SIEBTE ÜBUNG

ZUR EINFÜHRUNG IN DIE PROGRAMMIERUNG FÜR COMPUTERLINGUISTEN

Leonie Weißweiler

9.12.16

TWEEDBACK

71G

- lmu.twbk.de
- Lesson ID: **71G**

WIEDERHOLUNG: BINÄR

7IG

- Was ist **101010** im Dezimalsystem?

a) 10

b) 26

c) 42

d) 84

WIEDERHOLUNG: BINÄR

- Was ist **101010** im Dezimalsystem?

a) 10

b) 26

c) 42

d) 84

WIEDERHOLUNG: SPLIT

- Was ist “Are you saying ni to that old woman?”.split(”a”) ?

a) [“re you s”, “ying ni to th”, “t old wom”, “n?”]

b) [“Are you s”, “ying ni to th”, “t old wom”, “n?”]

c) [“A”, “re you s”, “a”, “ying ni to th”, “a”, “t old wom”, “a”, “n?”]

d) [“Are you s”, “aying ni to th”, “at old wom”, “an?”]

WIEDERHOLUNG: SPLIT

- Was ist “Are you saying ni to that old woman?”.split(”a”) ?

a) [“re you s”, “ying ni to th”, “t old wom”, “n?”]

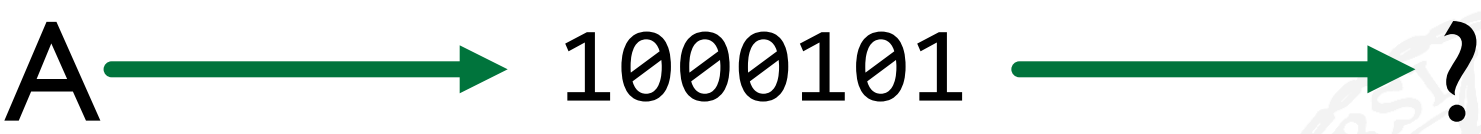
b) [“Are you s”, “ying ni to th”, “t old wom”, “n?”]

c) [“A”, “re you s”, “a”, “ying ni to th”, “a”, “t old wom”, “a”, “n?”]

d) [“Are you s”, “aying ni to th”, “at old wom”, “an?”]

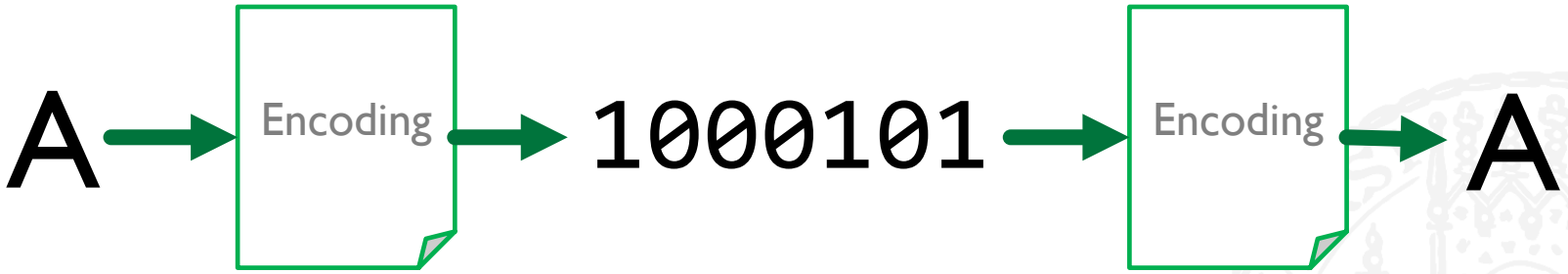
ENCODINGS: ASCII

- Computer können keine Buchstaben speichern, nur Zahlen
- Man braucht eine Vereinbarung welche Zahl zu welchem Buchstaben gehört



ENCODINGS: ASCII

- Computer können keine Buchstaben speichern, nur Zahlen
- Man braucht eine Vereinbarung welche Zahl zu welchem Buchstaben gehört



ENCODINGS: ASCII

- **ASCII**: 1963 u.a. für Fernschreiber entwickelt, 128 Zeichen auf 7 Bit
- `$ = 0100100`
- `A = 1000001`
- `z = 1111010`

ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol
0	0	NUL	16	10	DLE	32	20	(space)	48	30	0	64	40	@	80	50	P	96	60	`	112	70	p
1	1	SOH	17	11	DC1	33	21	!	49	31	1	65	41	A	81	51	Q	97	61	a	113	71	q
2	2	STX	18	12	DC2	34	22	"	50	32	2	66	42	B	82	52	R	98	62	b	114	72	r
3	3	ETX	19	13	DC3	35	23	#	51	33	3	67	43	C	83	53	S	99	63	c	115	73	s
4	4	EOT	20	14	DC4	36	24	$	52	34	4	68	44	D	84	54	T	100	64	d	116	74	t
5	5	ENQ	21	15	NAK	37	25	%	53	35	5	69	45	E	85	55	U	101	65	e	117	75	u
6	6	ACK	22	16	SYN	38	26	&	54	36	6	70	46	F	86	56	V	102	66	f	118	76	v
7	7	BEL	23	17	ETB	39	27	'	55	37	7	71	47	G	87	57	W	103	67	g	119	77	w
8	8	BS	24	18	CAN	40	28	(	56	38	8	72	48	H	88	58	X	104	68	h	120	78	x
9	9	TAB	25	19	EM	41	29	)	57	39	9	73	49	I	89	59	Y	105	69	i	121	79	y
10	A	LF	26	1A	SUB	42	2A	*	58	3A	:	74	4A	J	90	5A	Z	106	6A	j	122	7A	z
11	B	VT	27	1B	ESC	43	2B	+	59	3B	;	75	4B	K	91	5B	[	107	6B	k	123	7B	{
12	C	FF	28	1C	FS	44	2C	,	60	3C	<	76	4C	L	92	5C	\	108	6C	l	124	7C	|
13	D	CR	29	1D	GS	45	2D	-	61	3D	=	77	4D	M	93	5D	]	109	6D	m	125	7D	}
14	E	SO	30	1E	RS	46	2E	.	62	3E	>	78	4E	N	94	5E	^	110	6E	n	126	7E	~
15	F	SI	31	1F	US	47	2F	/	63	3F	?	79	4F	O	95	5F	_	111	6F	o	127	7F	

