# Python for AI

Дәуренов Ә.Д

# 7-дәріс. NumPy: Python тіліндегі массивтер және сандық есептеулер

## 1. Дәрістің мақсаты

NumPy кітапханасының сандық есептеулердегі рөлін, массивтердің құрылымын, деректерді компьютер жадында сақтау ерекшеліктерін және векторлық есептеулердің негізгі принциптерін түсіндіру.
Дәріс соңында білім алушылар:

- Сандық есептеулердің инженериядағы маңызын түсінеді.
- Бірөлшемді және көпөлшемді массивтерді ажыратады.
- NumPy массивінің Python тізімінен айырмашылығын түсіндіреді.
- Векторизация және broadcasting принциптерін сипаттайды.
- NumPy кітапханасының жасанды интеллект жүйелеріндегі рөлін біледі.

# Сандық есептеулер дегеніміз не?

Сандық есептеулер (Numerical Computing) — математикалық есептерді компьютер көмегімен сандық мәндер арқылы шешу процесі.

Инженерлік жүйелерде көптеген физикалық параметрлер үздіксіз өзгеріп отырады.

Мысалы, электрмобиль қозғалған кезде батарея кернеуі, ток күші, қозғалтқыш температурасы және айналу жиілігі өзгереді.

Компьютер бұл параметрлерді белгілі бір уақыт аралығында өлшенген сандық мәндер ретінде өңдейді.

## Сандық есептеулердің негізгі қолданылу бағыттары:

1. Физикалық процестерді модельдеу.
2. Өлшеу нәтижелерін өңдеу.
3. Сигналдарды талдау.
4. Статистикалық есептеулер жүргізу.
5. Машиналық оқыту алгоритмдерін іске асыру.

Мысалы, инженер батареяның бір ғана температурасын емес, бірнеше сағат ішінде тіркелген мыңдаған өлшеуді талдауы мүмкін.

Осындай есептер үшін тиімді **сандық есептеу құралдары қажет.**

# Массив ұғымы

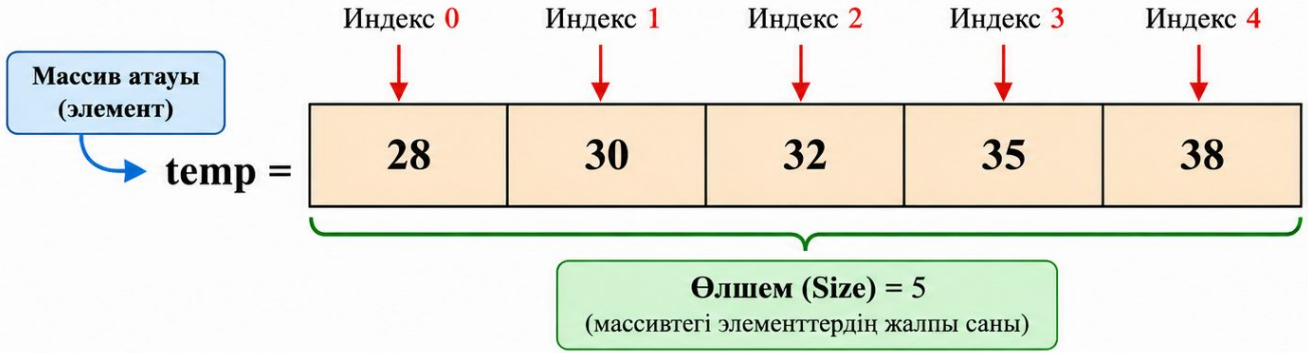
Массив (Array) — белгілі бір ретпен ұйымдастырылған элементтер жиынтығы.

Массивтің әр элементі белгілі бір орынға ие болады.

Элементтің орналасқан орны индекс деп аталады.

# Бірөлшемді және көпөлшемді массивтер

## Бірөлшемді массивтің құрылымы (мысал)




**Массив элементтері**

Массив элементтері — температура мәндері (°C).

Python тілінде индекстеу 0-ден басталады.

**Элемент (элемент)**

Массив ішінде сақталатын жеке мән.
Мысалы: 28, 30, 32, 35, 38.

**Индекс (index)**

Элементтің орналасқан орны.
Python тілінде индекс 0-ден басталады.

**Өлшем (size)**

Массивтегі элементтердің жалпы саны.
Бұл мысалда өлшем = 5.

