ИНТЕЛЛЕКТУАЛЬНЫЙ АНАЛИЗ ДАННЫХ

Лекция 10

# Даты и время в Python и pandas

Оспан Ә.Ғ., PhD, доцент-исследователь

КазНУ, кафедра ИИ и Big Data

2026-2027 учебный год

# Цель и содержание лекции

## Цель и результаты обучения

- Различать запись даты, момент и длительность.
- Использовать явные форматы, интервалы и часовые пояса.
- Проверять разбор и дневную агрегацию событий.

## Содержание лекции

1. **Типы и форматы**
2. **Календарная арифметика**
3. **Окна дат**
4. **UTC и Unix-время**
5. **Временные колонки pandas**

# Строка, момент и длительность - разные объекты

| Объект | Пример смысла | Что делаем |
|---|---|---|
| str | Текстовая запись даты | Разбираем по известному формату |
| datetime / Timestamp | Дата и время события | Сравниваем и выделяем компоненты |
| date | Календарная дата | Работаем с днями |
| timedelta | Длительность между моментами | Считаем полное число секунд |
| Unix timestamp | Число единиц от эпохи | Указываем единицу и UTC |

До преобразования установите смысл поля, единицы и часовой пояс.

# Шаблон даты: регистр меняет значение

| Обозначение | Смысл | Типичная путаница |
|---|---|---|
| %Y | Четырехзначный год | Не короткий %y |
| %m / %M | Месяц / минута | Регистр принципиален |
| %d | День месяца | Не номер месяца |
| %H | Час 00-23 | 24-часовая запись |
| %I + %p | Час 01-12 + AM/PM | Нужна часть суток |
| %S | Секунда | Не полная длительность |

# Разобрать дату и подготовить текст отчета

## Пример Python

```
from datetime import datetime, timedelta, timezone
import pandas as pd

text = '2019-07-07T18:59:33'
moment = datetime.strptime(text, '%Y-%m-%dT%H:%M:%S')
display_text = moment.strftime('%d.%m.%Y')
print(display_text)
print(moment.hour, moment.minute, moment.second)
```

Результат: 07.07.2019; 18 59 33

## Разбор

- strptime: str -> datetime.
- Шаблон явно задает смысл компонентов.

## Форматирование

- strftime: datetime -> str.
- Сокращенный вывод не меняет исходный moment.

# Неоднозначный формат и сортировка строк

## Неоднозначность

- 01/02/2024 требует соглашения.
- Автоматическое угадывание может быть неверным.
- Проверяем примеры после двенадцатого дня.

## Порядок

- Текстовый порядок не всегда календарный.
- Разбираем даты до сортировки и сравнения.
- Явный формат делает правило проверяемым.

Правдоподобная дата может быть результатом неправильной интерпретации исходного текстового формата.

# Длительность: разность полных моментов

**duration_seconds = (end - start).total_seconds()**

## Полная разность

- Учитывает переход минуты, дня, месяца и года.
- Считается по объектам времени, не по строкам.

## Проверка

- seconds хранит остаток, а не весь интервал.
- Отрицательный результат требует отдельного анализа.

timedelta(days=2, seconds=7): seconds=7; total_seconds()=172807.

Не заменяйте отрицательную длительность модулем без выяснения причины.

# Переход через полночь и 29 февраля

### Пример Python

```
start = datetime(2024, 2, 28, 23, 59, 50)
end = start + timedelta(seconds=20)
print(end.isoformat())
print((end - start).total_seconds())
long_gap = timedelta(days=2, seconds=7)
print(long_gap.seconds, long_gap.total_seconds())
```

Результат: 2024-02-29T00:00:10; 20.0; 7 172807.0

### Календарь

- После 28 февраля наступает 29 февраля.
- Переход месяца обрабатывается библиотекой.

### Интервал

- Разность равна двадцати секундам.
- Полные секунды учитывают и дни.

# Окно дат: включенность границ задается заранее

1

### Уточнить отчетный пояс

Определить, в каком поясе начинается календарный день.

2

### Задать начало и конец

Например, [1 марта 00:00, 2 марта 00:00).

3

### Разобрать и сравнить

Сопоставлять временные объекты, а не разнородные строки.

4

### Проверить границы

Событие ровно на конце входит в следующее окно.

- Соседние окна [start, end) не пересекаются.
- Три календарных даты и 72 часа - разные формулировки задачи.

# Наивное и осведомленное время

## Наивный datetime

- Пояс отсутствует.
- Нужна договоренность источника.
- Назначение tzinfo задает интерпретацию.

## Осведомленный datetime

- Момент связан с UTC.
- astimezone сохраняет момент, меняя часы.
- Отчетная дата может измениться при переводе.

