Programmierung von CAx-Systemen

Übung 1

David Straub

Einrichten der Python-Executable

- KCA-Rechner: Download von WinPython 3.13, entpacken ins Benutzerverzeichnis, z.B. `C:\Users\hm-abcd12ef`
- Windows: Download des Python Install Managers (Achtung: nicht Installers!)
- macOS: Download des Python Installers
- Debian/Ubuntu: `sudo apt install python python3-pip python3-venv`

Test: in der Python-Eingabeaufforderung

```
import sys
print(sys.executable)
```

Einrichten einer virtuellen Umgebung

Im Verzeichnis, in dem die virtuelle Umgebung angelegt werden soll, z.B. im Benutzer-Verzeichnis:

- Windows: `python -m venv cax-env`
- macOS/Linux: `python3 -m venv cax-env`

Aktivieren der virtuellen Umgebung:

- Windows: `cax-env\Scripts\activate`
- macOS/Linux: `source cax-env/bin/activate`

Installieren der benötigten Python-Pakete

```
python -m pip install jupyterlab cadquery build123d jupyter-cadquery ocp-vscode
```

Test der Jupyter-Installation

- `jupyter lab`
- Neues Notebook erstellen und speichern, z.B. `test.ipynb`

```
import jupyter_cadquery
import build123d as bd

bd.Box(30, 20, 10)
```

```
import jupyter_cadquery
from cadquery import func as cq_func

cq_func.box(30, 20, 10)
```

Einrichten des OCP-CAD-Viewers in Visual Studio Code

- Öffne VS Code
- Erweiterungen installieren
  - Python
  - Jupyter
  - OCP CAD Viewer

Hinweise:

- OCP ist der Python-Wrapper um OCCT, auf dem CadQuery und Build123d basieren
- Jupyter-CadQuery basiert auf dem Kern des OCP-CAD-Viewers

Einrichten von Python in VS Code und Test

- Select Python Interpreter → `cax-env\Scripts\python.exe` (Windows) oder `cax-env/bin/python` (macOS/Linux)
- File → New → Jupyter Notebook

```
import jupyter_cadquery
import build123d as bd


bd.Box(30, 20, 10)
```

Aufgabe: LEGO-Stein (Außenhülle)

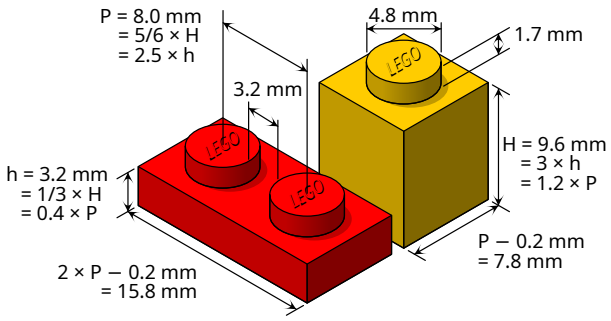

Figure 1: width:24cm

Grundkörper: `Box`

```
import build123d as bd

bd.Box(length=30, width=20, height=10)
```

- Erzeugt einen Quader mit den angegebenen Maßen in mm
- Der Mittelpunkt liegt im **Ursprung** (0, 0, 0)
- Parameter können auch ohne Namen angegeben werden: `bd.Box(30, 20, 10)`

Grundkörper: `Cylinder`

```
bd.Cylinder(radius=5, height=20)
```

- Erzeugt einen Zylinder mit dem angegebenen Radius und der Höhe
- Achse entlang der **Z-Achse**, Mittelpunkt im Ursprung

Positionierung: `Pos()`

```
zylinder = bd.Cylinder(radius=5, height=10)

bd.Pos(15, 0, 5) * zylinder
```

- `Pos(x, y, z)` definiert eine Position im Raum (in mm)
- Der `*`-Operator platziert den Körper an dieser Position

- Der Original-Körper bleibt unverändert

Boole'sche Operationen

```
vereinigung = quader + zylinder   # Vereinigung (Union)
differenz   = quader - zylinder   # Differenz (Difference)
schnitt     = quader & zylinder   # Schnitt (Intersection)
```

- Operationen erzeugen jeweils einen neuen B-Rep-Körper