**Деректер типі (data type)**

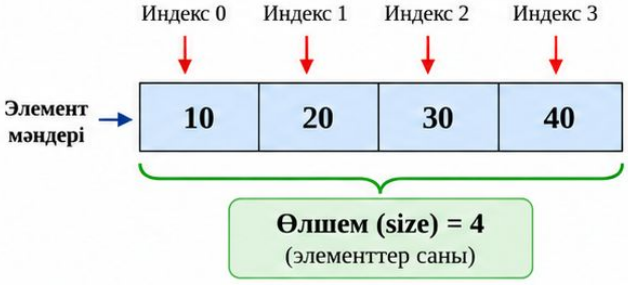
Сақталатын мәндердің түрі.
Бұл массивте — бүтін сандар (int) (температура °C).

# Бірөлшемді, екіөлшемді және үшөлшемді массивтер

## 1. Бірөлшемді массив (1D Array)

- Бір ось бойымен орналасқан элементтерден тұрады.
- Математикада **вектор** түрінде қарастырылады.

```
x = [10, 20, 30, 40]
```




## 2. Екіөлшемді массив (2D Array)

- Жолдар мен бағандардан тұрады.
- Математикада **матрица** деп аталады.

$$A = \begin{bmatrix} 30 & 380 \\ 35 & 375 \\ 40 & 370 \end{bmatrix}$$

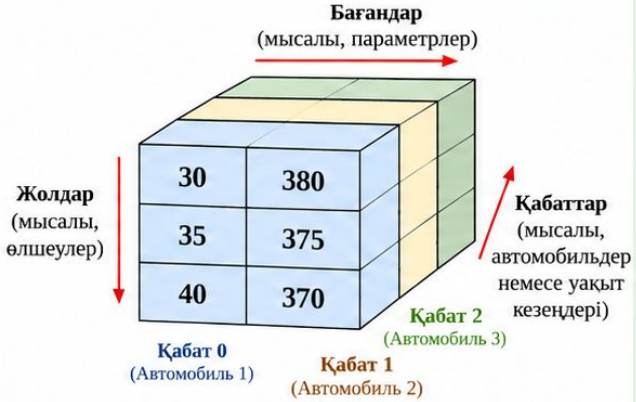
| | Баған 0 (Температура) | Баған 1 (Кернеу) |
|---|---|---|
| Жол 0 (Өлшеу 1) | 30 | 380 |
| Жол 1 (Өлшеу 2) | 35 | 375 |
| Жол 2 (Өлшеу 3) | 40 | 370 |

- Бірінші баған — температура (°C).
- Екінші баған — кернеу (В).
- Әр жол — жеке өлшеу.

```
A = [[30, 380],
     [35, 375],
     [40, 370]]
```

## 3. Үшөлшемді массив (3D Array)

- Бірнеше екіөлшемді матрицаның жиынтығы ретінде елестетуге болады.
- Бірдей өлшеулер бірнеше объектіден немесе бірнеше уақыт кезеңінен жиналса, ұйымдастыруға қолданылады.




```
data = [
   [[30, 380], [35, 375], [40, 370]],  # Қабат 0
   [[32, 390], [36, 372], [41, 368]],  # Қабат 1
   [[28, 385], [34, 378], [39, 365]]   # Қабат 2
]
```

# NumPy кітапханасы және оның мақсаты

NumPy (Numerical Python) — Python тілінде массивтермен және сандық деректермен тиімді жұмыс істеуге арналған ашық бастапқы кодты кітапхана.

NumPy-дың негізгі құрылымы — ndarray (N-dimensional Array), яғни N өлшемді массив.
NumPy келесі міндеттерді орындауға мүмкіндік береді:

- Үлкен сандық массивтерді сақтау.
- Массив элементтерімен математикалық операциялар жүргізу.
- Матрицалық есептеулер орындау.
- Статистикалық көрсеткіштерді есептеу.
- Сандық деректерді түрлендіру.
- Ғылыми модельдер мен машиналық оқыту жүйелеріне деректер дайындау.

NumPy — Python экожүйесіндегі ғылыми есептеулердің негізгі кітапханаларының бірі.

