

# SWP-PYTHON 5B

WS+SS 2025

Sabo RUBNER - RubnS - [sabo.rubner@htlinn.ac.at](mailto:sabo.rubner@htlinn.ac.at)

5. November 2025

# Übersicht

SWP 5B - Python

Test-Termine

Stoff-Übersicht

# Test-Termine

▶ siehe WebUntis

# Github Folien Link

► <https://github.com/htlinnrain/python>

# Git Repos G1

- ▶ Achrainer: [https://github.com/achrainer1112/SWP\\_Python5BHWII.git](https://github.com/achrainer1112/SWP_Python5BHWII.git)
- ▶ Andre: [https://github.com/benji0700/SWP1\\_Benjamin\\_Andre.git](https://github.com/benji0700/SWP1_Benjamin_Andre.git)
- ▶ Baumann: [https://github.com/baumann06/swp\\_web\\_python.git](https://github.com/baumann06/swp_web_python.git)
- ▶ Bobchev: <https://github.com/joanabcv/SWP-Rubner-Bobchev.git>
- ▶ Drexel: [https://github.com/annabelldrx/SWP\\_RUBNER\\_DREXEL](https://github.com/annabelldrx/SWP_RUBNER_DREXEL)
- ▶ Egger: [https://github.com/HannsDerKanns/SWP\\_25.git](https://github.com/HannsDerKanns/SWP_25.git)
- ▶ Embacher: <https://github.com/r6g3/Python-SWP-2025-26>
- ▶ Fitz: [https://github.com/derFitzi/SWP\\_25](https://github.com/derFitzi/SWP_25)
- ▶ Huber: <https://github.com/irihuber/Iris-Huber.git>

# Git Repos G2

- ▶ Ladinig: <https://github.com/raphaelladinig/sabo>
- ▶ Lindebner: <https://github.com/melli0-0/SWPPython25-26>
- ▶ Mrkonja:  
<https://github.com/Gmrkonja/SWP-PY-GMrkonja-5BHWII.git>
- ▶ Müller: <https://github.com/davidmuellerat/Rubner2025.git>
- ▶ Quickner:  
[https://github.com/DevilsBestFriend/SWP\\_Rubner\\_25\\_26.git](https://github.com/DevilsBestFriend/SWP_Rubner_25_26.git)
- ▶ Raffener: [https://github.com/junox0103/SWP\\_Rubna.git](https://github.com/junox0103/SWP_Rubna.git)
- ▶ Schennet: <https://github.com/deSilvasan/SWP2025-26>
- ▶ Steiner: [https://github.com/HendrikSteiner/SWP\\_Python\\_25-26](https://github.com/HendrikSteiner/SWP_Python_25-26)
- ▶ Vorhofer: [https://github.com/vorchris/SWP\\_25\\_26](https://github.com/vorchris/SWP_25_26)
- ▶ Wohlgemuth:  
<https://github.com/Werzu25/SWP-2025-26-Wohlgemuth.git>

# Stoff-Übersicht

- ▶ Python Theorie und einzelne Aufgaben
- ▶ verkettete Listen mit Python
- ▶ 2tes HJ individuelles Projekt in Python
- ▶ 2tes HJ Software Patterns wiederholen evtl. in ein anderem SWP-Fach
  - ▶ Singleton
  - ▶ Factory
  - ▶ Observer
  - ▶ Proxy
  - ▶ Strategy
  - ▶ Command
  - ▶ Composite

# Ablauf KW37

- ▶ Einführung
- ▶ Folien via Dropbox-Share-Ordner
- ▶ Git Repo für 2023 SWP-PYTHON anlegen
- ▶ Python2/Python3 (in P3 mit Abwärtskompatibilität gebrochen)
- ▶ Python downloaden, HalloWelt programmieren
- ▶ iterativ und rekursiv besprechen
- ▶ Unterschied Prozesse Threads



