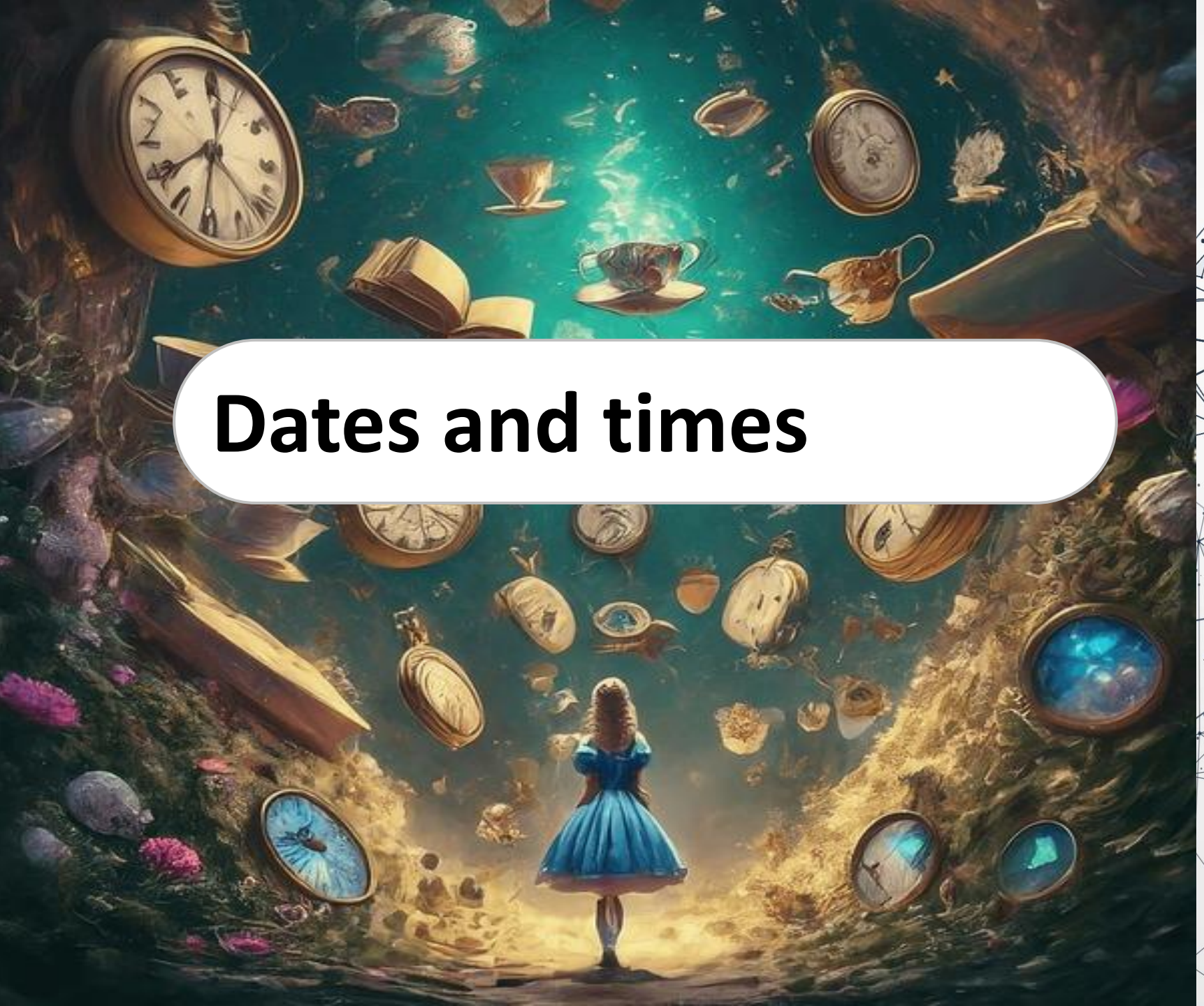


Dates and times

הטכניון
היחידה ללימודי המשך
ולימודי חוץ ע"ש עזריאלי

Python datatime
Pandas Timestamp
To-datetime
time_range

הטכניון
היחידה ללימודי המשך
ולימודי חוץ ע"ש עזריאלי

TECHNION
Azrieli Continuing Education and
External Studies Division

Python's datetime Module

Purpose:

Provides classes for manipulating dates and times.

