Teilchenwelt Masterclass

ALICE

CERN

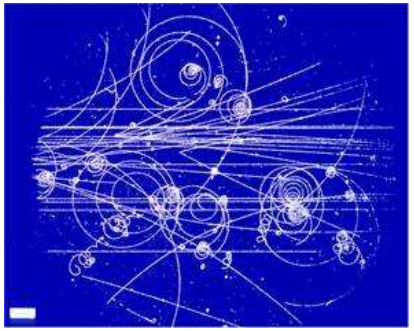
TECHNISCHE UNIVERSITÄT DRESDEN

NETZWERK TEILCHENWELT

Wie weist man Teilchen nach?

Bildgebende Detektoren

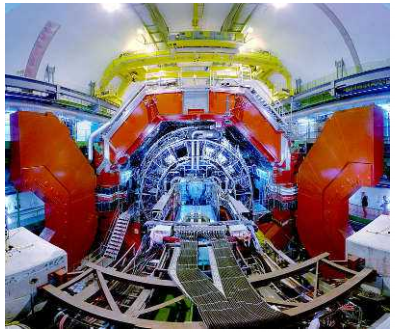
z.B.: Nebelkammer,
Blasenkammer



- sichtbare Teilchenspuren

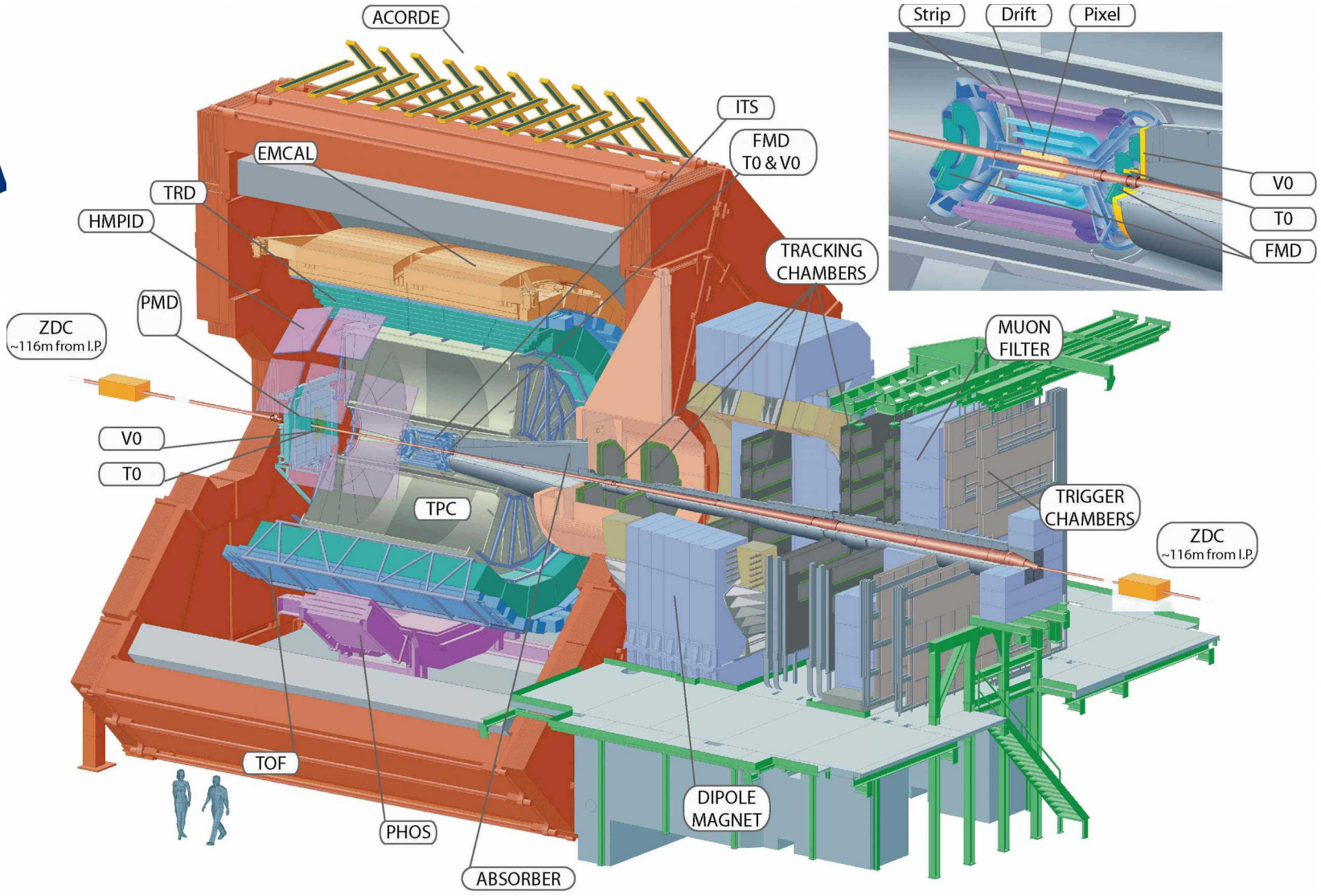
Elektronische Detektoren

z.B.: **ALICE**-Detektor,
Geigerzähler

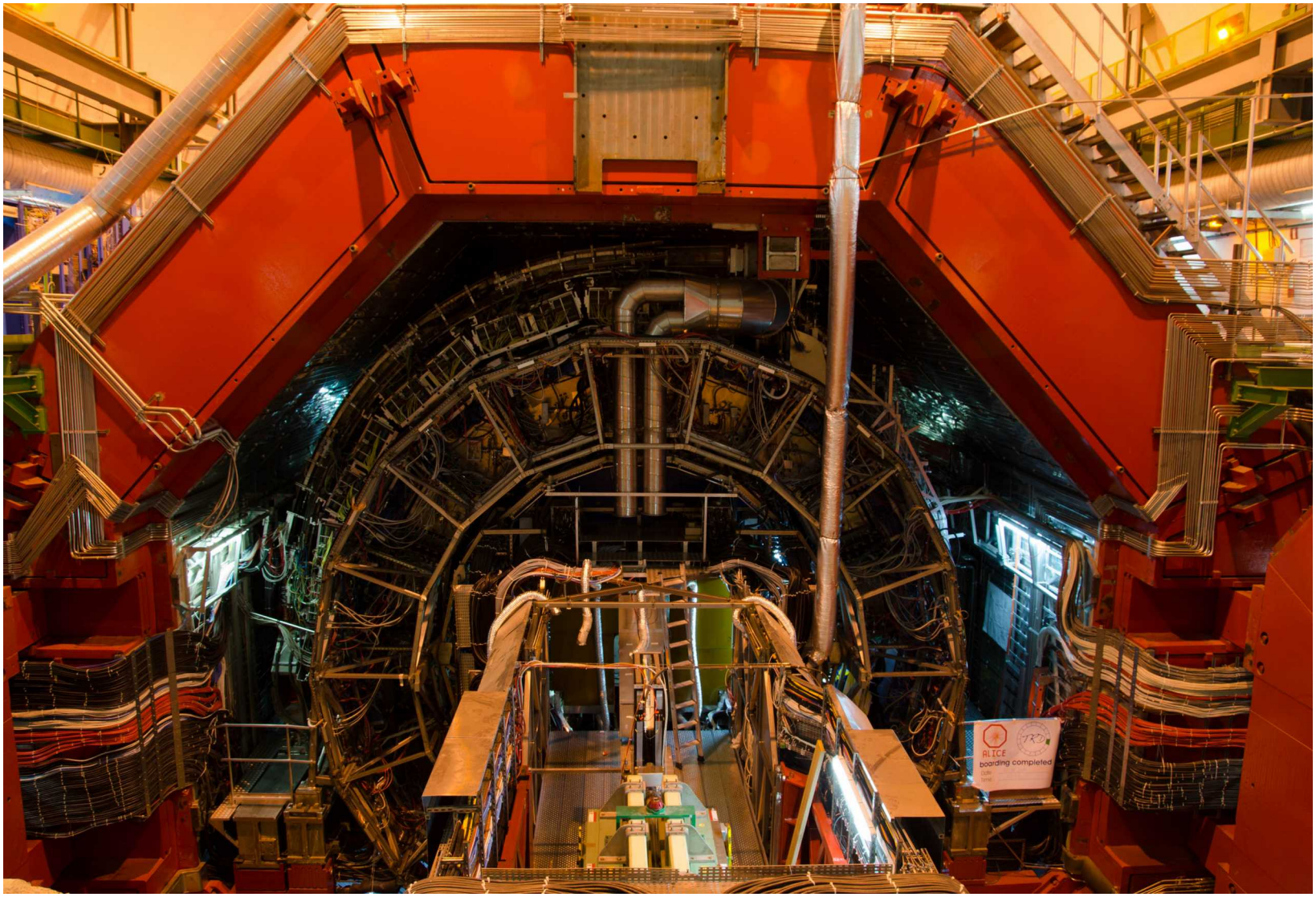


