| **Thema / Inhalt** | **Methode** | **Zeitbedarf** | **Hausaufgaben** |
| --- | --- | --- | --- |
| 1.Abend   * Sie können grundlegende UNIX-Befehle erklären und deren Bedeutung für die Python-Entwicklung beschreiben (K2). * Sie können den Aufbau einer Entwicklungsumgebung für Python auf einem BYOD-Gerät erklären (K2). | | | |
| **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 (f-Strings, Multiline-Strings) * User-Input (Wähle:) * If-then-elif-else Struktur * Loop mit 0 beenden * Behandlung von falsch Eingaben * Formeln implementieren (Variablen, Float-Input, Math-Operationen) | Nach Anleitung installieren und konfigurieren  Test-Driven Approach mit Reviews und Refactoring  Theoretischen Einschüben | 10‘    60’  130’ | Umrechner.py Menu implementieren |
| 2.Abend   * Sie können einfache Python-Skripte schreiben, um Daten zu verarbeiten und Funktionen effizient einzusetzen (K3). * Sie können API-Dokumentationen analysieren und daraus ableiten, wie externe Pakete in eigene Anwendungen integriert werden können (K4). | | | |
| **Aufgabe 1b (Umrechner.py)**   * 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 implementieren (Return container) | Test-Driven Approach mit Reviews und Refactoring  Theoretischen Einschüben    Test-Driven Approach mit Reviews und Refactoring  Theoretischen Einschüben | 100‘  100‘ | Umrechner.py and My\_XXX\_Library.py fertigstellen |
| 3.Abend   * Sie können verschiedenen Containers für Python kritisch vergleichen und Empfehlungen aussprechen (K5). * Sie können JSON- und XML-Daten verarbeiten (K3). | | | |
| **Leistungskontrolle 1 (Linux Commands)**  Elemete in den verschiedenen Containers zugreifen (lesen), hinzufügen/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.  Umrechner Erweiterungen:   * Quadratische Gleichung * Fakultät * Primzahlen Rechner * Primzahlen und Teiler Listen * Filter-Berechnungen | Formativer Test  Test-Driven Approach mit Reviews und Refactoring  Theoretischen Einschüben | 15’  60‘  140‘ | Weitere Funktionen für **umrechner.py** implementieren |
| 4.Abend   * Sie können die Funktionen des Moduls Sense\_Hat anhand der API Dokumentation richtig anwenden (K4). | | | |
| Linien Aufgaben lösen  **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) | SOL  Test-Driven Approach mit Reviews und Refactoring  Theoretischen Einschüben | 90‘  110‘ | Sense\_Hat Library mit setPixel(), drawLine(), drawCircle(), drawRectangle() |
| 5.Abend   * Sie können die Architektur einer Python-Anwendung analysieren, um Verbesserungspotenziale zu erkennen (K4). * Sie können verschiedene Programmieransätze bewerten und deren Effizienz vergleichen (K5). * Sie können JSON Responses von REST-Calls für eine Steuerung auswerten. (K4). | | | |
| **Leistungskontrolle 2**  Dictonaries / JSON-Strukturen | Formativer Test  Test-Driven Approach mit Reviews und Refactoring  Theoretischen Einschüben | 40’  160‘ |  |
| 6.Abend   * Sie können REST-Services in Python aufrufen und deren Respons-Daten (JSON, XML,….) in einer Anwendung verarbeiten (K3). * Sie können Filehandling-Mechanismen anwenden und bewerten (K5). | | | |
| 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) | Test-Driven Approach mit Reviews und Refactoring  Theoretischen Einschüben | 40‘  40‘  120‘ | Design und Implementation einer **Wetter-Logger**  (Wetterstation), welche Meteo-Daten von einem Ort / Lokation in ein File schreibt |
| 7.Abend   * Sie können die Vor- und Nachteile objektorientierter Programmierung in Python bewerten (K5). * Sie können eine eigene Klasse designen und diese modular in einer Anwendung nutzen (K3). | | | |
| Eine eigene, einfache Logger-Klasse mit log\_msg() ohne Strategien entwickeln und testen.  Anschliessend eigene Logger-Klasse in Meteo-App einsetzen. | Test-Driven Approach mit Reviews und Refactoring  Theoretischen Einschüben | 200‘ | Eigene einfache Loggerklasse in Wetter\_Logger verwenden |
| 8.Abend   * Sie können ein Konzept für die Arbeitsteilung in einem Python-Projekt entwickeln und umsetzen (K4). * Sie können die Effizienz und Sicherheit verschiedener Methoden für den Zugriff auf Raspberry Pi-Systeme beurteilen (K5). | | | |
| **Leistungsnachweis (Modullernzielkontrolle MILZ)**:  Eine Aufgabe unter Zeitdruck gemäss Spezifikationen nach dem Test-Driven Approach implementieren. | Selbstständiges programmieren und individuellen Review durch Dozenten. | 190’ |  |
| 9.Abend   * Sie können Bilder und PDFs entwickeln (K4). * Sie können unterschiedliche Automatisierungsansätze bewerten und für spezifische Szenarien adaptieren (K5). | | | |
| **Fachgespräche über MLZ**  **Bildbearbeitung mit Python** | Einzelgespräche gemäss Zeitplan  Selbstorganisiertes Lernen (SOL) | 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 (**MLZ**) am 8.Abend ist in Einzelarbeit in der vorgegebenen Zeit zu erstellen |