Key Features:

- **datetime.date**: Represents a calendar date (year, month, day).
- **datetime.datetime**: Represents a date and time.
- Access individual components (e.g., year, month, day).

TECHNION
Azrieli Continuing Education and External Studies Division

Python's datetime Module

```python
import datetime as dt
dt.date(2022, 1, 1)  # Creates a date object
dt.datetime(2022, 1, 1, 15, 30)  # Includes time
now = dt.datetime.now()  # Current date and time
```

Advantages:

- Handles basic date/time manipulations.
- Compatible with standard Python libraries.

TECHNION
Azrieli Continuing Education and External Studies Division

Timestamp

pd.Timestamp: A pandas equivalent of datetime.

- Handles flexible date formats.
- Integrates seamlessly with pandas DataFrames and Series.

Advantages:

- Simplifies date/time operations.
- Supports advanced indexing and slicing.

```
pd.Timestamp("2024-01-01")  # Creates a timestamp
pd.Timestamp("3/3/2020 15:00:01")  # Flexible parsing
```

TECHNION
Azrieli Continuing Education and External Studies Division

Creating Date Ranges with pd.date_range

Purpose:

Generate sequences of dates.

Key Parameters:

start, end: Define the range.

freq: Frequency of dates (e.g., daily, monthly).

periods: Number of dates.

```
pd.date_range(start="2024-01-01", periods=7, freq="D")
```

Use Cases:

- Scheduling events.
- Creating time-based indexes.

to_datetime

Purpose: Convert strings or objects to datetime.

Syntax Options:

pd.to_datetime: Handles ambiguous formats.

errors: Control error handling (coerce, ignore).

```
pd.to_datetime(arg ,errors)
```

to_datetime

Advantages:

- Simplifies working with inconsistent data.
- Ensures uniform datetime representation.

Code example

```
pd.to_datetime("2024-12-19")  # Standard format
pd.to_datetime(["12/19/2024", "31/10/1983"], dayfirst=True)
```

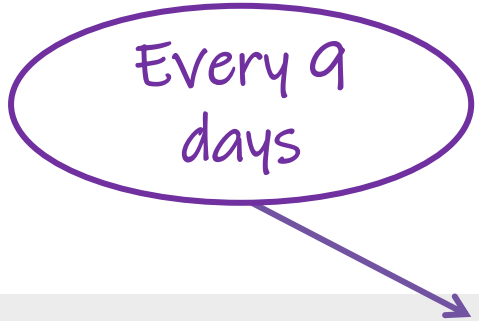
to_datetime

Custom Frequencies:

- **D**: Daily.
- **W**: Weekly.
- **M:** Month-end
- **Q**: quarter-end.

Use Cases:

- Business-specific scheduling.
- Custom time intervals for analysis.




```
pd.date_range("2024-12-13", "2024-12-31", freq="9D")
```

Boolean Indexing with Dates

Advantages:

- Easily filter data by date.
- Combine with other pandas operations.

Use Cases:

- Tracking task completion.
- Filtering rows by specific dates.

```
date_range = pd.date_range("2024-01-01", "2024-01-31")
completed = [True, False, True, False, True]
date_series = pd.Series(completed, index=date_range)
```

TECHNION
Azrieli Continuing Education and External Studies Division

Error Handling in to_datetime

Common Issues:

- Invalid dates (e.g., "2024-05-35").
- Ambiguous formats.

Solutions:

- errors='coerce': Converts invalid dates to NaT.
- errors='ignore': Leaves invalid entries as strings.

```
dates = ["2024-05-35", "2024-12-19"]
pd.to_datetime(dates, errors="coerce")
```

Invalid becomes NaT

TECHNION
Azrieli Continuing Education and External Studies Division

Summary

Key Concepts:

- Basic date manipulation with datetime.
- Advanced date handling with pandas.
- Generating, converting, and analyzing date ranges.

Advantages:

- Simplifies date/time operations.
- Powerful tools for real-world scenarios.