# Конспект по теме "Изменение типов данных"

## Как читать файлы из Excel

Для прочтения файлов Excel есть особый метод `read_excel()`. Он похож на `read_csv()`, но в отличие от него, `read_excel()` нужно два аргумента: строка с именем самого файла или пути к нему, и имя листа `sheet_name`. Если аргумент `sheet_name` пропущен, то по умолчанию прочитается первый по счёту лист

```
import pandas as pd
df = pd.read_excel('file.xlsx', sheet_name='List1')
```

## Перевод строковых значений в числа

Для того, чтобы перевести строковые значения в числа, есть стандартный метод Pandas — **to_numeric()**. Он превращает значения столбца в числовые типы *float64* (вещественное число) или *int64* (целое число) в зависимости от исходного значения.

У метода *to_numeric()* есть параметр **errors**. От значений, принимаемых *errors*, зависят действия *to_numeric* при встрече с некорректным значением:

- `errors='raise'` — дефолтное поведение: при встрече с некорректным значением выдается ошибка, операция перевода в числа прерывается;
- `errors='coerce'` — некорректные значения *принудительно* заменяются на NaN;
- `errors='ignore'` — некорректные значения *игнорируются*, но остаются.

Для того, чтобы перевести данные в нужный тип, применяется метод `astype()`. В качестве аргумента передаётся строка с названием типа.

# Методы Pandas для работы с датой и временем

Для работы с датой и временем в Python существует особый тип данных — **datetime**.

Чтобы перевести строку в дату и время, используется метод **to_datetime()**. Параметрами метода являются: столбец, содержащий строки, и формат даты в строке.

Формат даты задаётся с помощью специальной системы обозначений, где:

- %d — день месяца (от 01 до 31)
- %m — номер месяца (от 01 до 12)
- %Y — год с указанием столетия (например, 2019)
- %H — номер часа в 24-часовом формате
- %I — номер часа в 12-часовом формате
- %M - минуты (от 00 до 59)
- %S - секунды (от 00 до 59)

```
date['column']= pd.to_datetime(date['column'], format='%d.%m.%YZ%H:%M:%S')
```

Среди самых разнообразных способов представления даты и времени особое место занимает формат **unix time**. Его идея проста — это количество секунд, прошедших с 00:00:00 1 января 1970 года. *Unix*-время соответствует Всемирному координированному времени, или *UTC*.

Метод `to_datetime()` работает и с форматом *unix time*. Первый аргумент — это столбец со временем в формате unix time, второй аргумент `unit` со значением `'s'` сообщит о том, что нужно перевести время в привычный формат нужно с точностью до секунды.

Часто приходится исследовать статистику по месяцам, дням, годам. Чтобы осуществить такой расчёт, нужно поместить время в класс *DatetimeIndex* и применить к нему атрибут *month, day, year*:

```
date['column'] = pd.DatetimeIndex(date['column']).month
```

# Обработка ошибок с помощью try-except

Выгружая данные из разных систем, нужно быть готовым к следующим трудностям:

- Некорректный формат приводит к невыполнению кода. Говорят, что *«код падает с ошибкой»*.
- Ошибки в данных встречаются ближе к концу файла, и код на строках с неверными значениями не выполняется. Значит, пропадают расчёты для предыдущих, правильных строк;
- Данные могут поменяться.

К сожалению, предсказать все потенциальные ошибки невозможно. Для работы с непредсказуемым поведением данных есть конструкция **try-except**. Принцип работы такой: исходный код помещают в блок *try*. Если при выполнении кода из блока *try* возникнет ошибка, воспроизведётся код из блока *except*.

```
try:
  # код, где может быть ошибка
except:
  # действия если возникла ошибка
```

# Метод merge()

Данные хранятся в excel-таблице из нескольких листов. Для того, чтобы использовать данные на всех листах, нужно склеить таблицы.

Объединить несколько таблиц в одну поможет метод `merge()`.

Аргументы:

- *right* - имя DataFrame или Series, который нужно присоединить к исходной таблице
- *on* - общее поле в двух таблицам, по которым происходит соединение
- *how* - какие *id* включены в итоговую таблицу. Может принимать значения *left* - id из левой таблицы будут включены в итоговую таблицу или *right* - id из правой таблицы будут включены в итоговую таблицу.

# Сводные таблицы

Сводная таблица - это ваш помощник для обобщения данных и их наглядного представления.

В Pandas для подготовки сводных таблиц есть метод `pivot_table()`

Аргументы метода:

- *index* - столбец или столбцы, по которым происходит группировка данных
- *columns* - столбец по значениям которого будет происходить группировка
- *values* - значения, по которым мы хотим увидеть сводную таблицу
- *aggfunc* - функция, которая будет применяться к значениям