Effective Programming Practices for Economists

Software engineering

Which errors to handle?

Janoś Gabler and Hans-Martin von Gaudecker

Reminder of Example

```
def create_markdown_table(data):
    """Create a markdown table from a list of dictionaries or a dictionary of lists.

    """
    if isinstance(data, dict):
        lod = convert_dol_to_lod(data)
    else:
        lod = data

    keys = list(lod[0])

    lines = [
        _create_header(keys),
        _create_separator(len(keys)),
    ]
```

Which errors to handle?

- If your function is correct the only source of errors is `data`
- To make sure your function is correct, testing is better than error handling
- So what could go wrong with `data`?
  - `data` is neither a list nor a dict
  - `data` is a dict but contains values that are not lists
  - `data` is a dict of lists but the lists have different lengths
  - `data` is a list, but contains entries that are not dicts
  - `data` is a list of dicts but the dicts have different keys

Goals

- Raise errors as early as possible
- Absolutely avoid duplicated code for error handling
- Try to avoid running checks repeatedly

Where to handle errors in the example?

- in `create_markdown_table`
  - `data` is neither a list nor a dict
- in `convert_dol_to_lod`:
  - `data` is a dict but contains values that are not lists
  - `data` is a dict of lists but the lists have different lengths
- in `create_markdown_table`, branch of if-statement that gets called if data is a list:
  - `data` is a list, but contains entries that are not dicts
  - `data` is a list of dicts but the dicts have different keys