

The ABC of Computational Text Analysis

#6 INTRODUCTION TO PYTHON IN VS CODE

Alex Flückiger
Faculty of Humanities and Social Sciences
University of Lucerne

28 March 2024

Recap last lecture

- perform a frequency analysis using the Shell  specific words or entire vocabulary
- describe text as pattern using RegEx 
 - literal: `media a b c`
 - meta: `. \w \s [0-9] *`
 - power of `.` $*$

Outline

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



Python

Python is ...

a programming language that is ...

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



The most common programming and markup languages

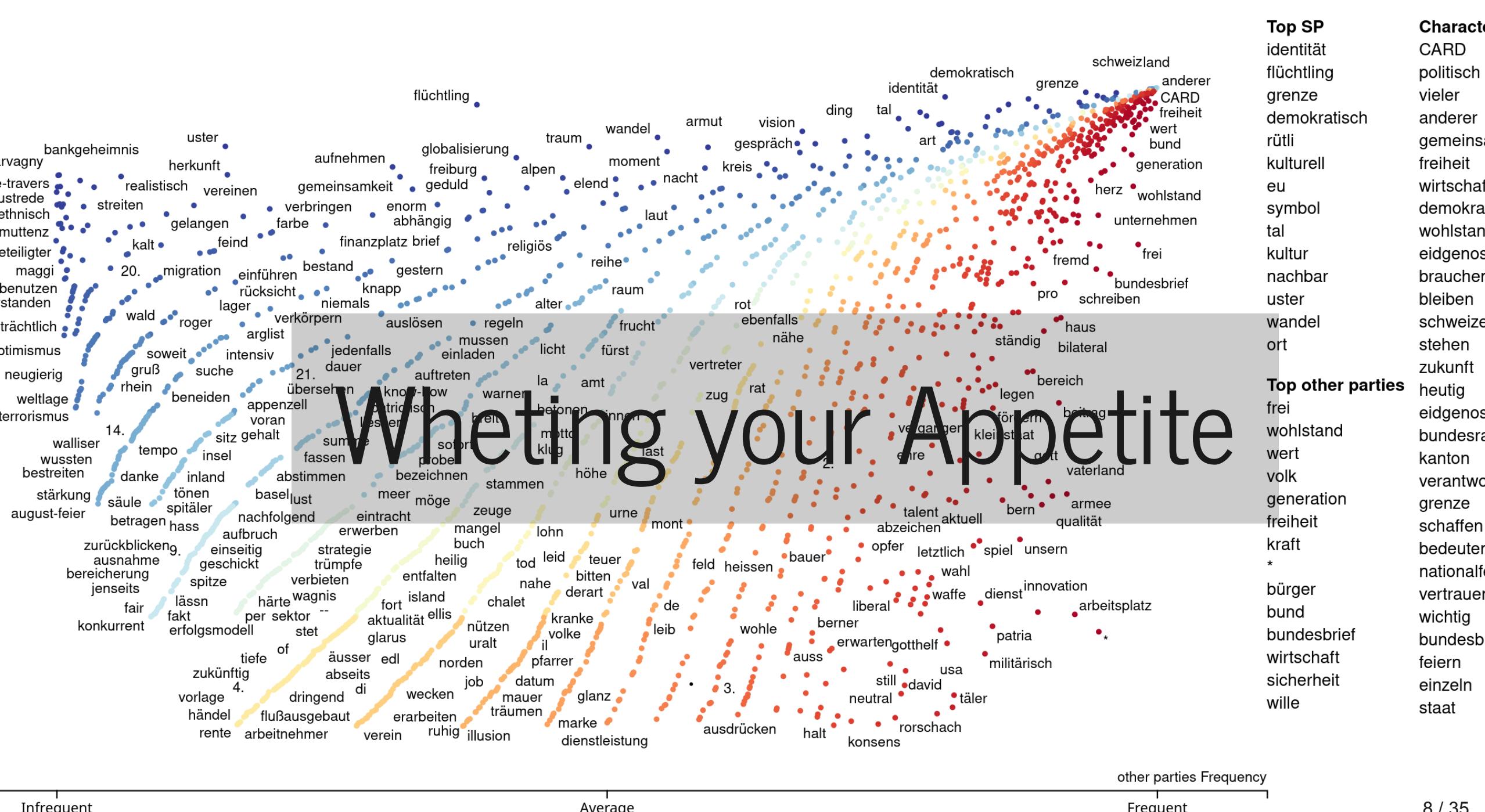
How to learn Programming?

