Office Automatisierung mit Python

Image: Dall-E 3

Stefan Baerisch, stbaer.com, enterPy 2022

Warum Office Automatisierung? Warum Python

Automatisierung

- Die verschiedenen Office Formate sind allgegenwärtig.
  - Als Datenquellen
  - Als Arbeitsergebnisse
- Viele Tätigkeit wiederholen sich.

Python

- Python ist leichtgewichtig genug, um schnell Ergebnisse zu liefern
  - Speziell für interne Prozesse, die nützlich, nicht nötig sind.
  - Es gibt eine Reihe von hilfreichen Modulen zum Umfang mit Office Formaten und für Datenmanipulation, Analyse, etc.

Was ist Office Automatisierung?

- Formate:
  - Excel
  - Powerpoint
  - Docx
  - PDF
- Aufgaben:
  - Lesen: Extraktion von Texten oder strukturieren Daten
  - Schreiben: Erstellung von neuen Dokumenten
  - Ändern: Einfügen von Inhalten
  - Transformieren: Umwandeln in andere Formate

Ziel des Vortrags

- Überblick der Technik: Welche Module gibt es, was leisten sie
- Gefühl für die APIs: Speziell die Office Open XML Module spiegeln die Annahmen des Dokumentenformats.
- NICHT: Detailierte Darstellung konkreter Szenarien. Details werden schnell aufwendig und zeitraubend.

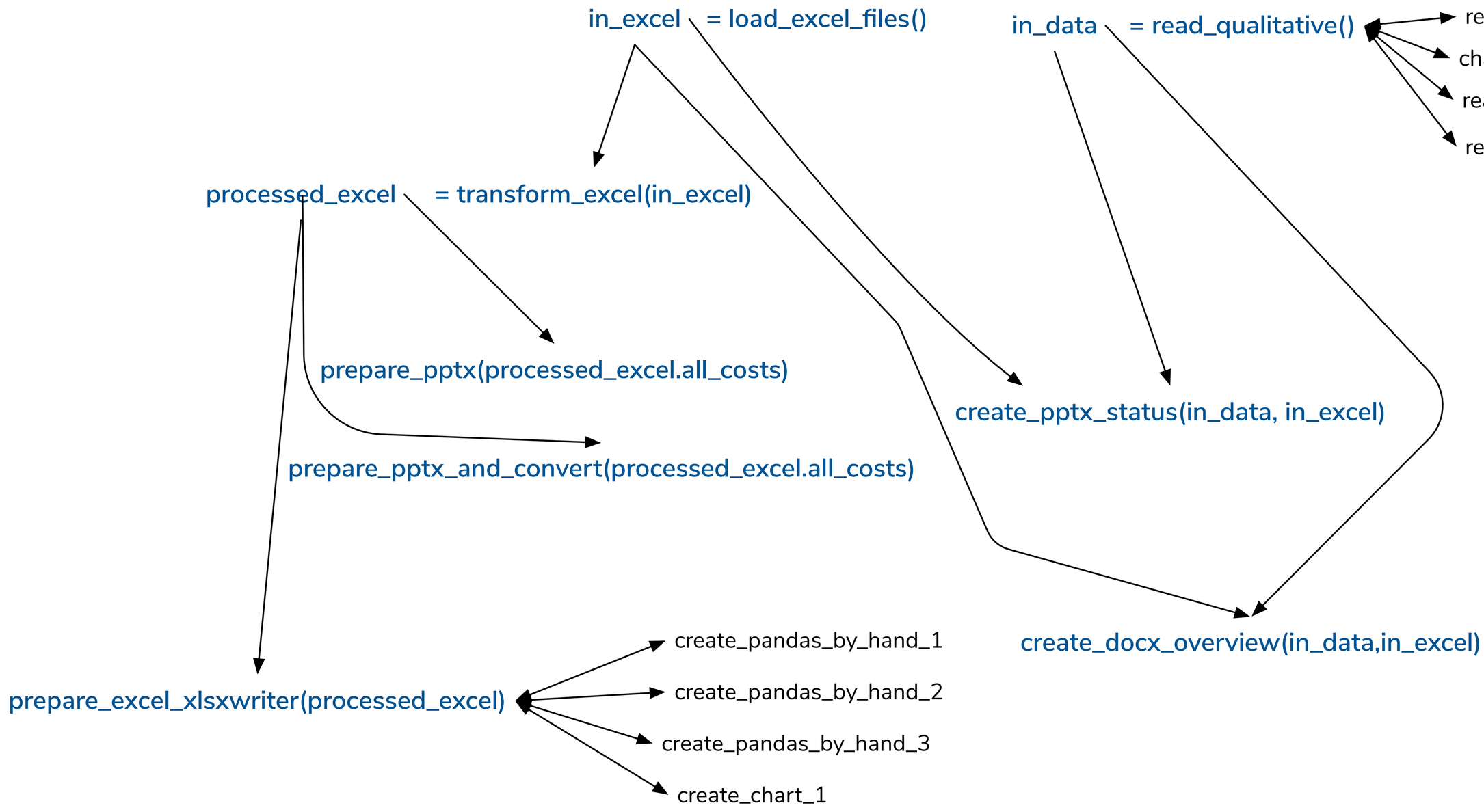
Python Module

	XLSX	DOCX	PPTX	PDF
Lesen / Datenextraktion	openpyxl / xlsxrw	python-docx	python-pptx	PyPDF
Erstellen	openpyxl / xlsxrw	python-docx	python-pptx	reportlab / LaTex / pandoc (*)
Ändern	openpyxl / xlsxrw	python-docx	python-pptx	
Transformieren	openoffice headless	openoffice headless	openoffice headless	pdfrw

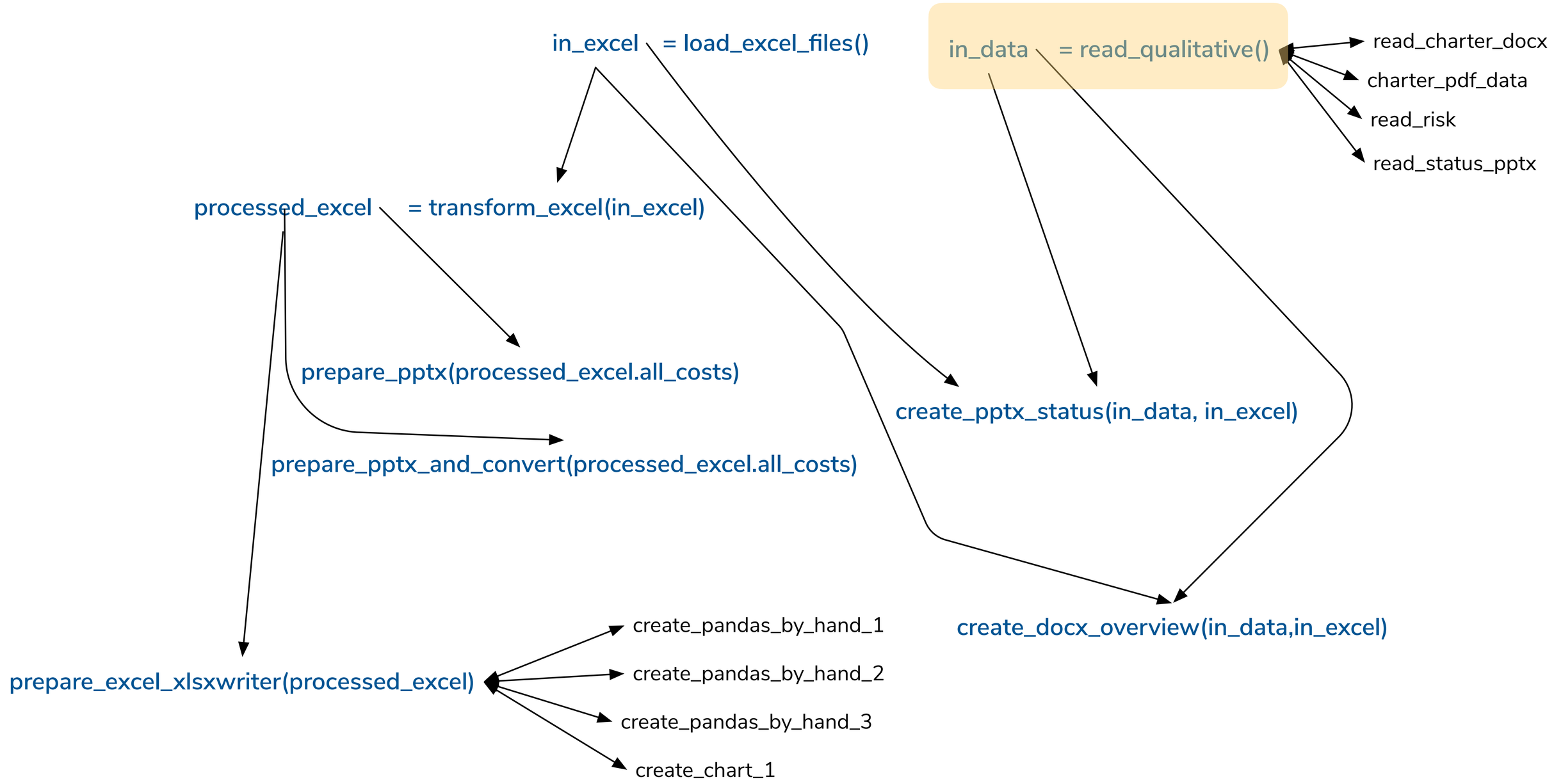
(*) hier nicht behandelt

Beispielprogramme

- Motto: Nicht schön, aber viel



Datenbereitstellung - Qualitative Daten



Datenbereitstellung - Qualitative Daten

```
def read_qualitative():
    charter_docx_data = read_charter_docx(in_fp("project_charter.docx"))
    charter_pdf_data = read_charter_pdf(in_fp("project_charter.pdf"))
    risk_data = read_risk(in_fp("project_riscs.xlsx", ))
    project_status_pptx = read_status_pptx(in_fp("project_status.pptx"))
    result = InData(charter_pdf_data, risk_data, project_status_pptx)

    return result
```

read_charter_docx() - Beispieldokument

Project Name: Kaffee kochen	Project Number: 3141
Date: 2022-10-07	Revision Number: 1

1. PROJECT GOALS

Der Kaffe in der Kantine muss besser werden.

