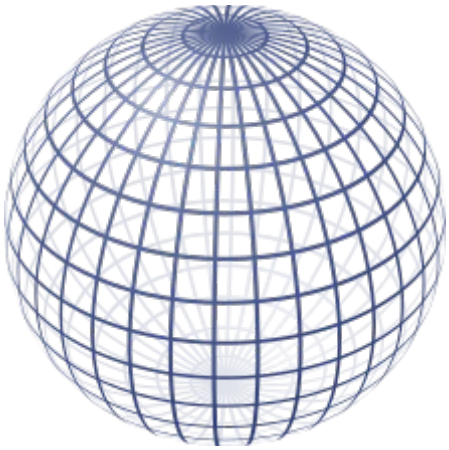
# LS 3 – Sensor-Monitoring-System

Sensordaten soll über einen längeren Zeitraum aufgezeichnet werden. Die Sensordaten lassen sich sich mit Python bestimmen. Für die dauerhafte Aufzeichnung kann man eine Datenbank nutzen. Können Python und Datenbank zusammenarbeiten?

**Aufgabe 1a - Kugelberechnung (I)**



Erstellen Sie ein Programm, welches nach Eingabe des Radius die Oberfläche und das Volumen einer Kugel ausrechnet. Die Berechnungen soll jeweils in einer separaten Unterfunktion erfolgen. Das Einlesen des Radius und die Ausgaben sollen im Hauptprogramm erfolgen. Speichere das Programm unter dem Namen **`Aufgabe_ls3_04a.py`** ab.

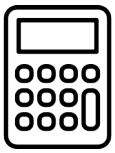
**Aufgabe 1b - Kugelberechnung (II)**

Modifizieren Sie das Programm aus (1a) so, dass das Einlesen des Radius und die Ausgaben ebenso in eigenen Unterprogrammen erfolgen. Speichere das Programm unter dem Namen **`Aufgabe_ls3_04b.py`** ab.

**Aufgabe 1c - Kugelberechnung (III)**

Modifizieren Sie das Programm aus (1b) so, dass es auf 3 Module für Steuerung, Funktionalität und Benutzerinteraktion aufgeteilt wird. Speichere das Programm unter dem Namen **`Aufgabe_ls3_04c...py`** ab.

## Aufgabe 2 – Modularer Taschenrechner



Erstellen Sie einen modular aufgebauten Taschenrechner.

Funktionale Anforderungen

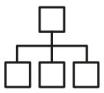
- Grundrechenarten ( +, -, *, /)
- Trigonometrische Funktionen ( sinus, cosinus, tangens, arcus sinus, arc cosinus, arc tangens)

Strukturelle Anforderungen

- Taschenrechner läuft in einer Schleife, bis der Benutzer ihn explizit beendet. In einem Schleifendurchlauf kann der Benutzer die Rechenart (Funktion) wählen, seine Zahleneingaben machen und das Ergebnis sich ansehen.
- Für jede Rechenart bzw. Rechenfunktion ist eine eigene Funktion zu schreiben.
- Alle Benutzerinteraktionen sollen in eigene Funktionen ausgelagert werden.
- Der Taschenrechner ist in 4 Module aufzuteilen. Ein Modul enthält alle trigonometrischen Funktionen, ein Modul enthält alle Grundrechenarten, ein Modul enthält nur Funktionen zur Benutzerinteraktion. Diese drei ersten Module enthalten nur Funktionen. Das vierte Modul enthält eine Steuerfunktion, die innerhalb des Moduls aufgerufen wird.

Speichern Sie das Programm unter dem Namen **`aufgabe_ls3_03....py`** ab. Dokumentieren Sie die Modul- und Funktionsaufrufhierarchie in geeigneten Diagrammen. Dokumentieren Sie die Steuerfunktion mit einem Struktogramm.

