| Abend | **Lernziel** | **Thema / Inhalt** | **Methode** | **Zeitbedarf** | **Hausaufgaben** |
| --- | --- | --- | --- | --- | --- |
| 1. Abend | RaspberryPi mit Sense-Hat in Betrieb nehmen und erstes «Hello.py» zur Ausführung bringen.  Ablaufstrukturen im Programm gezielt und richtig verwenden, *print()* und *input()* Funktionen sicher anwenden.  Umrechner.py (ohne eigene functions und ohne Bildschirmsteuerung) | **Vorstellung (Wer bin ich? Problem-Based Learning)**  **Installieren der Entwicklungsumgebung**   * VNC / Notepad++ / PuTTY / FTP Client * Nano auf RPi, die wichtigsten Befehle * 1.Progrogramm   + File erstellen und editieren (Console, Nano)   + print(), #! /usr/bin/pyhton3   + LINUX Befehle:     - cd, ls -al, pwd, rm, mv, cp, mkdir,     - LINUX file-system (chmod, filepath)   + Execution   + Fehlermeldungen interpretieren können und Lösungen implementieren * Notepad++, Putty * Programm erweitern (Print(), String-Operationen, Input())   **Aufgabe 1a (Umrechner.py)**   * Menu * User-Input (Wähle:) * If-then-elif-else Struktur * Loop mit 0 beenden * Behandlung von falsch Eingaben * Formeln implementieren (Variablen, Float-Input, Math-Operationen) * format() Methode | Nach Anleitung installieren und konfigurieren  Selber versuchen, Vormachen, Nachmachen mit theoretischen kurzen Einschüben | 10‘    60’  130’ | Umrechner.py alle Formeln implementieren und alle Menu-Punkte vollständig implementieren. |
| 2. Abend | Eigenen Funktion unter dem Aspekt der Re-Usability implementieren können.  Den Unterschied zwischen positional und named Parameters bei den Functions-Interfaces wie beim Aufruf sicher und gezielt anwenden können.  Exception-Handling in Python sicher anwenden und den Unterschied zwischen pre-condition check and exception sicher anwenden können. | **Aufgabe 1b (Umrechner.py)**   * Bildschirmsteuerung (cls(), halt()) implementieren * import math * Formeln in Funktionen implementieren * Funktionen abwärtskompatible erweitern * Exception Handling mit Pre-Checks und try-catch   **Aufgabe 1c (Umrechner.py, xxLibrary.py)**   * Funktion in eigene Library auslagern * Refactoring Umrechner.py verwendet eigene Library * Weitere Functionen readInt(), readFloat() implementieren, testen, anwenden und in eigene Library übernehmen. * Neuer Menu-Punkt: Quadratische Gleichung | Test-Driven Approach mit theoretischen Einschüben | 100‘  100‘ | Neue Funktionen entwickeln, testen und in eigene Lib übernehmen. |
| 3. Abend | LINUX Basics  Alle Methoden im SenseHat Module (gemäss API doc) erfolgreich selbst getestet.  Methoden Sense-Hat und Sense Klasse (API) mit LED Matrix verwenden. | **Leistungskontrolle 1**  **Aufgabe 2a (LED\_Matrix.py)**   1. setPixel(), setPixels(), clear(), sleep(), showMessage() 2. Eventhandling (Joystick) 3. IMU- und Meteo-Sensoren   **Aufgabe 2b (xx\_SenseHat\_Librarie.py)**   1. setPixel() mit clipping 2. drawLine(), drawRecantgle(), drawCircle() 3. Functions erweitern mit fillColor und borderColor 4. drawCompassNeedle(azimutInGrad) | Moodle Test  Test-Driven Approach mit theoretischen Einschüben  Test-Driven Approach mit theoretischen Einschüben | 15’  60‘  140‘ | Design und Implementation eines analogen **Kompasses (mit Nadel)** |
| 4. Abend | Containers in Python kennen und in eigenen Applikationen anwenden können. | Elemete in den verschiedenen Containers zugreifen (lesen), zufügen/ändern und löschen. Listen[], Tupels(), Dictonaries{}  Sub-Listen mit [1:-1] ranges lesen resp verarbeiten/ändern.  for – Loops   * Listen und Tuples * Dictonaries (keys())   Comprehensions mit Filter und ZIP für eigene Anwendungen einsetzen können. | Probieren, Vormachen, Nachmachen mit theoretischen kurzen Einschüben | 90‘  110‘ | **Meteo-App**  oder einer **Snake-App** oder  Linien Aufgaben |
| 5. Abend | Containers  Algorithmen in Funktionen umsetzen  Parameterübergaben \* (listen) \*\* (dictonaries) | **Leistungskontrolle 2**   * Fakultät * Primzahlen Rechner * Primzahlen und Teiler Listen * Filter-Berechnungen | Formativer Test  Probieren, Vormachen, Nachmachen mit theoretischen kurzen Einschüben | 40’  160‘ |  |
| 6. Abend | REST-Service mit JSON Response nutzen  Filehandling und direct EXCEL Zugriff erfolgreich anwenden | Open-Weather REST Service mit eigenem Token (AppID) aus Python aufrufen (requesten) und response als JSON Struktur verarbeiten.  Filehandling open() for read, write and append (inkl UTF and ASCII) | Probieren, Vormachen, Nachmachen mit theoretischen kurzen Einschüben | 40‘  40‘  120‘ | Design und Implementation einer **Meteo-Logger**  (Wetterstation), welche Metoe-Daten von einem Ort / Lokation optimiert und ohne «Löcher» loggen. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 7. Abend | Klassenkonzept in Python in einer konkreten Anwendung kennen lernen und anwenden können. | Eine eigene, allgemein einsetzbare Logger-Klasse gemäss Spezifikation entwickeln und testen.  Anschliessend eigene Logger-Klasse in Meteo-App einsetzen. | Probieren, Vormachen, Nachmachen mit theoretischen kurzen Einschüben | 200‘ |  |
| 8. Abend | Multi-Treathing und Timer-Events in Python kennen lernen.  Eine Wrapper-Class für einen Wetterdienst allgemein und nach OO Ansätzen designen und implementieren. | **Leistungsnachweis (Modullernzielkontrolle MILZ)**:  Eine allgemeine Weather-Class designen und implementieren, welche eine Wetterstation an einem bestimmten Ort kapselt. | Konzept anhand einiger Beispiele erklären (Walk-Through)  Selbststänges programmieren und individuelle Reviews durch Dozenz. | 10‘  190’ |  |
| 9. Abend | REST Service komponieren aus bestehendeREST Services | **Gebäudeautomation**  Selecta-Automat steuern  Rolladensteuerung anhand Wettervorhersagen  PiPlates  Shellys | Probieren, Vormachen, Nachmachen mit theoretischen kurzen Einschüben | 200‘ |  |

|  |  |
| --- | --- |
| **Bemerkungen:** | * Jeder Abend dauert 4 Lektionen. * Der Unterrichtsplan kann bei Bedarf dem vorhandenen Wissen der Klasse angepasst werden. * Die Studierenden lösen die Übungen auf ihren privaten Notebooks. * Der Leistungsnachweis am 8.Abend ist in Einzelarbeit in der vorgegebenen Zeit zu erstellen |