QUIZ

- Was ist **B** in ASCII?

a) 100010

b) 10010

c) 1000010

d) 10000010

QUIZ

- Was ist **B** in ASCII?

a) 100010

b) 10010

c) 1000010

d) 10000010

ENCODINGS: ISO 8859

- ASCII enthält nur englische Buchstaben und Sonderzeichen
  - Was ist mit anderen Sprachen? äüÂøáË ĪKËÅõ
- Computer arbeiten mit 8-Bit ➔ Es sind noch 128 Möglichkeiten übrig
- $ = **0**0100100
- A = **0**1000001
- z = **0**1111010
- Ö = **1**??????
- å = **1**??????

ENCODINGS: ISO 8859

- ISO 8859 enthält 15 verschiedene Belegungen für die übrigen Plätze
  - ISO 8859-1 (Westeuropäisch)
  - ISO 8859-5 (Kyrillisch)
  - ISO 8859-11 (Thai)
- A = 01000001
- z = 01111010
- Ä = 11000100
- ü = 11111011

QUIZ

- Wieviele Iso-Latin Buchstaben sind hier?

  01000101 11111100 01000111 01100011 11110110

a) 3

b) 4

c) 2

d) 1

QUIZ

- Wieviele Iso-Latin Buchstaben sind hier?