- elektrische Signale
- Eigenschaften der Teilchen werden daraus rekonstruiert

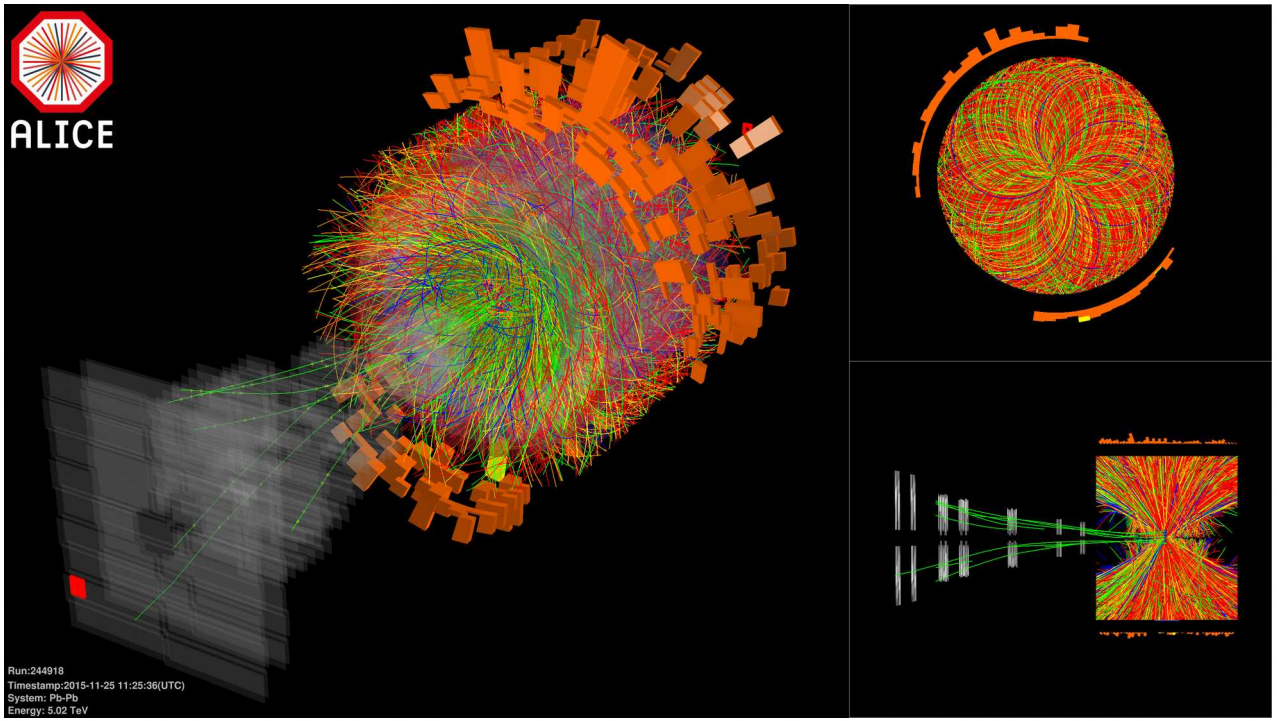
Der ALICE-Detektor



Der ALICE-Detektor



Zentrale Pb-Pb Kollision



- Manuelles Zählen der Teilchenspuren insbesondere für zentrale Pb-Pb Kollisionen nahezu unmöglich
- Des Weiteren müssen Milllionen von Ereignissen analyisert werden

➢ Automatische Analyse mittels Programme nötig!

