KIT-Fakultät für Physik
Institut für Experimentelle Teilchenphysik
Institut für Theorie der Kondensierten Materie
Prof. Dr. Ulrich Husemann (ETP)
Dr. Nils Faltermann (ETP)
Dr. Andreas Poenicke (TKM)

Karlsruher Institut für Technologie

Praktikum: Computergestützte Datenauswertung

Sommersemester 2021

Übungsblatt Nr. 2 Bearbeitung bis: 28.4., 18 Uhr

Sollten Sie Hilfe bei der Bearbeitung der Aufgaben benötigen, schauen Sie sich das Zusatzmaterial im Ordner "Vorlesungsmaterial" auf ILIAS an. Es ist gerade am Anfang normal, dass Sie sich nicht alle Befehle auswendig merken können und auch später werden Sie immer wieder einzelne Befehle oder Optionen nachschlagen müssen. Ihre Bearbeitung reichen Sie bitte als `ipynb`-Datei (eine Datei pro Aufgabe) bis zum Ablauf der Abgabefrist in ihrem Tutoriumsbereich auf ILIAS ein.

Aufgabe 2.1: Erste Schritte in Python

In der Vorlesung haben Sie die weitverbreitete Programmiersprache *Python* kennen gelernt. Außerdem kennen Sie bereits vom ersten Übungsblatt das Konzept der Jupyter-Notebooks, mit denen sich Python-Code interaktiv ausführen und dokumentieren lässt.

Erstellen Sie nun Ihr erstes eigenes Notebook und berechnen Sie dort die Fakultät $n!$ für die Werte $n = 2, \ldots, 10$. Implementieren Sie hierzu entweder die explizite oder rekursive Definition der Fakultät in Python und verwenden Sie keine externen Bibliotheken zur Berechnung. Lassen Sie sich jeweils die Wertepaare $(n, n!)$ auf dem Bildschirm ausgeben.

Hinweis zur Vorgehensweise beim Programmieren: Während der Programmentwicklung empfiehlt es sich, neu eingegebenen Code regelmäßig zu testen. Es ist auch üblich und sinnvoll, mit jeweils geeignet platzierten `print`-Befehlen zu überprüfen, ob wirklich genau das geschieht, was Sie sich vorgestellt hatten. Es ist normal, dass Python Sie gelegentlich mit Fehlermeldungen konfrontiert, die auf den ersten Blick nicht immer einsichtig sind. Korrigieren und testen Sie Ihren Code und bauen Sie Ihre Programme so schrittweise aus gut getesteten Einzelkomponenten auf, bis Sie am Ende ein zufriedenstellendes Gesamtergebnis erhalten.

Aufgabe 2.2: Datenauswertung: numpy

Vieles lässt sich bereits mit dem Standardbefehlssatz von Python erreichen. Im Bereich der Datenauswertung hat sich jedoch die Bibliothek `numpy` zu einem Standard entwickelt, da diese nicht nur zusätzliche Funktionen bietet, sondern auch, gerade bei größeren Datenmengen, deutlich schneller arbeitet.

Schreiben Sie ein neues Notebook und simulieren Sie den Wurf eines Würfels mit den Augenzahlen $\{1, \ldots, 6\}$ insgesamt 10 mal. Übeberlegen Sie sich, wie wahrscheinlich jeder Wert auftreten kann und welche Verteilung von Zufallszahlen Sie hierfür benötigen. Suchen Sie eine passende Funktion in der Dokumentation von `numpy` oder verwenden Sie die Ihnen aus der Vorlesung bekannten Beispielfunktionen. Speichern Sie die Häufigkeit jeder Augenzahl in einem `numpy.array` und lassen Sie sich die Werte ausgeben. Wiederholen Sie Ihre Simulation mit 100 bzw. 1000 Würfen. Erhalten Sie das erwartete Ergebnis?

Aufgabe 2.3: Darstellungen von Daten: matplotlib

Ein wichtiger Schritt bei der Datenauswertung ist die Visualisierung der Daten. Durch die Bibliothek `matplotlib` steht Ihnen hierzu ein mächtiges Werkzeug zur Verfügung, welches auch oft in Kombination mit `numpy` verwendet wird.

Stellen Sie eine Parabel $f(x) = x^2$ im Wertebereich $x \in [0, 5]$ grafisch dar. Erzeugen Sie "Datenpunkte" mit Unsicherheiten für $x \in \{1, 2, 3, 4\}$, die jeweils dem Wert $f(x)$ mit einer gau ssförmigen Unsicherheit von 10% des wahren Werts entsprechen. Tragen Sie die Datenpunkte mit Fehlerbalken in die eigene Grafik ein.

Tipp: Erzeugen Sie mit Hilfe von `numpy` ein `numpy.array` mit vier Zufallszahlen aus einer Standard-Normalverteilung. Wenn Sie diese Werte mit der gewünschten Unsicherheit (also $0.1 \cdot x^2$) multiplizieren, erhalten Sie die Zufallskomponente eines jeden Datenpunktes, die Sie zu den berechneten y-Werten addieren.

Hinweis: Ähnliche Problemstellungen, also der Vergleich von Daten mit einer Modellfunktion, treten in der Datenauswertung sehr häufig auf. Es lohnt sich daher, an Ihrem Code sehr sorgfältig zu arbeiten, damit Sie ihn oder Teile davon später weiter verwenden können. Trennen Sie auch die Erzeugung der darzustellenden Daten von der eigentlichen Darstellung; Sie erhalten dann Ihr eigenes, flexibles Grafik-Werkzeug für den Vergleich von Daten mit Modellen, auf das Sie immer wieder zurückgreifen können.