

# The ABC of Computational Text Analysis

## #9 *INTRODUCTION TO PYTHON*

Alex Flückiger  
Faculty of Humanities and Social Sciences  
University of Lucerne

27 April 2024

# Recap last Lecture

- from unique words to embeddings recontextualized word meaning
- today's data-driven NLP is both powerful and biased data is never raw but depends on many decisions

# Outline

- enter the shiny world of Python 😎  
programming basics
- get familiar with Visual Studio Code



Python

# Python is ...

a programming language that is...

- general-purpose
  - not specific to any domain
- interpreted
  - no compiling
- very popular in data science



*Common (programming) languages*

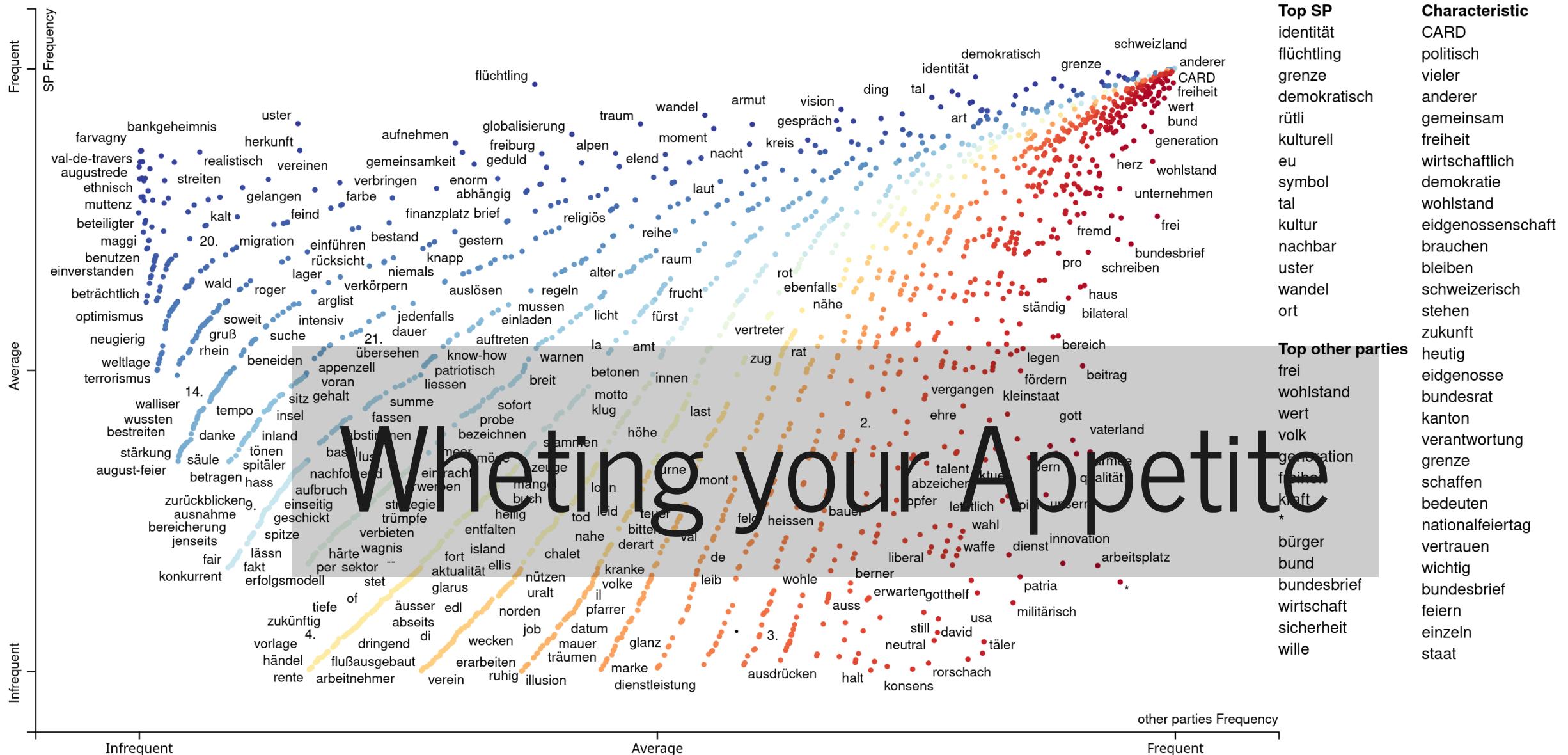
# How to learn Programming?