  01000101 11111100 01000111 01100011 11110110

a) 3

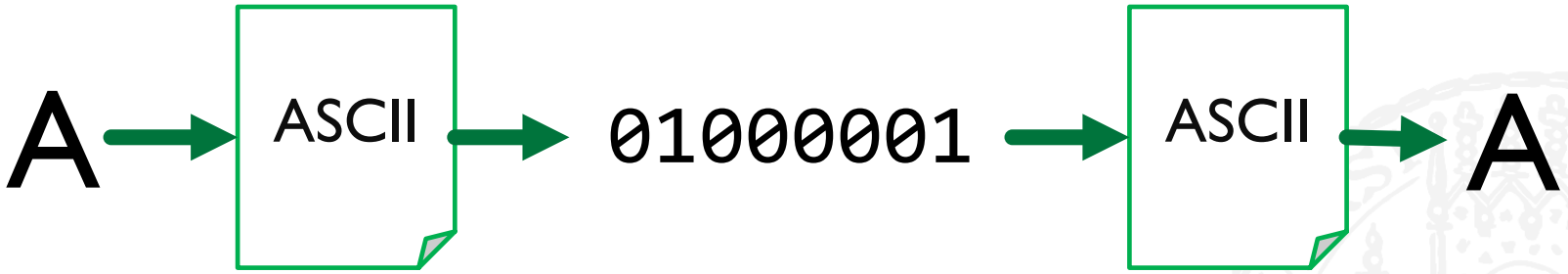
b) 4

c) 2

d) 1

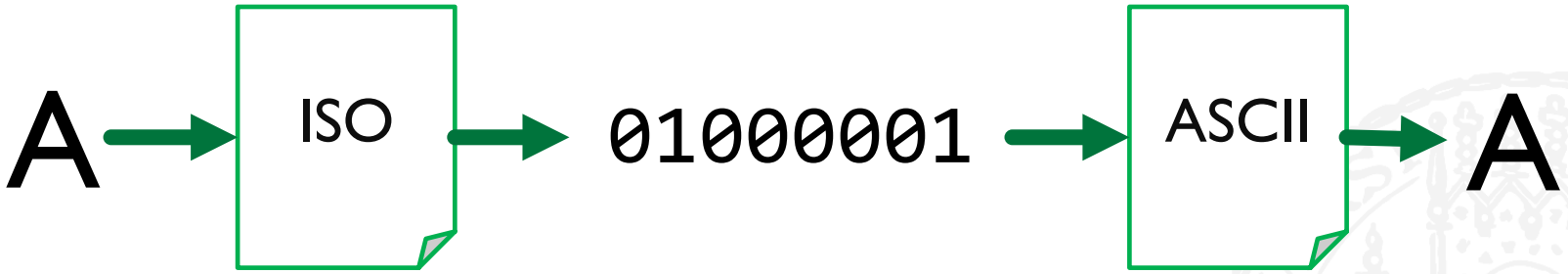
ENCODINGS: ISO 8859-1

- Durch das Ergänzen Funktioniert ISO „mit“ ASCII zusammen



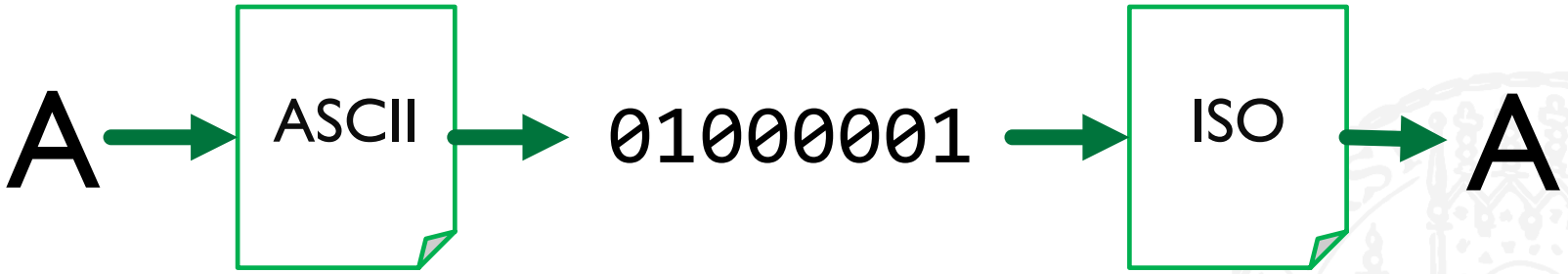
ENCODINGS: ISO 8859-1

- Durch das Ergänzen Funktioniert ISO „mit“ ASCII zusammen



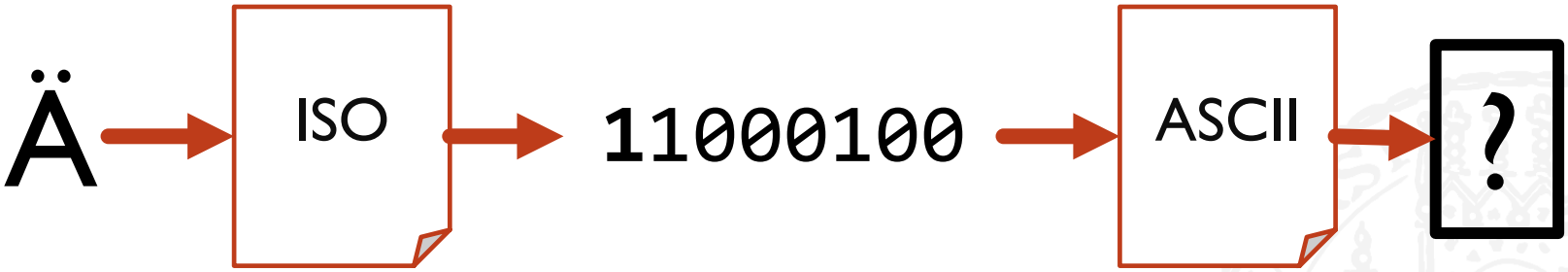
ENCODINGS: ISO 8859-1

- Durch das Ergänzen Funktioniert ISO „mit“ ASCII zusammen



ENCODINGS: ISO 8859-1

- Durch das Ergänzen Funktioniert (fast immer) ISO „mit“ ASCII zusammen



ENCODINGS: UNICODE

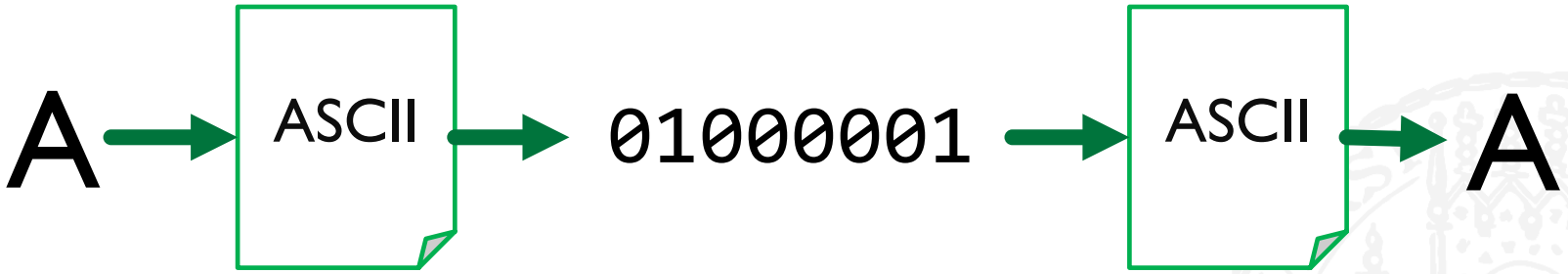
- ISO 8859 enthält jeweils nur 256 Zeichen
  - Was ist mit asiatischen Sprachen? ごみ 废话 🎄❤️🐧🦄💻
  - Was ist mit Dokumenten mit kyrillischen **und** deutschen „Sonderbuchstaben“?
- Es gibt mehr als $2^8 = 256$ Zeichen auf der Welt
- Es werden zwei Bit benötigt um alle Zeichen abzubilden
- In $2^{16} = 65.536$ ist genügend Platz für (fast) alle Zeichen

ENCODINGS: UTF 8

- Immer zwei Byte verwenden ist keine optimale Lösung
  - Platzverschwendung
  - Inkompatibel zu ASCII
  - Was ist wenn noch mehr Emojis erfunden werden...
  - ➢ Variable Länge
- Die ersten 127 Zeichen sind identisch zu ASCII und werden so gespeichert
  - **0**xxxxxxx = 00000000 0xxxxxxx
- Zeichen die mehr Platz benötigen werden in zwei/drei... Byte codiert
  - **110**xxxxx **10**xxxxxx = 00000xxx xxxxxxxx
  - **1110**xxxx **10**xxxxxx **10**xxxxxx = xxxxxxxx xxxxxxxx

