Python & pandas

A one day course

Uwe Ziegenhagen

`github.com/UweZiegenhagen/OneDayPythonPandasCourse`

Cologne, 6. Juli 2022

Introduction

Why Python/pandas?

- You *have* a CSV-file with semicolon as column separator and comma as decimal separator
- You *need* a CSV-file with comma as column separator and dot as decimal separator

```
1 import pandas as pd
2
3 df = pd.read_csv('myfile.csv', sep=';', decimal = ',')
4 df = pd.to_csv('myfile.csv', sep=';', decimal = ',')
```

Limits of this Course

- It is not a *full* course, we would need a whole week for this.
- We will skip many interesting things (that you do not necessarily need for your job)
- Goal: Teach you enough Python to a) read and b) understand Python-Code and c) write smaller programs relevant for your job

Python

- Invented by Guido van Rossum at the „Centrum Wiskunde & Informatica“ in Amsterdam as successor for the teaching language ABC
- Current version is 3.11
- For a long time, Python 3.x and Python 2.7 existed together
- Python 2.7 support expired in 2019:
- How to spot 2.7 code: → `print 'hello'` instead of `print('hello')`

Python versus Java & C

Python code is often much slower than C or Java but...

- the implementation time for Python is way faster
- speeds only matters sometimes, not always
- many computing-intensive Python modules use C/C++ modules „under the hood“
- bad C-Code is slower than good Python-code

pandas

- A Python library for data wrangling and management
- Invented by Wes McKinney during his time at AQR Capital Management
- In his own words: „I tell them that it enables people to analyze and work with data who are not expert computer scientists," he says. „You still have to write code, but it's making the code intuitive and accessible. It helps people move beyond just using Excel for data analysis."[1]

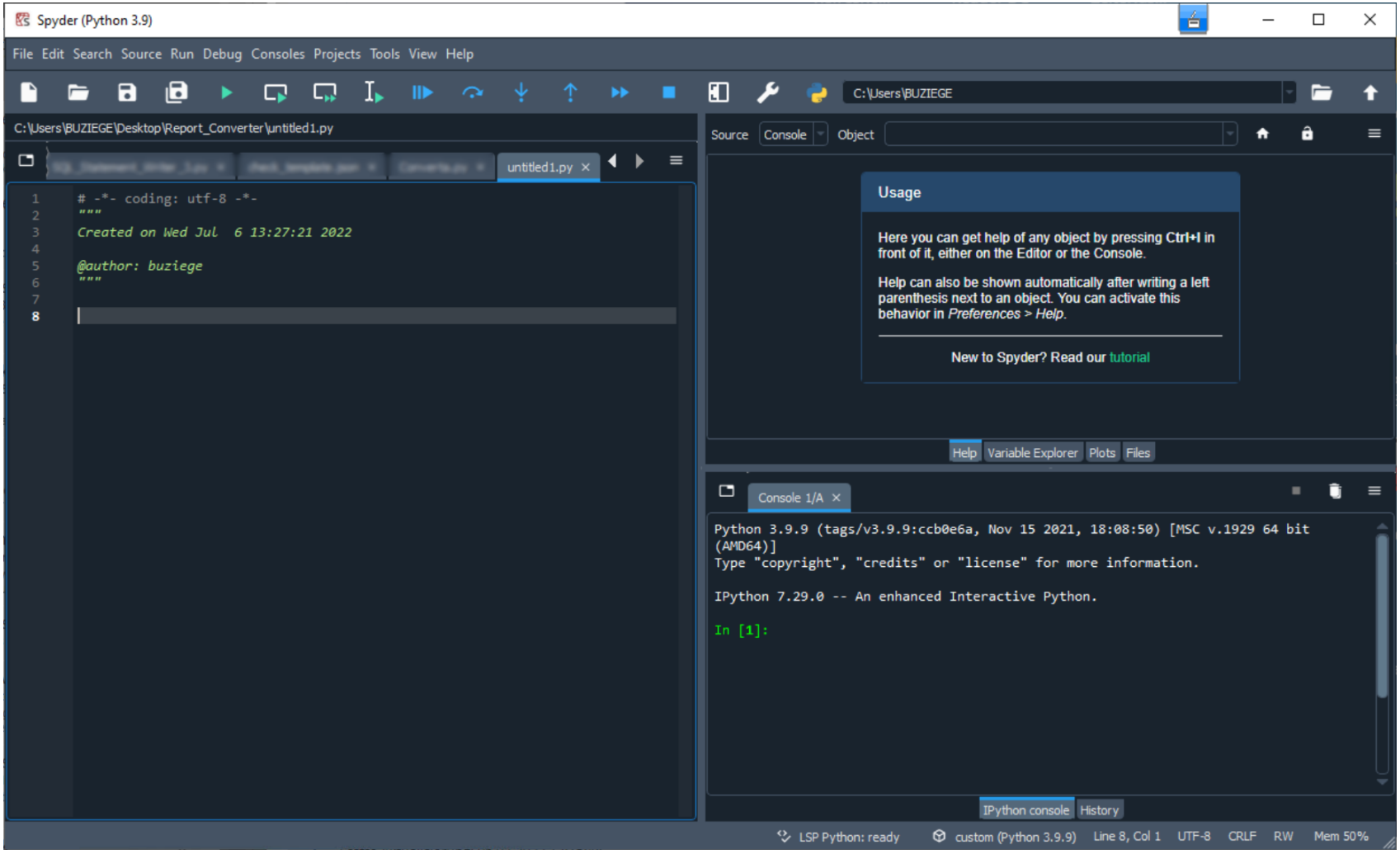
[1] qz.com/1126615/the-story-of-the-most-important-tool-in-data-science/