Один момент может иметь разные календарные даты в разных поясах и отчетных окнах.

# Перевод пояса и обратимость Unix-времени

## Пример Python

```
utc_moment = datetime(2024, 3, 1, tzinfo=timezone.utc)
plus_five = timezone(timedelta(hours=5))
local_moment = utc_moment.astimezone(plus_five)
stamp = utc_moment.timestamp()
restored = datetime.fromtimestamp(stamp, tz=timezone.utc)
print(local_moment.isoformat())
assert restored == utc_moment
```

Результат: 2024-03-01T05:00:00+05:00

## Перевод

- UTC 00:00 становится 05:00 при +05:00.
- Момент остается тем же.

## Обратимость

- Unix-число создается из осведомленного объекта.
- Восстановление явно указывает UTC.

# Unix-время и календарные интервалы

| Задача | Явное соглашение | Типичная ошибка |
|---|---|---|
| Unix-секунды | unit="s", UTC | Принять число за миллисекунды |
| Unix-миллисекунды | unit="ms", UTC | Принять число за секунды |
| Последние 72 часа | Фиксированная длительность | Подменить календарными датами |
| Следующий месяц | Календарное смещение | Всегда прибавлять 30 дней |
| День отчета | Границы в отчетном поясе | Группировать в случайном поясе |

# Временная колонка pandas: разбор и контроль

1

### Сохранить исходный текст

Нужен для диагностики и объяснения причины NaT.

2

### Задать формат и UTC

Параметры должны соответствовать договоренности источника.

3

### Посчитать ошибки

Разделить корректные строки и карантин до агрегации.

4

### Упорядочить и агрегировать

Использовать временной индекс и заданные календарные окна.

- dt.hour и dt.date зависят от выбранного пояса.
- Число включенных событий должно совпадать с суммой дневных count.

# Учебные события: NaT и дневные суммы

## Пример Python

```
events = pd.DataFrame({
    'time_text': ['2024-02-28T23:00:00Z',
                  '2024-02-29T01:00:00Z',
                  '2024-02-30T12:00:00Z'],
    'amount': [10, 20, 30]
})
events['time'] = pd.to_datetime(
    events['time_text'], format='%Y-%m-%dT%H:%M:%SZ',
    errors='coerce', utc=True)
valid = events.loc[events['time'].notna()].copy()
daily = valid.set_index('time')['amount'].resample('D').sum(
    min_count=1)
print(events['time'].isna().sum())
print(daily)
```

Результат: Число NaT: 1; дневные суммы UTC: 10 и 20.

## Контроль

- Один NaT: несуществующее 30 февраля.
- Сохранены две корректные учебные записи.

## Дневной отчет

- 28 февраля: 10; 29 февраля: 20.
- Пустое окно не объявляем нулем автоматически.

# Ошибки времени и проверочные случаи

| Ошибка | Последствие | Контрольный случай |
|---|---|---|
| %m вместо %M | Месяц вместо минут | Известные компоненты |
| seconds вместо total_seconds | Потеря целых дней | Интервал более суток |
| Назначить пояс вместо перевода | Другой момент | Обратный перевод в UTC |
| Перепутать s и ms | Другая дата | Известная метка и диапазон |
| Включить обе границы | Двойной учет | Событие ровно в полночь |
| Потерять NaT до счетчика | Нарушение баланса | Вход = корректные + карантин |

# Основные результаты лекции

1. Строка, момент времени и длительность имеют разный смысл.
2. Явный формат предотвращает неоднозначное угадывание даты.
3. total_seconds учитывает полную длительность, включая дни.
4. Перевод пояса сохраняет момент, но может изменить отчетную дату.
5. NaT и границы окон проверяются до дневной агрегации.

# Контрольные вопросы и самостоятельная работа

1. Чем strptime отличается от strftime?
2. Почему %H и %I нельзя выбирать по одному утреннему примеру?
3. Что теряется при использовании timedelta.seconds?
4. Чем назначение пояса отличается от перевода момента?
5. Почему [start, end) удобно для соседних окон?

**Самостоятельная работа**

Создайте два события по разные стороны полуночи UTC. Переведите в фиксированный +05:00, сравните дневные частоты и докажите сохранение моментов.

# Источники и документация

## Документация

### 1. Python: datetime

https://docs.python.org/3/library/datetime.html

### 2. pandas: временные ряды

https://pandas.pydata.org/docs/user_guide/timeseries.html

### 3. pandas: интервалы времени

https://pandas.pydata.org/docs/user_guide/timedeltas.html

## Основа лекции

- Лекция 10 Работа с датами+.pptx
- Лабораторная работа 10.docx