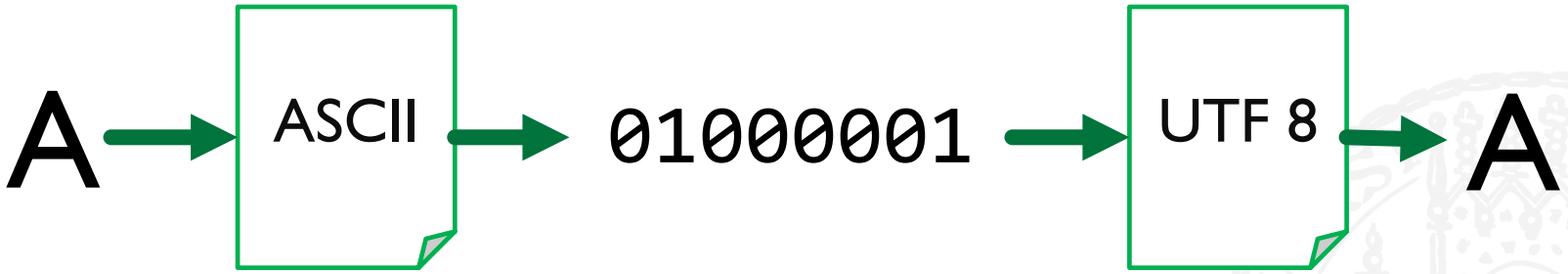
ENCODINGS: UTF 8

- Durch das Ergänzen Funktioniert UTF 8 „mit“ ASCII zusammen



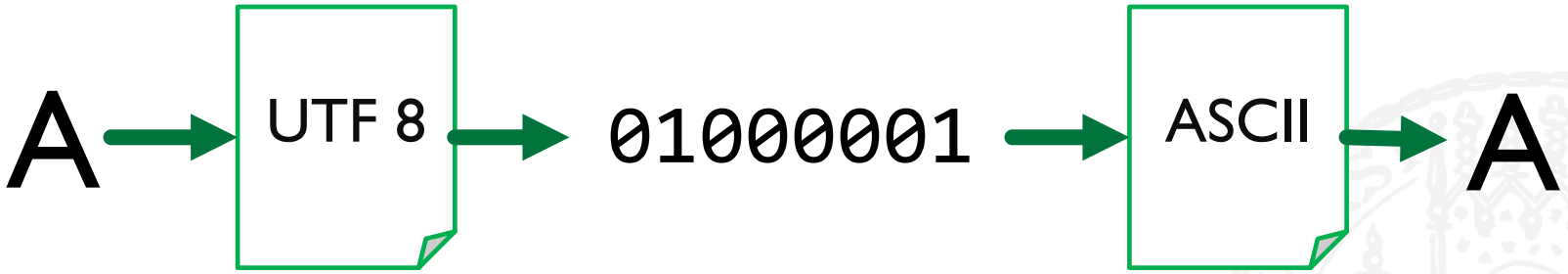
ENCODINGS: UTF 8

- Durch das Ergänzen Funktioniert UTF 8 „mit“ ASCII zusammen



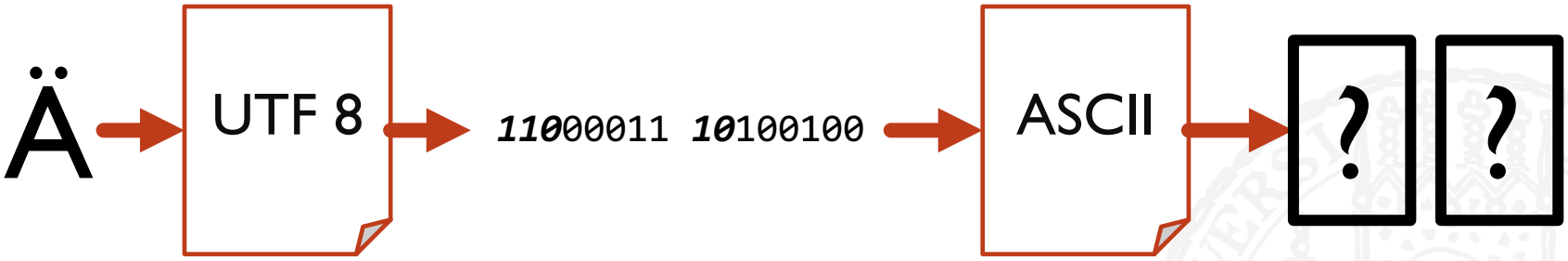
ENCODINGS: UTF 8

- Durch das Ergänzen Funktioniert UTF 8 „mit“ ASCII zusammen



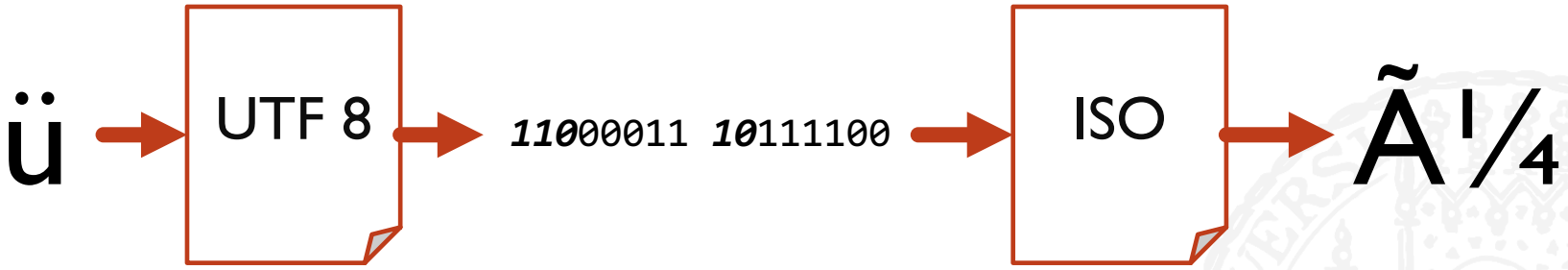
ENCODINGS: UTF 8

- Durch das Ergänzen Funktioniert UTF 8 „mit“ ASCII zusammen



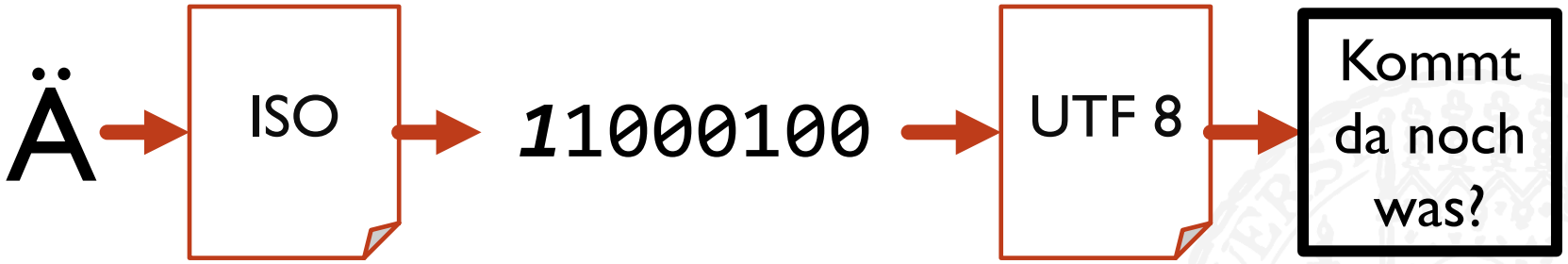
ENCODINGS: UTF 8

- Durch das Ergänzen Funktioniert UTF 8 **nicht** mit ISO Latin zusammen



ENCODINGS: UTF 8

- Durch das Ergänzen Funktioniert UTF 8 **nicht** mit ISO Latin zusammen



ENCODINGS: UTF-16

- UTF-16 belegt pauschal 2 Byte (16 Bit) pro Zeichen
- Inkompatibel zu allen anderen Encodings
- Programmierer sind sich bis heute nicht einig welches Byte zuerst kommt
- Es gibt deswegen zwei „Varianten“ von UTF-16:
  - **UTF-16 L**ittle**E**ndian (zuerst das “hintere“/“niederwertige“ Byte)
  - **UTF-16 B**ig**E**ndian (zuerst das „vordere“/“hochwertige“ Byte)
- Manchmal wird als erstes ein **B**yte**O**rder**M**ark gespeichert: `11111111 11111110` (LE)
- Sonst muss man raten, aber da die meisten Texte größtenteils aus englischen Buchstaben bestehen ist das hochwertige Byte sehr häufig 00000000

QUIZ

- Was ist hier passiert?

Das ist nÃ¤mlich doof!

a) ISO-Latin als ASCII angezeigt

b) ASCII als ISO-Latin angezeigt

c) UTF-8 als ISO-Latin angezeigt

d) UTF-8 als ASCII angezeigt

QUIZ

- Was ist hier passiert?

Das ist nÃ¤mlich doof!