2. DELIVERABLES

Espresso und Filterkaffee von akzeptabler Qualität mit weniger als 10% Kaffeesatz

3. SCOPE DEFINITION

Auswahl und Anschaffung einer Kaffeemaschine
Erwerb von 2Kg Arabica Kaffee
NICHT enthalten ist die Beschaffung von Sojamilch.

4. PROJECT MILESTONES

- *Kaffee kaufen*
- *Kaffee machen*
- *Kaffee trinken*

5. ASSUMPTIONS, CONSTRAINTS & DEPENDENCIES

Keine Teetrinker im Projekt.

6. RELATED DOCUMENTS

Reference any related documents that were used to define scope and assumptions – e.g., RFQ, RFP, Sales Proposal, etc.

7. PROJECT ORGANIZATIONAL STRUCTURE

Identify the key stakeholders and team members by function, name and role.

Function	Name	Role
Kaffeeröster	Katrin Kaffee	PM
Wasserentkalker	Klaus Kaffee	DEV

8. PROJECT AUTHORIZATION

Approved by:	Business Manager	Date
Approved by:	Project Manager	Date

read_charter_docx() - Code

```
def read_charter_docx(fp):
    doc_result = docx2python(fp).document
    po = doc_result[1][0][0][0]
    ms = doc_result[3][7][0]
    ms = [v.replace('--\t', '') for v in ms if v]
    return {
        'ms': ms,
        'po': po
    }
```

read_charter_pdf() - Code

```
def read_charter_pdf(fp):
    texts = []
    with open(fp, 'rb') as inf:
        pdf = PyPDF2.PdfFileReader(inf)
        for i in range(0, pdf.numPages):
            page = pdf.getPage(i)
            texts.append(page.extractText())
    texts = " ".join(texts)

    rx = r"Project Name:(.*?)Project Number:"
    po = re.search(rx, texts)
    po = po.group(1).strip()

    ms = texts.split("PROJECT MILESTONES ")[1].split("5.     ASSUMPTIONS, CONSTRAINTS & DEPENDENCIES")[0]
    ms = [v.strip() for v in ms.split(" • ") if v.strip()]
    return {
        'ms': ms,
        'po': po
    }
```

read_risk() - Beispieldokument

RISK MANAGEMENT LOG

Project Name:	Kaffee Kochen
Project Manager Name:	Ernst Espresso
Project Description:	Kaufen, Kochen, Trinken

ID	Current Status	Risk Impact	Probability of Occurrence	Risk Map	Risk Description	Project Impact
1	Open	High	Low	Yellow	**Unsere Kaffeemaschine explodiert**	**Wir müssen Tee trinken**
2	Open	Medium	Medium	Yellow	**Kaffee wird Teurer**	**Blümchenkaffee**

read_risk() - Code

```
def read_risk(fp):
    def get_valus(ws):
        result = []
        for row in ws.iter_rows(min_row=6, max_row=ws.max_row, min_col=2, max_col=6):
            row_result = []
            for cell in row:
                v = cell.value
                if v is None:
                    return result
                row_result.append(v)
            result.append(row_result)
        return result

    result = []
    wb = openpyxl.load_workbook(fp)
    ws = wb['Risk_Tracking_Log']

    vals = get_valus(ws)
    for val in vals:
        del val[3]
        result.append({k: v for k, v in zip(['status', 'impact', 'prob', 'text'], val)})
    return result
```

read_status_pptx() - Beispieldokument

Overall Project Status Report

UNIVERSITY OF TASMANIA

Project Name
dd/mm/yy

UTAS

Key achievements in the last [week]

- [Detail the key achievements in the last period]

Key actions planned for the next [two weeks]

- [Detail the key actions planned for the next period of work]

Key risks / issues / scope changes

- [Detail key risks and issues]

Project Metrics	Status
Budget	O1
Schedule	G2
Risk Management	G3
Stakeholder Management	R4

Overall Status: G

Project milestones *	Target date	Expected completion	Status
	dd/mm/yy	dd/mm/yy	✓
	dd/mm/yy	dd/mm/yy	G
	dd/mm/yy	dd/mm/yy	G
	dd/mm/yy	dd/mm/yy	G

Stakeholder Management Issues

- [Any key stakeholder issues]

Key Discussion Areas

- [Key discussion areas]

Legend: Major issues R Some issues O Satisfactory G

Internal document not for distribution

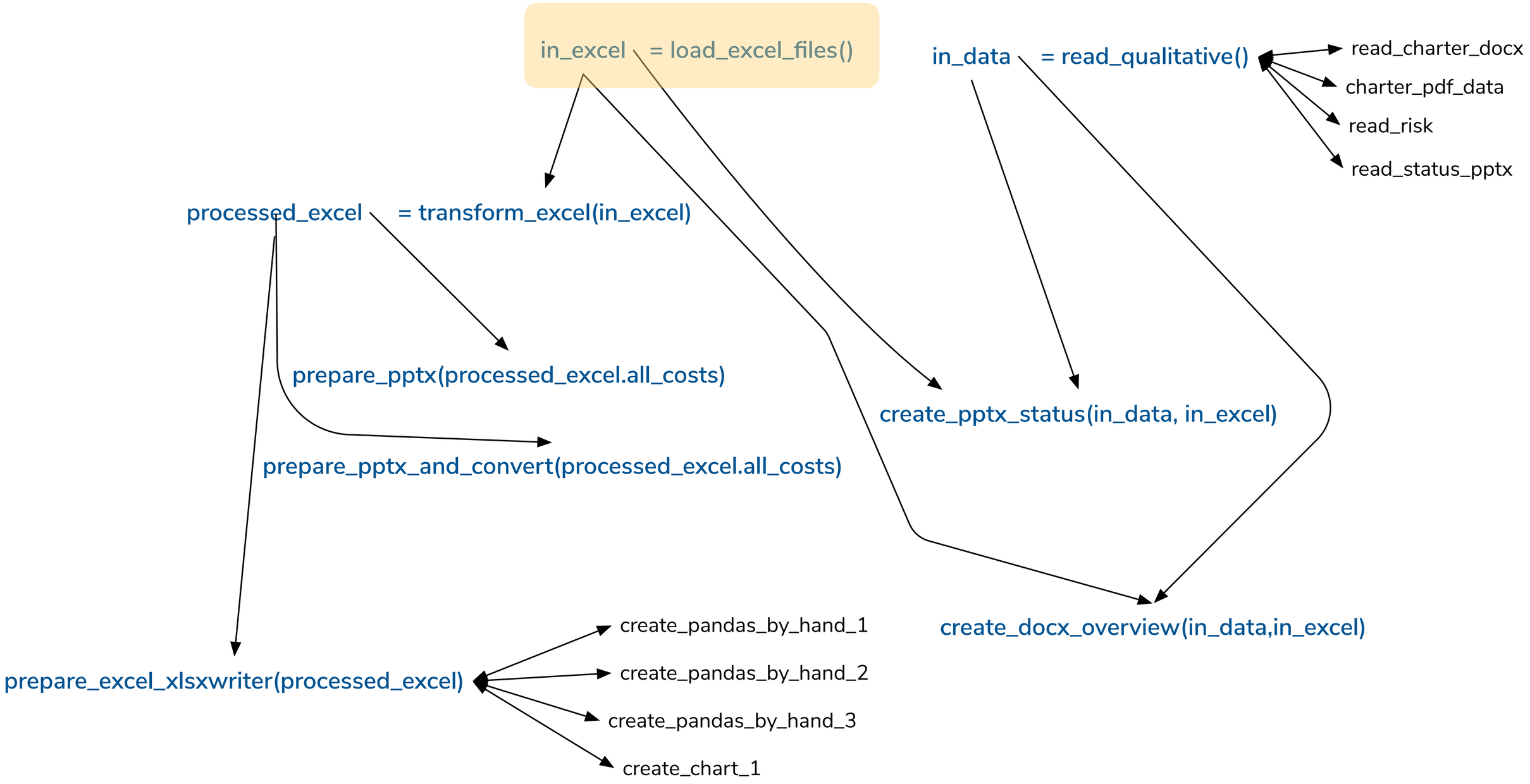
read_status_pptx() - Code

```
def read_status_pptx(fp):
    def get_color_name(rgp):...

    def get_status_shapes(shapes):...

    shapes = []
    prs = Presentation(fp)
    slide = prs.slides[0]
for shape in slide.shapes:
    if shape.has_text_frame:
        shapes.append([shape, pu.Emu(shape.left).cm, pu.Emu(shape.top).cm])

status_shapes = get_status_shapes(shapes)
return status_shapes
```

read_status_pptx() - Code (2)

```
def get_status_shapes(shapes):
    result = {}
    coords = [
        ['Budget', 19.495941666666667, 3.853822222222222],
        ['Risk', 19.479947222222222, 5.203886111111111],
        ['Stakeholder', 19.487525, 5.960566666666667],
        ['Schedule', 19.479936111111112, 4.492372222222222]
    ]

    for coord in coords:
        temp = []
        for shape in shapes:
            dist = math.sqrt(((shape[1] - coord[1]) ** 2) + ((shape[2] - coord[2]) ** 2))
            temp.append([shape, dist])
        best_match = sorted(temp, key=lambda v: v[1])[0]
        best_shape = best_match[0][0]
        result[coord[0]] = get_color_name(best_shape.fill.fore_color.rgb)
    return result
```

