Data Science Technologies I

Dominik Matula

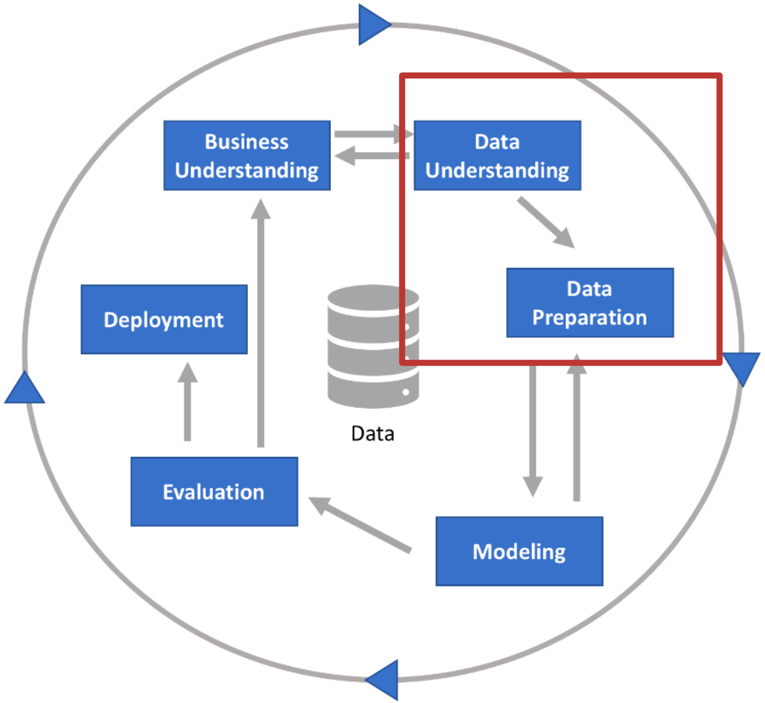
September 2023

Outline

1. A tour of Python ecosystem
2. How to store data
3. How to read and transform data using python

Data science process

PROFINIT



Getting Python

- **Python.org**
  - https://www.python.org/downloads/

- **Anaconda**
  - https://www.anaconda.com/products/individual

PROFINIT

Developing in python

PROFINIT

- Jupyter
- **VSCode**
  - + Jupyter, +Quarto extensions

- Some alternatives
  - PyCharmPro (free for students)
  - Jetbrains DataSpell (EAP) - Jupyter on steroids
  - Spider

VSCode cool things

PROFINIT

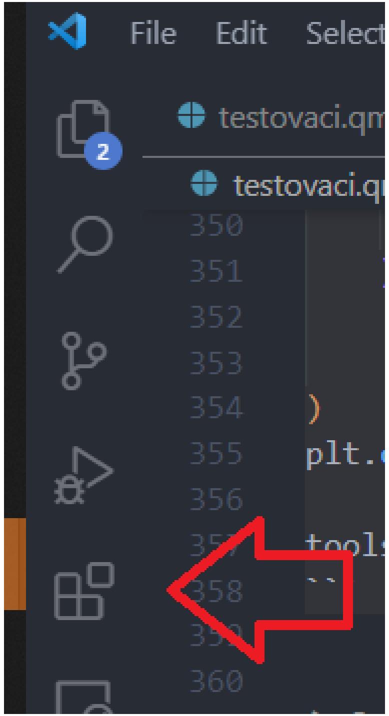
- Devtools integration:
  - git, docker, lint
- Quite smart python autocompletion
  - plus Copilot/Tabnine integration
- Interactive debugger
- Database plugin
- Unit-tests runner
- Remote development
- Quarto integration

Note: available in PyCharm as well except Quarto integration

VSCode setup

PROFINIT

- Get & install VSCode:
  - code.visualstudio.com/Download
- Setup project dir
- Install extensions
  - Python
  - Jupyter
  - Quarto