Grundlegendes zu Python

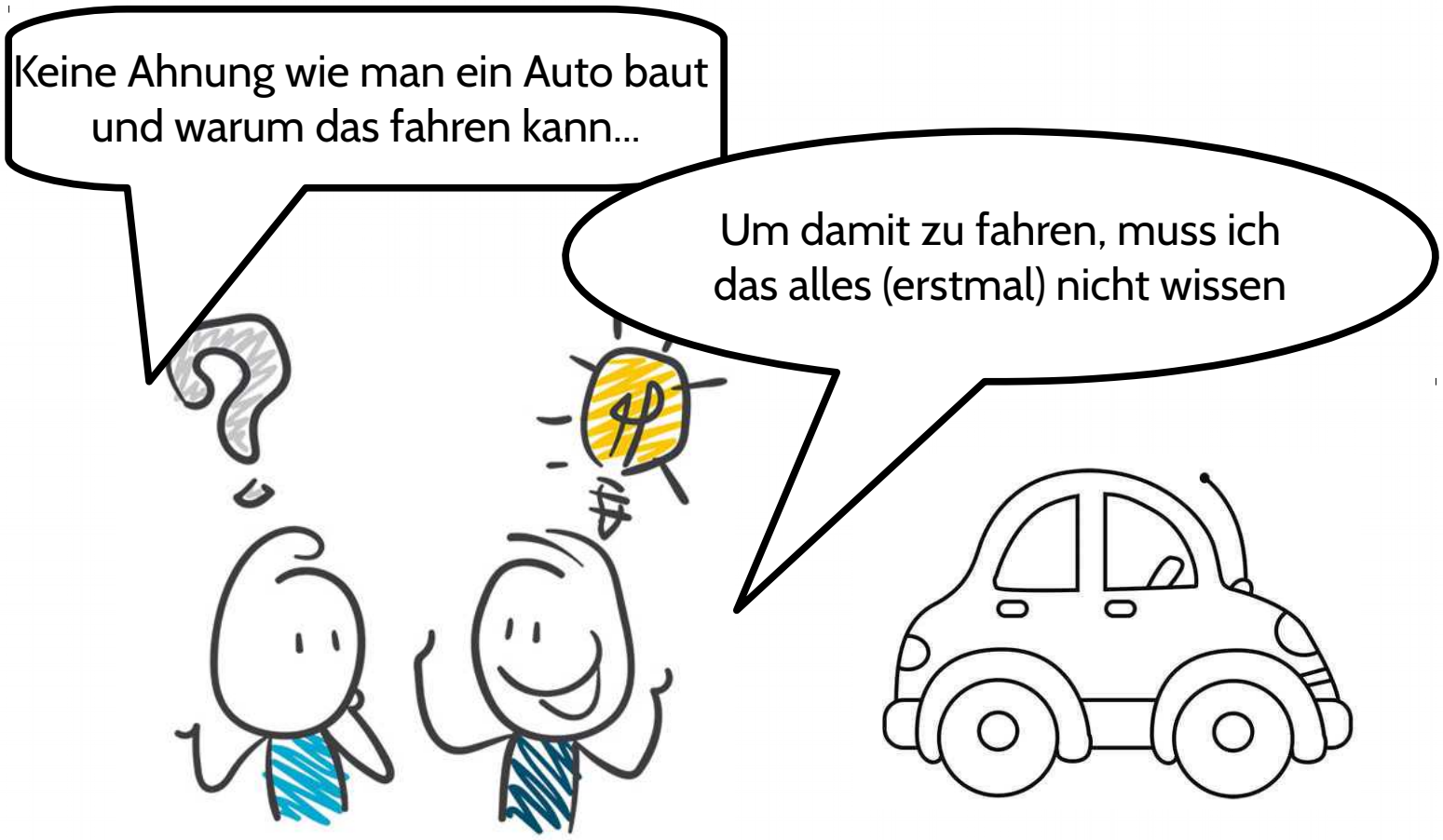
- Entwickelt in den 1990er Jahren von ‘Guido van Rossum’ mit dem Ziel größter Einfachheit und Übersichtlichkeit

- Sehr stark verbreitet, für verschiedenste Anwendungen, das “Schweizer Messer” der Programmierung

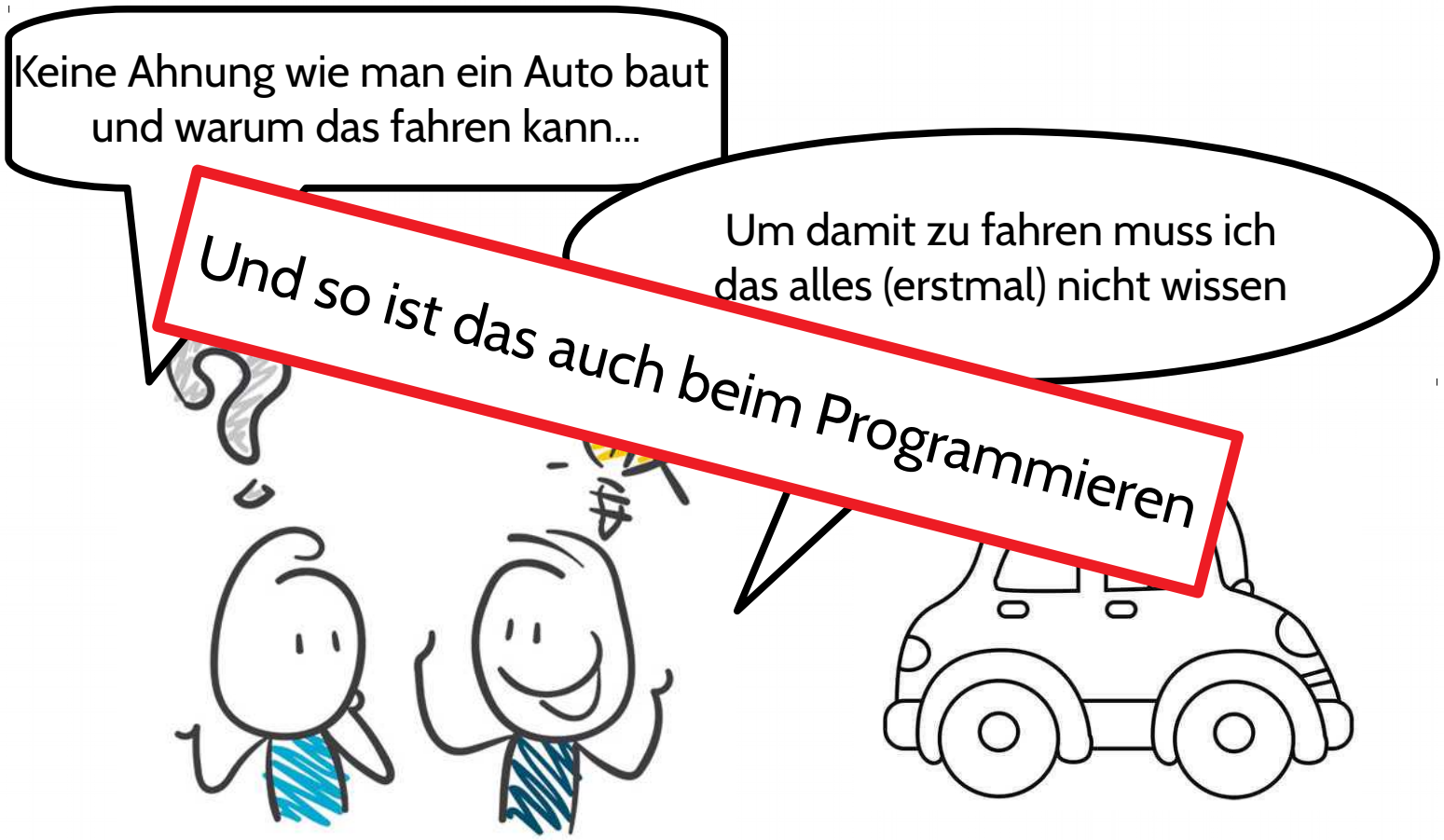
- Universelle Programmiersprache mit vielen Standardbibliotheken
  - Oft benötigte Standardaufgaben bereits verfügbar (Bildschirm Ein- und Ausgabe, Sortieren, Zeichenketten, Container,…)