## Three inconvenient truths 😰

- programming cannot be learnt in a course alone
  - I try to make the start as easy as possible!
- frustration is part of learning
  - fight your way!
- the Python ecosystem is huge
  - grow skills by step-by-step

...

Programming can be absolutely captivating! 🤜



# Programming Concepts & Python Syntax

# Variables

Variables are kind of storage boxes 

```
# define variables
x = "at your service"
y = 2
z = ", most of the time."

# combine variables
int_combo = y * y      # for numbers any mathematical operation
str_combo = x + z       # for text only concatenation with +
                        # show content of variable
print(str_combo)
```

# Data Types

The type defines the object's properties

Name	What for?	Type	Examples
String	Text	str	"Hi!"
Integer, Float	Numbers	int, float	20, 4.5
Boolean	Truth values	bool	True, False
:	:	:	:
List	List of items (ordered, mutable)	list	["Good", "Afternoon", "Everybody"]
Tuple	List of items (ordered, immutable)	tuple	(1, 2)
Dictionary	Relations of items (unordered, mutable)	dict	{"a":1, "b": 2, "c": 3}

# Data Type Conversion

Combine variables of the same type only

```
# check the type
type(YOUR_VARIABLE)

# convert types (similar for other types)
int('100') # convert to integer
str(100) # convert to string

# easiest way to use a number in a text
x = 3
mixed = f"x has the value: {x}"
print(mixed)
```

# Confusing Equal-Sign

= vs. == contradicts the intuition

```
# assign a value to a variable
x = 1
word = "Test"

# compare two values if they are identical
1 == 2          # False
word == "Test"   # True
```

# Comments

- lines ignored by Python
- write comments, it helps you...



to learn initially

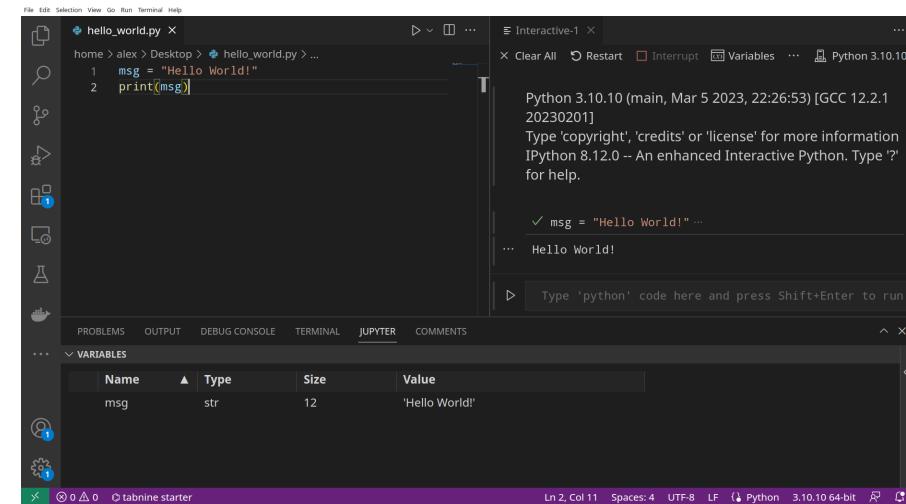
to understand later

```
# single line comment  
"""  
comment across  
multiple  
lines  
"""
```

# Visual Studio Code

## The (best) editor to program in Python

- VS Code features
  - interactive programming
  - integrated development environment (IDE)
  - similar to RStudio
- 3 views in editor
  - programming (left)
  - output (right)
  - additional information (bottom)
- use **tab** for autocomplete



A screenshot of the Visual Studio Code interface. On the left, there's a code editor with a Python file named 'hello\_world.py' containing the code: 

```
msg = "Hello World!"\nprint(msg)
```

. To the right of the code editor is a terminal window titled 'Interactive-1' showing the output of the code execution: 

```
Python 3.10.10 (main, Mar 5 2023, 22:26:53) [GCC 12.2.1\n20230201]\nType 'copyright', 'credits' or 'license' for more information\nIPython 8.12.0 -- An enhanced Interactive Python. Type ?\nfor help.\n\n✓ msg = "Hello World!" ...\n... Hello World!
```

. At the bottom, there's a 'VARIABLES' panel showing a single variable 'msg' with the value 'Hello World!'. The status bar at the bottom indicates the code is at line 2, column 11, with 4 spaces, using UTF-8 encoding, and is a Python 3.10.10 64-bit build.