## Aufgabe 3 – Reflexion Modularisierung



a) Was versteht man bei der Programmierung unter einer Funktion?
b) Schreiben Sie eine Python Funktion welche zwei Zahlen addiert. Die beiden Summanden werden als Parameter mitgegeben, die Summe als Return-Wert zurückgegeben.
c) Kennzeichnen Sie an der Funktion aus (b) die Teile Funktionskopf, Funktionskörper, Funktionsname und Funktionsparameter.
d) Zeigen Sie mit einem Beispiel, wie man für Parameter Default-Werte festlegen kann.
e) Visualisieren Sie die Funktionsaufrufhierarchie von `beispiel_ls3_02.py` mit einem geeigneten Diagramm.
f) Mit welchem Schlüsselwort werden Funktionsdefinitionen in Python gekennzeichnet?
g) Python besitzt einen Satz von fest vordefinierten Funktionen (build in functions). Nennen Sie drei und beschreiben Sie kurz, was diese machen. Wo finden Sie Dokumentationen dieser Funktionen?
h) Was versteht man unter einem Modul? Wie ist der Zusammenhang zwischen Modul und Datei in Python?
i) Visualisieren Sie die Modulaufrufhierarchie von `beispiel_ls3_03....py` mit einem geeigneten Diagramm.
j) Visualisieren Sie die Funktionsaufrufhierarchie von `beispiel_ls3_03....py` mit einem geeigneten Diagramm.
k) Mit welcher Anweisung werden andere Module in Python eingebunden?

**Aufgabe 4 – Reflexion Testen**


a) Was ist der Unterschied zwischen einem White-Box-Test und einem Black-Box-Test?
b) Was versteht man unter einem Komponenten-Test?
c) Was unterscheidet Komponenten-Test und Integrations-Test?
d) Was versteht man unter einem Regressionstest?
e) Nennen Sie Werkzeuge, welche es zur Testautomatisierung bei Python gibt?
f) Welche Fehler werden typischerweise von einem Compiler erkannt?
g) Was versteht man unter Test-Driven-Development?

**Aufgabe 5 – Datenbankfunktionen extrahieren**

Bauen Sie `aufgabe_ls3_00.py` so um, dass die 4 allgemeinen Zugriffsfunktionen in ein eigenes Modul ausgelagert sind, die importiert wird. Die Funktion `dbZugriffsWerte` ist spezifisch und bleibt. Verlagern Sie den restlichen Skript-Code in eine Funktion. Speichern Sie das Datenbankmodul unter dem Namen **`dblib.py`** ab, das umgebaute Beispiel soll unter dem Namen **`aufgabe_ls3_01.py`**. Dokumentieren Sie die Modul- und Funktionsaufrufhierarchie.

**Aufgabe 5 – Sensor-Monitoring-System**

Sensordaten (drei Sensoren: Alpha-Wert, Beta-Wert, Gamma-Wert) sollen über einen längeren Zeitraum aufgezeichnet werden. Hierzu ist ein Aufzeichnungssystem zu erstellen, welches aus mehreren Komponenten besteht:

1. Einer zentralen Datenbank zur Speicherung der Sensordaten
2. Ein Python Programm, welches in regelmäßigen Intervallen die Sensordaten ermittelt (Zufallszahlen zwischen 0 und 1000 generieren) und diese in der Datenbank abspeichert.
3. Ein Python Programm, welches die in der Datenbank enthaltenen Werte anzeigt. Wenn möglich sollen sinnvolle Auswahlkriterien unterstützt werden.
4. Ein Python Programm, welches die in der Datenbank enthaltenen Werte als CSV-Datei exportiert, so dass eine Weiterverarbeitung und Visualisierung mit Hilfe eines Tabellenkalkulationsprogramms möglich ist.

Erstellen Sie eine aktuelle Übersichtsdokumentation bestehend aus Kontextdiagramm, Blockdiagramm mit Modulen sowie der Funktionsaufrufhierarchie.

Mögliche Erweiterungen:

- Erweitern Sie Komponente 3, so dass neben der Anzeige auch eine einfache statistische Auswertung möglich ist.
- Erweitern Sie Komponente 3, so dass auch ein Löschen der Daten möglich ist.

Hinweise:

- Benutzen XAMP Portable, zum Aufbau einer MySQL- bzw. MariaDB-Datenbank.
- Die Komponenten 3 und 4 können zu einer zusammengefasst werden.

- Speichern Sie die Programme unter dem Namen **`aufgabe_ls3_02a.py`**, **`aufgabe_ls3_02b.py`**, **`...`** ab. Das Skript zur Erstellung der Datenbank soll **`aufgabe_ls3_02.sql`** heißen.