# Ablauf KW39

- ▶ Schülergruppen Einteilung kontrollieren webuntis
  - ▶ keine Datentypen (type) dynamische/statische Typisierung
  - ▶ aber type hints
  - ▶ funktionale und objektorientierte Programmierung möglich
- ▶ NichtInterpreter und Interpreter Sprachen
- ▶ Cpython (normal), PyPy (eigene JIT), Jython (benutz JVM), IronPython (.NET)
- ▶ Interpreter in der Kommandozeile
- ▶ Was ist eine Instanz/Referenz//Referenzzähler/GarbageCollection/del
- ▶ <https://www.learnpython.org/>
- ▶ <https://www.programiz.com/python-programming>
- ▶ python syntax kontrollstrukturen (if, schleifen, break, pass, try-except) besprechen und erarbeiten je ein Beispiel
- ▶ bedingte Ausdrücke
  - [https://openbook.rheinwerk-verlag.de/python/05\\_001.html](https://openbook.rheinwerk-verlag.de/python/05_001.html)
- ▶ <https://www.datacamp.com/tutorial/python-switch-case>
- ▶ <https://www.programiz.com/dsa/binary-search> wegen Syntax besprechen

## KW40

```
#include <stdio.h> // includes "printf" command ;

int add(int varA, int varB ) {
    // inline assembler starts here
    __asm {
        mov eax, varA    ; // move first argument to EAX register
        mov ecx, varB    ; // move second argument to ECX register
        add eax, ecx      ; // EAX = EAX + ECX
    }
}

int main() {
    int result = add(1, 2);    // call C-function "add"
    printf( "1 + 2 = %d\n", result); // output result to console
    return 0;
}
```

Abbildung: Assembler und C++ Beispiel

# KW40

- ▶ Wiederholung
- ▶ <https://www.youtube.com/watch?v=fM409bModsE&t=13s>
- ▶ Arraylist in java, listen, echte arrays
- ▶ Zuordnungs Datenstrukturen (dict)  
[https://openbook.rheinwerk-verlag.de/python/14\\_001.html](https://openbook.rheinwerk-verlag.de/python/14_001.html)
- ▶ Aufgabe Programmiere Lottoziehung als Methode:
  - ▶ `random.getrand()`
  - ▶ Algorithmus zum Zufallszahlenziehen muss so programmiert sein, dass keine Zufallszahl zweimal gezogen werden kann.
  - ▶ Das heisst, wenn Sie alle 45 Zahlen ziehen müssten, Würden Sie den Zufallszahlengenerator nur 45 mal benutzen dürfen.
  - ▶ Ziehe die sechs Zahlen und gib Sie am Bildschirm aus
  - ▶ Aufgabe Programmier Lottoziehung Statistik als Methode:
  - ▶ Mach 1000 Ziehungen
  - ▶ Erstelle Dictionary für Statistik, wie oft welche Zahl gezogen wurde
  - ▶ Ruf die Statistikmethode nach jede Ziehung auf und inkrementiere den Zähler

# KW41

- ▶ Lotto aufgabe besprechen
- ▶ interaktiver Modus  
[https://www.python-kurs.eu/python3\\_interaktiv.php](https://www.python-kurs.eu/python3_interaktiv.php)
- ▶ Datentypen (ganze Zahlen (Division?), Gleitkommazahl, Zeichenketten (addieren), Listen, Dictionarys)
- ▶ divisions <https://www.codingninjas.com/studio/library/floor-division-in-python>
- ▶ Variablen sind case sensitiv
- ▶ Schlüsselwörter  
[https://openbook.rheinwerk-verlag.de/python/A\\_001.html](https://openbook.rheinwerk-verlag.de/python/A_001.html)
- ▶ logische Ausdrücke e.g. and / or / lt / eq
- ▶ Programmdateien Endung \*.py
- ▶ Shebang #!/usr/bin/env python3 (MagicLine UNIX)  
<https://www.delftstack.com/de/howto/python/python-shebang/>
- ▶ keine TABS, nur 4 Leerzeichen als Einrückung, Lange Zeilen umbrechen max 80Zeichen (Backslash benutzen)
- ▶ Kommentare mit # oder drei Anführungszeichen, pass Anweisung
- ▶ list comprehensions  
<https://realpython.com/list-comprehension-python/>