Interface of *Visual Studio Code*

# In-class: Get started for Python

1. Make sure that your local copy of the Github repository KED2024 is up-to-date with `git pull` in your command-line.
2. Open the Visual Studio Editor.
3. Windows User only: Make sure that you are connected to WSL: Ubuntu (green badge lower-left corner, see image on the slide after the next). If not, click on the badge and select `New WSL Window`.

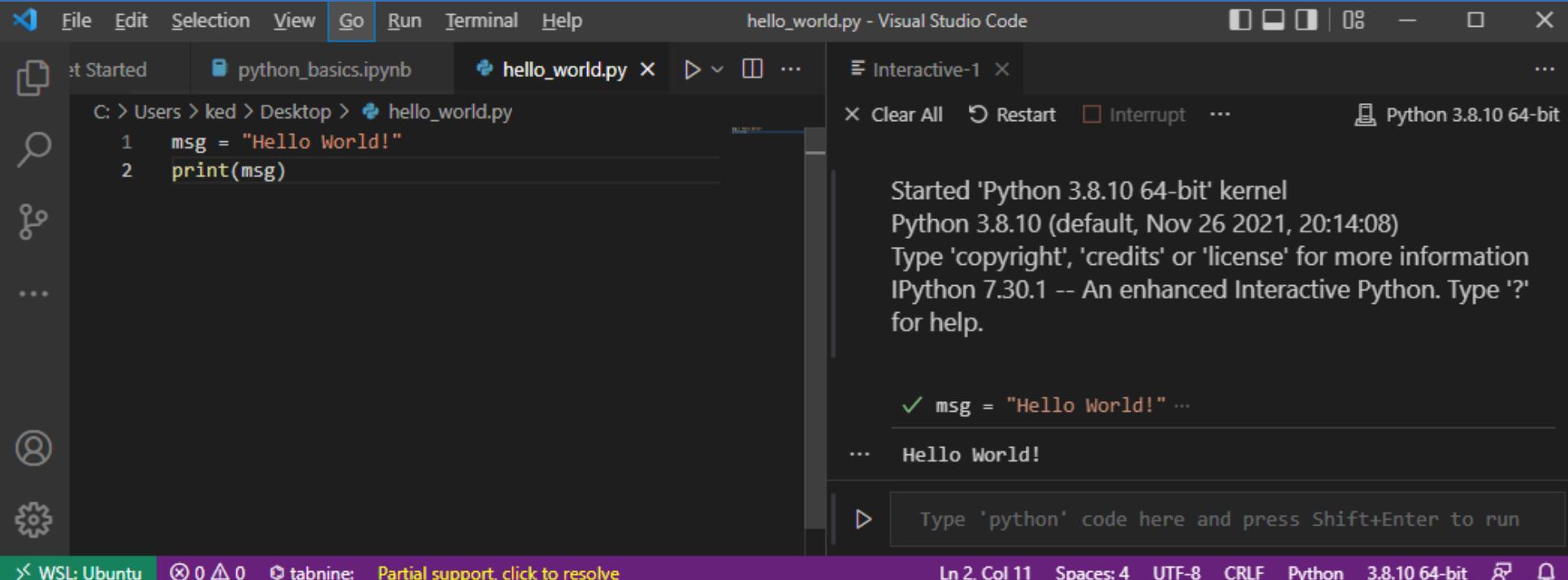
# In-class: Run your first Python Program

1. Create a new file with the following content, save it as `hello_world.py` in the KED2024 folder. Then, execute it by a right click on the code and select `Run current file in interactive window`.

```
# print out a message
msg = "Hello World!"
print(msg)
```

2. Does the output looks like the screenshot on the next slide? If the execution doesn't work as expected, ask me or your neighbour. There might be a technical issue.

# In-class: Expected Output



The screenshot shows a Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** hello\_world.py - Visual Studio Code.
- Left Sidebar:** Shows icons for File Explorer, Search, Open, and Help.
- Central Area:** A code editor with the file `hello_world.py` containing the following code:

```
1 msg = "Hello World!"  
2 print(msg)
```
- Terminal Tab:** Interactive-1 (Python 3.8.10 64-bit) showing the output of the code execution:

```
Started 'Python 3.8.10 64-bit' kernel  
Python 3.8.10 (default, Nov 26 2021, 20:14:08)  
Type 'copyright', 'credits' or 'license' for more information  
IPython 7.30.1 -- An enhanced Interactive Python. Type '?'  
for help.  
  
✓ msg = "Hello World!" ...  
... Hello World!
```
- Bottom Status Bar:** WSL: Ubuntu, 0 △ 0, tabnine: Partial support, click to resolve, Ln 2, Col 11, Spaces: 4, UTF-8, CRLF, Python 3.8.10 64-bit.