Python

Datentypen
Funktionen

Spyder

- We will use the Spyder5 IDE with Python 3.9.9, make sure it is installed



Python as a Calculator

- Spyder5 runs an IPython kernel, this runs our programs
- We can also use it as a calculator

```
In [1]: 4*5.4
Out[1]: 21.6

In [2]: 4/12
Out[2]: 0.3333333333333333

In [3]: _*3
Out[3]: 1.0

'hello'
Out[4]: 'hello'
```

Python as a Calculator

```
1
2  In [1]: 4%2
3  Out[1]: 0
4
5  In [2]: 5%2 # Modulo
6  Out[2]: 1
7
8  In [3]: 3**3
9  Out[3]: 27
10
11 In [4]: 5//2
12 Out[4]: 2
```

Priority of Operators

- Round brackets have highest priority
- followed by Power
- followed by multiplication and division
- followed by addition and substraction

TODO4U:

$\Rightarrow$ Solve exercise sheet 1!

Basic Input & Output

```
1  print('Hello World')
2
3  yourName = input('Tell me your name: ')
4
5  print('Hello ' + yourName + ', welcome to this class')
6
7  print('Hello ', yourName,', welcome to this class', sep='')
8
9  print(f'Hello {yourName}, welcome to this class')
```

- `input()` only reads strings
- If you need a number, you need to convert it
- there are better ways than `print()` for logging, but it works...
- f-Strings (last row) are recommended for mixed output!

Rules for Variables

- must start with a letter or _
- Case-sensitivity: 'A' is not 'a'
- Recommendation: small letters
- Let them speak for themselves: 'diameter' is good, 'd' is bad

Reserved Keywords

The following keywords are reserviert and must not be used for variables' names.

and	as	assert	break	class
continue	def	del	elif	else
except	False	finally	for	from
global	if	import	in	is
lambda	None	nonlocal	not	or
pass	raise	return	True	try
while	with	yield		

Datentypen

- Integer (ganze Zahlen)
- Float (Fließkommazahlen)
- Zeichenketten
- Boolesche Werte
- Komplexe Zahlen

Integer

- unbegrenzte Länge, im Gegensatz zu anderen Sprachen
- dürfen nicht mit 0 beginnen, wenn es sich um Zahl im Dezimalsystem handeln soll
- Führende 0 bei Darstellung in Hexadezimal-, Binär- und Oktalsystem:

  `0b/0B` Binärzahl
  `0x/0X` Hexadezimalzahl
  `0o/0O` Oktalzahl

- Funktionen `hex()`, `bin()`, `oct()` für Umwandlung in passenden String
- Interne Darstellung als Dezimalzahl

Float

- Fließkommazahlen
- `3.1415927`
- `3.1e8`
- Hinweis: Nicht jede Fließkommazahl kann genau dargestellt werden („Floating-Point Arithmetic“)
- `docs.python.org/3/tutorial/floatingpoint.html`