**Aufgabe 7 – Reflexion Datenbanken**

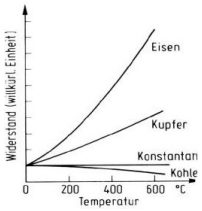
a) Was versteht man unter einer Datenbank? Was versteht man unter einem Datenbankmanagementsystem? Was versteht man unter einem Datenbanksystem?
b) In der Praxis werden häufig sogenannte relationale Datenbanken eingesetzt. Was versteht man hierunter?
c) Welche relationalen Datenbanken kennen Sie (Hersteller/Produkt)?
d) Was ist die Beziehung zwischen MySQL und MariaDB?
e) Was ist XAMPP?
f) Mit welchem in XAMPP enthaltenen Werkzeug können Sie die datenbank administrieren?
g) Gibt es auch nicht-rationale Datenbanken? Wenn ja, nennen Sie eine und worin unterscheidet sich diese von einer relationalen Datenbank?
h) Skizzieren Sie den Aufbau eines Python Programms, welches auf eine MariaDB-Datenbank zugreift.
i) Mit welcher Sprache programmiert man Zugriffe auf relationale Datenbanken? Welche Abkürzung wird hierfür verwendet und wofür steht die Abkürzung?
j) Mit welcher Anweisung kann man Daten in einer Datenbank abspeichern?
k) Mit welcher Anweisung kann man Daten aus einer Datenbank auslesen?
l) Welches Modul müssen Sie in Python importieren, um auf eine MySQL-/MariaDB-Datenbank zuzugreifen?

**Zusatzaufgaben (Binnendifferenzierung)**


## Aufgabe 8 - Lambda-Funktionen

In Python gibt es Lambda-Funktionen. Erarbeiten Sie ein Beispiel, aus dem hervorgeht, was Lamda-Funktionen sind und wie man sie nutzen kann.

## Aufgabe 9 - Widerstandsrechner




In letzter Zeit muss Klaus Merkenix öfter die Temperaturabhängigkeit von Widerständen berücksichtigen. Die von seinem Freund Hans Klugmeier zur Verfügung gestellten Formeln und Tabellen sind umständlich zu nutzen und ein Programm zur Berechnung würde viel Zeit sparen.

**Funktionale Anforderungen:**

- Erstellen Sie für Klaus Merkenix ein Programm zur Berechnung temperaturabhängiger Widerstände. Das Programm soll in zwei verschiedenen Situationen helfen:
  1. Der genormte Widerstandswert $R_{20}$, das Material und die Temperatur $\theta$ sind bekannt und es soll der temperaturabhängige Widerstand $R_\theta$ berechnet werden.
  2. Die Länge, der Querschnitt, das Material und die Temperatur $\theta$ sind bekannt und es soll der temperaturabhängige Widerstand $R_\theta$ berechnet werden.
- Als Hilfsmittel stehen die Formeln und Tabellen von Hans Klugmeier (FS-LF3-LS3-Text1) zur Verfügung.

**Strukturelle Anforderungen:**

- Der Wiederstandsrechner läuft in einer Schleife, bis der Benutzer ihn explizit beendet. In einem Schleifendurchlauf kann der Benutzer zwischen den beiden Varianten wählen, seine Zahleneingaben machen und das Ergebnis sich ansehen.
- Alle Formeln sind in eigenen Funktionen zu isolieren.
- Alle Benutzerinteraktionen sollen in eigene Funktionen realisiert werden.
- Der Widerstandsrechner ist in 4 Module aufzuteilen. Ein Modul enthält Funktionen, die die materialspezifischen Parameter liefern, ein Modul enthält alle benutzte Formeln, ein Modul enthält alle Funktionen zur Benutzerinteraktion. Diese drei ersten Module enthalten nur Funktionen. Das vierte Modul enthält die main-Funktion und je nach Zerlegung eine oder mehrere Steuerfunktionen.

**Vorgehensweise:**

a) Schritt 1: Erstellen Sie zuerst ein Kontextdiagramm, einen Entwurf der Benutzerschnittstelle, einen Funktionsbaum und die Funktionsprototypen. Dies soll auf Papier geschehen. Stellen Sie auch die benötigten Formeln zusammen, so dass klar ist, wie aus den Eingangsgrößen die Ausgangsgrößen berechnet werden können.
b) Schritt 2: Erstellen Sie das Programm und testen Sie es.

**Hinweise:**

- Dokumentieren Sie die Modul- und Funktionsaufrufhierarchie graphisch.

- Dokumentieren Sie die Steuerfunktion mit einem Struktogramm oder einem PAP.
- Speichern Sie das Programm unter dem Namen **`aufgabe_ls3_05....py`** ab.