a) ISO-Latin als ASCII angezeigt

b) ASCII als ISO-Latin angezeigt

c) UTF-8 als ISO-Latin angezeigt

d) UTF-8 als ASCII angezeigt

QUIZ

- Aus „abcdefg“ ist „愀戀挀挠攀昀□“ geworden. Was ist passiert?

a) UTF-8 als UTF-16LE angezeigt

b) UTF-16LE als UTF-9 angezeigt

c) UTF-16LE als UTF-16BE angezeigt

d) UTF-16 als ISO-Latin angezeigt

QUIZ

- Aus „abcdefg“ ist „愧戀振搀攀昀□“ geworden. Was ist passiert?

a) UTF-8 als UTF-16LE angezeigt

b) UTF-16LE als UTF-9 angezeigt

c) UTF-16LE als UTF-16BE angezeigt

d) UTF-16 als ISO-Latin angezeigt

QUIZ


```
Leonie@Laptop $ hexdump -C datei.txt
00000000  4c 00 61 00 74 00 65 00  69 00 6e 00 20 00 69 00  |L.a.t.e.i.n. .i.|
00000010  73 00 74 00 20 00 65 00  69 00 6e 00 65 00 20 00  |s.t. .e.i.n.e. .|
00000020  74 00 6f 00 74 00 65 00  20 00 53 00 70 00 72 00  |t.o.t.e. .S.p.r.|
00000030  61 00 63 00 68 00 65 00  0a 00                    |a.c.h.e...|
0000003a
```

- Welches Encoding hat die Datei?

a) ISO-Latin

b) UTF-8

c) UTF-16LE

d) UTF-16BE

QUIZ


```
Leonie@Laptop $ hexdump -C datei.txt
00000000  4c 00 61 00 74 00 65 00  69 00 6e 00 20 00 69 00  |L.a.t.e.i.n. .i.|
00000010  73 00 74 00 20 00 65 00  69 00 6e 00 65 00 20 00  |s.t. .e.i.n.e. .|
00000020  74 00 6f 00 74 00 65 00  20 00 53 00 70 00 72 00  |t.o.t.e. .S.p.r.|
00000030  61 00 63 00 68 00 65 00  0a 00                    |a.c.h.e...|
0000003a
```

- Welches Encoding hat die Datei?

a) ISO-Latin

b) UTF-8

c) UTF-16LE

d) UTF-16BE

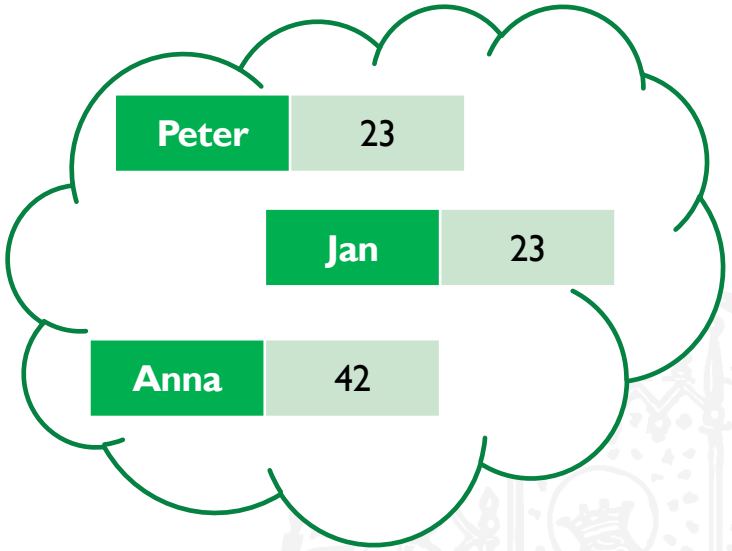
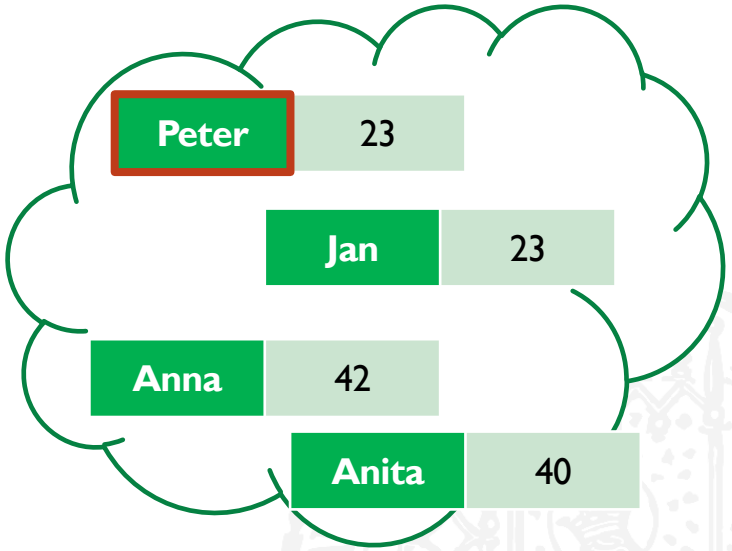
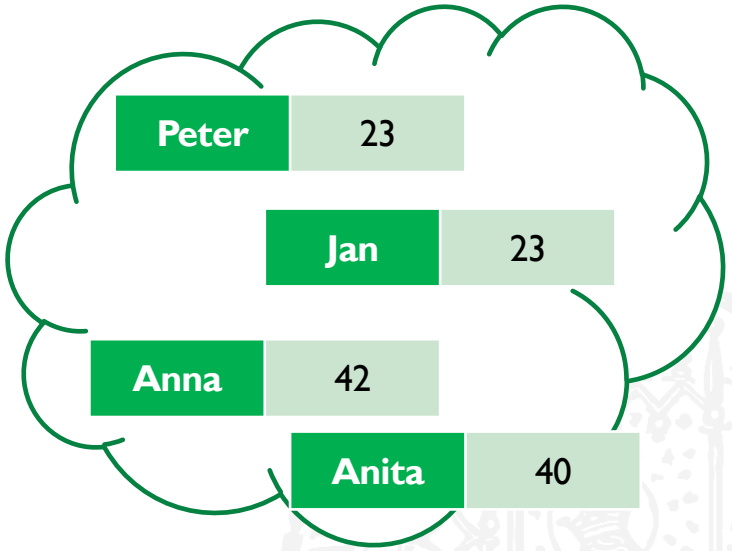
RECODE