Datenbereitstellung - Qualitative Daten



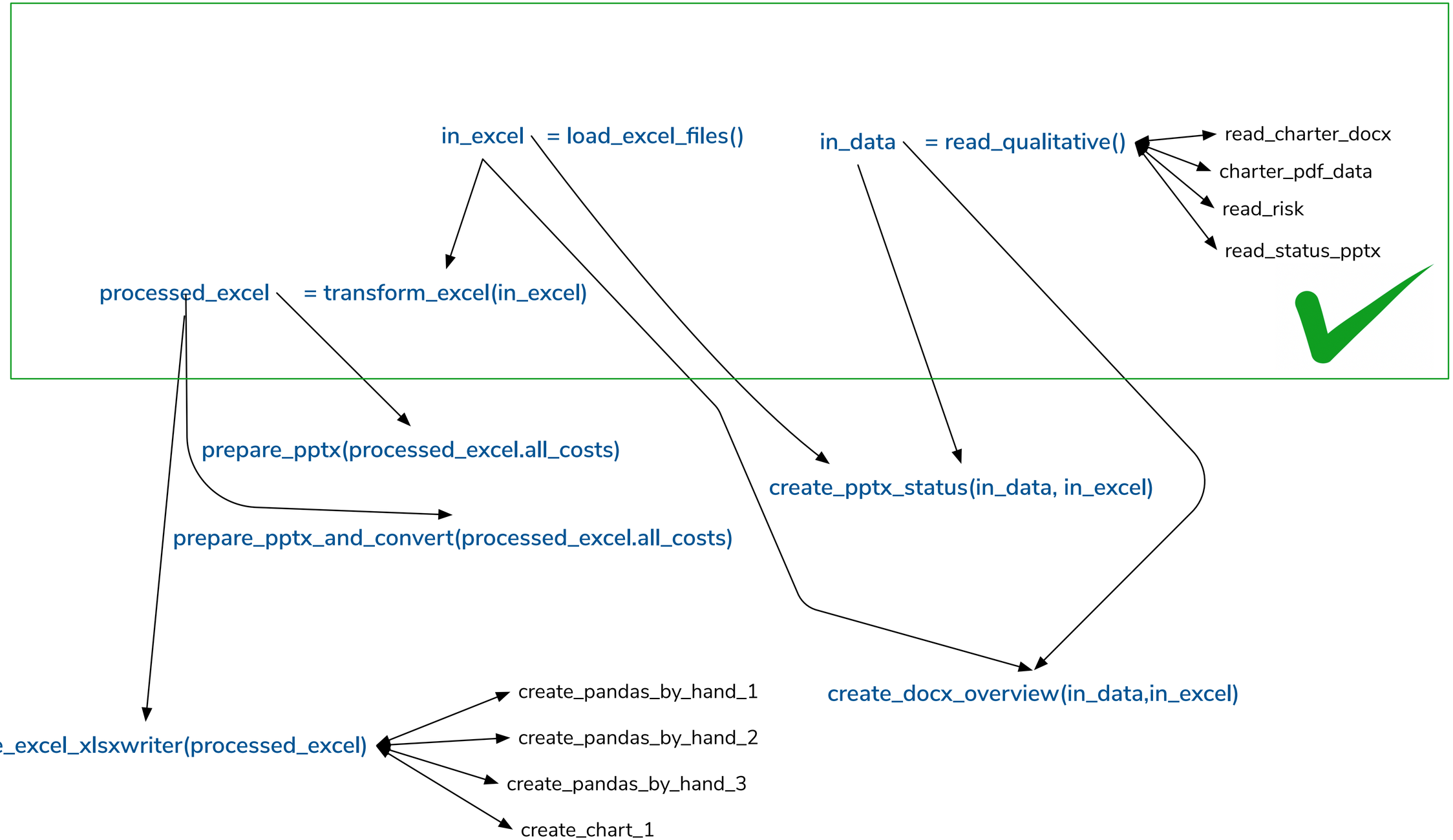
load_excel_files() - Code

```
def load_excel_files():
    df_times = pd.read_excel(in_fp("project_hours.xlsx"))
    df_expenses = pd.read_excel(in_fp("project_expenses.xlsx"))
    df_expenses.rename({"Amount": "Cost"}, axis="columns", inplace=True)
    df_rates = pd.read_excel(in_fp("project_rates.xlsx"))
    result = ExcelData(df_times, df_expenses, df_rates)
    return result
```

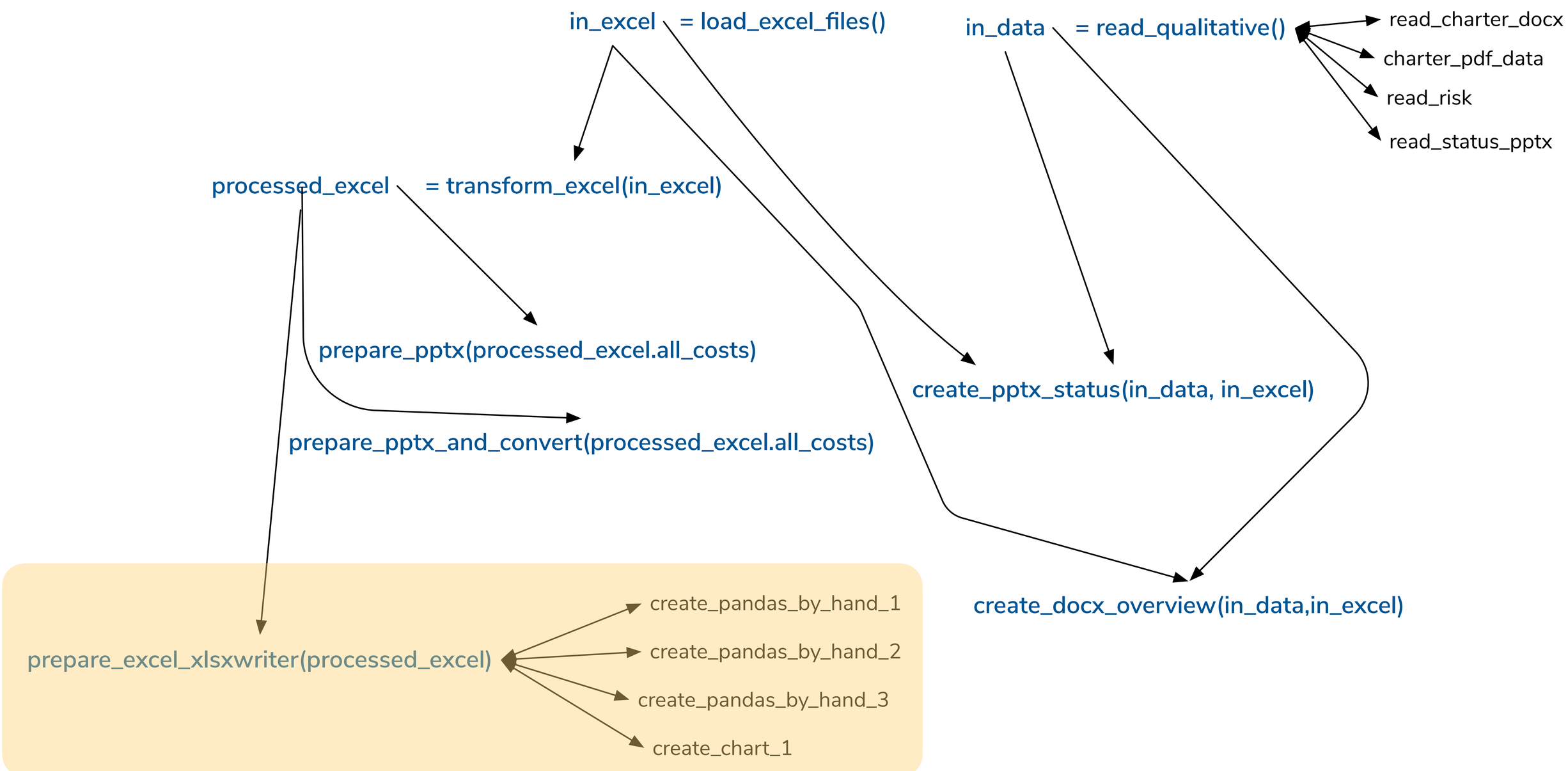
transform_excel_files() - Code

```
def transform_excel(inexcel):
    df_times_rate = inexcel.hours.merge(inexcel.rates, how="outer", on="Person")
    times_diff = df_times_rate["TimeStop"] - df_times_rate["TimeStart"]
    df_times_rate["Cost"] = times_diff * df_times_rate["Rate"]
    df_times_cost_pivot = df_times_rate.pivot_table(
        values="Cost", index=["Project", "Person"]).reset_index()
    df_times_cost_pivot["Cost Type"] = "hours"
    df_expenses_pivot = inexcel.expenses.pivot_table(
        values="Cost", index=["Project", "Person"]).reset_index()
    df_expenses_pivot["Cost Type"] = "expenses"
    df_all_costs = pd.concat([df_expenses_pivot, df_times_cost_pivot], sort=True)

    result = ExcelResults(df_times_cost_pivot, df_expenses_pivot, df_all_costs)
    return result
```

