Работа со Временем

Даты и Время в Питоне

Единица дат

Единица времени

Дата и время (Datetime)

- Сочетает в себе дату (**Date**), время (**Time**), а также опционально часовой пояс (**Timezone**)
- Стандарт **ISO 8601** предлагает формат для кодирования даты и времени
- Пример: **2018-10-13T15:53:20**

Работа с модулем Питона **datetime**

```
from datetime import datetime, date


new_date = datetime(year=2018, month=10, day=13)
new_date.year
new_date.month
new_date.hour
```

Работа с модулем Питона **dateutil**

```
from dateutil import parser

new_date = parser.parse("13th October 2018")
parser.parse("2018-10-13T15:53:20")
```

Практика

python_datetime.ipynb

Поддержка дат и времени в Numpy

- более эффективное представление в памяти
- особенно важно для *списков* дат

- см. **`scripts/datetime_sizer.py`**

Работа с типом **`datetime64`** в Numpy (1/2)

```
import numpy as np
import datetime

np.datetime64("2018-11-03")
np.datetime64("2018-10-03 12:00")
np.array(['2018-11-02', '2018-10-02', '2015-11-03'], dtype='datetime64')
current = np.datetime64(datetime.datetime.now())
```

Работа с типом **`datetime64`** в Numpy (2/2)

```
import pandas as pd


pd.to_datetime(current).year
np.datetime64("2018-11-03") - np.datetime64("2018-11-01")
np.datetime64('2018-11-03') + np.timedelta64(14, 'D')
np.datetime64('2018-10-03 12:00') + np.timedelta64(6, 'h')
np.datetime64('2018-11-03') + np.arange(10)
```

Практика

numpy_datetime.ipynb

Поддержка дат и времени в Pandas

- сочетает в себе простоту использования объектов **`datetime`** и **`dateutil`** Питона (например, методы и атрибуты доступа)

- эффективное представление в памяти и манипуляция при помощи **`numpy`**

- интеграция с **`Dataframe`** в Pandas

Основные классы для дат и времени в Pandas

Название класса	Описание
`Timestamp`	Представляет собой дату и время (точку во времени)
`DatetimeIndex`	Индекс `Timestamp`
`Period`	Представляет собой временной интервал (период времени, интервал с заданным ритмом повторений)
`PeriodIndex`	Индекс Period

Работа с датами и временем в Pandas

```
import pandas as pd

pd.to_datetime("14th of October, 2018")
pd.Timestamp(year=2018, month=10, day=14, hour=12, minute=0, second=30)
datetimes = pd.DatetimeIndex(['2014-07-04', '2014-08-04',
                              '2015-07-04', '2015-08-04'])
series = pd.Series([10, 4, 14, 30], index=datetimes)
series['2015']
pd.date_range('2015-07-03', '2015-07-10')
pd.date_range('2018 Oct 1', periods = 10, freq = 'W')
```

Работа с временными дельтами (**`Timedelta`**)

- **`Timedelta`** представляет собой временной интервал между двумя точками во времени (**`Timestamp`**)

- объект **`Timedelta`** есть результат некоторых арифметических операции над объектами класса **`Timestamp`**

- объекты класса **`Timestamp`** также можно создать с нуля

Работа с Timedelta в Pandas (1/2)

```
import pandas as pd


diff = pd.Timestamp("2019-07-07") - pd.Timestamp("2019-07-05")
diff.components
    > Components(days=2, hours=0, minutes=0, seconds=0, milliseconds=0,
    microseconds=0, nanoseconds=0)

diff.days
    > 2
```

Работа с Timedelta в Pandas (2/2)

```
import pandas as pd


diff = pd.Timestamp("2019-07-07") + Timedelta(days=10, minutes=10)
diff
    > Timestamp('2019-07-17 00:10:00')
```

Работа с часовыми поясами (timezones)

- в конструкторе объекта **`Timestamp`** можно указать часовой пояс (timezone) при помощи параметра **`tz="..."`**

- список всех часовых поясов можно получить из модуля **`pytz`**

```
import pytz

pytz.all_timezones
    > ['Africa/Abidjan', 'Africa/Accra', 'Africa/Addis_Ababa',
       'Africa/Algiers', 'Africa/Asmara', 'Africa/Asmera',
       'Africa/Bamako',  'Africa/Bangui', 'Africa/Banjul',
       'Africa/Bissau',  'Africa/Blantyre', 'Africa/Brazzaville',
       'Africa/Bujumbura' ...
```

Работа с часовыми поясами (timezones) в Pandas

```
import pandas as pd

ts = pd.Timestamp("2019-07-07 11:00", tz="Europe/Paris")

ts.tzinfo
ts.tz_convert("Europe/London")
    > Timestamp('2019-07-07 10:00:00+0100', tz='Europe/London')

ts = pd.Timestamp("2019-07-07 11:00")
ts.tz_localize("Chile/Continental")
```

Практика

pandas_datetime.ipynb

Выводы

- **`datetime` в Питоне**
  - поддержка «из коробки»
  - неэффективно для работы с временными рядами в анализе данных
- **`datetime64` в Numpy**
  - эффективное представление
  - ограниченный набор операций
- **Pandas**
  - эффективное представление в памяти
  - интеграция с функциями Pandas
  - простые методы доступа
  - временной ряд может быть представлен в Pandas объектом Series, в котором индексом являются точки во времени