- Man muss **häufig** Encodings umwandeln
- Hierzu gibt es das Tool **recode** `ENCODING..ENCODING DATEI`

```
Leonie@Laptop $ recode -l
ASCII-BS BS
ASMO_449 arabic7 iso-ir-89 ISO_9036
AtariST
baltic iso-ir-179
Bang-Bang
Leonie@Laptop $ recode UTF-8..ISO-8859-1 file.txt

Leonie@Laptop $ recode ISO-8859-1..ASCII-BS file.txt

Leonie@Laptop $ recode ASCII-BS..UTF-8 file.txt
```

DICTIONARIES

- Beispiel: Alter von Personen speichern:
  - „Jan“ ➔ 23
  - „Anna“ ➔ 42
  - „Peter“ ➔ 23
- Lösung mit zwei Listen:
- ```
  namen = [„Jan“, „Anna“, „Peter“]
  alter = [27,    42,     23]
  ```
- Schwierig zu verwalten!
  - Dopplungen vermeiden
  - Werte finden (einmal durchsuchen?)


  ```

DICTIONARIES

- Es gibt in Python sog. **Dictionaries**
- In einem Dictionary kann man Werte unter **Schlüsseln** speichern
- Hier ist **Peter** ein Schlüssel und **23** der Wert
- Jeder Schlüssel der vorkommt hat genau einen Wert
- Jeder Wert kann beliebig oft vorkommen



DICTIONARIES

- Man kann die Werte über die Schlüssel erreichen
- `>>> print( dict["Peter"] )`
  23
- `>>> dict["Anita"] = 40`



DICTIONARIES

- Dictionaries müssen wie Listen initialisiert werden
  - `dict = {}`
- Mit eckigen Klammern kann man einfügen/auslesen
  - `dict[„Peter“] = 23`
  - `print( dict[„Peter“] )`
- Mit `print` wird eine Textdarstellung ausgegeben
  - `{‘Peter‘: 23, ‘Jan‘: 23, ‘Anna‘: 42, ‘Anita‘: 40}`



QUIZ

- Was ist das Ergebnis dieses Codes?
- ```
  dict = {}
  dict['a'] = 2
  dict['b'] = 3
  dict['a'] = 4
  print( dict['a'] + dict['b'] )
  ```

a) 4

b) 5

c) 6

d) 7


  ```

QUIZ

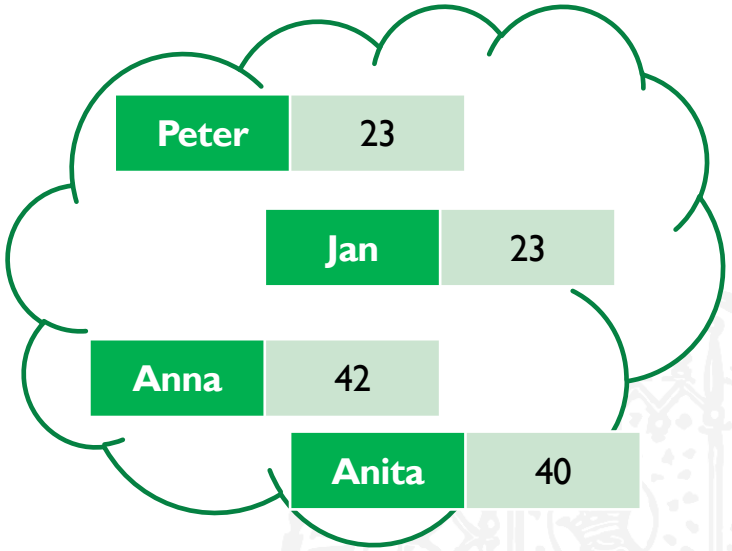
- Was ist das Ergebnis dieses Codes?
- ```
  dict = {}
  dict['a'] = 2
  dict['b'] = 3
  dict['a'] = 4
  print( dict['a'] + dict['b'] )
  ```

