

**Yannick Frommherz**

Research Associate / Chair of Applied Linguistics

# "Hello humanities"

A modular Python programming course targeting the  
specific needs & requirements of humanities students

Programming and Data Infrastructure in Digital Humanities // 27<sup>th</sup> of March 2023

# Who we are

**Simon Meier-Vieracker**

Professor of Applied Linguistics



**Yannick Frommherz**

PhD candidate in Linguistics



first steps in  
programming

## Programming for Digital Humanities (4ME501)

Fall Term, September 5<sup>th</sup> 2018  
Introduction Meeting

Marcelo Milrad, Dan Kohen, Koraljka Golub & Ilir Jusufi

[marcelo.milrad@lnu.se](mailto:marcelo.milrad@lnu.se), [koraljka.golub@lnu.se](mailto:koraljka.golub@lnu.se),  
[dan.kohen@lnu.se](mailto:dan.kohen@lnu.se) & [ilir.jusufi@lnu.se](mailto:ilir.jusufi@lnu.se)

Linnæus University



# What we do

- **Design materials for humanities students to learn programming with Python**
- **Targeting their specific needs & requirements**
  - no prior knowledge
  - little technical know-how
  - reservations about technology & logic
  - wide variety of backgrounds and interests
- **Background**
  - embedded in the [virTUos project](#) researching digital teaching methods
  - new master's in Digital Humanities at TU Dresden, offering a perfect testbed

# How we do it

## First up: Motivation

- Why (on earth) should I as a humanist learn programming?
- What can I (not) do with programming skills?

### Einführung

Dies ist unser erstes Notebook. Hier lernen wir ganz grundsätzliche Dinge übers Programmieren und die Programmiersprache Python. Zuerst beantworten wir die Frage, warum Programmierkenntnisse überhaupt sinnvoll sind:

**Programmieren lohnt sich immer dann, wenn wir es mit repetitiven Aufgaben zu tun haben, die eine gewisse Regelmäßigkeit aufweisen.**

Machen wir's konkret: Stell Dir vor, Du möchtest für ein Forschungsprojekt Bildunterschriften von sämtlichen Artikeln auf ZEIT Online im Jahr 2021 sammeln. Du müsstest jeden einzelnen Artikel im Browser aufrufen und die paar Wörter unter den (oftmals mehreren) Bildern in einem Artikel markieren, kopieren und, sagen wir, in eine Excel-Tabelle einfügen. Damit nicht genug: Du müsstest jeweils auch noch den Artikellink, -titel und das Publikationsdatum ablegen. Das würde Ewigkeiten in Anspruch nehmen! 🤖

Da es sich aber um eine repetitive, regelhafte Aufgabe handelt (also ein sehr ähnlicher Ablauf stets wiederholt wird), ist dies ein Paradebeispiel dafür, wie wir uns das Leben mit Programmierkenntnissen leichter machen können. Gewiss, wenn Du das erste Mal einen Web Scraping-Code schreibst, so nennt sich diese Herangehensweise, dann wird das auch einige Stunden oder Tage dauern. Aber selbst das steht in keinem Verhältnis zur Zeit, die Du bei einer manuellen Bewältigung dieser Aufgabe benötigen würdest. Und beim nächsten Mal codest Du dann auch bereits schneller! 😊

**Hier noch ein paar weitere Beispiele für repetitive, regelhafte Aufgaben, bei denen Du von Programmierkenntnissen profitieren wirst:**

- Aufbereiten und Bearbeiten von Daten, z.B. alle Wörter in einem bestimmten Text sollen großgeschrieben werden
- Anreichern von Daten, z.B. alle Eigennamen in einem Text sollen getaggt werden, je nach dem, ob es sich um eine Person, einen Ort oder eine Institution handelt
- Filtern von Daten, z.B. alle Bilder, die einen R-Mittelwert (im RGB-Farbraum) von über 150 aufweisen
- Auswerten von Daten, z.B. Zählen des Vorkommens bestimmter Formulierungen in einem Text, Errechnen von Schlüsselwörtern für einen Text

Darüber hinaus lassen sich Daten mithilfe von Programmierkenntnissen auch sehr gut statistisch auswerten sowie visualisieren. All diese Techniken werden wir im Verlauf des Kurses kennenlernen.

