Лабораторна робота №1

Жук Дмитро РА-241

17 листопада 2025 р.

Тема: Основи Python – змінні, введення-виведення, типи даних, базові обчислення.

Мета: Ознайомитися із середовищем Jupyter Notebook. Навчитися працювати зі змінними та базовими типами даних. Використовувати print(), input(), type(). Виконувати прості арифметичні операції.

Хід роботи

Згідно комплексу лабораторних робіт буде роглянуто роботу програми ”UART Data Visualizer”.

У програмі використовується велика кількість змінних для зберігання станів, конфігурацій та даних.

1 Глобальні змінні (константи)

- `FRAME_SIZE`: Зберігає очікуваний розмір одного кадру даних (пакету байтів).
- `TERMINATOR`: Зберігає спеціальну послідовність байтів, яка означає кінець кадру.
- `UART_AVAILABLE`: Змінна-прапор (flag), яка показує, чи вдалося імпортувати бібліотеку `serial`.

Змінні-атрибути (у класах): Це змінні, що належать конкретному об’єкту (наприклад, `self.port`).

- `self.running`: Зберігає стан потоку (чи він повинен працювати).
- `self.dummy`: Зберігає вибір режиму (чи генерувати випадкові дані).

- `self.port`, `self.baudrate`: Зберігають налаштування COM-порту.
- `self.ser`: Зберігає об'єкт для роботи з COM-портом.
- `self.root`: Зберігає головне вікно програми (GUI).
- `self.config_data`: Зберігає дані, завантажені з конфігураційного JSON-файлу.
- `self.lines`, `self.subplots`: Зберігають об'єкти графіків для їх оновлення.
- `self.data`: Зберігає масив (історію) отриманих числових даних.

Локальні змінні (у функціях):

- `buffer`: Тимчасово накопичує байти, прочитані з порту.
- `dummy_floats`: Тимчасовий список для згенерованих випадкових чисел.
- `path`: Зберігає шлях до файлу, який вибрав користувач.
- `f`: Об'єкт відкритого файлу для читання.
- `baud`: Зберігає швидкість (baudrate), яку ввів користувач, після перетворення у число.

2 Типи Даних

Числові (`int` та `float`):

- **`int`** (ціле число) використовується для лічильників, індексів та налаштувань (наприклад, `baudrate = 115200`, `self.history_length = 200`).
- **`float`** (число з плаваючою комою) використовується для значень, що можуть бути нецілими, наприклад, час затримки (`time.sleep(0.1)`) або коефіцієнти масштабування (`cfg.get("scale 1.0)`).

Логічний (`bool`):

- Тип **`bool`** (значення `True` або `False`) використовується для керування станом програми. Наприклад, `self.running = True` або `self.dummy = dummy or not UART_AVAILABLE`.

Послідовності:

- **`str`** (рядок): Для будь-якої текстової інформації — заголовки вікон (”UART Data Visualizer”), мітки, повідомлення про помилки.
- **`bytes`** (байтовий рядок): Критично важливий тип для роботи з ”сирими”даними, що надходять з COM-порту. Наприклад, `TERMINATOR = b'\xAA\xBB'` та `buffer = b''`.
- **`list`** (список): Динамічний масив для зберігання об'єктів, наприклад, `self.lines`

= [] (список ліній графіка) або ports = [] (список доступних портів).

- **tuple** (кортеж): Незмінний список. Використовується для групування пов'язаних даних, наприклад (line, ch, ch_idx).

Словники (dict):

- Тип dict є ключовим для зберігання конфігурації. Він дозволяє зберігати дані у форматі "ключ: значення"(наприклад, self.config_data = json.load(f)).

Інші типи:

- None (тип NoneType) використовується для ініціалізації змінних, які ще не мають значення (self.config_data = None).
- **numpy.ndarray** (масив numpy) використовується для ефективного зберігання та обробки числових масивів self.data.

3 Введення-Виведення (Input/Output)

Введення (Input):

- **Введення від користувача (GUI):** Користувач вводить дані через графічні елементи: ttk.Combobox (вибір порту), ttk.Entry (введення тексту швидкості), ttk.Checkbutton (вибір режиму) та ttk.Button (старт/стоп).
- **Введення з файлу:** Програма відкриває діалогове вікно filedialog.askopenfile для вибору файлу, а потім зчитує (вводить) дані з нього за допомогою with open(path, 'r') as f:.
- **Введення з пристрою (UART):** Основне джерело даних. Команда data = self.se виконує низькорівневе читання (введення) байтів із зовнішнього пристрою.

Виведення (Output):

- **Виведення на екран (GUI):** Основне виведення — це оновлення графіків (self.ca Також програма виводить інформацію у спливаючих вікнах (messagebox.showinfo та messagebox.showerror(...)).
- **Виведення у консоль:** Використовується для налагодження та інформування про критичні помилки, які не бачить користувач, наприклад, print(f"UART open error: {e}").

4 Базові Обчислення

Арифметичні операції:

- Використовуються операції множення (`*`), додавання (`+`) та віднімання (`-`). Наприклад, `FRAME_SIZE = 16 * 4 + 2` (обчислення розміру), `buffer += data` (додавання байтів до буфера), `scaled = self.data[ch_idx] * cfg.get("scale 1` (масштабування даних для графіка).

Операції порівняння:

- Використовуються для прийняття рішень в умовних операторах `if`. Наприклад: `if frame[-2:] == TERMINATOR:` (перевірка на рівність), `if len(buffer) >= FRAME_SIZE:` (більше або дорівнює), `if subplot_idx < 0:` (менше).

Логічні операції:

- Оператори `and`, `or`, `not` використовуються для комбінування умов. Наприклад, `if self.reader and self.reader.running:` (перевіряє, що об'єкт існує **і** він активний) або `dummy or not UART_AVAILABLE` (використовує **або** та **не**).

Обчислення за допомогою функцій:

- Використовуються вбудовані функції, що виконують обчислення: `len()` (обчислення довжини послідовності), `int()` (обчислення цілого числа з рядка), `struct.unpack()` (складне обчислення для перетворення байтів у числа `float`).

Висновок

У ході лабораторної роботи досягнуто мети: ознайомлення з Jupyter Notebook, робота зі змінними, типами даних (int, float, bool, str, bytes, list тощо), функціями print(), input(), type() та базовими обчисленнями. Аналіз програми "UART Data Visualizer"і застосування змінних для станів і даних, введення-виведення через GUI, файли та UART, а також арифметичні, порівняльні й логічні операції. Робота закріпила фундаментальні навички Python для подальших завдань.