Strings

- Doppelte oder einfache Anführungsstriche
- Mehrzeilige Strings:
  - Dreifache doppelte oder einfache Anführungsstriche
  - Alternativ Backslash am Ende der Zeile
- Zahlreiche Funktionen zur String-Verarbeitung, Details später

```
1  a = "Ich bin ein String"
2
3  b = 'Ich auch'
4
5  c = """Ich bin auch
6  ein String"""
7  # 'Ich bin auch\nein String'
```

Boolesche Werte

- Benannt nach George Boole
- 1854: „An investigation into the Laws of Thought“
- Herzstück moderner Computertechnik
- Boolesche Operatoren or ($\cup$), and ($\cap$), `not`

```
1 a = True
2 b = False
3
4 a == b #False
5 a or b # True
6 a and b # False
7 a and not b # True
8 not a and b # False
```

Typumwandlungen

- Das Mischen von Strings und Float/Integer erfordert explizite Typumwandlung mittels `str()` Funktion
- Hinweis: pandas hat dazu `.astype(<Datentyp>)`

```
>>> a + str(b)
'abc123'
>>> a+str(c)
'abc3.141'
>>> a*str(b)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can't multiply sequence by non-int of type 'str'
```

Funktionen

- Bereits bekannte Funktionen: `id()` , `len()` und `type()`
- Funktion: Benannte Sequenz von Befehlen
- Zweck: Code kapseln, um mehrfachen Aufruf zu erleichtern
- Können Argumente als Input bekommen, können Rückgabewerte zurückgeben
- Definition einer neuen Funktion:
  - Namen festlegen
  - Argumente festlegen
  - Befehlssequenz festlegen

Eingebaute Funktionen

abs	delattr	hash	memoryview	set
all	dict	help	min	setattr
any	dir	hex	next	slice
ascii	divmod	id	object	sorted
bin	enumerate	input	oct	staticmethod
bool	eval	int	open	str
breakpoint	exec	isinstance	ord	sum
bytearray	filter	issubclass	pow	super
bytes	float	iter	print	tuple
callable	format	len	property	type
chr	frozenset	list	range	vars
classmethod	getattr	locals	repr	zip
compile	globals	map	reversed	__import__
complex	hasattr	max	round	

Eingebaute Funktionen I

* steht für „vermutlich keine Relevanz in der Vorlesung“

abs()	Absolutwert einer Zahl
all()	prüft, ob alle Elemente eines iterable True sind*
any()	prüft, ob wenigstens Elemente eines iterable True ist*
ascii()	ASCII Darstellung eines Objekts
bin()	wandelt in Binärzahl um
bool()	gibt Boole’schen Wert für Ausdruck zurück

Eingebaute Funktionen II

breakpoint()	im Debugging genutzt*
bytearray()	erzeugt Array of Bytes*
bytes()	erzeugt neues Byte-Objekt*
callable()	Prüfung, ob Objekt aufrufbar ist*
chr()	Unicode-String-Repräsentation einer Zahl
classmethod()	wandelt Methode in Klassenmethode um*

Eingebaute Funktionen III

compile()	übersetzt String in exec-baren Code*
complex()	erzeugt komplexe Zahl*
delattr()	löscht Attribut aus Objekt
dict()	erstellt ein dictionary Objekt
dir()	Variablenliste im aktuellen Scope
divmod()	gibt Tupel aus Integer Division und Rest zurück

Eingebaute Funktionen III

enumerate()	erzeugte Liste von Zahl, Item aus Iterables
eval()	parst und führt String-Ausdruck aus
exec()	führt dynamisch erzeugten Code aus
filter()	genutzt in funktionaler Programmierung
float()	erstellt float aus Zahl oder String
format()	erstellt formatierten String

Eingebaute Funktionen IV

frozenset()	erzeugt unmutuable Set*
getattr()	gibt Wert von Attribut eines Objekts zurück
globals()	gibt dict der globalen Symboltabelle zurück
hasattr()	prüft ob Objekt ein Attribut hat
hash()	gibt integer Hash-Wert für Objekt zurück
help()	ruft die Hilfe auf

Eingebaute Funktionen V

hex()	Hexadezimaldarstellung
id()	Gibt die interne ID eines Objekts zurück
input()	liest von der Tastatur einen String
int()	erzeugt integer aus Zahl oder String
isinstance()	prüft, ob Objekt von Typ x ist
issubclass()	prüft, ob Klasse Subklasse von x ist