- Dynamische Typisierung, Objektorientierung

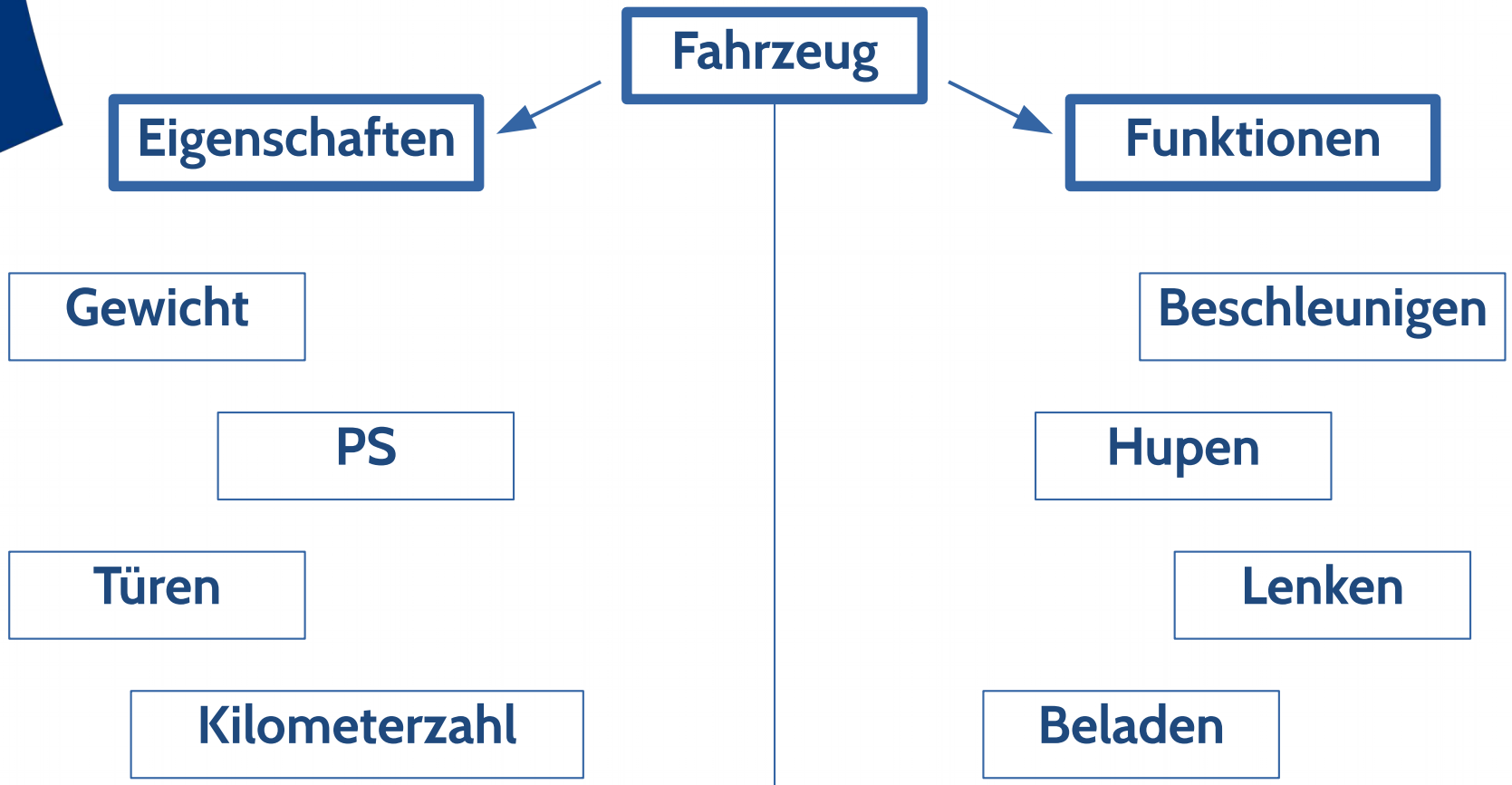
Objektorientierte Programmierung



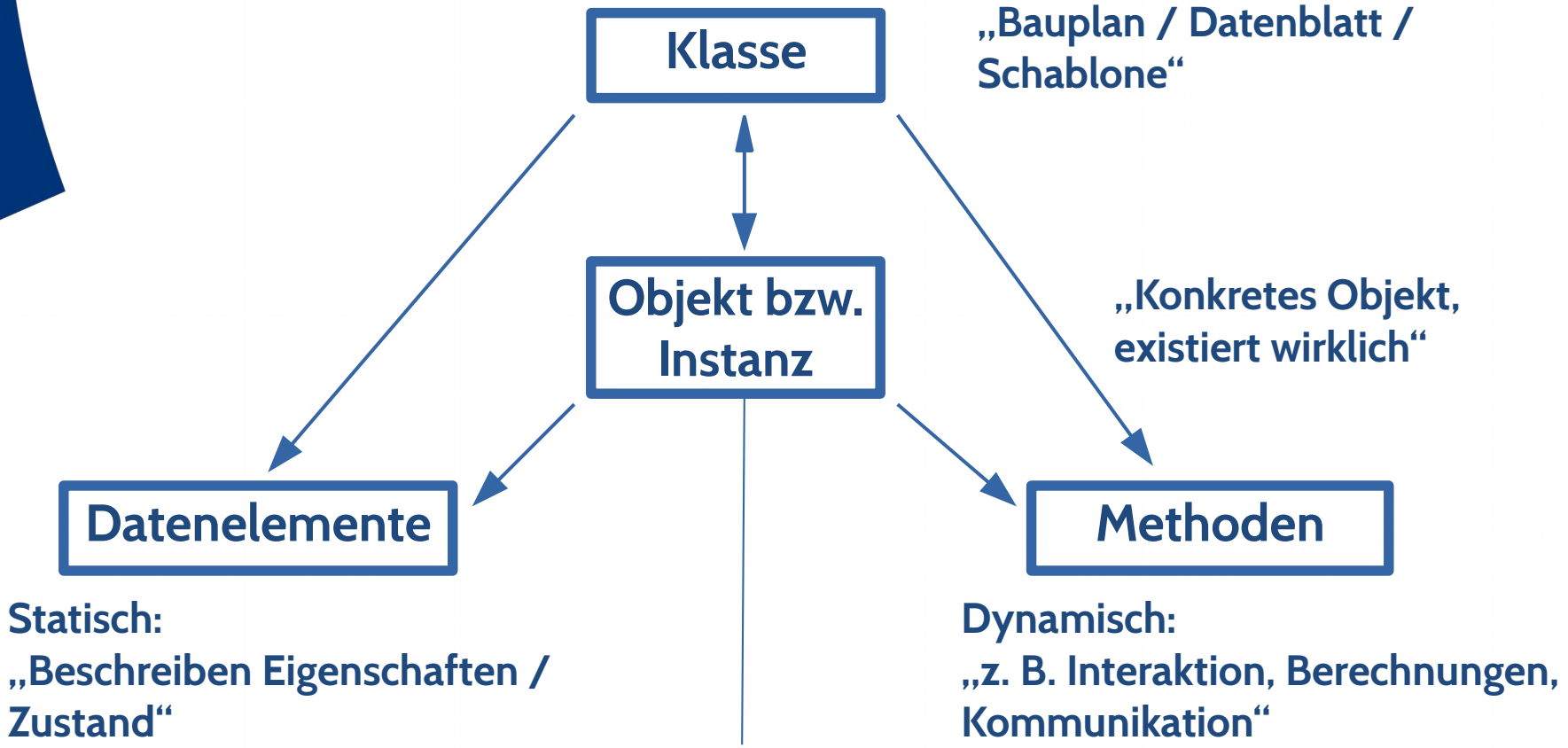
Objektorientierte Programmierung



Objektorientierte Programmierung



Objektorientierte Programmierung



- Von einer Klasse kann es beliebig viele Objekte geben!
- Änderung der Eigenschaften/Ausführen von Methoden eines Objektes beeinflussen erst einmal nicht die anderen Objekte!

Grundlegendes zu Python

- Kommentare:
  - Beginnen mit **#**
- Befehle
  - Stehen in einer Zeile

ALICE_RAA_Tools.py | ALICE_RAA.py

```
1 import ALICE_RAA_Tools as at
2 import matplotlib.pyplot as plt
3 import matplotlib as mpl
4 import pandas as pd
5 import numpy as np
6
7 listCentralities = [(0,5), (30,40), (40,50), (70,80)]
8 centralities = at.create_dictionary(listCentralities)
9
10 # read events
11 event_lines = at.count_lines('event_information.csv')
12 print("Number of events: '" + str(event_lines) + "'!")
```

- Dateien (üblicherweise *.py)
  - Öffnen und editieren mit beliebigem Texteditor möglich
- *"import Beispiel"* zu Beginn
  - Dienen zum "Laden" bereits verfügbarer Programmpakete