# How we do it

## Second: An introduction to algorithmic thinking

(shown to increase subsequent learning success, [Koulouri et al., 2015](#))

- What is an algorithm even?
- What is algorithmic thinking?
- How do computers “think”?
- How does natural language relate to (pseudo-)code?

**„Was soll ich anziehen?“-Algorithmus I**

Deutsch	Schau aus dem Fenster. Falls es regnet, zieh eine Regenjacke an. Gehe raus.
Pseudocode	Schau aus dem Fenster. Falls es regnet: Zieh eine Regenjacke an. Gehe raus.

(nach Writing Problem 0-2: Everyday Algorithms. Harvard University. <https://cdn.cs50.net/ap/1516/problems/0/2/0-2.html>)

TECHNISCHE UNIVERSITÄT DRESDEN

Folie 6

DRESDEN concept

13:39 / 21:10 • 1. Beispiel >

Algorithmisches Denken

# How we do it

## Third: Setting the right expectations: you will fail (badly and repeatedly 🤖)

```
print(Print_out)
```

```
-----
NameError                                Traceback (most recent call last)
/var/folders/dy/1d_wflpx5_57vhd_6cdbfk0m0000gn/T/ipykernel_23940/346808566.py in <module>
----> 1 print(Print_out)

NameError: name 'Print_out' is not defined
```

Hier sehen wir eine Fehlermeldung, von denen uns noch ganz viele begegnen werden. Erstens sehen wir, über welche Zeile Python den Fehler gemeldet hat. In dem Screenshot ist eine 1, bei längeren Codes ist diese Angabe aber natürlich sehr hilfreich. Zweitens steht zu unterst jeweils ein `NameError`, was bedeutet, dass Python die Variable `Print_out` nicht kennt. Durch Korrigieren des Variablennamens wird die Fehlermeldung beseitigt. Voraussetzung, dass die Code-Zelle oben bereits einmal ausgeführt wurde, d.h. die Variable `print_out` mit kleinem "p" in

### Exkurs: Hilfe holen

Bis zu diesem Punkt im Notebook bist Du sicherlich auf das ein oder andere Problem gestoßen, das Du nicht selbst lösen konntest. In diesen Fällen Hilfe bei Google gesucht. Das ist genau die richtige Strategie, denn in den seltensten Fällen bist Du mit Deinem Programm alleine. Ein sehr beliebtes Forum für Fragen rund um Python, das zumeist auch das erste Suchergebnis bei Google ist, heißt [Stack Overflow](#).

Angenommen Du stündest vor der Herausforderung, Duplikate aus einer Liste zu entfernen und kämst nicht weiter, so würdest du über ["remove duplicates in lists"](#) (oder so ähnlich) zu diesem [Beitrag](#) gelangen. Die Frage des Nutzers steht auch im Screenshot unten:

## Removing duplicates in lists

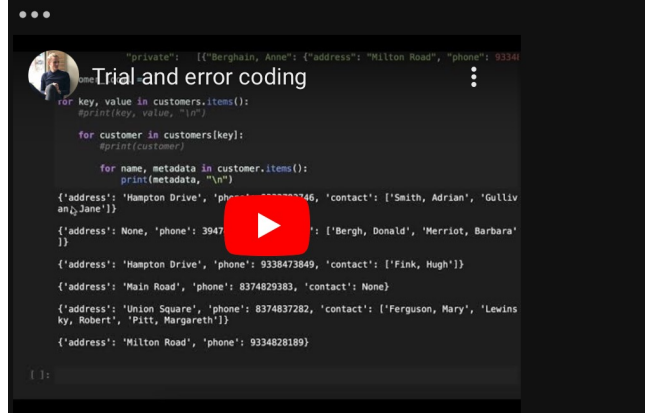
Asked 10 years, 6 months ago   Modified 1 month ago   Viewed 1.9m times

### Exkurs: Trial-and-error-Coding

Unsere Codes werden zunehmend komplexer: In der letzten Übung hast Du etw algorithmischem Denken gelernt haben, ist es wichtig, sich stets zu überlegen, Herangehensweise nimmt mit zunehmender Komplexität der Aufgaben natürlic

Die einzelnen Schritte jedoch löst man dann meistens durch "trial and error". C was man erwartet. Manchmal klappt's, manchmal nicht.