# Python тізімі мен NumPy массивінің айырмашылығы

Python тілінде деректерді сақтау үшін list құрылымы қолданылады. Алайда үлкен көлемдегі біртекті сандық деректерді өңдеу кезінде NumPy массивтері тиімді болуы мүмкін.

| Сипаттама | Python List | NumPy Array |
|---|---|---|
| Негізгі мақсаты | Жалпы деректерді сақтау | Сандық есептеулер |
| Элементтердің типтері | Әртүрлі болуы мүмкін | Әдетте біртекті |
| Жадыны пайдалану | Әр элементке жеке объект сілтемелері болуы мүмкін | Біртекті сандық мәндер ықшам сақталады |
| Математикалық операциялар | Көбіне циклдер қолданылады | Массивтік операциялар қолданылады |
| Үлкен массивтермен есептеу | Кейде баяуырақ | Көптеген операцияларда жылдамырақ |

Бұл NumPy кез келген жағдайда Python тізімінен жылдамырақ дегенді білдірмейді. Артықшылық әсіресе үлкен біртекті сандық массивтерге қолданылатын есептеулерде байқалады.

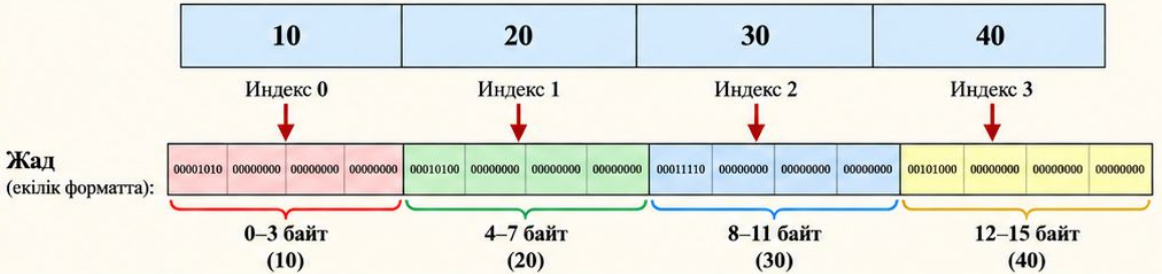
# NumPy массивтерінің компьютер жадында сақталуы

## Негізгі идея

- Компьютерде барлық деректер **екілік форматта** сақталады.
- NumPy массивтері — біртекті сандық элементтер жадта **біртұтас (үздіксіз) аймақта** сақталады.
- Массивтің элементтері жадта **бірінен кейін бірі** орналасады (gap жоқ).
- Бұл құрылым көптеген сандық элементтерді тиімді өңдеуге мүмкіндік береді.

## Массив элементтерінің жадта орналасуының моделі

**Массив:** `arr = np.array([10, 20, 30, 40], dtype=np.int32)`




**Мысал:** төрт int32 элементі, әрқайсысы 4 байт. Бұл үздіксіз орналасқан массивтің қарапайым көрінісі. NumPy массивтері кейде жадтың ортақ бөлігін пайдаланатын көріністер (views) болуы мүмкін.

## NumPy деректер типтері

| Тип | Сипаттамасы | Әдеттегі көлемі |
|---|---|---|
| `int8` | Бүтін сан | 1 байт |
| `int16` | Бүтін сан | 2 байт |
| `int32` | Бүтін сан | 4 байт |
| `int64` | Бүтін сан | 8 байт |
| `float32` | Жылжымалы нүктелі сан | 4 байт |
| `float64` | Жылжымалы нүктелі сан | 8 байт |

## Жад көлемінің мысалы

1 миллион өлшеуді `float64` форматында сақтау үшін тек сандық элементтердің өзіне шамамен 8 МБ жады қажет.

**1 000 000 × 8 = 8 000 000 байт**
**≈ 8 МБ**




Массивтің қосымша қызметтік ақпараты (shape, dtype және т.б.) бұл есепке кірмейді.

# Векторизация принципі (Vectorization)

Векторизация — математикалық операцияны массивтің көптеген элементіне бірден қолдану тәсілі.

Дәстүрлі бағдарламалауда әр элементті өңдеу **үшін цикл пайдаланылуы** мүмкін.

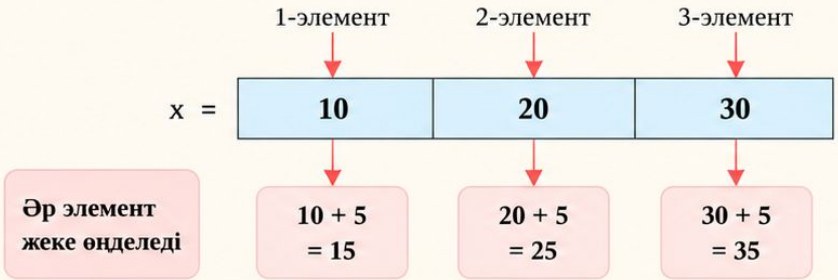
Векторизацияда – бағдарламашы массивке орындалатын **операцияны тұтас көрсетеді,**