Wo wir sind


in_excel = load_excel_files()
in_data = read_qualitative()
read_charter_docx
charter_pdf_data
read_risk
read_status_pptx
processed_excel = transform_excel(in_excel)
prepare_pptx(processed_excel.all_costs)
create_pptx_status(in_data, in_excel)
prepare_pptx_and_convert(processed_excel.all_costs)
create_docx_overview(in_data,in_excel)
prepare_excel_xlsxwriter(processed_excel)
create_pandas_by_hand_1
create_pandas_by_hand_2
create_pandas_by_hand_3
create_chart_1

Excel Dateien Erstellen



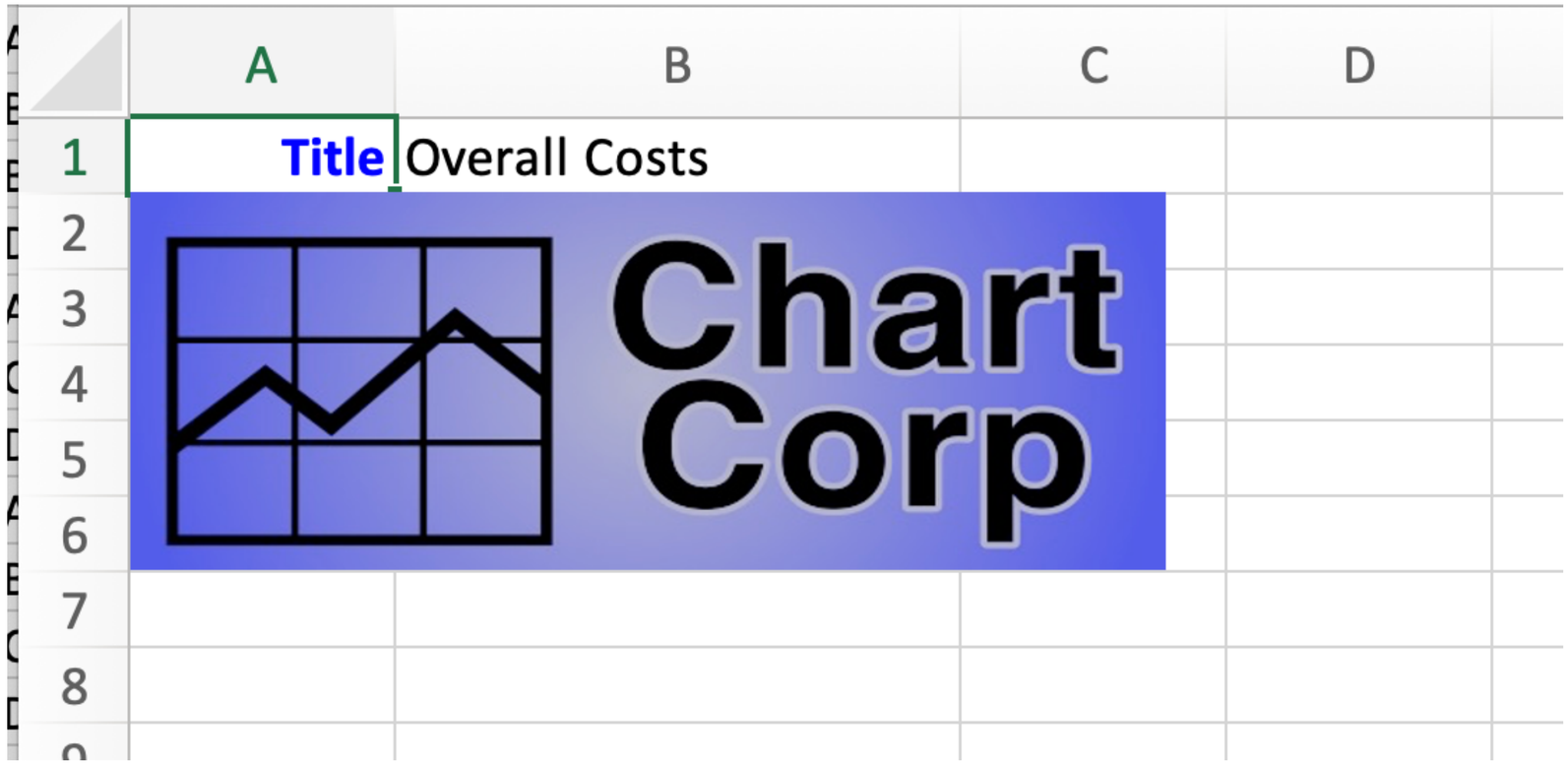
prepare_excel_xlsxwriter() - Code

```
def prepare_excel_xlsxwriter(excel_data):
    export_to_xlsx_sheets(excel_data)
    create_sheets_from_pandas_intro(excel_data)
    writer = pd.ExcelWriter(out_fp('numbers.xlsx'), engine='xlsxwriter')
    workbook = writer.book
    create_pandas_by_hand_1(workbook, "All Costs", excel_data.all_costs)
    create_pandas_by_hand_2(workbook, "All Costs 2", excel_data.all_costs)
    create_pandas_by_hand_3(workbook, "All Costs 3", excel_data.all_costs)
    create_chart_1(workbook, "Sheet with Chart 1", excel_data.all_costs)
    writer.close()
```

export_to_xlsx_sheets() - Code

```
def export_to_xlsx_sheets(excel_data):
    writer = pd.ExcelWriter(out_fp('simple_numbers.xlsx'), engine='xlsxwriter')
    excel_data.all_costs.to_excel(writer, index=False, sheet_name='df_all_costs')
    excel_data.expenses_pivot.to_excel(writer, index=False, sheet_name='df_expenses_pivot')
    excel_data.times_cost_pivot.to_excel(writer, index=False, sheet_name='df_times_cost_pivot')
    writer.close()
```

create_sheets_from_pandas_intro() - Ergebnis



create_sheets_from_pandas_intro() - Code

```
def create_sheets_from_pandas_intro(excel_data):
    writer = pd.ExcelWriter(out_fp('simple_intro.xlsx'), engine='xlsxwriter')
    workbook = writer.book
    create_introsheet(workbook)
    excel_data.all_costs.to_excel(writer, index=False,
                                  sheet_name='df_all_costs')
    excel_data.expenses_pivot.to_excel(writer, index=False,
                                       sheet_name='df_expenses_pivot')
excel_data.expenses_pivot.to_excel(writer, index=False)
excel_data.times_cost_pivot.to_excel(writer, index=False, sheet_name='df_times_cost_pivot')
writer.close()
```

create_sheets_from_pandas_intro() - Code

```
def create_introsheet(workbook):
    introsheet = workbook.add_worksheet("Introduction")
    bold = workbook.add_format(
        {'bold': True, "align": "right", "font_color": "blue"})
    introsheet.write(0, 0, 'Title', bold)
    intro_text = 'Overall Costs'
    introsheet.write(0, 1, 'Overall Costs')
    introsheet.set_column(1, 1, len(intro_text) + 5)
    introsheet.insert_image(1, 0, asset_fp("logo.jpg"),
                            {'x_scale': 0.5, 'y_scale': 0.5})
```

create_pandas_by_hand_1() - Ergebnis

Cost	Cost Type	Person	Project
20	expenses	Anna	Kaffee Auswählen
10	expenses	Ben	Kaffee Auswählen
15	expenses	Cato	Kaffee Auswählen
20	expenses	Anna	Kaffee Kaufen
2,5	expenses	Ben	Kaffee Kaufen
5	expenses	Cato	Kaffee Kaufen
50	expenses	Anna	Kaffee Kochen
2	expenses	Cato	Kaffee Kochen
20	expenses	Ben	Kaffee Mahlen
30	expenses	Ben	Kaffee Trinken
845	hours	Anna	Aufwischen
385	hours	Ben	Aufwischen
770	hours	Ben	Kaffee Auswählen
953,3333	hours	Daniele	Kaffee Auswählen
130	hours	Anna	Kaffee Kaufen
385	hours	Cato	Kaffee Kaufen
563,3333	hours	Daniele	Kaffee Kaufen
823,3333	hours	Anna	Kaffee Kochen

create_pandas_by_hand_1() - Code

```
def create_pandas_by_hand_1(workbook, sheet_title, dataframe):
    sheet = workbook.add_worksheet(sheet_title)
    sheet.write_row(0, 0, dataframe.columns)
    for i, row in enumerate(dataframe.values):
        sheet.write_row(i + 1, 0, row)
```

create_pandas_by_hand_2() - Ergebnis

Cost	Cost Type	Person	Project
20	expenses	Anna	Kaffee Auswählen
10	expenses	Ben	Kaffee Auswählen
15	expenses	Cato	Kaffee Auswählen
20	expenses	Anna	Kaffee Kaufen
2,5	expenses	Ben	Kaffee Kaufen
5	expenses	Cato	Kaffee Kaufen
50	expenses	Anna	Kaffee Kochen
2	expenses	Cato	Kaffee Kochen
20	expenses	Ben	Kaffee Mahlen
30	expenses	Ben	Kaffee Trinken
845	hours	Anna	Aufwischen
385	hours	Ben	Aufwischen
770	hours	Ben	Kaffee Auswählen
953,333333	hours	Daniele	Kaffee Auswählen
130	hours	Anna	Kaffee Kaufen
385	hours	Cato	Kaffee Kaufen
563,333333	hours	Daniele	Kaffee Kaufen
823,333333	hours	Anna	Kaffee Kochen
770	hours	Ben	Kaffee Kochen
440	hours	Cato	Kaffee Kochen
693,333333	hours	Daniele	Kaffee Kochen
823,333333	hours	Anna	Kaffee Mahlen
660	hours	Ben	Kaffee Mahlen