*The Output "Hello World!" on the right side should look like in the screenshot*

# Iterations

## for-loop

do something with each element of a collection

```
sentence = ['This', 'is', 'a', 'sentence']

# iterate over each element
for token in sentence:

    # do something with the element
    print(token)
```

# Conditionals

## if-else statement

condition action on variable content

```
sentence = ['This', 'is', 'a', 'sentence']

if len(sentence) < 3:
    print('This sentence is shorter than 3 tokens')

elif len(sentence) == 3:
    print('This sentence has exactly 3 tokens')

else:
    print('This sentence is longer than 3 tokens')
```

# Indentation matters!

- intend code within code blocks  
loops, if-statements etc.
- press **tab** to intend



```
if 5 > 2:  
    print('5 is greater than 2')
```



```
if 5 > 2:  
print('5 is greater than 2')
```

# Methods: Create new Objects

Build your world! 

```
sentence = 'This is a sentence'  
  
# split at whitespace and save result in new variable  
tokens = sentence.split(' ')  
  
# check the content and type variables  
print(sentence, type(sentence), tokens, type(tokens))
```

# Methods: Change Objects

Depending on the object, you can do different things 

```
# add something to a list
tokens.append('.')

# concatenate elements to string
tokens = ' '.join(tokens)
print(tokens, type(tokens))
```

# Functions and Arguments

## DRY: Don't Repeat Yourself

- functions have a name and optional arguments

`function_name(arg1, ..., argn)`

- some functions are predefined

`print(), len() and sorted()`

```
def get_word_properties(word): # define function
    """
    Print properties for a word given as an argument.
    """
    length = len(word)
    sorted_letters = sorted(word, reverse=True)
    print(f"{word} has {length} letters ({sorted_letters}).")

get_word_properties('computer') # call function with any word
```

# Indexing

Computers start counting from zero! 😵

```
sentence = ['This', 'is', 'a', 'sentence']

# element at position X
first_tok = sentence[0]      # 'This'

# elements of subsequence [start:end]
sub_seq = sentence[0:3]       # ['This', 'is', 'a']

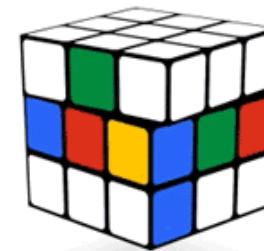
# elements of subsequence backwards
sub_seq_back = sentence[-2:]  # ['a', 'sentence']
```

# Errors

A myriad of things can go wrong



1. read the message
2. find the source of the error  
script name + line number
3. paste message into Google



Play with more cubes at [Chrome Cube Lab](#)

*Learning by doing, doing by googling*

# Modules/Packages

No programming from scratch 

- packages provide more objects and functions
- packages need to be installed additionally

```
# import third-party package
import pandas
import spacy
import plotnine
```

# Jupyter Notebooks

- combine text and code in a single document
  - similar to [R Markdown](#)
- open in [VS Code](#)
- run code with the play button left to the cell or with [CTRL+Enter](#)

# In-class: Exercises I

1. Open the script with the basics of Python in your Visual Studio Editor:  
`KED2024/materials/code/python_basics.ipynb`
2. Try to understand the code in each cell and run them by clicking the play symbol left to them. Check the output. Modify some code as well as data and see how the output changes. Initially, this try-and-error is good strategy to learn. Some ideas:

Combine a string and an integer variable without converting it. What error do you get? How can you avoid it?

Select `is` `a` from the variable `sentence` using the right index.

# In-class: Exercises II

1. Write a Python script that

takes text (a string)

splits it into words (a list)

iterates over all the tokens and print all tokens that are longer than 5 characters

Bonus: wrap your code in a function.

2. Go to the next slide. Start with some of the great interactive exercises out there in the web.

# Resources

## Get more explanations

- Google's Python Class
- Introduction of Python for Social Scientists (Walsh 2021)
- Official Python introduction

## Learn basics interactively

- Introduction to Python by IBM Cognitive Class
- LearnPython



Questions?

# References

Walsh, Melanie. 2021. "Introduction to Cultural Analytics & Python." Zenodo.  
<https://doi.org/10.5281/ZENODO.4411250>.