**ал кітапхана – есептеуді оңтайландырылған ішкі процедуралар арқылы** орындайды.

# Циклдік өңдеу мен векторизацияны салыстыру

## 1. Циклдік өңдеу (элементтерді жеке өңдеу)

Массивтің әр элементі цикл арқылы жеке өңделеді.




```
# Цикл арқылы әр элементке +5 қосу
import numpy as np
x = np.array([10, 20, 30])
y = np.empty_like(x)  # нәтижені сақтау үшін бос массив
for i in range(len(x)):
    y[i] = x[i] + 5
print(y)  # [15 25 35]
```

## 2. Векторизация (бір операция, барлық элементтерге)

Бір массивтік операция барлық элементтерге бір уақытта қолданылады.




```
# Векторизация арқылы бірден барлық элементтерге +5 қосу
import numpy as np
x = np.array([10, 20, 30])
y = x + 5
print(y)  # [15 25 35]
```

**Нәтиже:**

- Циклдік өңдеу — әр элемент жеке өңделеді (жылдамдығы төмен).
- Векторизация — бір операция барлық элементтерге бірден қолданылады (жылдам және тиімді).
- NumPy-де векторизация ұсынылады, себебі ол үлкен массивтермен жұмыс істегенде айтарлықтай жылдам.

Векторизацияның артықшылықтары:

- Бағдарлама коды қысқарады.
- Көптеген есептеулер тиімді орындалады.
- Python циклдерінің қосымша шығындары азаяды.
- Математикалық формулаларды бағдарламада көрсету жеңілдейді.

# Broadcasting принципі

## 1. Негізгі идея

- **Broadcasting** — NumPy кітапханасында өлшемдері әртүрлі, бірақ өзара үйлесімді массивтерге математикалық операцияларды қолдану механизмі.
- Кіші өлшемді массив (мысалы, бір сан немесе бір өлшемді массив) үлкен өлшемді массивтің бойынша автоматты түрде “кеңейтіледі”.
- Осылайша, барлық элементтерге бірдей операция қолданылады.

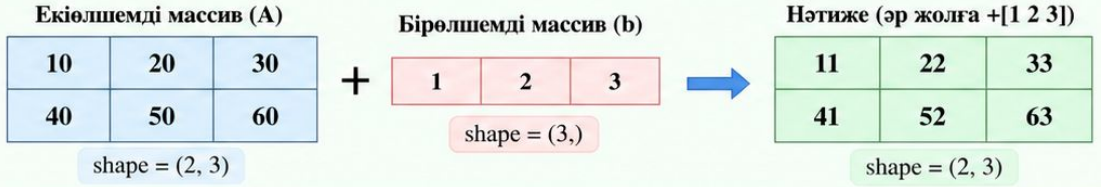
**Ереже:**
Өлшемдер оң жақтан салыстырылады және әр сәйкес өлшем бірдей немесе олардың біреуі 1 болуы керек.

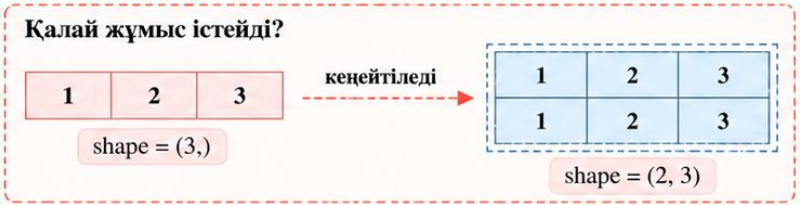
## 2. 1D массивке broadcasting (скалярмен мысал)




```
import numpy as np
x = np.array([20, 25, 30])
y = x + 5  # broadcasting
print(y)
# [25 30 35]
```

## 3. 2D массивке broadcasting (жол бойынша мысал)




```
import numpy as np
A = np.array([[10, 20, 30],
              [40, 50, 60]])
b = np.array([1, 2, 3])
C = A + b  # broadcasting
print(C)
#  [[11 22 33]
#   [41 52 63]]
```

**Қорытынды:**

- Broadcasting әртүрлі өлшемдегі, бірақ үйлесімді массивтерге операцияларды автоматты түрде қолдануға мүмкіндік береді.
- Өлшемдер оң жақтан салыстырылады, әр сәйкес өлшем бірдей немесе олардың біреуі 1 болуы керек.

