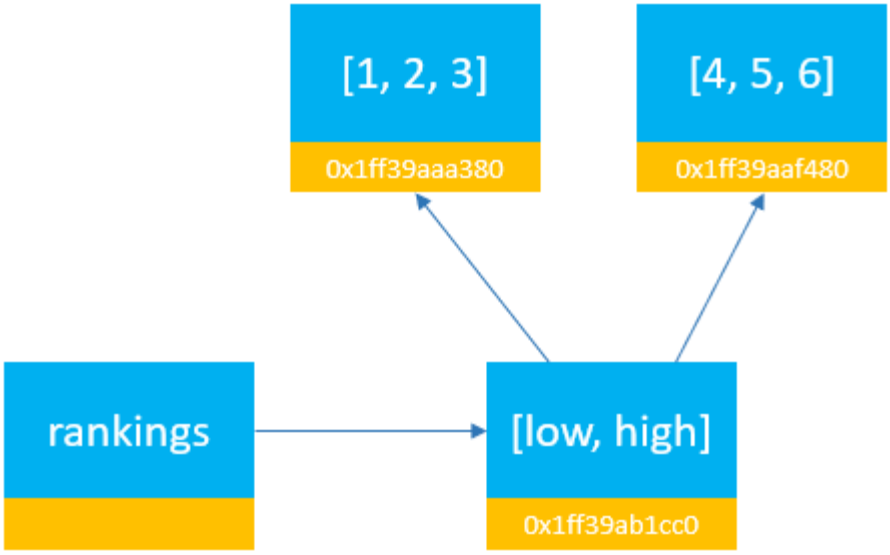
high.append(6)

print(rankings)

Code language: Python (python)

And the rankings tuple changes to the following:

([1, 2, 3], [4, 5, 6])



Python Memory Management

You can begin by thinking of a computer's memory as an empty book intended for short stories. There's nothing written on the pages yet. Eventually, different authors will come along. Each author wants some space to write their story in.

Since they aren't allowed to write over each other, they must be careful about which pages they write in. Before they begin writing, they consult the manager of the book. The manager then decides where in the book they're allowed to write.

Since this book is around for a long time, many of the stories in it are no longer relevant. When no one reads or references the stories, they are removed to make room for new stories.

In essence, computer memory is like that empty book. In fact, it's common to call fixed-length contiguous blocks of memory pages, so this analogy holds pretty well.

The authors are like different applications or processes that need to store data in memory. The manager, who decides where the authors can write in the book, plays the role of a memory manager of sorts. The person who removed the old stories to make room for new ones is a garbage collector.

Private heap and Stack

Memory management in Python involves a private heap containing all Python objects and data structures. The management of this private heap is ensured internally by the *Python memory manager*.

Whenever a new function or class is declared it is very common to have some variable declaration inside them, these declarations are associated with the function itself and do not change in the runtime. They occupy a fixed memory size. Hence these are stored in the stack area.

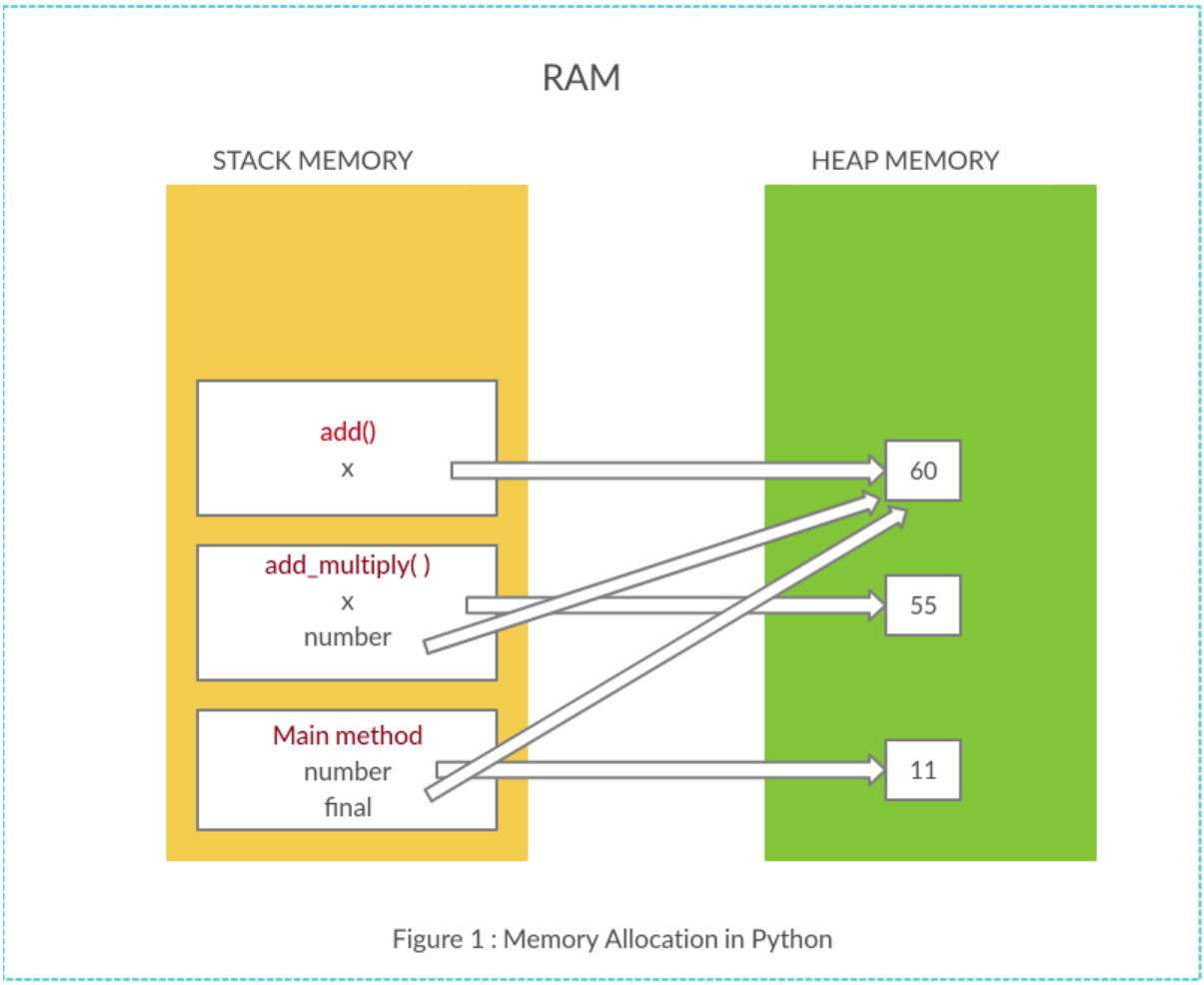


Figure 1 : Memory Allocation in Python

Coding problems solved:

1. Check whether a string is palindrome or not e.g., radar, level etc
2. Given a list, write a Python program to swap first and last element of the list.
3. Write a python program to find mean of two numbers
4. Write a python program to print maximum of two numbers