Some 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 through!
- the Python ecosystem is huge
 - grow** your 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 (mutable)	list	["Good", "afternoon", "everybody"]
Tuple	List of items (immutable)	tuple	(1, 2)
Dictionary	Relations of items (mutable)	dict	{"a":1, "b": 2, "c": 3}

Converting data types

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
- use **tab** for autocompletion



Visual Studio Code

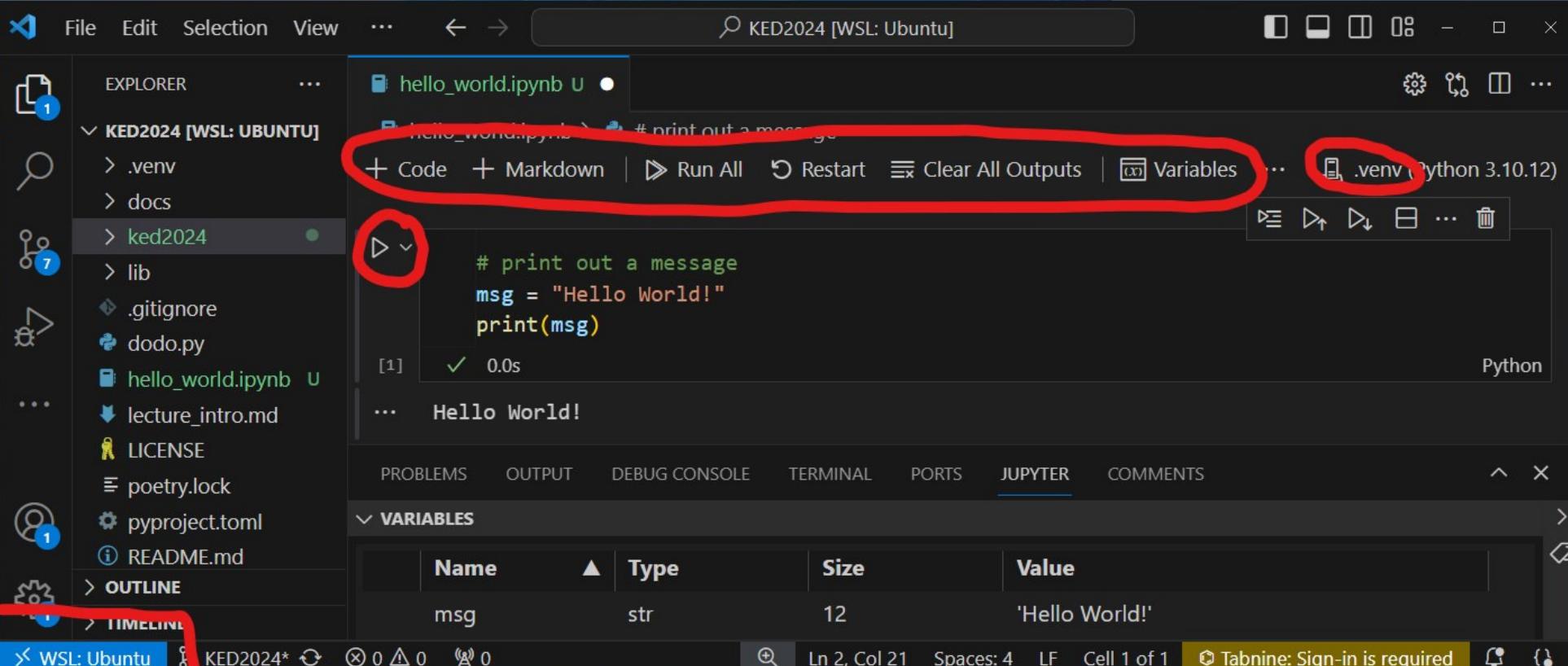
Jupyter Notebooks .ipynb

- combine text and code in a single document
 - similar to R Markdown
- see also explainer for usage in VS Code