Grundlegendes zu Python

- Verschiedene Datentypen
  - Text *(str)*
    - z.B. “Hallo Welt!”; “1” Zeichen und Zeichenketten
  - Numerisch *(int, float, complex)*
    - z.B. 1; 3.1415, 1+i*2 Dezimalzahlen
  - Folgen *(list, tuple)*
    - Liste = [0, 1, 2, 3]; Tuple = (2.45, 87)
  - Abbildung *(dict)*
    - z.B. G = {‘Daniel’: 1,90, ‘Anderer Name’: 1,87,...}
  - Boolean *(bool)*
    - 2 Möglichkeiten Wahr oder falsch
      - True - “1”
      - False - “0”

Grundlegendes zu Python

- Einige wichtige Operatoren
  - Zuweisung: "="

    z.B.: ***a = 2, text = "Hallo Welt!"***
  - Arithmetische Operatoren: "+,-,*,/"

    z.B.: ***c = 2 * a***
  - Inkrement/Dekrement:

    z.B.: ***c += 1*** → Abkürzung für c = c + 1
  - Vergleiche: "<, >, ==, !=, <=, >="

    z.B.: ***if(c > a and a == 2):***

    a += 1
  - Zugriff auf Methoden eines Objektes:
    - z.B. ***Auto.Hupen()***

      pd.read_csv()

Jupyter - Notebooks

- Python und Funktionen

(1) Rückgabewert
   (1) Name
   (2) Parameter
   (3) Anweisungsblock

https://github.com/dmuhlhei/ALICE_MC_RAA

Bitte beachten:

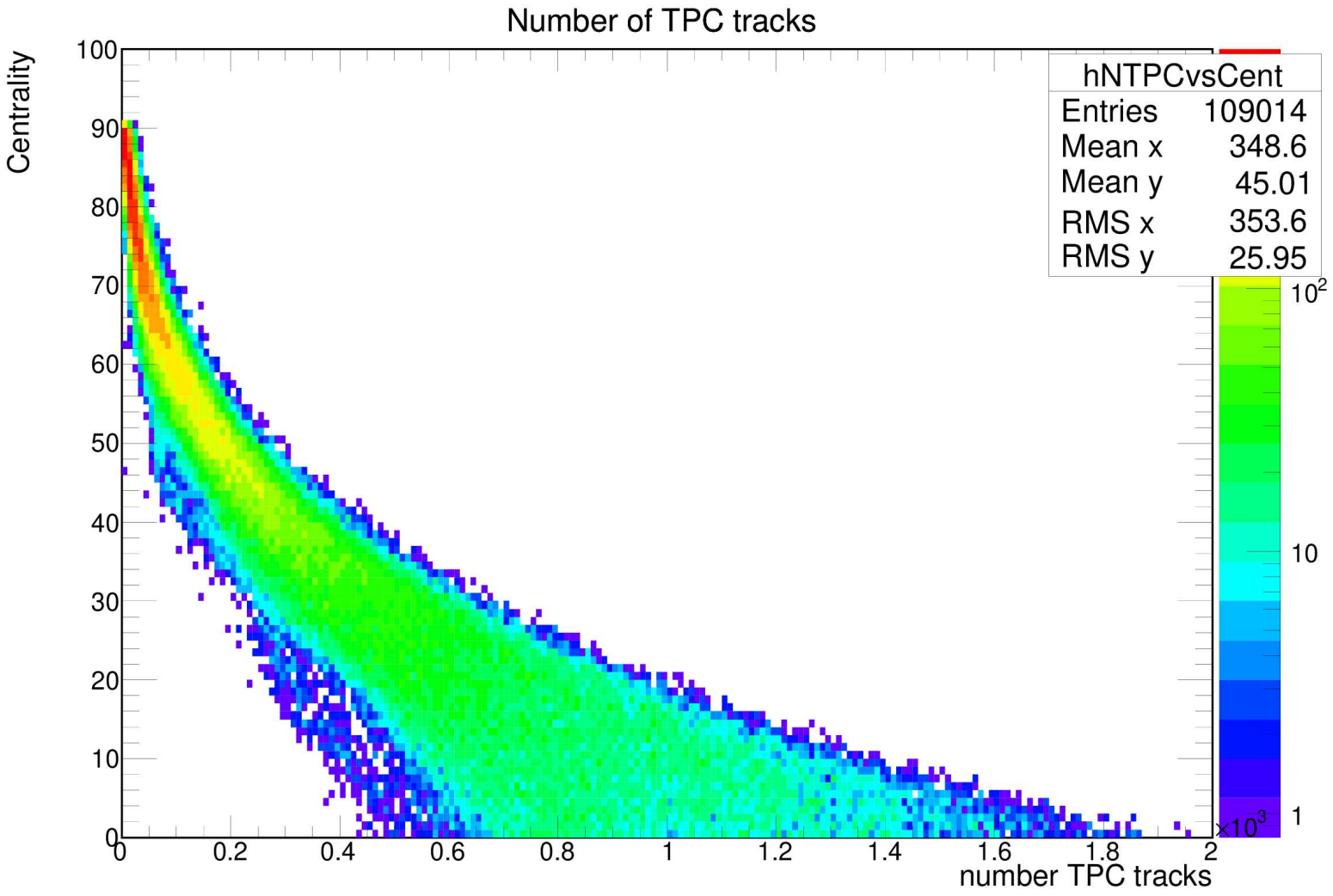
Automatisches Timeout nach ca. 10 Minuten ohne Aktivität!

