Третій етап.

**3.3 Описати абстрактний тип даних який необхідно для вирішення задачі дослідження.**

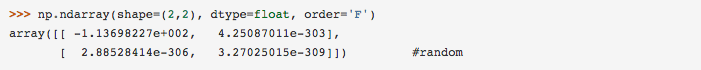
Для нашого проекту Visitors Tracker ми використовуємо вже існуючу структуру даних – numpy.ndarray. Ця структура даних входить до зовнішньої бібліотеки для мови програмування python – numpy. Основними особливостями numpy.ndarray і перевагами над звичайним ADT Array і стандартним list є набагато ширший функціонал та поле застосування. Бібліотека numpy широко застовується у data science, а numpy.ndarray це один з найбільш використовуваних основних типів даних цієї бібліотеки через велику кількість методів, які дозволяють маніпулювати структурою даних.

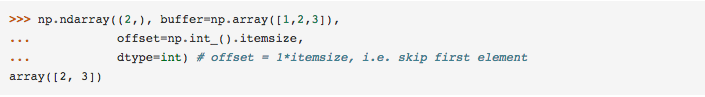
Numpy.ndarray приймає 6 аргументів:

1. Shape – кортеж цілих чисел, який показує розміри та виміри масиву.
2. Dtype (data-type) – необов’язковий аргумент, який вказує бажаний тип даних для numpy.array. Цей аргумент використовується щоб розширювати тип даних в масив, обернена функція - .astype(t), де t – тип.
3. Buffer – використовується для заповнення numpy.ndarray певними даними.
4. Order – необов’язковий аргумент, який відповідає за спосіб індексації елементів в numpy.ndarray (Row-major або Column-major).

Найчастіше використовувані методи numpy.ndarray:

* Astype(dtype) – створює копію масиву з елементами вказаного типу.
* Copy() – створює копію масиву.
* Fill(value) – заповнює всі елементи масиву скалярним значенням.
* Flatten() – створює одновимірний масив з усіх вкладених масивів.
* Item() – перетворює елемент numpy.ndarray в скалярне значення.
* Max(axis), min(axis) – повертає найбільше та найменше значення по вказаній осі, або на всіх осях по замовчуванню.
* Mean(axis) – повертає середнє значення масиву по вказаній осі, або по всіх осях по замовчуванню.
* Reshape() – повертає масив з тими самими даними, але іншою формою.
* Resize() – змінює розмір та форму масива.

Кілька прикладів використання numpy.ndarray (взято з офіційної документації numpy): 



Оскільки наша курсова робота передбачає роботу з відеозаписами, тобто обробку кожного кадру то представлення в числовому вигляді цих зображень реалізоване через numpy.ndarray. Це дозволяє дуже по-різному маніпулювати зображеннями і їх перетворенням в цілях оптимізації якості кінцевої теплової карти. Нижче наведений фрагмент модуля, в якому всі пікселі зображення піддаються спочатку перетворенням в тип float64 для детальнішої обробки, а потім в uint8 для полегшення візуалізації теплової карти.

gray = self.prepare\_frame(*frame*)  
processed = self.compare\_with\_prev(gray)  
processed = processed.astype(np.float64)  
self.res += (50 \* processed + gray) \* 0.1  
show\_res = (self.res) / self.res.max()  
show\_res = np.floor(show\_res \* 255)  
show\_res = show\_res.astype(np.uint8)  
show\_res = cv2.applyColorMap(show\_res, cv2.COLORMAP\_JET)  
cv2.imshow("res", show\_res)

Також завдяки numpy.ndarray ми можемо додавати різні коефіцєнти для варіації пікселів, що дозволяє змінювати насиченість кольорів на тепловій карти чи тривалість охолодження на статичних зонах на відео.

Отже, ми описали таку структуру даних як numpy.ndarray і її функціонал, який дуже ефективно оптимізує роботу з такими файлами як зображення (кадри з відео).