create_pandas_by_hand_2() - Code

```
def create_pandas_by_hand_2(workbook, sheet_title, dataframe):
    sheet = workbook.add_worksheet(sheet_title)
    large_text = workbook.add_format({'bold': True, "font_size": 14})
    red_bold = workbook.add_format({'bold': True, "font_color": "red"})
    sheet.write_row(0, 0, dataframe.columns, large_text)

    for i, header in enumerate(dataframe.columns):
        sheet.set_column(i, i, len(header) * 1.2 + 5)

    percentile75 = dataframe["Cost"].describe()["75%"]
    for i, row in enumerate(dataframe.values):
        for i2, value in enumerate(row):
            if i2 == 0:
                if value > percentile75:
                    sheet.write_number(i + 1, i2, value, red_bold)
                else:
                    sheet.write_number(i + 1, i2, value)
            else:
                sheet.write_string(i + 1, i2, value)
```

create_pandas_by_hand_3() - Ergebnis

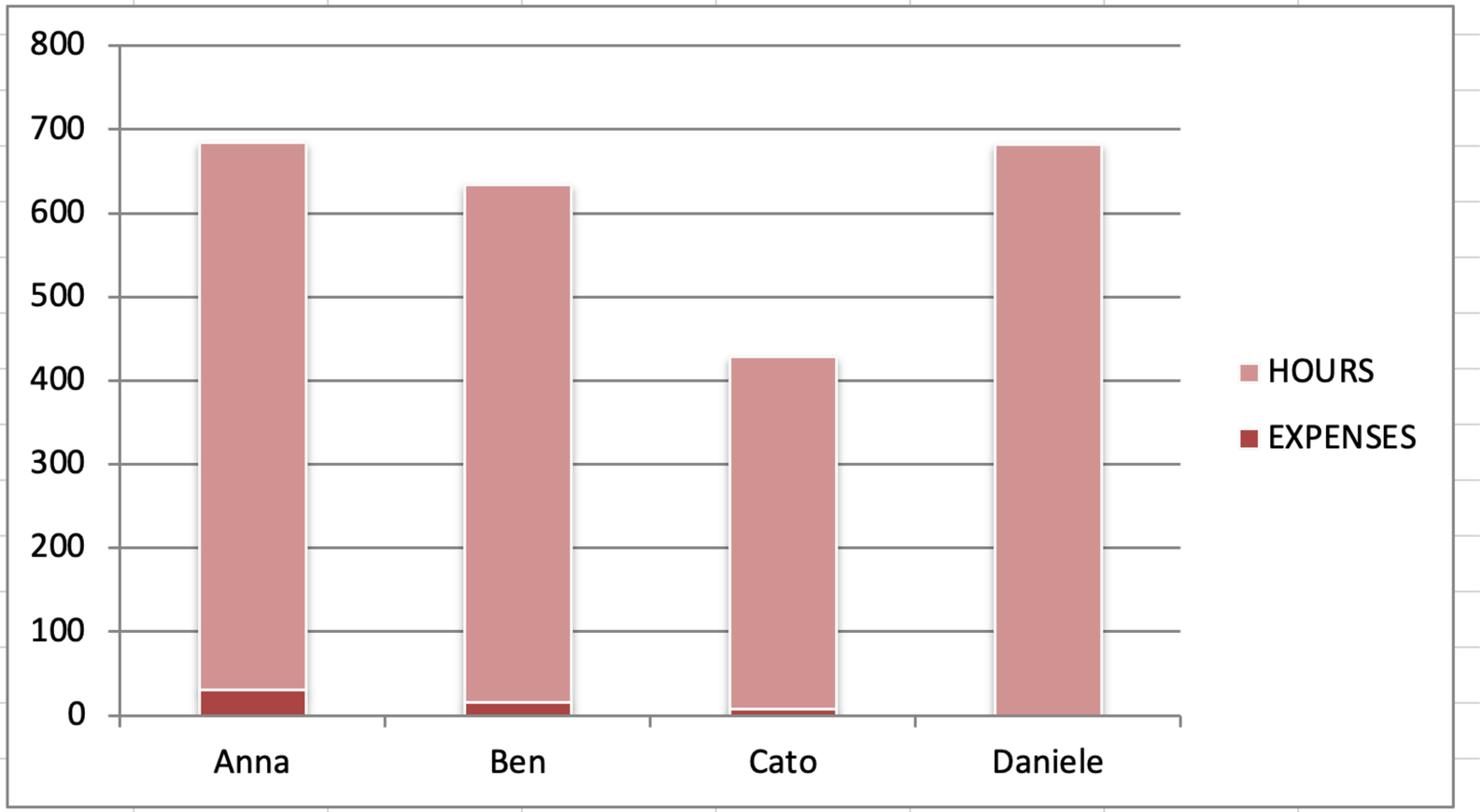
Cost	Cost Typ	Person	Project
20,	expenses	Anna	Kaffee Auswählen
10,	expenses	Ben	Kaffee Auswählen
15,	expenses	Cato	Kaffee Auswählen
20,	expenses	Anna	Kaffee Kaufen
2,5	expenses	Ben	Kaffee Kaufen
5,	expenses	Cato	Kaffee Kaufen
50,	expenses	Anna	Kaffee Kochen
2,	expenses	Cato	Kaffee Kochen
20,	expenses	Ben	Kaffee Mahlen
30,	expenses	Ben	Kaffee Trinken
845,	hours	Anna	Aufwischen
385,	hours	Ben	Aufwischen
770,	hours	Ben	Kaffee Auswählen
953,3	hours	Daniele	Kaffee Auswählen
130,	hours	Anna	Kaffee Kaufen
385,	hours	Cato	Kaffee Kaufen
563,3	hours	Daniele	Kaffee Kaufen
823,3	hours	Anna	Kaffee Kochen
770,	hours	Ben	Kaffee Kochen
440,	hours	Cato	Kaffee Kochen
693,3	hours	Daniele	Kaffee Kochen
823,3	hours	Anna	Kaffee Mahlen

create_pandas_by_hand_3() - Code

```
def create_pandas_by_hand_3(workbook, sheet_title, dataframe):
    num_format = workbook.add_format({'num_format': "####.#"})
    sheet = workbook.add_worksheet(sheet_title)
    nrows, ncols = dataframe.shape
    columns_desc = [{"header": v} for v in dataframe.columns]
    sheet.add_table(0, 0, nrows, ncols - 1, {"data": dataframe.values,
                                             "columns": columns_desc})
    sheet.set_column(0, 0, 10, num_format)


    conditional_options = {
        'type': '3_color_scale',
        "min_color": "green",
        "mid_color": "yellow",
        "max_color": "red"
    }
    sheet.conditional_format(1, 0, nrows, 0, conditional_options)
```

create_chart_1() - Ergebnis

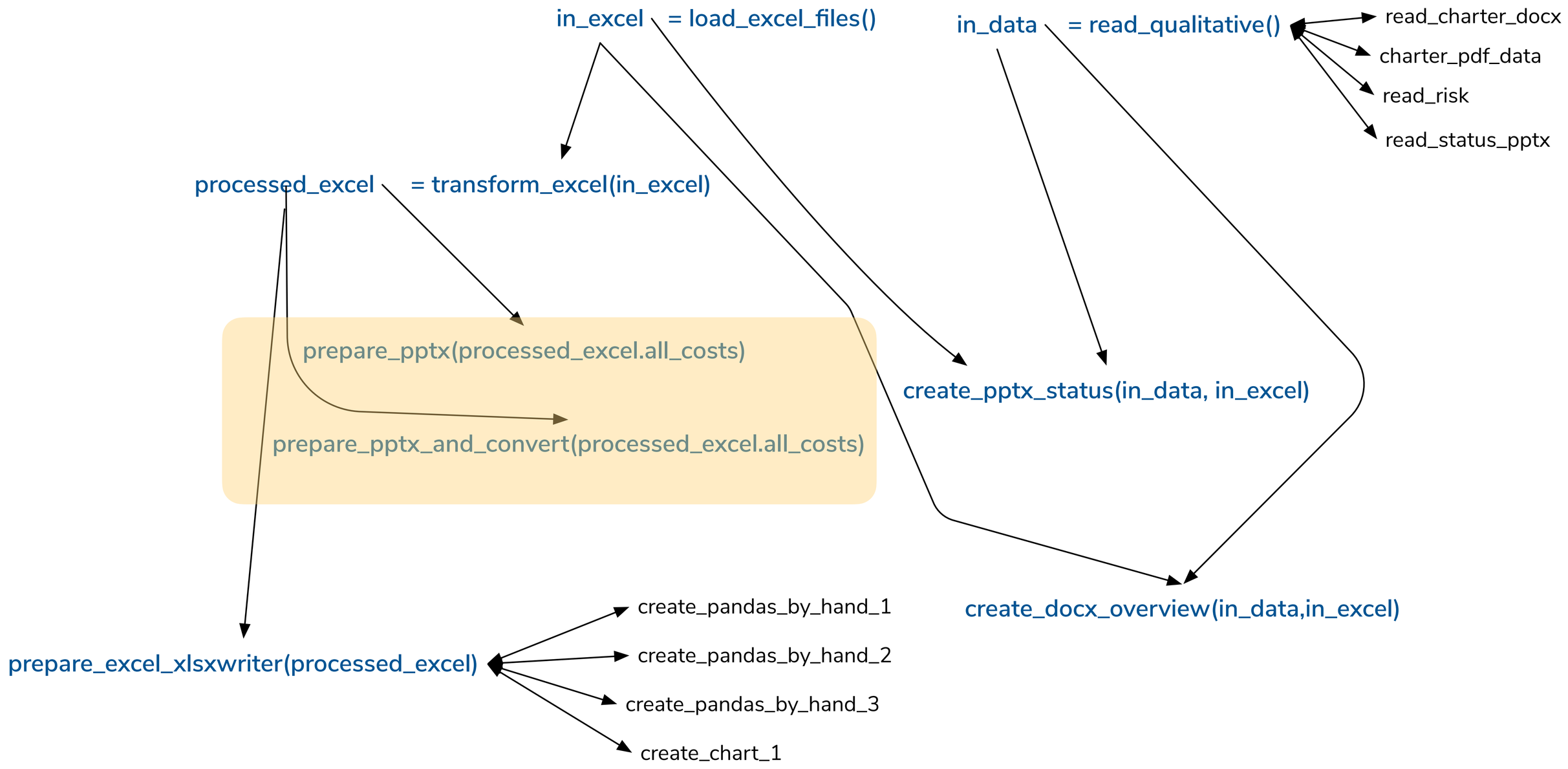
PERSON	EXPENSES	HOURS
Anna	30	654,3333
Ben	15,625	618,2
Cato	7,333333	421,6667
Daniele	0	682,5