# Математикалық және статистикалық есептеулер (NumPy)

## Негізгі идея

- NumPy инженерлік өлшеулерді математикалық және статистикалық талдауға мүмкіндік береді.
- Негізгі сипаттамалар:
  - Орташа арифметикалық мән (mean)
  - Медиана (median)
  - Минимум, максимум
  - Стандартты ауытқу
  - Басқа статистикалық көрсеткіштер

Бұл көрсеткіштер деректерді талдауда, жүйелерді бағалауда кеңінен қолданылады.

## 10.1 Орташа арифметикалық мән (Mean)

Орташа мән — өлшеулердің жалпы деңгейін сипаттайды.

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

**Мысал: батарея температурасы**

| Өлшемдер (температура, °C) | 20 | 21 | 22 | 23 | 80 |
|---|---|---|---|---|---|

**Орташа мән:**

$$\overline{x} = \frac{20 + 21 + 22 + 23 + 80}{5} = 33.2$$

Орташа мән барлық өлшеулердің жаллы деңгейін көрсетеді. Бұл мысалда орташа температура 33.2 °C.

## 10.2 Медиана (Median)

Медиана — реттелген деректер жиынындағы орталық мән.

1. Мәліметтерді өсу ретімен орналастырамыз:

| 20 | 21 | 22 | 23 | 80 |
|---|---|---|---|---|

**Медиана:**

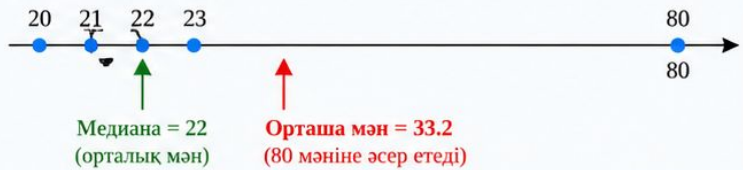
$$Me = 22$$

80 мәні қалған өлшеулерден едәуір жоғары болғандықтан, орташа мәнге қатты әсер етеді. Ал медиана мұндай шеткі мәндерге төзімдірек.

## Нәтижені салыстыру

| | | |
|---|---|---|
| **Орташа мән** | $\overline{x} = 33.2$ | — барлық мәндерге әсер етеді (шеткі мәндерге сезімтал). |
| **Медиана** | $Me = 22$ | — шеткі мәндерге төзімді. |

**Деректердің көрінісі**



# 10.3 Стандартты ауытқу (Standard Deviation)

## 1. Негізгі идея

Стандартты ауытқу өлшеулердің орташа мәннен қаншалықты таралғанын (шашыраңқылығын) көрсетеді. Мәндер неғұрлым алшақ болса, стандартты ауытқу соғұрлым үлкен болады.

**Формула (бас жиынтық үшін):**

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}$$

**Мұндағы:**

- $\sigma$ — стандартты ауытқу.
- $x_i$ — жеке өлшеу.
- $\mu$ — орташа мән.
- $n$ — өлшеулер саны.

## 2. Есептеу мысалы

Екі қатардың орташа мәні 40°C, бірақ таралу деңгейлері әртүрлі.

### Аз ауытқуы бар деректер (тұрақты)

| Өлшеулер ($x_i$) | 39 | 40 | 41 | 40 | 39 |
|---|---|---|---|---|---|

**1) Орташа мән:**

$$\mu = \frac{39 + 40 + 41 + 40 + 39}{5} = 40$$

**2) Стандартты ауытқу:**

$$\sigma = \sqrt{\frac{1}{5}\left[(39-40)^2 + (40-40)^2 + (41-40)^2 + (40-40)^2 + (39-40)^2\right]}$$

$$= \sqrt{\frac{1}{5}\left(1 + 0 + 1 + 0 + 1\right)} = \sqrt{\frac{3}{5}} \approx 0.77$$

### Үлкен ауытқуы бар деректер (құбылмалы)

| Өлшеулер ($x_i$) | 25 | 55 | 30 | 50 | 40 |
|---|---|---|---|---|---|

**1) Орташа мән:**

$$\mu = \frac{25 + 55 + 30 + 50 + 40}{5} = 40$$

**2) Стандартты ауытқу:**

$$\sigma = \sqrt{\frac{1}{5}\left[(25-40)^2 + (55-40)^2 + (30-40)^2 + (50-40)^2 + (40-40)^2\right]}$$

$$= \sqrt{\frac{1}{5}\left(225 + 225 + 100 + 100 + 0\right)}$$

$$= \sqrt{130} \approx 11.40$$

## 3. График арқылы салыстыру




## 4. Нәтижені салыстыру

| Қатар | Орташа мән ($\mu$) | Стандартты ауытқу ($\sigma$) | Сипаттамасы |
|---|---|---|---|
| Аз ауытқу | 40 | 0.77 | Мәндер ортаға жақын, тұрақты. |
| Үлкен ауытқу | 40 | 11.40 | Мәндер кең таралған, құбылмалы. |