Get started in VS Code

1. *Windows user only:* connect to WSL Ubuntu left-lower corner
2. open KED2024 via menu `File > Open Folder`
3. load Python environment `.venv`
4. create notebook via menu `File > New File > Jupyter Notebook`
5. run code with  left to the cell or with `CTRL+Enter`
output shows below cell
6. check out the control bar for global actions at the top

How it should look like



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

- File Bar:** File, Edit, Selection, View, ..., KED2024 [WSL: Ubuntu].
- Explorer:** Shows a file tree for "KED2024 [WSL: UBUNTU]" containing .venv, docs, ked2024, lib, .gitignore, dodo.py, hello_world.ipynb, lecture_intro.md, LICENSE, poetry.lock, pyproject.toml, and README.md.
- Code Editor:** A Jupyter notebook cell with the following code:

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

The cell output shows: [1] ✓ 0.0s ... Hello World!
- Toolbar:** Includes buttons for Code, Markdown, Run All, Restart, Clear All Outputs, Variables, and a dropdown menu.
- Bottom Status Bar:** WSL: Ubuntu, KED2024*, Ln 2, Col 21, Spaces: 4, LF, Cell 1 of 1, Tabnine: Sign-in is required.

Red boxes highlight the following areas:

- The "Variables" button in the toolbar.
- The "Variables" tab in the bottom navigation bar.
- The "TIMELINE" icon in the sidebar.

The output "Hello World!" should look like in the screenshot above.

In-class: Get started with 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 (blue badge in lower-left corner). If not, click on the badge and select `New WSL Window`.

In-class: Run your first Python program

1. Follow the steps described on the slide Get started in VS Code.
2. Create a new file Jupyter Notebook with the following content, save it as `hello_world.ipynb` in the KED2024 folder. Then, run the code.

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

3. Does the output looks like the screenshot on the previous slide? If the execution doesn't work as expected, ask me or your neighbour. There might be a technical issue.
4. When you are done already, you can experiment with the data types that we have discussed.

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 an 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 (**shift + tab** to unintend)



```
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)

predefined functions: print() len() sorted() ...

```
def get_word_properties(word): # define function
    """
    Print properties for a word and return its length.
    """
    length = len(word)
    sorted_letters = sorted(word, reverse=True)
    print(f"{word} has {length} letters: {sorted_letters}.")
    return length # optional return value

word_length = get_word_properties("computer") # call function
```

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. check the underlined (i.e., erroneous)
code

2. read the error message

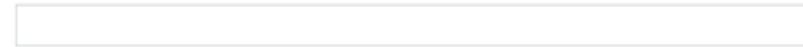
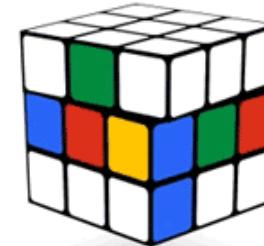
3. find the source of the error

line number given in code

4. paste error into Google

5. restart interactive environment

you may have left-over variables defined



Google Search I'm Feeling Lucky

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
```

In-class: Exercises I

1. Open the Jupyter Notebook with the basics of Python in your Visual Studio Editor:

KED2024/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 and data to see how the output changes. Initially, this try-and-error is good strategy to learn. Some more 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 Python code 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?



Have a nice
Easter break!

References

Walsh, Melanie. 2021. "Introduction to Cultural Analytics & Python." Zenodo.

<https://doi.org/10.5281/ZENODO.4411250>.