Konkret heißt das für Dich: führe Deinen Code immer wieder aus, lasse Dir einz die Lösung heran.



# How we do it

## Now to the actual materials:

- **Interactive Jupyter Notebooks**
  - elegantly interweaving learning content with hands-on code
  - students can change, complement, fill, and, of course, execute code cells and get immediate feedback
- **Videos embedded in the notebooks on installation and onboarding, algorithmic thinking, and trial-and-error coding**
- **Designed for asynchronous learning: Students learn whenever, wherever and at their own pace**

## Bedingte Anweisungen

Wir widmen uns als Erstes den bedingten Anweisungen und definieren dafür einen simplen string. Führe wie immer die Zelle aus, um `sentence` zu initialisieren.

```
sentence = "Der morgige Tag wird schön."
```

Eine bedingte Anweisung wird mit `if` eingeleitet, z.B.:

```
if sentence.startswith("Der"):  
    print("Der Satz fängt mit einem Artikel im Maskulinum an.")
```

Der Satz fängt mit einem Artikel im Maskulinum an.

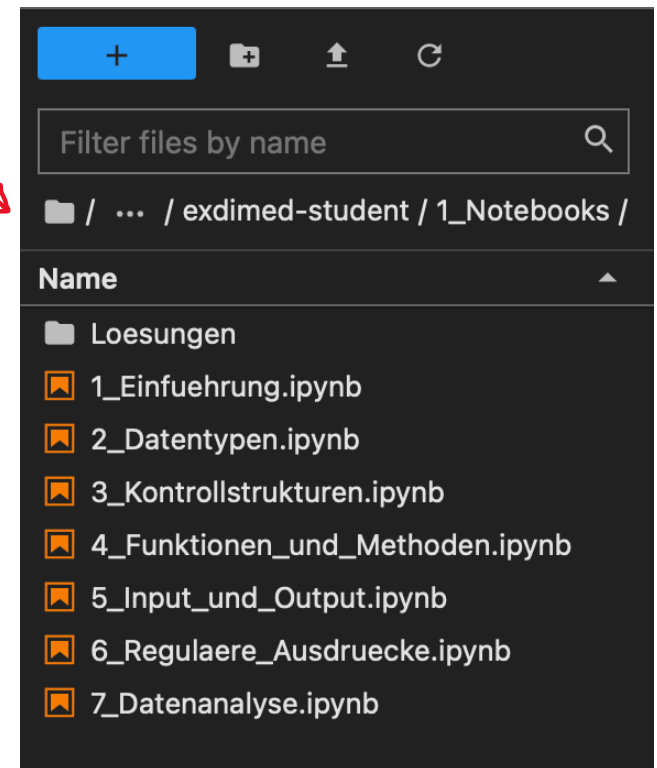
Natürlichsprachlich formuliert liest sich der obige Code: "Wenn der Satz mit 'Der' anfängt, dann geben wir '...' zurück".

Übrigens haben wir gerade eine sog. string-Methode kennengelernt, nämlich `startswith`, die überprüft, ob ein string (hier: `sentence`) mit der in der Klammer definierten Zeichenkette beginnt. Diese und andere Methoden besprechen wir im Detail im nächsten Notebook.