create_chart_1() - Code

```
def create_chart_1(workbook, sheet_title, df_all_costs):
    sheet = workbook.add_worksheet(sheet_title)
    df_chart = df_all_costs.pivot_table(
        values="Cost", index="Person", columns="Cost Type").fillna(0)
    df_chart.reset_index(inplace=True)
    sheet.write_row(0, 0, [s.upper() for s in df_chart.columns])
    sheet.write_column(1, 0, df_chart['Person'])
    sheet.write_column(1, 1, df_chart['expenses'])
    sheet.write_column(1, 2, df_chart['hours'])
    chart = workbook.add_chart({'type': 'column', 'subtype': 'stacked'})
    chart.set_style(12)
    nrows = df_chart.shape[0]
    for i in [1, 2]:
        chart.add_series({
            'name': [sheet.get_name(), 0, i],
            'categories': [sheet.get_name(), 1, 0, nrows, 0],
            'values': [sheet.get_name(), 1, i, nrows, i]})
    sheet.insert_chart('A8', chart, {'x_offset': 25, 'y_offset': 10})
```

Powerpoint Dateien erstellen



Die Ergebnisse

Introduction

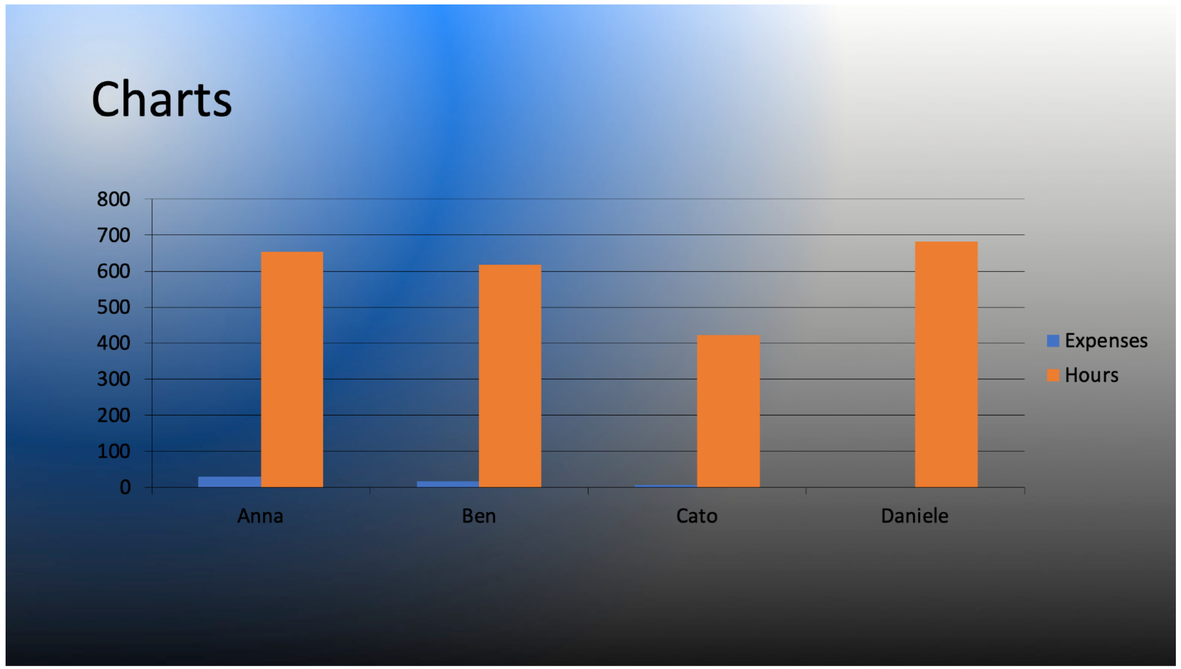
A Short but meaningful text for the slide

Chart Corp

Data Table

Cost	Cost Type	Person	Project
20.0	expenses	Anna	Kaffee Auswählen
10.0	expenses	Ben	Kaffee Auswählen
15.0	expenses	Cato	Kaffee Auswählen
20.0	expenses	Anna	Kaffee Kaufen
2.5	expenses	Ben	Kaffee Kaufen
5.0	expenses	Cato	Kaffee Kaufen
50.0	expenses	Anna	Kaffee Kochen
2.0	expenses	Cato	Kaffee Kochen
20.0	expenses	Ben	Kaffee Mahlen
30.0	expenses	Ben	Kaffee Trinken
845.0	hours	Anna	Aufwischen
385.0	hours	Ben	Aufwischen
770.0	hours	Ben	Kaffee Auswählen

Charts



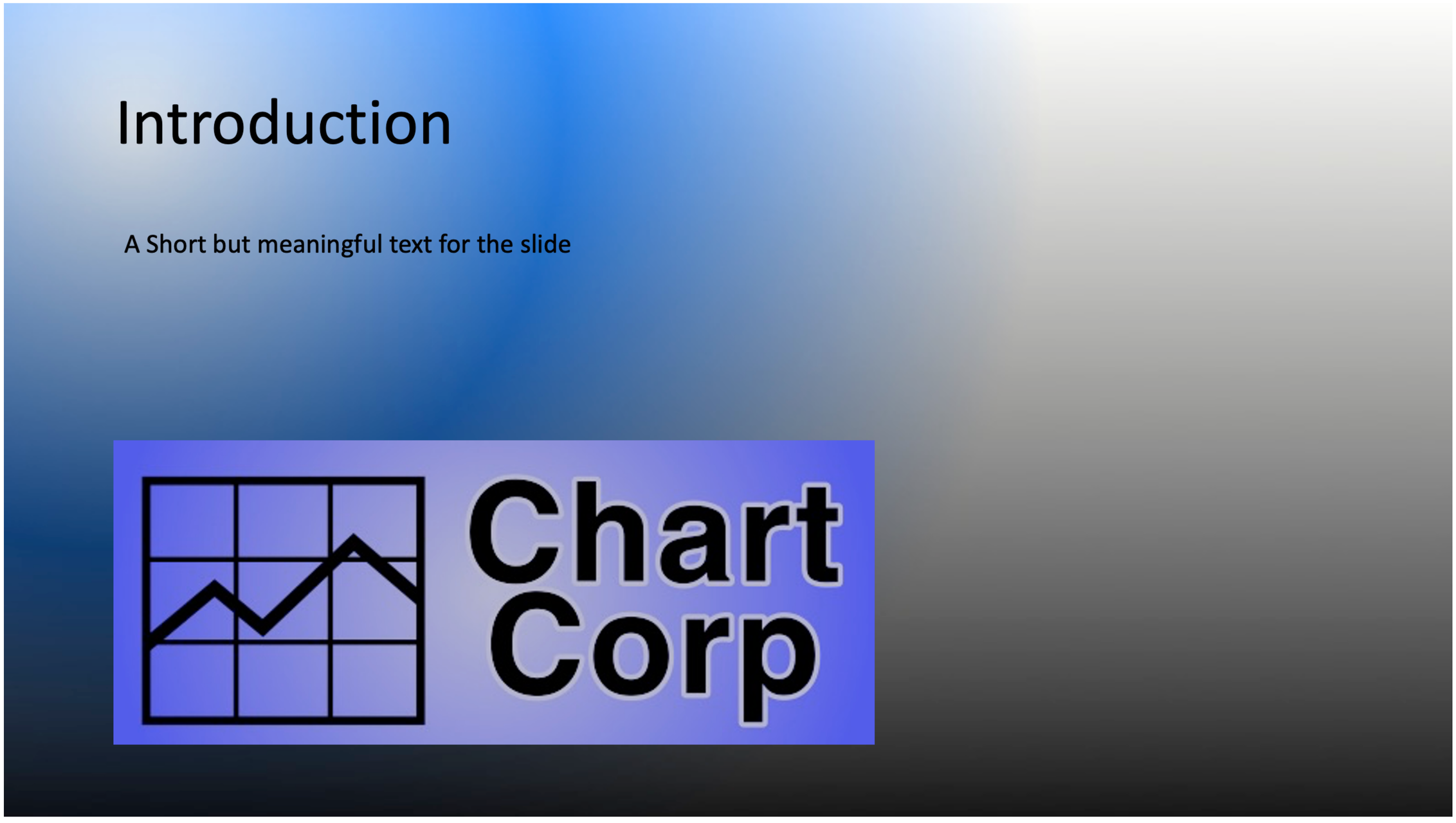
prepare_pptx() - Code 1

```
def prepare_pptx(df_all_costs):
    presentation = pptx.Presentation(asset_fp("template.pptx"))

    slide = create_slide(presentation, "Introduction")
    create_intro_slide_with_graphic(slide)

    slide = create_slide(presentation, "Data Table")
    create_table_slide(df_all_costs, slide)

    slide = create_slide(presentation, "Charts")
    create_chart_slide(df_all_costs, slide)

    presentation.save(out_fp('test.pptx'))
```