a) 4

b) 5

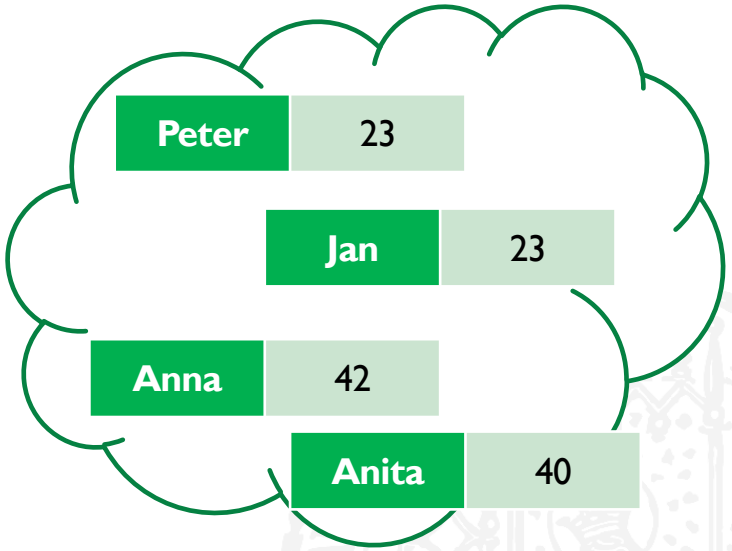
c) 6

d) 7


  ```

DICTIONARIES

- Mit einer for schleife kann man iterieren
- ```
  for key, value in dict.items():
      print(key, value)
  ```
- ```
  for key in dict.keys():
      print(key, dict[key])
  ```
- ```
  for value in dict.values():
      print(value)
  ```


  ```

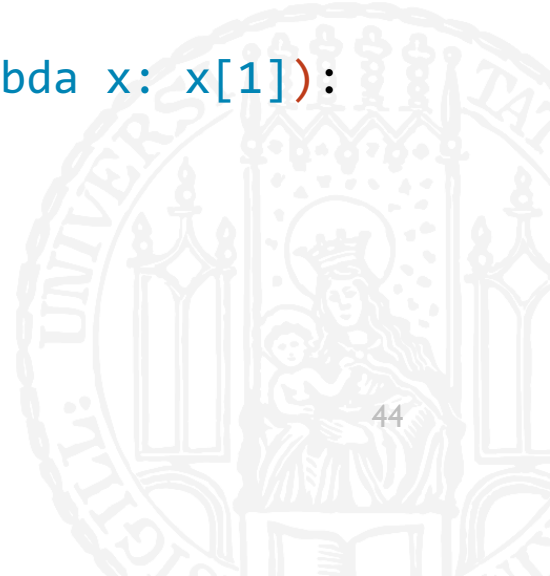
DICTIONARIES

- Dictionaries haben **keine Reihenfolge**
- Beim Ausgaben oder iterieren ist die Reihenfolge **zufällig**
- Man kann sich die Werte aber sortieren lassen



DICTIONARIES


- ```
  #nach Keys sortieren
  for key, value in sorted(dict.items()):
      print(key, value)
  ```
- ```
  #nach Value sortieren
  for key, value in sorted(dict.items(), key=lambda x: x[1]):
      print(key, value)
  ```



QUIZ

- Wie gibt man die Werte eines Dictionaries sortiert nach den Keys aus?

a)
```
for key in sorted(dict)
    print(key)
```

b)
```
for key in sorted(dict.keys())
    print(key)
```

c)
```
for key in sorted(dict.keys())
    print(dict[key])
```

d)
```
for key, value in sorted(dict.keys())
    print(value)
```

QUIZ

- Wie gibt man die **Werte** eines Dictionaries **sortiert nach den Keys** aus?

a)
```
for key in sorted(dict)
    print(key)
```

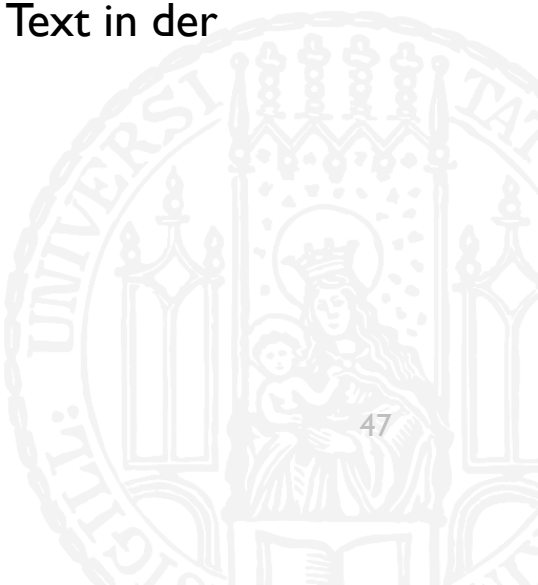
b)
```
for key in sorted(dict.keys())
    print(key)
```

c)
```
for key in sorted(dict.keys())
    print(dict[key])
```

d)
```
for key, value in sorted(dict.keys())
    print(value)
```

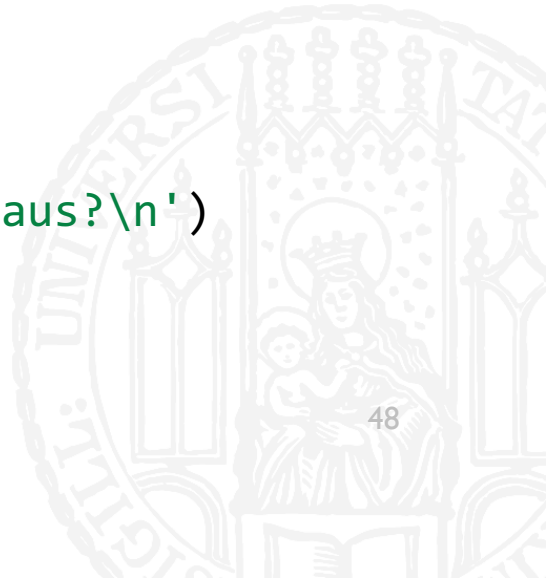
MUSTERLÖSUNG 7-1

- Schreiben Sie ein Programm, das aus gegebenem Anlass fragt, wie oft der Text Wann kommt endlich der Nikolaus? auf dem Terminal ausgegeben und gleichzeitig in die Datei "nikolaus.txt" geschrieben werden soll. Schreiben Sie den Text in der gewünschten Anzahl auf die Konsole und die Datei.