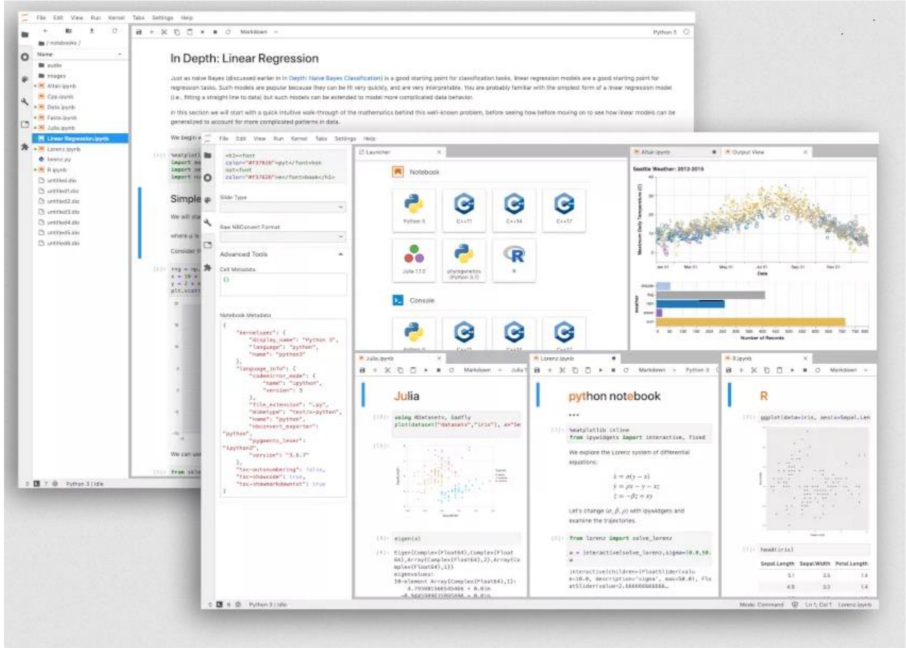


Note: You can use your favourite IDE, of course

Jupyter

PROFINIT

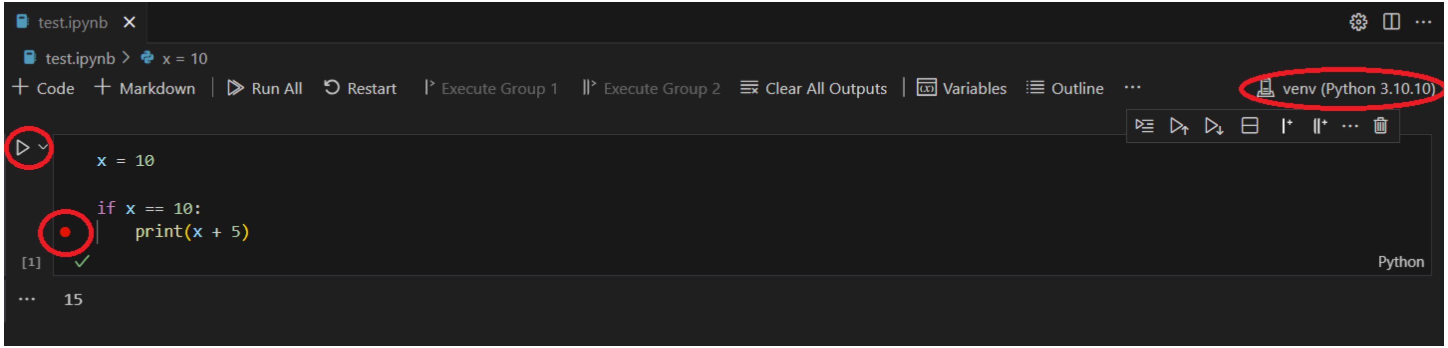
- › Web-based IDE
- › Report oriented
  - – But handles .py etc. as well
  - – Code & Markdown cells
- › Install & run
  - – `pip install jupyterlab`
  - – `jupyter lab`
- › Key shortcuts
  - – Esc; (A)bove, (B)elow, (DDelete)
  - – Make cell (M)arkdown, (Y)code
  - – (Ctrl + enter) to run



Debugging a Jupyter notebook

PROFINIT

- Open Jupyter notebook file
- Select Kernel
  = Python that executes the notebook
- Click on breakpoint
- Run cell with Debug



