Contribution ID: **1** Type: **not specified**

Introducing Immortal Objects: building block towards a multi-core Python runtime.

In this presentation, I’ll cover memory management in Python starting from the fundamentals. I’ll explain the rationale behind the need for PEP 683: “–Immortal Objects, Using a Fixed Refcount” , discussing about this change may be unlocking exciting avenues for true parallelism in python.

Objective of talk is to discuss on how and why we need pure immutable objects and fixing the old time issue with them can make Python more powerful‼ in terms of code pure parallel execution.
Certain objects such as ‘None’, ‘True’, and ‘False’ act as global singletons, shared across the interpreter instead of creating fresh copies each time. However there is a catch even though they are considered immutable these objects were not out of editing their reference count for any new reference or de reference.

The takeaway of this presentation would be :
a) Refresher on object(mutable vs immutable and their significance)
b) Refresher on Life cycle of a python object.
c) Refresher on python memory management with garbage collection
d) How overcoming with pseudo immutable object will be unlocking exciting avenues for true parallelism in Python.

The target audience would be intermediate and advanced Pythonista’s. However I will present the information in simple and subtle manner for better understanding of python newbies .

Timeline of a talk :
Introduction(1 min)
What is a python object(3 min)
Life cycle of a python object(3 min)
Python object Runtime state(3 min)
Explain Garbage collection based on reference count(4 min)
Motivation for PEP-0683(3 min)
implementation summary for PEP-0683(4 min)

for you reference: This is the article I wrote on the same topic in my weekly newsletter:
https://technikal.substack.com/p/how-instagram-introduction-immortal

Author: MEHRA, Aditya

Presenter: MEHRA, Aditya