# KW42

- ▶ Division ausprobieren `a / b` und `a // b` (floor division)
- ▶ Auswertungsreihenfolge der Operatoren
- ▶ `https://cito.github.io/byte_of_python/read/operator-precedence.html`
- ▶ Referenz/Instanz (besteht aus Identität/Typ/Wert)
- ▶ `type` ausprobieren
- ▶ Identität ist Ganzzahl und wie ein Fingerabdruck vergleich mit `is`
- ▶ `https://openbook.rheinwerk-verlag.de/python/07_001.html`
- ▶ `id(ref1) == id(ref2)` oder `ref1 is ref2`
- ▶ Instanzen freigeben mit `del`
- ▶ `https://openbook.rheinwerk-verlag.de/python/07_002.html`
- ▶ `https://www.geeksforgeeks.org/garbage-collection-python/`,  
`https://stackify.com/python-garbage-collection/`
- ▶ `mutable/immutable`
- ▶ `https://openbook.rheinwerk-verlag.de/python/07_003.html`
- ▶ `tuple`, `range`, `set`
- ▶ `https://openbook.rheinwerk-verlag.de/python/13_001.html`

# KW43

- ▶ Dict-Comprehension Aufgabe ansehen
- ▶ Pokerspielsimulator als Aufgabe über Herbstferien:
  - ▶ überlege wie man die Pokerkarten modellieren könnte (vier Farben, 13 Symbole)
  - ▶ gib zufällig fünf Karten
  - ▶ recherchiere welche Kombinationen beim Pokerspiel existieren
  - ▶ schreibe Funktionen für die Kombinationen Paar, Drillinge, Poker, Flash, Strasse usw.
  - ▶ spiele 100000 mal und zähle die Anzahl der verschiedenen Kombinationen
  - ▶ berechne den prozentuellen Anteil einer Kombination zu der Gesamtspieleanzahl
  - ▶ recherchiere die richtige Anteile im Netz und vergleiche die Ergebnisse
  - ▶ Git!
  - ▶ Codingtips beherzigen!!!

# KW45

- ▶ Stoff Wiederholung?
- ▶ isinstance
- ▶ [https://www.w3schools.com/python/ref\\_func\\_isinstance.asp](https://www.w3schools.com/python/ref_func_isinstance.asp)
- ▶ arrays <https://www.geeksforgeeks.org/python-arrays/>
- ▶ Poker besprechen

# KW46

- ▶ TEST
- ▶ Poker abgabe und besprechen



# KW47

- ▶ Compiler-Interpreter <https://www.data-science-architect.de/python-compiliert-interpretiert/>
- ▶ mutable-immutable <https://www.data-science-architect.de/mutable-und-immutable-objects/>
- ▶ [https://www.python-kurs.eu/python3\\_deep\\_copy.php](https://www.python-kurs.eu/python3_deep_copy.php)
- ▶ <https://www.python.org/dev/peps/pep-0008/>
- ▶ praxisbezogene Beispiele:  
<https://www.delftstack.com/de/howto/python/>
- ▶ eingebaute Funktionen:  
[https://openbook.rheinwerk-verlag.de/python/19\\_008.html#u19.8](https://openbook.rheinwerk-verlag.de/python/19_008.html#u19.8)
- ▶ DUNDER (double underscore) Variablen, kommen bei Objektorientierung
- ▶ Exceptions raisen  
[https://www.w3schools.com/python/python\\_try\\_except.asp](https://www.w3schools.com/python/python_try_except.asp)
- ▶ Debugging scripts with pdb  
<https://docs.python.org/3/library/pdb.html>
- ▶ Debugging scripts with pdb  
<https://realpython.com/python-debugging-pdb/>
- ▶ Ternärer Operator  
<https://www.geeksforgeeks.org/ternary-operator-in-python/>

# KW47

- ▶ Pokererweiterungen ansehen
- ▶ `lambda` <https://databasecamp.de/python/python-lambdas>
- ▶ Was ist der Unterschied zwischen `lambda` und `def`?
  - ▶ `def` ist eine Anweisung (Statement)
  - ▶ Eine Anweisung führt eine Aktion aus und hat keinen Wert.
  - ▶ Es erzeugt ein Objekt vom Typ `function`, aber die Definition selbst ist kein Ausdruck.
  - ▶ `lambda` ist ein Ausdruck (Expression)
  - ▶ Ein Ausdruck hat einen Wert und kann direkt ausgewertet werden.
  - ▶ Es kann in jeder Position verwendet werden, wo ein Ausdruck erwartet wird.
- ▶ `map`+`lambda` <https://www.digitalocean.com/community/tutorials/how-to-use-the-python-map-function-de>
- ▶ `filter` function <https://www.ionos.de/digitalguide/websites/web-entwicklung/python-filter-function/>
- ▶ `zip` function  
<https://marketmix.com/de/python-die-zip-funktion-erklaert/>
- ▶ Virtual Environments  
<https://realpython.com/python-virtual-environments-a-primer/>

