Лабораторна робота №3

Жук Дмитро РА-241

18 листопада 2025 р.

Тема: Списки, кортежі, робота з рядками, операції індексації та зрізів.

Мета: Ознайомитися з методами створення, обробки та маніпуляції списками, кортежами й рядками в Python. Навчитися використовувати індексацію, зрізи, функції та методи для роботи з цими типами даних.

Хід роботи

1 Списки (Lists)

Списки — це змінювані послідовності, які можуть містити елементи будь-якого типу. Їх створюють за допомогою квадратних дужок [] або функцій типу `list()`. Обробка включає додавання (`append, extend`), видалення (`pop, remove`), сортування (`sort`) тощо. Маніпуляція — це зміна елементів на місці. Індексація: `list[index]` для доступу (індекси починаються з 0, негативні — з кінця). Зрізи: `list[start:end:step]` для витягнення підсписків.

У програмі списки використовуються часто для зберігання даних, конфігурацій і елементів інтерфейсу.

Наприклад:

- **Створення списків:** Порожні списки створюються просто як [], а заповнені — з елементами або через list comprehension (генератори списків).

```
self.lines = []
self.subplots = []
```

Тут в `init` класу App створюються порожні списки для зберігання ліній графіків і `subplot`. Вони потім наповнюються в циклі.

Приклад створення через `list comprehension`:

```
dummy_floats = [random.uniform(0, 10) for _ in range(16)]
```

У методі `_run_dummy` класу `UARTReader` це створює список з 16 випадкових `float`. Функція random.uniform генерує числа, а `comprehension` — зручний спосіб заповнити список в один рядок.

Через `list()`:

```
self.channel_keys = list(channels_cfg.keys())
```

`setup_plots` перетворює ключі словника на список, щоб зберегти порядок каналів.

- **Обробка та маніпуляція:** Додавання елементів через `append`, доступ і зміна через індексацію.

```
self.subplots.append(ax)
```

У циклі for в `setup_plots` додається об'єкт ax `(subplot)` до списку. Це маніпуляція — список росте динамічно.

```
self.lines.append((line, ch, ch_idx))
```

Тут додається кортеж до списку `lines`. Потім в `update_plot` цей список **обробляється в циклі:**

```
for item in self.lines:
    line, cfg, ch_idx = item
    scaled = self.data[ch_idx] * cfg.get("scale", 1.0)
    line.set_ydata(scaled)
```

Ітерація по списку, розпаковка кортежу `(item - (line, cfg, ch_idx))`, обчислення `scaled` і оновлення даних лінії. Це показує, як списки використовуються для групування даних і їхньої обробки.

- **Індексація та зрізи:** Доступ до елементів за індексом, зрізи для підчастин.

```
ax = self.subplots[subplot_idx]
```

У `setup_plots: subplot_idx` — індекс, беремо конкретний `subplot` зі списку. Якщо індекс виходить за межі, код раніше коригує його if-умовами.

```
ports = [p.device for p in serial.tools.list_ports.comports()]
```

У `refresh_ports:` створює список портів. **Потім:**

```
if ports:
    self.port_var.set(ports[0])
```

Індексація `ports[0]` — бере перший елемент, якщо список не порожній.Зрізів у списків тут менше, але приклад з numpy-масивом (який подібний до списку) в `on_new_data`:

```
self.data[:, -1] = floats
```

Це індексація: всі рядки (:), останній стовпець (-1). Numpy-масив `self.data` створено як `np.zeros((16, self.history_length))`, але принцип схожий на списки списків.

2 Кортежі (Tuples)

Кортежі — незмінні послідовності, створюються круглими дужками () або без них для розпаковки. Вони швидші за списки, корисні для фіксованих даних. Обробка: не можна змінювати, але можна ітерувати. Маніпуляція обмежена (немає append), але підтримують індексацію та зрізи як списки.

- **Створення кортежів:** Прямо в дужках або через розпаковку.

```
self.lines.append((line, ch, ch_idx))
```

Тут створюється кортеж `(line, ch, ch_idx)` і додається до списку. Кортежі зручні для ”упаковки”кількох змінних разом.

```
frame, buffer = buffer[:FRAME_SIZE], buffer[FRAME_SIZE:]
```

У `_run_uart:` це розпаковка кортежу з двох зрізів. Python автоматично створює кортеж з правої сторони і розпаковує в дві змінні.

```
floats = struct.unpack('<16f', frame[:64])
```

Функція `unpack` повертає кортеж з 16 `float`. Це створення через функцію — типовий спосіб для фіксованих даних.

- **Обробка та маніпуляція:** Оскільки незмінні, маніпуляція — це створення нових, але обробка через ітерацію чи розпаковку.

```
line, cfg, ch_idx = item
```

У `update_plot: item` — це кортеж з `lines`, розпаковуємо в три змінні. Це зручна обробка — розбиваємо на частини для використання.

```
self.fig, self.ax = plt.subplots(figsize=(8, 6))
```

У `init`: `subplots` повертає кортеж `(fig, ax)`, розпаковуємо одразу. Це автоматизує присвоєння кількох значень.

- **Індексація та зрізи:** Як у списків, але без змін.

```
if frame[-2:] == TERMINATOR:
```

Тут `frame` — байтовий рядок `(bytes)`, але принцип той же: [-2:] — зріз останніх двох елементів (індексація з кінця). Порівнюємо з кортежем байтів `TERMINATOR = b'\xAA\xBB'`.

У unpack: `floats[0]` міг би бути, але тут весь кортеж присвоюється `self.data[:, -1] = floats` — це присвоєння кортежу масиву.

3 Рядки (Strings)

Рядки — послідовності символів, створюються лапками `""` або `''`. Обробка: конкатенація (+), форматування `(f-strings)`. Маніпуляція: створення нових рядків методами `(upper, split)`. Індексація: `str[index]`, зрізи: `str[start:end]`.

- **Створення рядків:** Прямо в лапках, або через форматування.

  Простий рядок для заголовка вікна. `f-string` для форматування:

```
label=f"{ch.get('name', ch_name)} [{ch.get('unit','')}]"
```

`setup_plots`: вставляє значення з словника в рядок. Це створення динамічного рядка з змінними.

```
print(f"UART open error: {e}")
```

`run: f-string` для логування помилки з змінною `e`.

- **Обробка та маніпуляція:** Методи типу `get` для словників, але для рядків — конкатенація, `format`.

```
buffer += data
```

`_run_uart: buffer - bytes` (схожий на рядок), `+=` додає нові дані. Для `str` це створює новий рядок (бо незмінні).

```
path = filedialog.askopenfilename(filetypes=[("JSON Files", "*.json")])
```

`load_config: filetypes` — список кортежів, але `"*.json"` — рядок для фільтра.

- **Індексація та зрізи:** Для доступу до частин.

```
frame[:64]
```

Зріз перших 64 байтів для unpack. `frame[-2:]` — останні два, як вище.

```
self.ax.text(0.5, 0.5, "Load config to start", ha='center', va='center')
```

`"Load config to start"` — рядок, але індексація не використовується; зрізи могли б бути для підрядків, наприклад, якщо б треба було витягти частину.

Висновок

У ході виконання лабораторної роботи №3 розглянуто основні послідовні типи даних мови Python: списки, кортежі та рядки.

Звіт оформлено в LaTeX з використанням класу `extarticle (14 pt)` та компілятора `XeLaTeX`. Це забезпечило професійну українську типографіку, чітке відображення коду з підсвіткою синтаксису (пакет listings), коректні відступи, клікабельні посилання та загалом значно вищий рівень презентації матеріалу порівняно зі звітами у текстових редакторах.