що тип операції можна визначити за записаним акустичним сигналом з точністю до 90% істинних прогнозів (ідентифікацій).

Докладно отримані результати доповідались на конференції [7] та були опубліковані в статті [8].

Висновки та перспективи подальших досліджень. Методи звукової фіксації та аналізу отриманих записів можуть бути використані під час розслідування злочинів та судово-медичної експертизи, для визначення позиції стрільця, категорії та моделі використаної зброї, а також для судової ідентифікації вогнепальної зброї.

Подальше застосування наведеного методу та дослідження ширшого спектру зброї дасть можливість виявляти конструктивні особливості механізму (справність, рівень зносу) зброї лише за характеристиками звуку, що ним видаються в процесі використання.

Список використаних джерел

1. *Татарнікова Т. О. Експертні дослідження матеріалів та засобів цифрового аудіозапису, Академія внутрішніх справ України, 2009.*
2. *Raponi S., Oligeri G., Ali I. M.. Sound of guns: digital forensics of gun audio samples meets artificial intelligence, Springer Nature, 2022.*
3. *Maciąg P., Chałko L. Use of sound spectral signals analysis to assess the technical condition of mechanical devices, MATEC Web of Conferences 290, 01006, 2019.*
4. *Paredes D. M., Apolinario J. A. Shooter localization using microphone arrays on elevated platforms. 2014 IEEE Central America and Panama Convention (CONCAPAN XXXIV). IEEE, 11.2014.*
5. *Hardle W., Simar L. Applied Multivariate Statistical Analysis. Springer, 2007*
6. *Chen T., Guestrin C. XGBoost: A Scalable Tree Boosting System, 2016.*
7. *Varer B., Giverts P. The comparison of feature engineering methods used for acoustic identification of firearms. IEEE UkrMiCo, 2021.*
8. *Giverts P., Sofer S., Solewicz Y., Varer B. Firearms identification by the acoustic signals of their mechanisms. Forensic Science International. Vol. 306, 2020.*

УДК 004.01

Грабік К.І., здобувач кафедри
інформаційних технологій

ВИДІЛЕННЯ ОПЕРАТИВНОЇ ПАМ’ЯТІ У МОВІ ПРОГРАМУВАННЯ PYTHON

Донецький національний університет імені Василя Стуса, м. Вінниця

Python – це мова програмування, яка максимально відгороджує програміста від роботи з пам'яттю, це є певним плюсом, оскільки дозволяє зосередитись на вирішенні завдання, але мені як людині, яка доволі багато

писала на мові C++ було цікаво, як працює виділення пам'яті (також цей процес називається алокація пам'яті).

Почати потрібно з того, що в операційній системі є абстракція, яка називається "процес", яка займається виділенням пам'яті з метою спрощення алокацію для сторонніх програм, в тому числі і для інтерпретатора Python. Програма при старті створює процес, просить в ОС пам'ять, заповнює її даними, після того, як програма завершилися – повертає пам'ять в операційну систему. Цей підхід має декілька мінусів, суттєвим з яких є час виконання таких запитів, бо ОС має провести певний аналіз, чи може вона надати програмі пам'ять, чи все ж таки має нам відмовити і аварійно зупинити процес виконання коду.

Треба зазначити, що під Python в даній роботі мається на увазі шестидесяти чотирьох бітна реалізацію інтерпретатора під назвою CPython (в інших реалізація інтерпретатора правила виділення пам'яті можуть відрізнятися).

Через те, що виділення пам'яті у ОС невиликими обсягами не є дуже ефективним, розробники Python пішли на хитрість, інтерпретатор при старті виконання програми робить запит на виділення фрагмента пам'яті розміром 256 кілобайт, його зазвичай називають ареною (рис. 1). В середині арена розбивається на так звані пули, розмір яких 4 кілобайта.

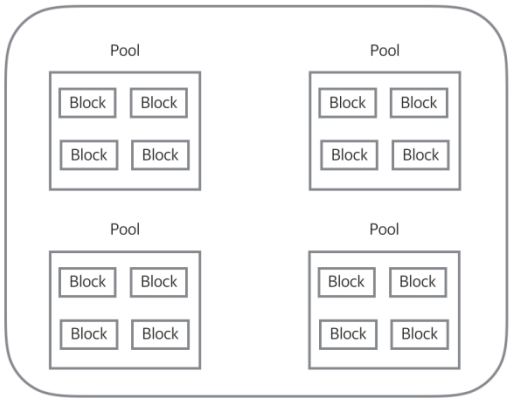

Рис.1

В середині пулів вже знаходяться блоки певного розміру (в кожному пулі можуть знаходитися блоки лише одного розміру, при чому їх розмір завжди буде кратний 2). Таких арен Python може запросити скільки завгодно, при чому пули в арені завжди мають однаковий розмір, проте розділяються на блоки різного розміру. Коли інтерпретатору потрібно записати якісь нові дані, він шукає арену, яка на даний момент найбільш заповнена ну і в якій звичайно є пул з блоком потрібного розміру. Взагалі з точки зору інтерпретатора всі арени – це зв’язні списки, при цьому вони поділяються на три види: full, used, та empty (повність заповнена арена, частково заповнена і порожня відповідно).