## 5. Қорытынды

- Екі қатардың орташа мәні бірдей (40°C), бірақ олардың таралу деңгейі әртүрлі.
- Стандартты ауытқу мәндердің шашыраңқылығын көрсетеді.
- Мәндер неғұрлым алшақ болса, стандартты ауытқу соғұрлым үлкен болады.
- Инженерлік талдауда тек орташа мәнді емес, деректердің таралуын да ескеру қажет.

# 11. Нормализация ұғымы (Normalization)

## 1. Негізгі идея

**Нормализация (Normalization)** — сандық мәндерді белгілі бір ортақ масштабқа келтіру процесі.

- Мақсаты — әр түрлі өлшем бірліктеріндегі мәндерді салыстыруға мүмкіндік беру.
- Нәтижесінде барлық мәндер бір диапазонға (әдетте 0 мен 1 арасына) түрлендіріледі.
- Бұл көптеген машиналық оқыту алгоритмдерінің тиімді жұмыс істеуіне көмектеседі.

## 2. Мысал: автомобиль жүйесіндегі параметрлер

| | |
|---|---|
| **Температура:** | 20 – 80 °C |
| **Кернеу:** | 300 – 450 В |
| **Жылдамдық:** | 0 – 150 км/сағ. |

Бұл параметрлер әртүрлі өлшем бірліктерімен және масштабтармен беріледі, сондықтан оларды нормализациялау қажет.

## 3. Min-Max нормализациясы

**Формула:**

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

- $x$ — бастапқы мән.
- $x_{\min}$ — минимал мән.
- $x_{\max}$ — максимал мән.
- $x'$ — нормализацияланған мән (0 мен 1 арасында).

Бұл формула арқылы мәндер әдетте 0 мен 1 аралығына түрлендіріледі.

## 4. Нормализациялау мысалы

Бір мәннің нормализациясын есептейік.

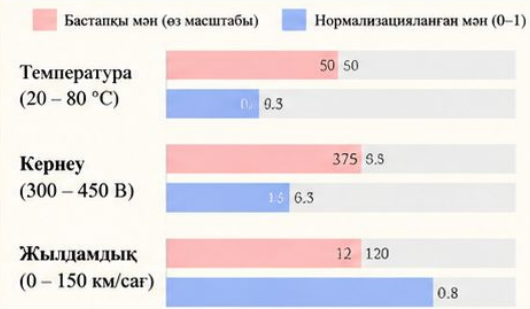
| | | |
|---|---|---|
| **Температура (20 – 80 °C)** | $x' = \frac{50-20}{80-20} = \frac{30}{60} = 0.5$ | **50 °C → 0.5** |
| **Кернеу (300 – 450 В)** | $x' = \frac{375-300}{450-300} = \frac{75}{150} = 0.5$ | **375 В → 0.5** |
| **Жылдамдық (0 – 150 км/сағ)** | $x' = \frac{120-0}{150-0} = \frac{120}{150} = 0.8$ | **120 км/сағ → 0.8** |

## 5. Нормализацияның әсері

Нормализациялауға дейін және кейінгі мәндердің салыстырмасы:

| Параметр | Бастапқы мән | Нормализацияланған мән (0–1) |
|---|---|---|
| Температура (°C) | 50 | 0.5 |
| Кернеу (В) | 375 | 0.5 |
| **Жылдамдық (км/сағ)** | **120** | **0.8** |

## 6. Визуализация




## 7. Қорытынды

- Нормализация әр түрлі өлшем бірліктеріндегі сандық мәндерді бір ортақ масштабқа келтіреді.
- Нәтижесінде барлық параметрлер 0 мен 1 аралығында болады.
- Бұл машиналық оқыту алгоритмдерінің тиімді жұмыс істеуіне көмектеседі, бірақ барлық алгоритмдер үшін міндетті емес.
- Нормализацияны қолдану қажеттілігі модельдің ерекшелігіне байланысты.