Eingebaute Funktionen VI

iter()	iteriert über Sequenz
len()	Länge eines Objekt
list()	erzeugt Liste
locals()	erzeugt Update für die
map()	genutzt in funktionaler Programmierung
max()	gibt Maximum zurück

Eingebaute Funktionen VII

memoryview()	gibt memoryview eines Objekts wieder*
min()	gibt Minimum zurück
next()	gibt nächstes Objekt von Iterator zurück
object()	gibt neues Objekt zurück*
oct()	Oktaldarstellung einer Zahl*
open()	öffnet Datei zum Lesen/Schreiben

Eingebaute Funktionen VIII

ord()	Inverse von chr(), gibt Zahl für Zeichen aus
pow()	analog zu x**y
print()	gibt Objekte aus auf Kommandozeile, in Datei
property()	gibt Property-Attribut aus*
range()	erzeugt eine Zahlenliste
repr()	gibt ASCII-Repräsentation eines Objekts wieder*

Eingebaute Funktionen IX

reversed()	gibt umgekehrte Sequence zurück
round()	rundet Zahl
set()	erstellt Menge
setattr()	Gegenstück zu getattr()
slice()	gibt slice-Objekt zurück*
sorted()	gibt sortierte Liste zurück

Eingebaute Funktionen X

staticmethod()	wandelt Methode in statische Methode um*
str()	erzeugt String aus Objekt
sum()	berechnet Summe aus
super()	gibt Proxy-Objekt zurück*
tuple()	erzeugt Tupel aus Iterable
type()	gibt Typ eines Objekts wieder

Eingebaute Funktionen XI

vars()	gibt dict-Attribut eines Objekts wider
zip()	aggregiert iterables in einen Iterator
__import()__	Funktion zum Anpassen von import-Statements*

Funktionen in C und Python

```c
#include<stdio.h>
long add(long a, long b){
    long ergebnis;
    ergebnis = a + b;
    return ergebnis;}

int main(){
   int a, b, c;
   printf("Geben Sie zwei Zahlen ein\n");
   scanf("%d%d", &a, &b);
   long meinergebnis = add(a,b);
   printf("Summe der Zahlen = %d\n", meinergebnis);
   return 0;}
```

Listing 1: addTwoNumbers.c Code

```python
def add(a, b):
    return int(a) + int(b)

a, b = input('Geben Sie zwei Zahlen ein!').split()
print('Die Summe beträgt: {}'.format(add(a,b)))
```

Listing 2: addTwoNumbers.py Code

Einfache Funktionen

Leere Funktionen

- `pass` wird oft genutzt, wenn Funktionen noch nicht fertig sind
- Sinnvoll, wenn z. B. zuerst das Grundgerüst einer Anwendung entworfen werden soll
- ohne `pass` kommt `IndentationError: expected an indented block` Fehler

Einfache Funktionen

- Kein Parameter
- Kein Rückgabewert (`void`)

```
1 Hallo
```

Funktionen mit Argumenten

- Ein Argument `text` wird übergeben
- Fehlermeldung, wenn Argument fehlt

```
1 Hallo FOM!
```

Funktionen mit Argumenten

- Zwei Argumente, `text` und `anzahl`, werden übergeben

```
1
2 Hallo FOM!
3 Hallo FOM!Hallo FOM!Hallo FOM!
```

Funktionen mit Argumenten

- Setzen von Standardwerten für die Parameter
- Erlaubt Aufruf der Funktion ohne Parameter

```
1 Hallo FOM!
2 Hallo FOM!Hallo FOM!Hallo FOM!
3 Hello Köln!
4 Hello Köln!Hello Köln!
```

Funktionen mit Rückgabewerten

- Funktionen können Wert zur weiteren Verarbeitung zurückgeben

```
1 HuhuHuhu
```

Funktionen mit mehreren Rückgabewerten

- Funktionen können mehr als einen Rückgabewert haben
- Funktion liefert dann ein Tupel, eine unveränderliche Liste der Werte, zurück (dazu später mehr)
- Art des Umgangs mit dem Tupel nennt man „Unpacking“
- Hinweis: Parameter `sep` im Beispiel ist Parameter der `print()` Funktion

```
1 2>HuhuHuhu
```

pandas

Links

Links

- https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html
- https://pandas.pydata.org
-
-
-
-