Der Nukleare Modifikationsfaktor in ALICE

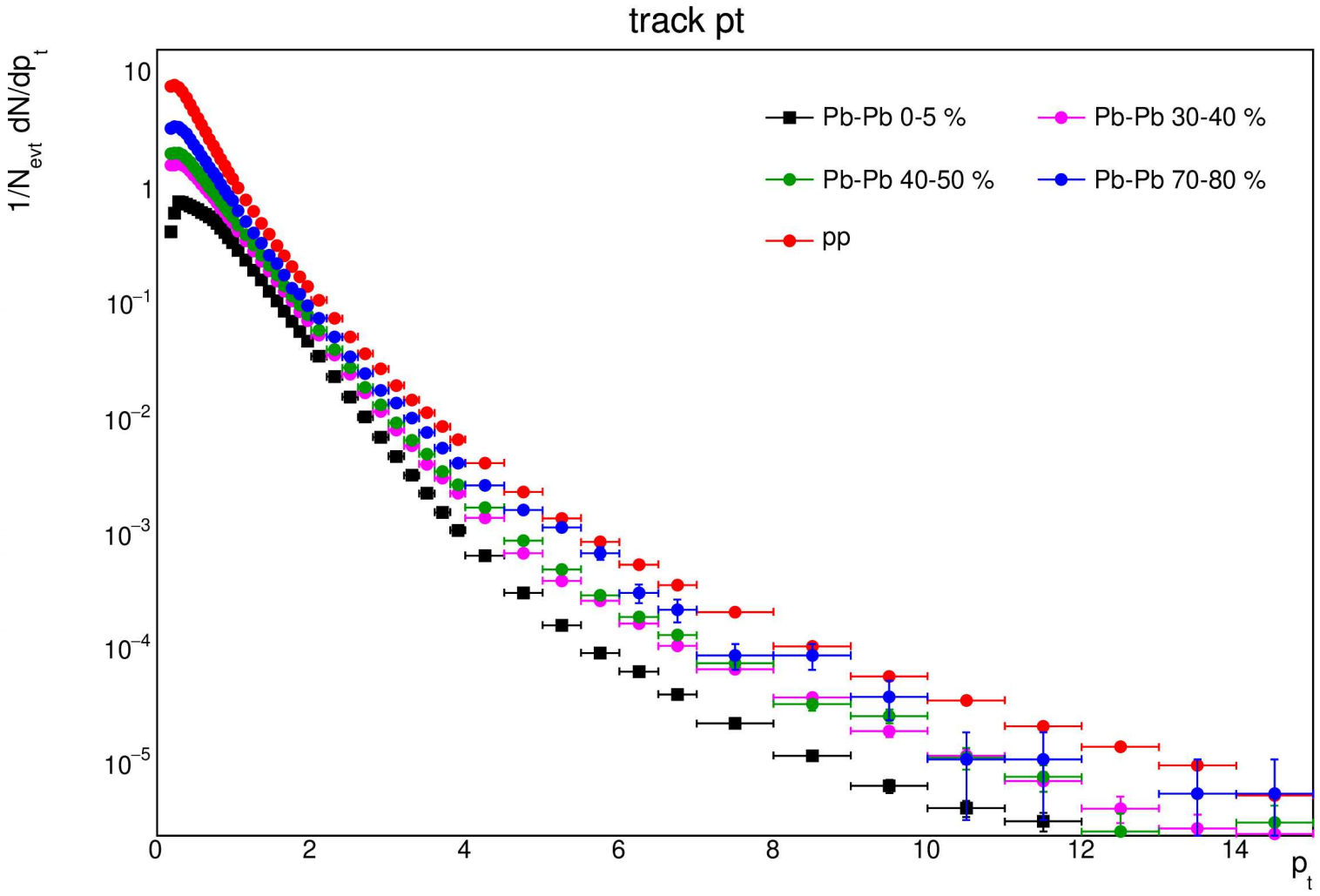
- Aufgabe: **Sehr viele** Kollisionen analysieren, die von ALICE aufgenommen worden sind
- Ziel: i) Darstellung der Multiplizitäten sowie Impulsspektren in pp und Pb-Pb Kollisionen
  ii) Bestimmung der nuklearen Modifikationsfaktoren R_{CP} und R_{AA}
- Weg: 1) (Kurze) Einführung in Python
  2) Objektorientierung in der Programmierung
  3) Los geht's....