Jupyter nbconvert

PROFINIT

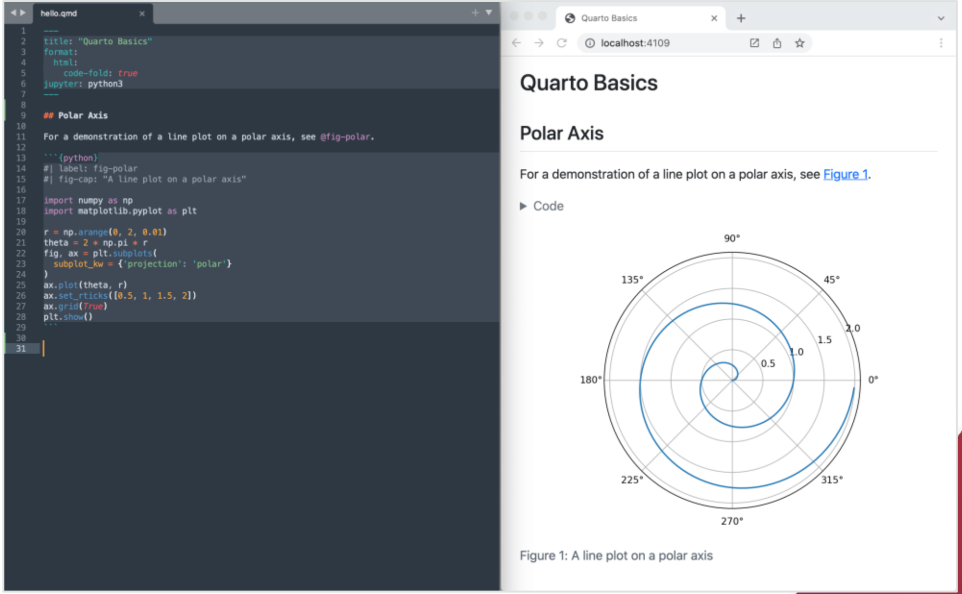
- Do not share your resutls as a source code
- `jupyter nbconvert <yourntb.ipynb>`
  - + hide unnecessary code cells (--no-input)
  - TIP: pretty-jupyter package

- **TIP**: Quarto can do that for you as well!
  - + shortcuts in VSCode (ctrl + shift + K)

Quarto

- "An open-source scientific and technical publishing system"
  - It's like Jupyter, but better :)
  - Based on Pandoc
  - One input, many outputs

1. Download + install Quarto
   - quarto.org/docs/get-started
2. Install VSCode extension
   - No PyCharm extension yet :(



Python toolkit [1]

PROFINIT

- **Data manipulation**
  - **Pandas**
  - Apache Spark (pyspark)

- **Data visualization**
  - matplotlib
  - **seaborn**
  - plotly

Python toolkit [2]

PROFINIT

- **Stat and Machine learning**
  - statsmodels
  - scikit-learn (and scikit-* family)
  - prophet (time-series data)
  - h2O
- **Gradient boosting**
  - catboost
  - lightgbm
  - xgboost
- **Neural networks**
  - tensorflow
  - pytorch

Python toolkit [3]

PROFINIT

- **NLP**
  - nltk
  - spacy
  - gensim
- **Vision**
  - opencv (cv2)
  - scikit-image
- **Graphs**
  - networkX
  - snap-stanford

Data Sources

Pandas I/O

PROFINIT

- **Files**
  - json, csv, tsv `pd.read_csv()` / `pd.read_json()`
  - parquet (pyarrow / fastparquet) `pd.read_parquet()`
  - excel (openpyxl) `pd.read_excel()`
  - pickle
- **Database**
  - Sqlalchemy create_engine (pyodbc string) + pandas
  - e.g.: `pd.read_sql_table(tabname, con=engine)` / `pd.read_sql_query()`
- **Writing**
  - to_*() (csv, json, parquet, excel)
  - use compression to reduce file-size and speedup IO (.gz, .zip)

Pandas

Basics

PROFINIT

- Series
  - 1d labeled array, may contain mixed data types
- DataFrame
  - 2d array, aka table
- Index
  - aka primary key for a row (without UNIQUE constraint)
- Columns
- Axis
  - 0 – columns (column-wise)
  - 1 – index (row-wise)