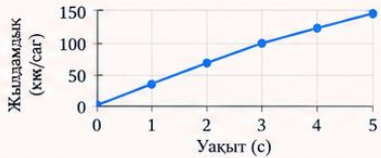
# 12. NumPy кітапханасының инженерияда қолданылуы

**NumPy** – тек деректерді сақтау құралы емес. Ол әртүрлі инженерлік есептеулердің математикалық негізін қамтамасыз етеді.

## 1. Автомобиль инженериясы

Жылдамдық пен үдеу деректерін өңдеу



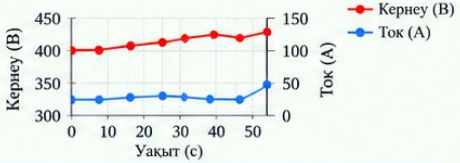



```
import numpy as np
t = np.array([0, 1, 2, 3, 4, 5])  # уақыт
v = np.array([0, 30, 60, 90, 120, 150])  # жылдамдық
a = np.gradient(v, t)  # үдеуді есептеу
```

## 2. Электромобиль жүйелері

Батарея кернеуі мен ток өлшеулерін талдау



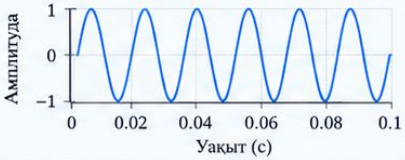



```
import numpy as np
voltage = np.array([...])  # кернеу (В)
current = np.array([...])  # ток (А)
power = voltage * current  # қуатты есептеу
```

## 3. Электроника

Электр сигналдарын сандық өңдеу

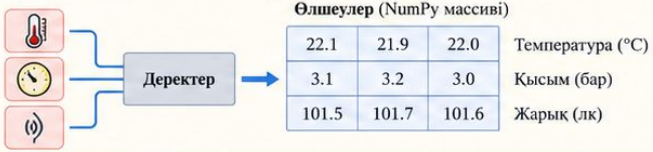





```
import numpy as np
t = np.linspace(0, 0.1, 1000)  # уақыт
signal = np.sin(2 * np.pi * 50 * t)  # 50 Гц сигнал
filtered = np.fft.fft(signal)  # жиілік талдауы (FFT)
```

## 4. Автоматика

Датчик өлшеулерін салыстыру

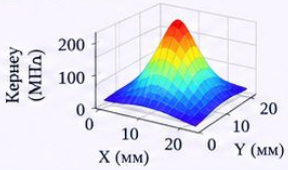



```
import numpy as np
sensors = np.array([[22.1, 21.9, 22.0],
                    [3.1, 3.2, 3.0],
                    [101.5, 101.7, 101.6]])  # датчик деректері
mean_values = sensors.mean(axis=1)  # орташа мәндер
```

## 5. Машина жасау

Механикалық параметрлерді есептеу






```
import numpy as np
force = np.array([...])  # күш (Н)
area = np.array([...])   # аудан (м²)
stress = force / area    # механикалық кернеу (МПа)
```

## 6. Жасанды интеллект

Сандық белгілер мен модель параметрлерін ұсыну




```
import numpy as np
X = np.array([0.2, 0.7, 0.5, 0.1],
             [0.3, 0.6, 0.4, 0.2], ...])  # белгілер
y = np.array([0, 1, 0, ...])  # мақсатты мәндер
```

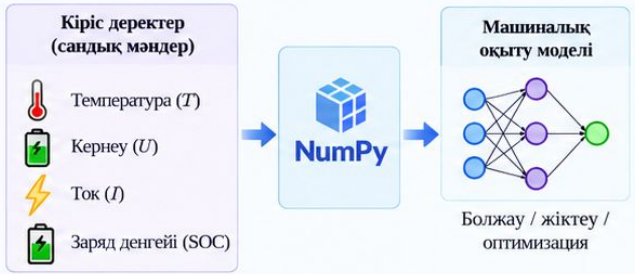
## Қорытынды:

- NumPy инженерлік есептеулерді тиімді, жылдам және дәл орындауға мүмкіндік береді.
- Ол деректерді талдау, модельдеу, сигналдарды өңдеу және машиналық оқытуда кеңінен қолданылады.

# 13. NumPy және жасанды интеллект

## 1. Негізгі идея

Машиналық оқыту алгоритмдері деректермен математикалық операциялар жүргізеді.
NumPy үлкен көлемдегі сандық деректерді тиімді сақтауға және өңдеуге мүмкіндік береді.




## 2. Бір өлшеу (бір объект)

Бір көлік жүйесінің параметрлері бір вектор түрінде беріледі:

$$\mathbf{x} = [T,\ U,\ I,\ SOC]$$

Мұнда:
$T$ — температура,
$U$ — кернеу,
$I$ — ток,
$SOC$ — заряд деңгейі.