# How we do it


## Modular setup

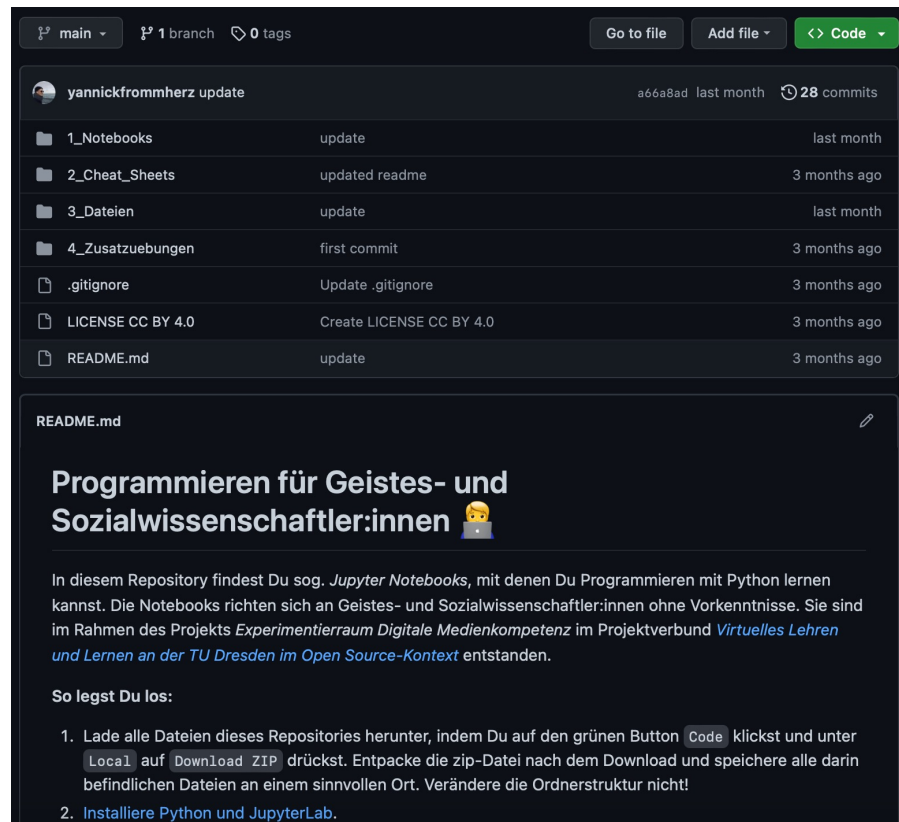
- **Basic module covering the foundations of programming with Python for *all* humanities subjects**
  - introduction, data types, control structures, functions and methods, input and output, regular expressions, data analysis
- **Advanced modules delving into subject-specific use cases, based on interviews with scholars from a variety of humanities subjects**
  - e.g., automated news factor analysis in communication studies, or keyness analysis in linguistics
- **Students with different backgrounds and at different levels can flexibly acquire programming skills that are relevant to them in particular**





# How we do it

- **Lecturers are welcome to use and adapt our materials under [CC BY 4.0](#)** 
- materials can be used as-is or easily extended thanks to their modular structure
- **Eventually, we will index our materials as Open Educational Resources**
- **Meanwhile, feel free to clone them from [GitHub](#)**



# Our own teaching

- **Materials were first used this winter semester in a joint course with literature studies in a “from humanists to humanists” atmosphere**
  - Blended Learning setup: Materials were studied asynchronously, but we met bi-weekly for tutorials
  - Tutorials were used to cultivate an open-minded, frustration-tolerant attitude towards programming
  - Mini-hackathons on preprocessing OCR data, computing ngrams and key-ngrams were organized as collaborative learning was shown to be beneficial in acquiring programming skills  
([Koulouri et al., 2016](#); [Beck and Chizhik, 2013](#); [Braught et al., 2011](#))
- **Learnings**
  - Attitude is everything: “errors are the normalest thing”, “everyone has had this issue at some point”, “there are many ways of solving a problem”, “it’s normal to code iteratively”, “help is just a google query away”...
  - Co-present, collaborative learning is unbeatable (ideally, make attendance compulsory)

**Thank you for your attention!**