DataFrame

PROFINIT

df.index

index labels

df.columns

column names

pd.Series

	Mountain	Height (m)	Range	Coordinates	Parent mountain	First ascent	Ascents bef. 2004	Failed attempts bef. 2004
0	Mount Everest / Sagarmatha / Chomolungma	8848	Mahalangur Himalaya	27°59′17″N 86°55′31″E	NaN	1953	>>145	121.0
1	K2 / Qogir / Godwin Austen	8611	Baltoro Karakoram	35°52′53″N 76°30′48″E	Mount Everest	1954	45	44.0
2	Kangchenjunga	8586	Kangchenjunga Himalaya	27°42′12″N 88°08′51″E	Mount Everest	1955	38	24.0
3	Lhotse	8516	Mahalangur Himalaya	27°57′42″N 86°55′59″E	Mount Everest	1956	26	26.0
4	Makalu	8485	Mahalangur Himalaya	27°53′23″N 87°05′20″E	Mount Everest	1955	45	52.0
5	Cho Oyu	8188	Mahalangur Himalaya	28°05′39″N 86°39′39″E	Mount Everest	1954	79	28.0
6	Dhaulagiri I	8167	Dhaulagiri Himalaya	28°41′48″N 83°29′35″E	K2	1960	51	39.0
7	Manaslu	8163	Manaslu Himalaya	28°33′00″N 84°33′35″E	Cho Oyu	1956	49	45.0
8	Nanga Parbat	8126	Nanga Parbat Himalaya	35°14′14″N 74°35′21″E	Dhaulagiri	1953	52	67.0
9	Annapurna I	8091	Annapurna Himalaya	28°35′44″N 83°49′13″E	Cho Oyu	1950	36	47.0

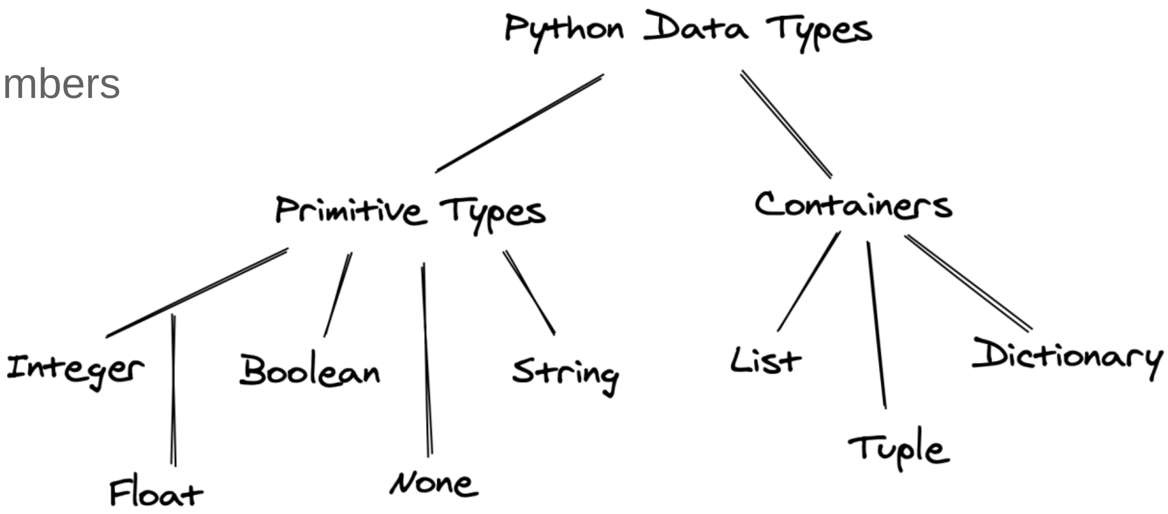
data

df.loc[2, “Mountain”]

Data types

PROFINIT

- Numbers
  - Integers, floating-point numbers
- Booleans
- Dates
- Categories
- Text



Hints:

- df.dtypes, df.select_dtypes

Getting the right data

PROFINIT

- **By columns**
  - A single column `df[“Age”]  --> pd.Series`
  - Many columns `df[[“Age”, “Sex”]]  --> pd.DataFrame`
  - By index of a column `df.iloc[:, 0], df.iloc[:, [2,3]], df.iloc[:, 2:6]`
- **By rows**
  - By index value `df.loc[0], df.loc[0:5]`
  - By integer `df.iloc[0], df.iloc[0:3]`
- **Subset**
  - By condition `df[df[“Age”] >= 30]`
  - By multiple conditions `df[(df[“Age”] >= 30) & (df[“Sex”] == “female”)]`
  - Boolean indexing operators `&(and) |(or) ~(not)`