prepare_pptx() - Code 2

```
def create_slide(presentation, title):
    layout = presentation.slide_layouts[5]
    slide = presentation.slides.add_slide(layout)
    if title is not None:
        slide_title = slide.shapes.title
        slide_title.text = title
    return slide
```

prepare_pptx() - Einführung



prepare_pptx() - Code Einführung

```
def create_intro_slide_with_graphic(slide):
    from pptx.util import Inches
    left = width = height = Inches(1)
    top = Inches(2)
    txBox = slide.shapes.add_textbox(left, top, width, height)
    tf = txBox.text_frame
    tf.text = "A Short but meaningful text for the slide"
    top = Inches(4)
    slide.shapes.add_picture(asset_fp("logo.jpg"), left, top)
```

prepare_pptx() - Ergebnis Tabelle

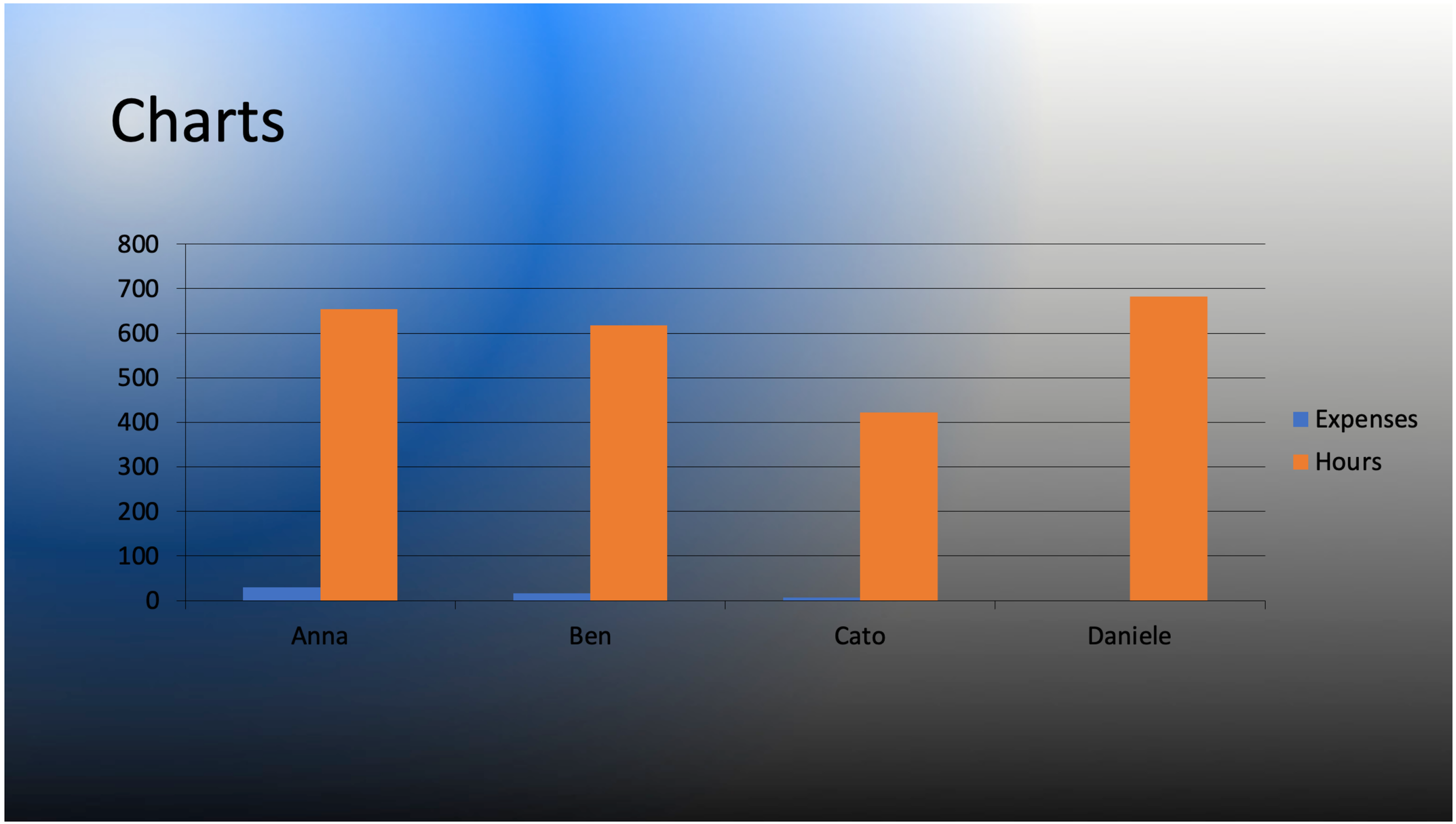
Data Table

Cost	Cost Type	Person	Project
20.0	expenses	Anna	Kaffee Auswählen
10.0	expenses	Ben	Kaffee Auswählen
15.0	expenses	Cato	Kaffee Auswählen
20.0	expenses	Anna	Kaffee Kaufen
2.5	expenses	Ben	Kaffee Kaufen
5.0	expenses	Cato	Kaffee Kaufen
50.0	expenses	Anna	Kaffee Kochen
2.0	expenses	Cato	Kaffee Kochen
20.0	expenses	Ben	Kaffee Mahlen
30.0	expenses	Ben	Kaffee Trinken
845.0	hours	Anna	Aufwischen
385.0	hours	Ben	Aufwischen
770.0	hours	Ben	Kaffee Auswählen

prepare_pptx() - Code Tabelle

```
def create_table_slide(df_all_costs, slide):
    table_left = Inches(1)
    table_top = Inches(2)
    table_width = Inches(12)
    table_height = Inches(4)
    pd2ppt.df_to_table(slide, df_all_costs, table_left, table_top,
                       table_width, table_height)
```

prepare_pptx() - Chart



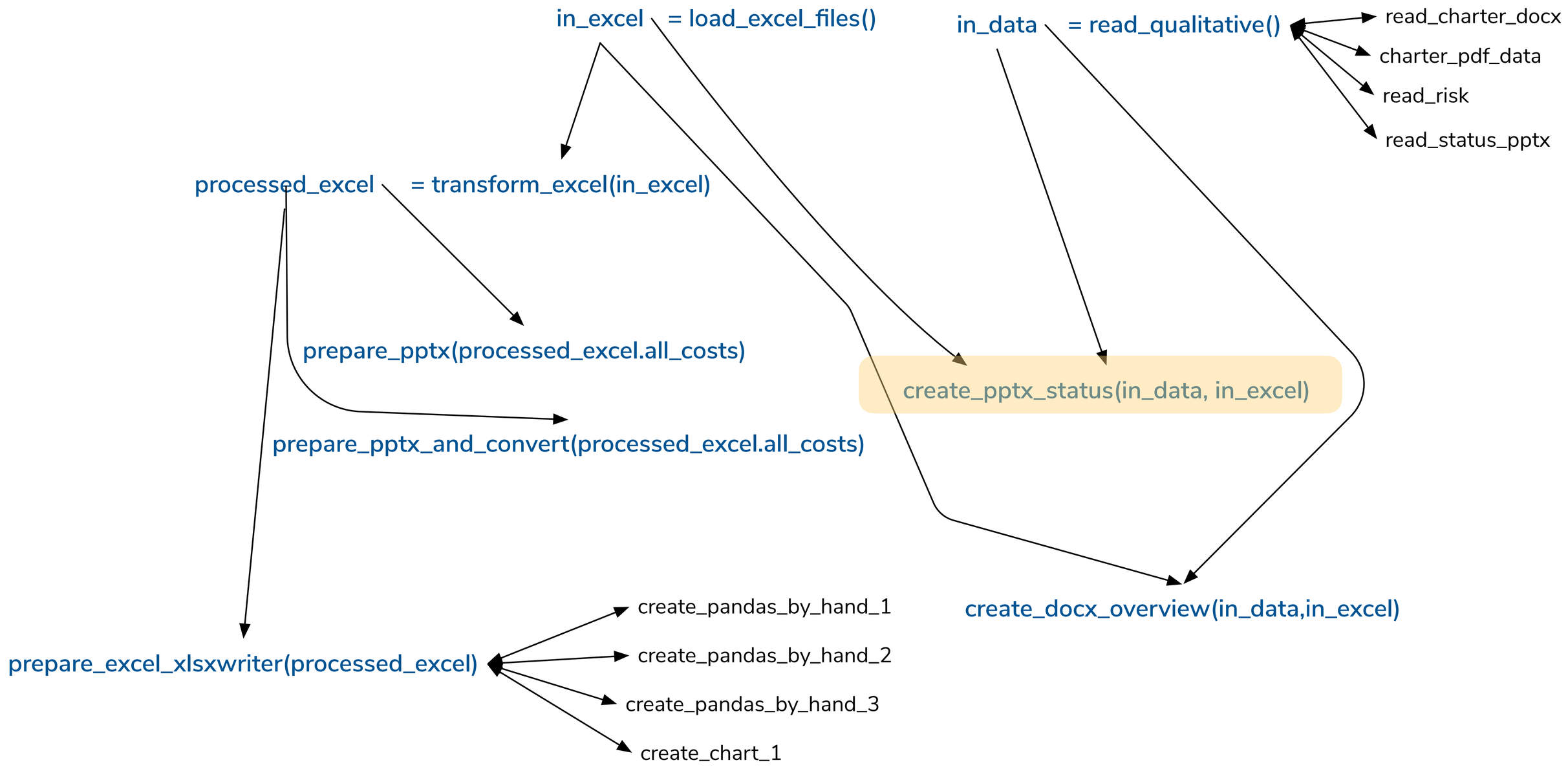
prepare_pptx() - Chart Code

```
def create_chart_slide(df_all_costs, slide):
    df_chart = df_all_costs.pivot_table(values="Cost",
                                        index="Person", columns="Cost Type")
    df_chart = df_chart.reset_index().fillna(0)
    chart_data = ChartData()
    chart_data.categories = list(df_chart['Person'])
    chart_data.add_series('Expenses', list(df_chart["expenses"]))
    chart_data.add_series('Hours', list(df_chart["hours"]))
    CHART_TYPE = XL_CHART_TYPE.COLUMN_CLUSTERED
    chart_left = Inches(1)
    chart_top = Inches(2)
    chart_width = Inches(12)
    chart_height = Inches(4)
    chart = slide.shapes.add_chart(CHART_TYPE, chart_left, chart_top,
                                   chart_width, chart_height, chart_data).chart
    chart.has_legend = True
    chart.legend.include_in_layout = False
```