Ergebnisse

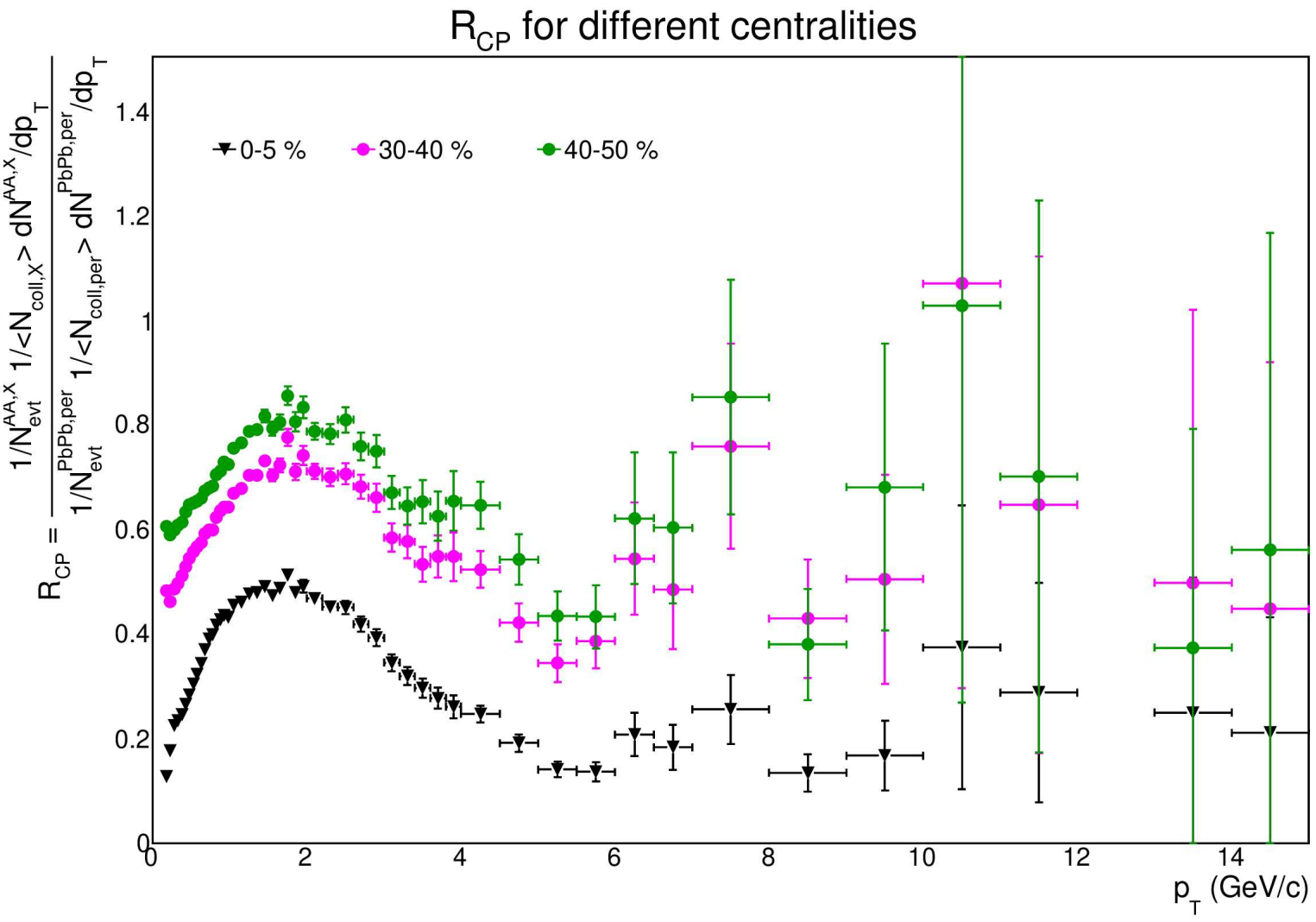
Teilchenspurmultiplizitäten



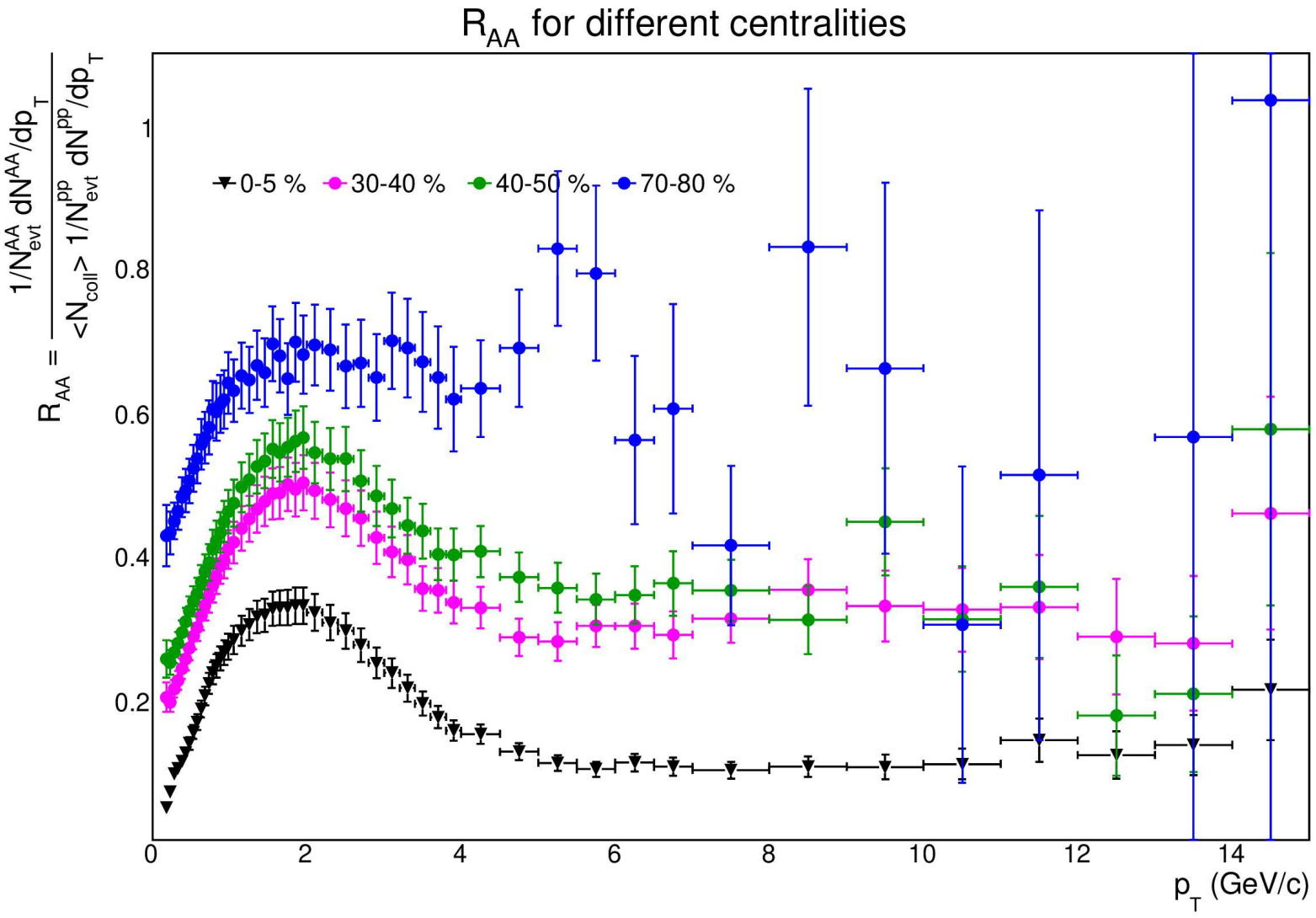
Transversalimpulsspektren



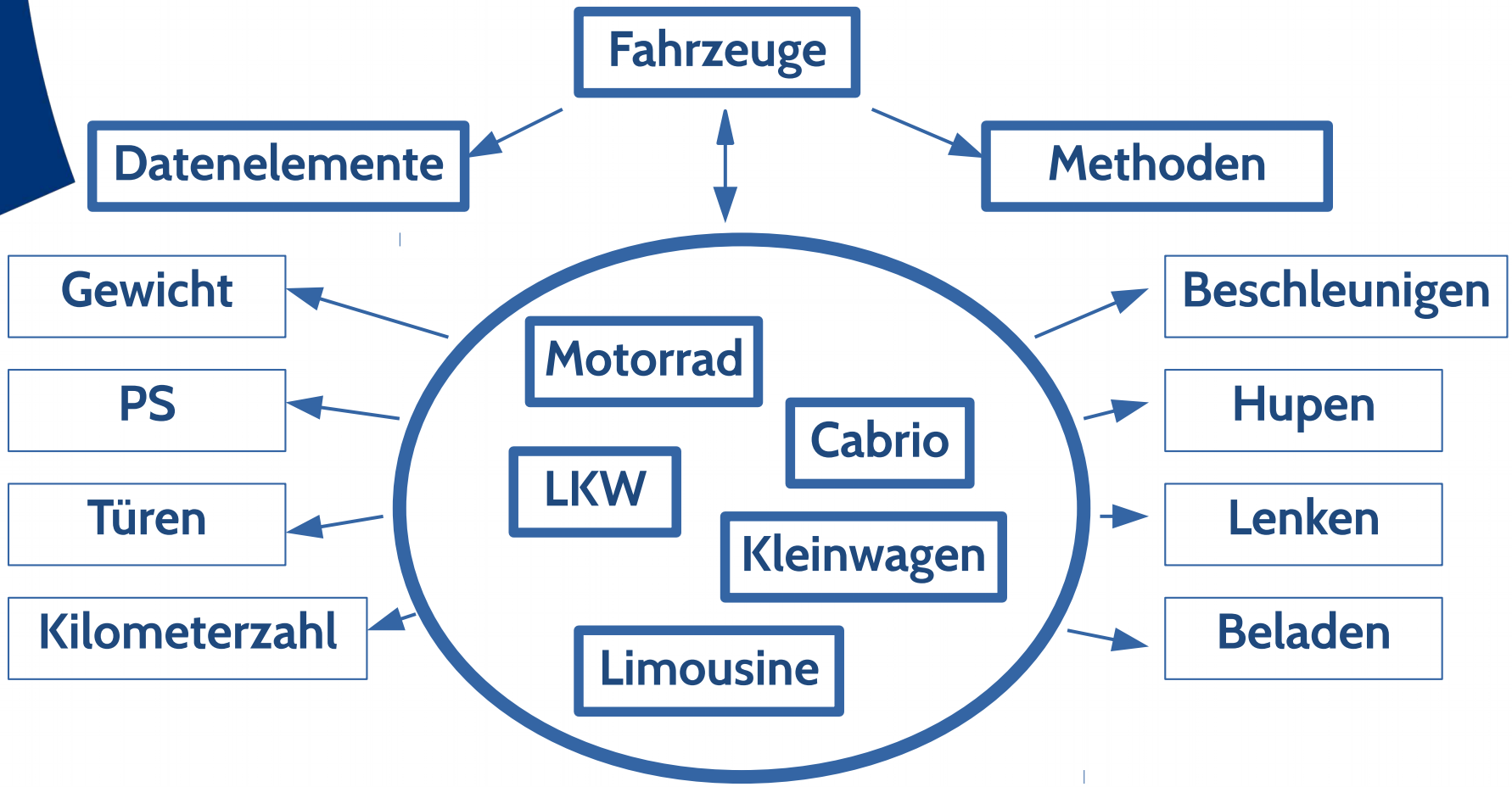
Nuklearer Modifikationsfaktor R_{CP}