**Мысал (бір көлік):**

| T (°C) | U (V) | I (A) | SOC |
|---|---|---|---|
| 30 | 380 | 25 | 0.8 |

```
import numpy as np
# Үш көлік деректері
x = np.array([30, 380, 25, 0.8])
print(x)
#  [ 30.  380.   25.    0.8]
```

## 3. Көп өлшеу (бірнеше объект)

Бірнеше көлік үшін параметрлер матрица түрінде сақталады:

$$X = \begin{bmatrix} T_1 & U_1 & I_1 & SOC_1 \\ T_2 & U_2 & I_2 & SOC_2 \\ T_3 & U_3 & I_3 & SOC_3 \end{bmatrix}$$

Бұл матрицада әр жол — бір көлік,
әр баған — белгілі бір параметр.

**Мысал (3 көлік):**

| T (°C) | U (V) | I (A) | SOC |
|---|---|---|---|
| 25 | 370 | 20 | 0.6 |
| 30 | 380 | 25 | 0.8 |
| 35 | 390 | 18 | 0.9 |

```
import numpy as np
# Үш көлік деректері
X = np.array([
    [25, 370, 20, 0.6],
    [30, 380, 25, 0.8],
    [35, 390, 18, 0.9]
])
print(X)
#  [ 25. 370.  20.  0.6]
#  [ 30. 380.  25.  0.8]
#  [ 35. 390.  18.  0.9]]
```

## 4. Машиналық оқытуда қолданылуы

Кіріс матрицасы (X) модельге беріледі, модель осы деректерден заңдылықтарды үйренеді және жаңа жағдайлар үшін нәтиже болжайды.

Кіріс деректері (X)

| T (°C) | U (V) | I (A) | SOC |
|---|---|---|---|
| 25 | 370 | 20 | 0.6 |
| 30 | 380 | 25 | 0.8 |
| 35 | 390 | 18 | 0.9 |
| ... | ... | ... | ... |




**Шығыс (мысалдар)**

- Жүріс қоры (км)
- Батарея жағдайы
- Ақауды анықтау
- Жүйені оңтайландыру

## 5. NumPy арқылы мүмкін болатын операциялар

NumPy модель құруда қажет негізгі математикалық операцияларды тез орындайды:

- Массивтерді сақтау және өңдеу
- Матричалық есептеулер
- Статистикалық талдау (орташа, ауытқу)
- Нормализация
- Деректерді масштабтау
- Фильтрация, сорттау
- Векторизация (жылдам есептеу)
- Көп өлшемді деректермен жұмыс

```
import numpy as np
# Мысал: нормализация (min-max)
X = np.array([[25, 370, 20, 0.6],
              [30, 380, 25, 0.8],
              [35, 390, 18, 0.9]])
X_min = X.min(axis=0)
X_max = X.max(axis=0)
X_norm = (X - X_min) / (X_max - X_min)
print(X_norm)
# Нәтиже: барлық мәндер 0 мен 1 арасында
```

**Қорытынды:**

NumPy сандық деректерді тиімді сақтауға, өңдеуге және математикалық операциялар жүргізуге мүмкіндік беріп, машиналық оқыту және жасанды интеллект модельдерінің негізін құрайды.

# Дәрістің қорытындысы

NumPy — ғылыми және инженерлік сандық есептеулердің маңызды құралы. Оның негізгі артықшылықтары массивтерді тиімді ұйымдастыруға, математикалық операцияларды векторизациялауға және көпөлшемді деректермен жұмыс істеуге байланысты.

NumPy принциптерін түсіну кейінгі машиналық оқыту, деректерді талдау және инженерлік модельдеу тақырыптарын меңгеру үшін маңызды.

# Бақылау сұрақтары

1. Сандық есептеулер дегеніміз не?
2. Массив пен вектордың қандай байланысы бар?
3. Бірөлшемді және екіөлшемді массивтердің айырмашылығы қандай?
4. NumPy массивтері компьютер жадында қалай ұйымдастырылады?
5. ndarray дегеніміз не?
6. Векторизацияның негізгі артықшылығы қандай?
7. Broadcasting қандай жағдайда қолданылады?
8. Медиана мен орташа мәннің айырмашылығы қандай?
9. Нормализация не үшін қажет?
10. NumPy жасанды интеллект жүйелерінде қандай рөл атқарады?