DataFrame basic operations and attributes

PROFINIT

- `df.head(), df.tail(), df.sample(n=12)` - a quick glimps at data
- `df.columns, df.shape` - data dim

- `df.count()`
- `df.describe()` - summary stats for numeric columns
- `df[“Sex”].value_counts()` - frequency table

- `df.sort_index()` or `df.sort_values(“column”, ascending=True)`
  - You can sort by multiple columns – `df.sort_values(["a", "b"])`

- `df[“val”].astype(“newtype”)` - change dtype

DataFrame basic operations, continued

PROFINIT

- `df.drop(labels, axis="columns")`
- `df.drop_duplicates()`
  - You can specify which columns check for duplicities via '`subset`'

- `df.rename({“A”: “B”}, axis="columns")` - rename
- `df[“Sex”].map({“male”: 0, “female”: 1})` - relevel
- `df["Age"].replace(0, 999)` - replace

- `pd.cut(df[“Age”], bins=[17, 21, 35, 45, 100])`

DataFrame piping

PROFINIT

- ```
  df2["is_male"] = df2["gender"] == 'male'
  male_proportion_by_class = df2.groupby("pclass").agg({"is_male": "mean"})
  male_proportion_by_class.sort_values().head(1)
  ```

- ```
  (
    df
    .assign(is_male = lambda d: d["sex"] == 'male')
    .groupby("pclass")
    .agg({"is_male": "mean")
    .sort_values()
    .head(1)
  )
  ```

Missing data

PROFINIT

Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
male	19.0	0	0	349212	7.8958	NaN	S
male	NaN	8	2	CA. 2343	69.5500	NaN	S
male	21.0	0	0	A/5. 13032	7.7333	NaN	Q
male	50.0	0	0	250643	13.0000	NaN	S
male	17.0	0	0	315090	8.6625	NaN	S
male	NaN	0	0	374910	8.0500	NaN	S
female	47.0	1	1	11751	52.5542	D35	S
male	NaN	0	0	2700	7.2292	NaN	C
male	26.0	1	0	350025	7.8542	NaN	S
male	NaN	0	0	12460	7.7500	NaN	Q

Missing data

PROFINIT

- Typical source of missing data
  - **Missing completly at random**
    - errors during data collection or data processing, in a non systematic way
  - **Missing at random**
    - Missings caused by known facts only (e.g., not having a wife --> unknown wife's age)
  - **Missing not at random**
    - Missings caused by unkown variables, too (e.g., rich people not motivated enough to fill a poll --> bias)
- Typical strategies to deal with missing data
  - Drop column `df.drop(columns=["colA"])`
  - Drop rows `df.dropna(subset=["colA"])`
  - Impute with constant (mean, mode, 0), e.g.: `df.fillna({"colA": 0})`
  - Impute with a model

Missing data in pandas

PROFINIT

- `None` – python general representation of missing value
- `Np.nan` – Numpy’s NaN is usually used in pandas

- ! By default, NaNs are excluded from aggregate functions
- To check whether a value is missing
  - `df["age"].isna()` or `df["age"].notna()`
- We can drop rows with missing values
  - `df.dropna(subset=["age"])`
- We can fill missing values with a constant
  - `df["age"].fillna(val)`

Summary tables

PROFINIT

› How to produce a summary table for two or more categorical variables?

– `pd.cross_table` – frequency table

– `df.pivot_table` – any aggregation fn

```
pd.crosstab(
    df['Sex'],
    df['Survived']
)
```

Survived Sex	0	1
female	81	233
male	468	109

```
pd.crosstab(
    df['Sex'],
    df['Survived'],
    normalize='index'
)
```