# KW48

- ▶ Wiederholung
- ▶ Python objektorientiert anfangen
- ▶ <https://realpython.com/python3-object-oriented-programming/>
- ▶ [https://realpython.com/python-classes/  
#special-methods-and-protocols](https://realpython.com/python-classes/#special-methods-and-protocols)
- ▶ [https://realpython.com/python-super/  
#an-overview-of-pythons-super-function](https://realpython.com/python-super/#an-overview-of-pythons-super-function)
- ▶ [https://realpython.com/python-double-underscore/  
#double-leading-underscore-in-classes-pythons-name-mangling](https://realpython.com/python-double-underscore/#double-leading-underscore-in-classes-pythons-name-mangling)
- ▶ HÜ beliebige Vererbung aufbauen, abstimmen so das jeder eine andere Thematik hat

# KW49

- ▶ Wiederholung
- ▶ siehe Textdatei
- ▶ Mitarbeiter Aufgabe siehe Dropbox bis KW50

# KW50

- ▶ Wiederholung
- ▶ individuelles OOP Beispiel zeigen
- ▶ Mitarbeiter OOP Aufgabe besprechen
- ▶ classmethod staticmethod unterscheidung
- ▶ pipreqs+venv
- ▶ pip ist veraltet, benutze uv:  
<https://www.youtube.com/watch?v=6pttmsBSi8M>

# KW51

## ► Weihnachtsstunde

# KW02

- ▶ Python Modules Packages  
<https://realpython.com/python-modules-packages/>
- ▶ Fehlerbehandlungs-arten
- ▶ HÜ 1: Magic-Methods Aufgabe
- ▶ HÜ 2: Fehlerbehandlung in eigene OOP Aufgaben einbauen und Art kommentieren

# KW03

- ▶ WDH
- ▶ HÜs besprechen
- ▶ Python Namespaces

<https://realpython.com/python-namespaces-scope/>



# KW04

- ▶ Namespaces `global()` und `local()` wiederholen
- ▶ magic-method Aufgabe ansehen

# KW05

## ► Prüfungen

# KW08

- ▶ noch 7x Unterricht (inkl.)
- ▶ arg-kwargs besprechen
- ▶ <https://www.data-science-architect.de/args-kwargs/>
  - ▶ Was passiert, wenn man `*args` und `**kwargs` vertauscht (`def test(**kwargs, *args)`)?
  - ▶ Warum sollte man `*args` nicht mit list oder dict verwechseln?
  - ▶ Wann ist `**kwargs` nützlich? (z. B. für flexible Konfigurationsparameter)
  - ▶ Wie kann man `*args` in einer rekursiven Funktion verwenden?
- ▶ Innere methoden
- ▶ <https://realpython.com/inner-functions-what-are-they-good-for/>
- ▶ Aufgabe...demonstriere in Pythoncode...args...kwargs...innere methoden

# KW09

- ▶ args, kwargs Beispiel ansehen
- ▶ inner function beispiel ansehen
- ▶ Python Decorators
- ▶ <https://realpython.com/primer-on-python-decorators/>
- ▶ Rest: Zeitmesser-Decorator in alte Pokeraufgabe einbauen

# KW10

- ▶ Dekorator Beispiel besprechen
- ▶ Einfach verkettete Liste Einführung + Aufgabe

# KW11

- ▶ verkettete Liste Beispiel besprechen
- ▶ iterables <https://realpython.com/python-iterators-iterables/>
- ▶ Aufwandsklassen besprechen
- ▶ <https://www.happycoders.eu/de/algorithmen/o-notation-zeitkomplexitaet/>
- ▶ Vergleiche Arrays mit Listen in Kontext Aufwandsklassen

# KW12

## ► Wiederholungsstunde

# KW13

## ► Wiederholungsstunde



# KW14

## ► Prüfungen