Nuklearer Modifikationsfaktor R_{AA}

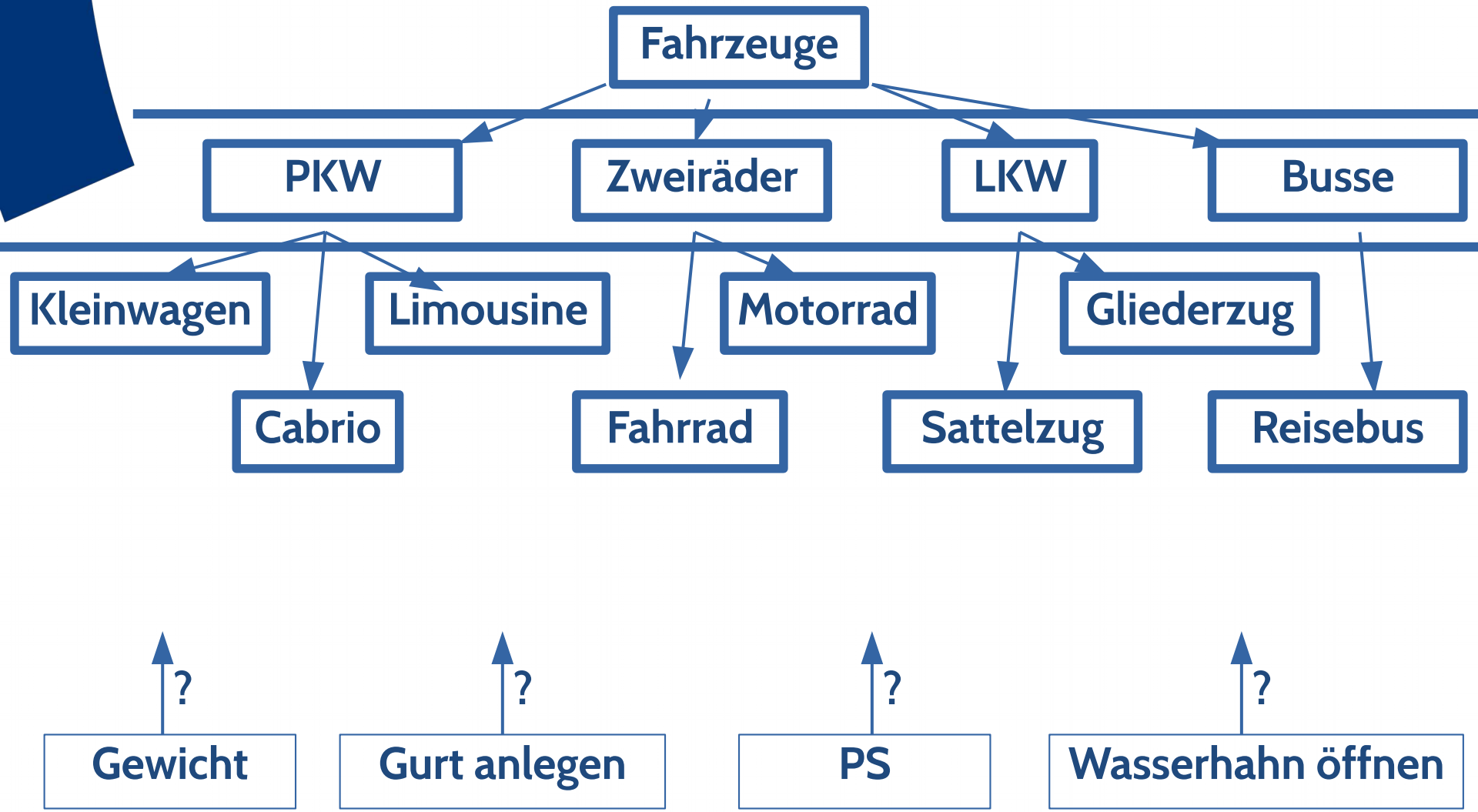


Objektorientierte Programmierung



Anhängerkupplung

Was passiert wenn?

Verdeck öffnen

Objektorientierte Programmierung