Для ефективної роботи потрібно не тільки виділяти пам'ять, а і очищати її від того, що ми не використовуємо, для цього в Python є 2 засоби: лічильник посилань і збірник сміття.

Перш ніж ми розбиратись, як працює збірник сміття, слід поглянути на структуру об'єкта в пам’яті (рис. 2).

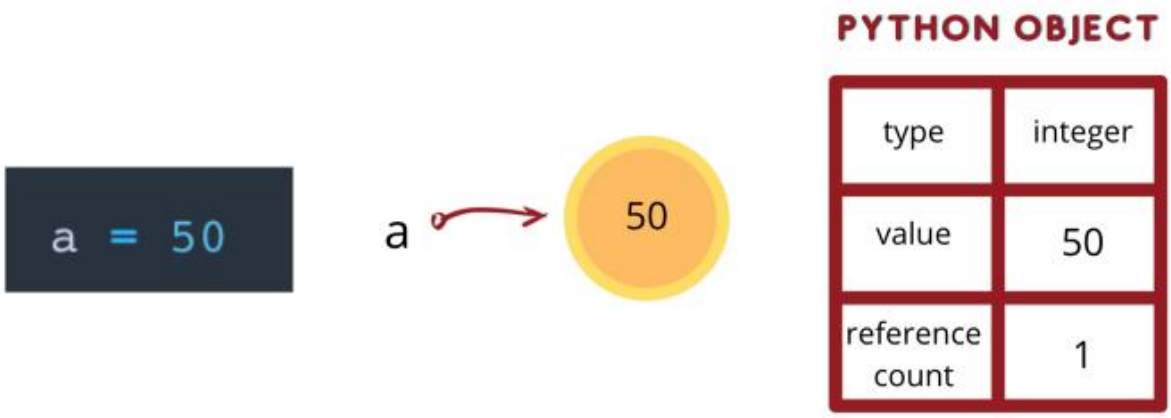

Рис. 2

Об’єкт складається з 3 частин: типу даних, значення і лічильника посилань, от як раз остання частина і є нашим першим механізмом з очистки даних. Працює він доволі просто, коли зберігається посилання на об'єкт в змінній, то цей лічильник збільшується на 1, а коли ми видаляємо це посилання з змінної, то зменшується, коли він досягає 0, то цей об'єкт повністю видаляється. Проблема цього механізму, коли об'єкт посилається сам на себе, або коли він посилається на інший об'єкт, котрий в свою чергу посилається на нього, тобто коли утворюється замкнуте коло посилань, як показано на схемах (рис. 3).

Звичайно, цей лічильник не зможе дійти до 0 ніколи і в такі моменти на допомогу приходить інший механізм видалення об'єктів з пам'яті, збірник сміття. Він в свою чергу працює зовсім по іншому, він дивиться, чи є об'єкт, що зберігає в собі посилання на інші об'єкти потрібним в майбутньому. якщо ні, то він буде видалений.

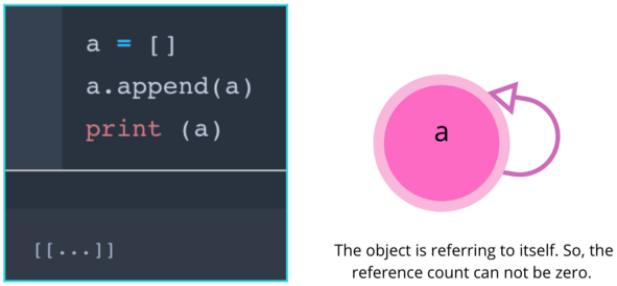

Рис. 3

Слід зазначити, що збірник сміття працює на постійно, а тільки коли потрібно і механізм виявлення потрібних даних в ньому доволі складний.

Отже, як можна побачити, розробники доволі добре продумали виділення їхньою мовою програмування оперативної пам'яті, проте в цього підходу є свій мінус, він потребує сам по собі набагато більше пам'яті, ніж класичний підхід з поступовою алокацією.

Список використаних джерел

1. *[Електронний ресурс]* https://docs.python.org/