Survived Sex	0	1
female	0.26	0.74
male	0.81	0.19

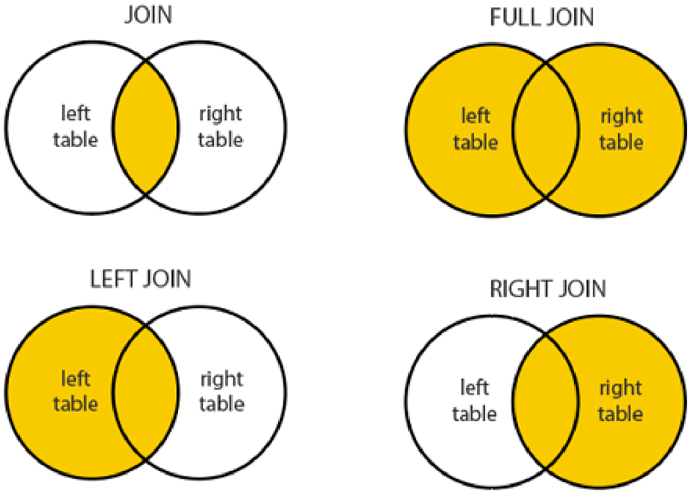
```
df.pivot_table(index='Pclass',
               columns='Sex',
               values='Age',
               aggfunc='mean')
```

Sex Pclass	female	male
1	34.61	41.28
2	28.72	30.74
3	21.75	26.51

Combining tables

PROFINIT

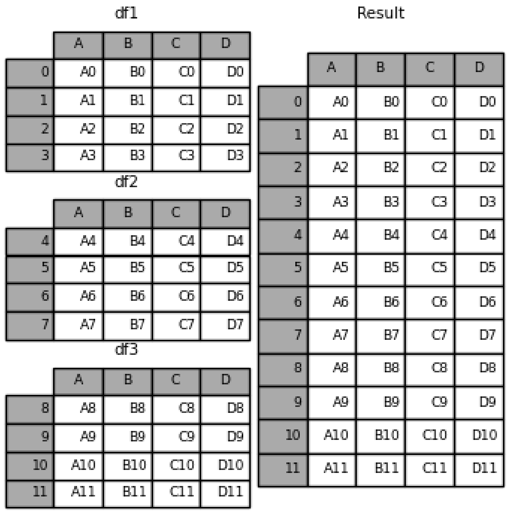
- Database like join – `df.merge`.
- Keyword argument `how` define the type of the join
  - Left
  - Right
  - Outer
  - Inner
  - Cross



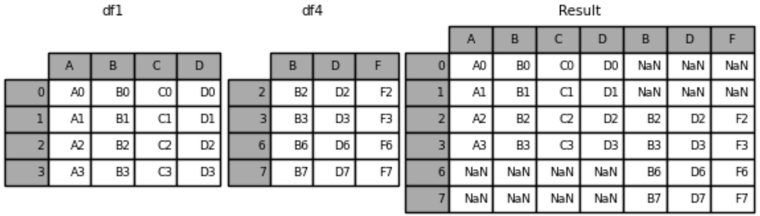
Hint: pandas.pydata.org/docs/reference/api/pandas.merge.html

Combining dataframes

- We can also glue dataframes together by rows/columns using `pd.concat`



`pd.concat([df1, df2, df3])`



`pd.concat([df1, df2], axis="column")`

Hint: pandas.pydata.org/docs/reference/api/pandas.concat.html

Aggregations

PROFINIT

- Split a data into groups and compute summary statistics for each group
- ```
  (
     df
     .groupby([group_key, another_one])
     .agg(example=(“column_name”, “agg_function”))
  )
  ```
- Aggregate functions
  - Min, max, average, nunique, sum, size, count, var, sem, descrie
  - First, last, nth
  ```

Visualizations

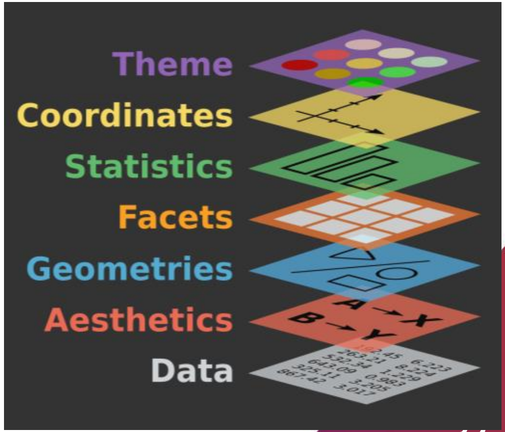
PROFINIT

Will be covered in next lessons.

matplotlib

seaborn

seaborn
Objects API



Visualizations

PROFINIT

Will be covered in next lessons.

PROFINIT

Profinit EU, s.r.o., Tychonova 2, 160 00 Praha 6
Tel.: + 420 224 316 016, web: www.profinit.eu