



Tag 5 Module und evt. Exceptions

17. März 2023



Ablauf

- Rückblick Vierter Tag
- Prüfung besprechen
- Erarbeiten und Präsentationen einiger wichtigen Module:
 - datetime
 - os
 - platform
 - path
 - random
 - requests
- Übung mit Einschüben
 - Module erstellen und verwenden
 - Exception Handling



Vorschau Übung

Kommando-Interpreter

- 1 ask for an amount of days and calculate and print the date from now (now + days)
- 2 ask for an amount of days and calculate and print when that date was (now days)
- 3 ask for a date and an amount of days to calculate and print a second date
- 4 ask for two dates and calculate and print the time delta in days
- 10 print current directory
- 11 print items of current directory ('ls'/'dir')
- 12 move one directory up ('cd ..')
- 13 ask for a sub-directory name and go into ('cd xyz')
- 30 print computer name
- 31 print operating system
- 32 print platform info
- 33 print Python version
- 40 download a web page and print it
- ? print this instructions
- e exit program



Gruppenarbeit Themen

w3schools	docs.python
(w3schools)	docs.python
·	docs.python
w3schools	docs.python
w3schools	readthedocs
w3schools	docs.python
	w3schools w3schools



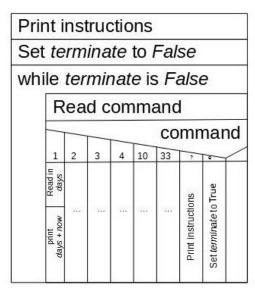
Ablauf

- Inhalt der Module erfassen, kondensieren und aufbereiten
- Kurzvortrag, 5-10 Minuten
 - Überblick
 - Vorstellung einzelne Funktionen
- Handout (Pdf, Word) 1-2 Seiten



Übung

- Kommando-Interpreter
 - 1) Gibt Instruktionen aus
 - 2) Lässt den Benutzer Funktion auswählen (Eingabe Zahl)
 - 3) Führt gewählte Funktion aus
 - 4) Gehe zu Schritt 2
- Erweiterung
 - Funktion 40 (download web page) \rightarrow pip install requests
 - Prüfe die Kommando-Zeilen Argumente (sys.argv). Falls dem Skript eine Zahl als Argument mitgegeben wurde, führe diese Funktion aus und beende das Programm





Anforderungen und Vorgehen

- Funktionen nach Thema in Module aufteilen (1-4, 10-13, usw.)
- Empfohlenes Vorgehen
 - 1) Main-Loop konstruieren
 - 2) Module mit je einer Funktion pro Thema erstellen, ohne 40 (z. B. 1, 10, 20, usw.)
 - 3) Weitere Funktionen hinzufügen



Exceptions

- Weshalb 'Ausnahmen'?
- Abfangen
- Auslösen
- Vordefinierte Exceptions



Exceptions

- Dienen der Fehlerbehandlung
- Lesen einer Datei: Datei kann ungültig sein (open() misslingt) oder während dem Lesen verschwinden, weil z. B. der USB-Stick aufgesteckt wird (readline() misslingt)

```
input_file = open('Data/2_test.txt', 'r')
while True:
    line = input_file.readline()
    if not line: break
    print(line.rstrip())
```



Alternative

- Zusätzlich zum eigentlichen Resultat (input_file resp. Line) müsste auch immer noch ein Status zurückgeliefert werden
- Dieser Status müsste jeweils auch immer geprüft werden
- Und würde ein zusätzliches Abbruchkriterium sein
 - → Code wäre vermutlich 2x so lange und deutlich schwieriger zu verstehen
 - → → produktiver> Code und Fehlerbehandlungscode vermischt



Fangen von Exceptions

```
try:
 code>
except:
 <Fehlerbehandlung>
try:
    input_file = open('test.txt', 'r')
   while True:
        line = input file.readline()
        if not line: break
        print(line.rstrip())
except: # catch all Exceptions
   print(f"reading file failed")
```



Werfen von Exceptions

raise <ExceptionType>(arguments)

```
def my_function(number):
    if not isinstance(number, int):
        raise TypeError("'number' must be a int")
    if number<0:
        raise ValueError("'number' must be positive")
    return number + 1</pre>
```



Exception Typen

(Auswahl)

TypeError Ungültiger Typ ValueError Ungültiger Wert

KeyError Schlüssel nicht vorhanden, z.B. in einem Dictionary IndexError Index ausserhalb des gültigen Bereiches, z.B. Liste

IOError Ein-/Ausgabe Fehler

Mehr: https://www.tutorialsteacher.com/python/error-types-in-python