prepare_pptx_and_convert() - Code

```
def prepare_pptx_and_convert(df_all_costs):

    pptx_filename = out_fp("status_2.pptx")

    import os
    import subprocess
    import pptx

    export_format = "pdf"
    presentation_plain = pptx.Presentation(asset_fp("template_plain.pptx"))
    slide = create_slide(presentation_plain, "Charts")
    create_chart_slide(df_all_costs, slide)
    presentation_plain.save(pptx_filename)

    libre_office_binary = "/Applications/LibreOffice.app/Contents/MacOS/soffice"
    cmd = [libre_office_binary, "--headless", "--convert-to", export_format,
           "--outdir", os.path.dirname(pptx_filename),
           pptx_filename]
    subprocess.run(cmd, check=True)
```

Powerpoint Dateien mit Platzhaltern erstellen



create_pptx_status() - Platzhalter und Ergebnis

Coffee Project Status Report

Risks
• STATUS

Numbers
• Numbers

Folien mit Platzhaltern

Coffee Project Status Report

Ausgefüllte Platzhalter

Risks
• **Unsere Kaffeemaschine explodiert**
• **Kaffee wird Teurer**

Numbers	
Aufwischen	**20**
Kaffee Auswählen	29
Kaffee Kaufen	21
Kaffee Kochen	61
Kaffee Mahlen	35
Kaffee Trinken	42

create_pptx_status() - Code

```
def create_pptx_status(indata, inexcel):
    df = pd.DataFrame(inexcel.hours)
    df['Hours'] = df['TimeStop'] - df['TimeStart']
    result = df.pivot_table(values='Hours', index='Project', aggfunc='sum').to_dict()['Hours']

    prs = Presentation(asset_fp("placeholder.pptx"))
    slide = prs.slides[0]
    placeholders = {shape.placeholder_format.idx: shape for shape in slide.placeholders}

    table_ph = placeholders[11]
    text_ph = placeholders[10]
```

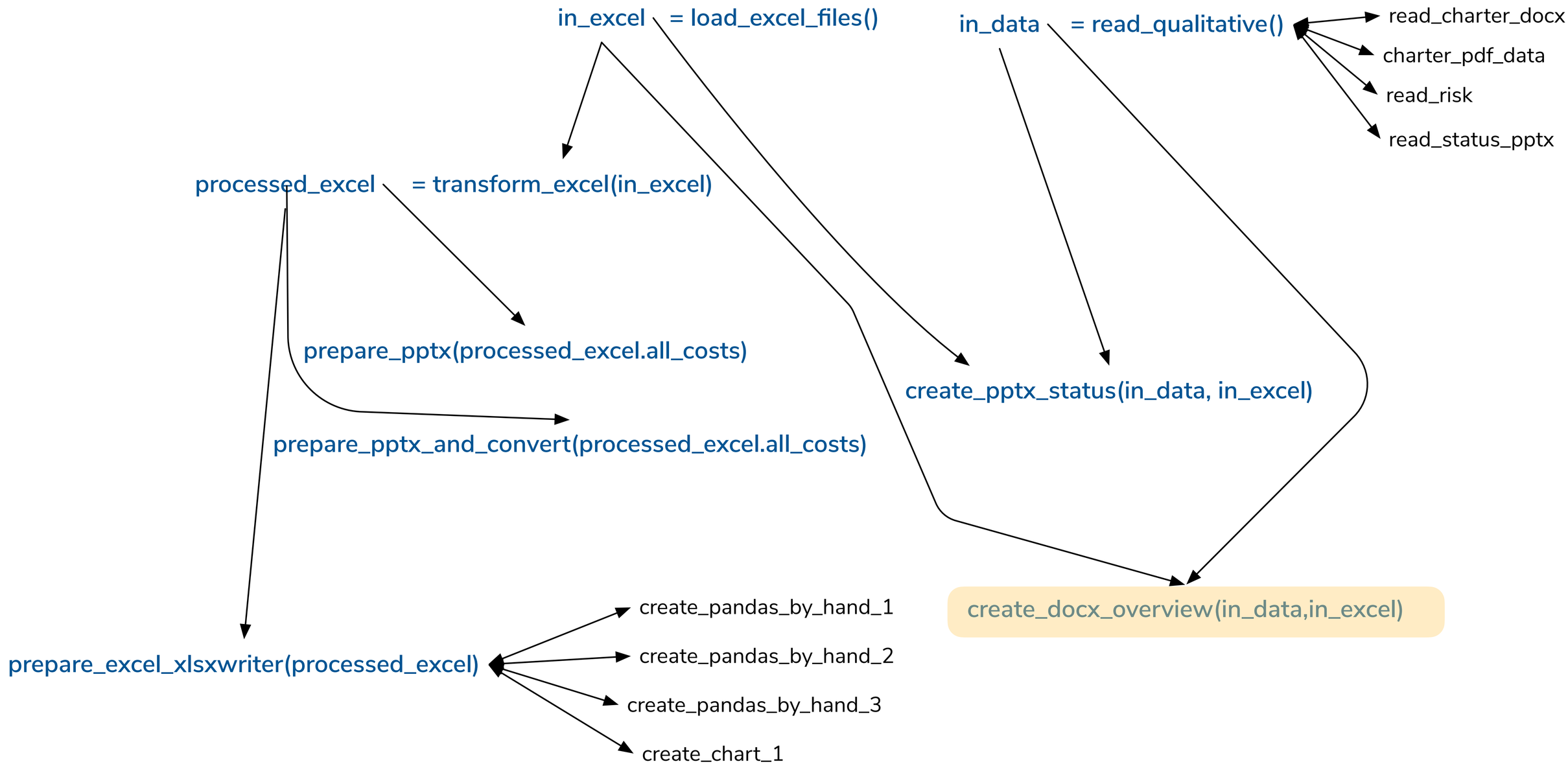
create_pptx_status() - Code

```
table_ph = placeholders[11]
text_ph = placeholders[10]

table = table_ph.insert_table(rows=len(result), cols=2).table
for i, (k, v) in enumerate(result.items()):
    for v2, i2 in zip([k, v], [0, 1]):
        cell = table.cell(i, i2)
        cell.text = str(v2)
```

create_pptx_status() - Code

```
colors = {
    'Low':RGBColor(124,252,0),
    'Medium': RGBColor(255, 194, 0),
    'High': RGBColor(136, 8, 8),
}

for risk in indata.risks:
    p = text_ph.text_frame.add_paragraph()
    run = p.add_run()
    run.text = risk['text']

    font = run.font
    font.size = Pt(20)
    font.bold = True
    font.color.rgb = colors[risk['impact']]
```

Word Dateien erstellen



create_docx_overview() - Ergebnis

Projekt Report

Project: Kaffee kochen

Milestones

1. Kaffee kaufen
2. Kaffee machen
3. Kaffee trinken

create_docx_overview() - Code

```python
def add_docx_header(doc, size, title, level=1):
    h = doc.add_heading(title, level=level)
    h.style.paragraph_format.space_after = dpt(6)
    h.runs[0].font.size = dpt(size)


def create_docx_overview(in_data, in_excel):
    document = Document()
    add_docx_header(document,18,"Projekt Report")

    document.add_paragraph(f"Project: {in_data.charter['po']}")
    document.add_paragraph("Milestones")

    for g in in_data.charter['ms']:
        document.add_paragraph(g,style='List Number')

    document.save(out_fp("project_status.docx"))
```

Zusammenfassung

- Python hat Module für alle Wesentlichen Aufgaben
  - Die Umwandlung in PDF ist eine Ausnahme
- Das Erstellen von Dokumenten ist einfach, auch im Zusammenspiel mit Formatierungen
- Die Extraktion von Daten und das Ändern bestehender Dokumente ist nicht trivial bzw. ist fragil in Bezug auf Änderungen an Dokumenten.