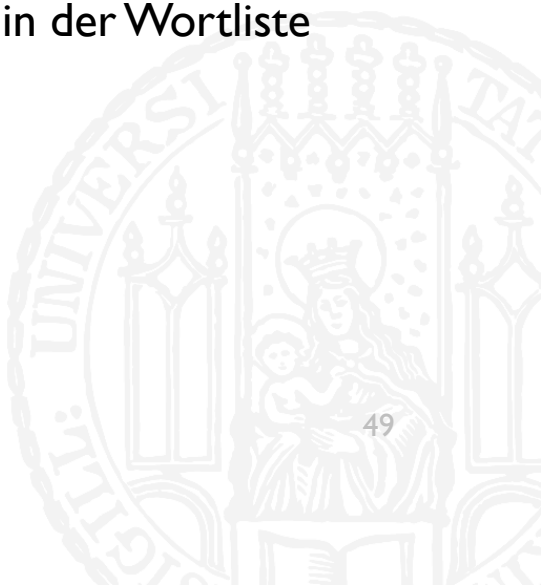
MUSTERLÖSUNG 7-1


```
#!/usr/bin/python3
#Aufgabe 7-1
#Autorin: Leonie Weißweiler

nikolaus = open('nikolaus.txt', 'w')
anzahl = int(input('Wie oft soll \'Wann kommt endlich der
Nikolaus\' ausgegeben werden?\n'))

for x in range(0,anzahl):
    print('Wann kommt endlich der Nikolaus?')
    nikolaus.write('Wann kommt endlich der Nikolaus?\n')

nikolaus.close()
```



MUSTERLÖSUNG 7-2

- Schreiben Sie ein Programm, das den Text aus der Datei liest und erzeugen Sie eine Frequenzliste (dictionary) aus den Wörtern. Speichern Sie die Einträge der Wortliste in die Datei 'frequenzliste.txt'. Das Format der einzelnen Zeile in der Wortliste frequenzliste.txt ist: Das Wort ... kommt ... mal vor.



MUSTERLÖSUNG 7-2


```
#!/usr/bin/python3
#Aufgabe 7-2
#Autorin: Leonie Weißweiler

verbrechen = open('Verbrechen_Strafe.txt', 'r')
frequenzoutput = open('frequenzliste.txt', 'w')
frequenzliste = {}

for line in verbrechen:
    for word in line.split(' '):
        word = word.strip()
        word = word.lower()

        if (word in frequenzliste):
            frequenzliste[word] = frequenzliste[word] + 1

        else:
            frequenzliste[word] = 1
```

MUSTERLÖSUNG 7-2

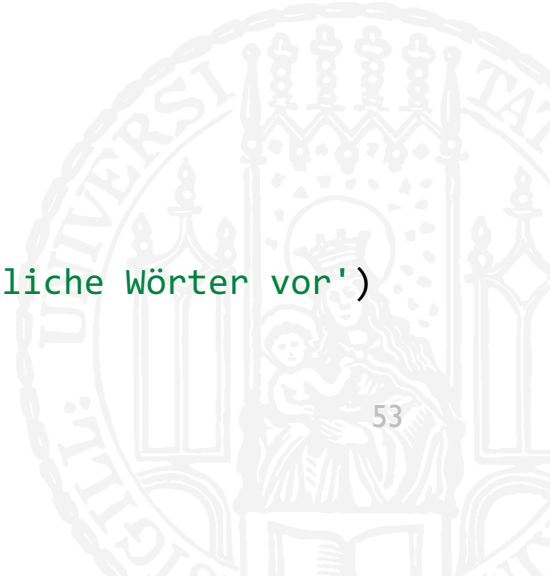

```
for wort, frequenz in frequenzliste.items():
    frequenzoutput.write('Das Wort ')
    frequenzoutput.write(wort)
    frequenzoutput.write(' kommt ')
    frequenzoutput.write(str(frequenz))
    frequenzoutput.write(' mal vor.\n')

verbrechen.close()
frequenzoutput.close()
```

MUSTERLÖSUNG 7-3

- Verwenden sie die erzeugte Frequenzliste um zu ermitteln, wie viele unterschiedliche Wörter in der Datei vorkommen.

MUSTERLÖSUNG 7-3


```
#!/usr/bin/python3
#Aufgabe 7-3
#Autorin: Leonie Weißweiler

verbrechen = open('Verbrechen_Strafe.txt', 'r')
frequenzliste = {}

for line in verbrechen:
    for word in line.split(' '):
        word = word.strip()
        word = word.lower()

        if (word in frequenzliste):
            frequenzliste[word] = frequenzliste[word] + 1

        else:
            frequenzliste[word] = 1

print ('In der Datei kamen', len(frequenzliste), 'unterschiedliche Wörter vor')
verbrechen.close()
```



MUSTERLÖSUNG 7-4

Verändern Sie das vorherige Programm dahingehend, dass Sie den Benutzer nach einem Wort fragen und ausgeben. wie oft das Wort in der Datei vorkommt.

a) Es wird Groß/Kleinschreibung aller Buchstaben variiert.

b) Es wird nur die Groß/Kleinschreibung des Anfangsbuchstabens variiert.

MUSTERLÖSUNG 7-4A


```
#!/usr/bin/python3
#Aufgabe 7-4a
#Autorin: Leonie Weißweiler

verbrechen = open('Verbrechen_Strafe.txt', 'r')
verbrechen = open('Verbrechen_Strafe.txt', 'r')
frequenzliste = {}
benutzerwort = input('Geben Sie ein Wort ein\n')

for line in verbrechen:
    for word in line.split(' '):
        word = word.strip()
        word = word.lower()

        if (word in frequenzliste):
            frequenzliste[word] = frequenzliste[word] + 1

        else:
            frequenzliste[word] = 1

print ('Das Wort', benutzerwort, 'kam', frequenzliste.get(benutzerwort.lower(), 0), 'mal
in der Datei vor.')
```

MUSTERLÖSUNG 7-4B


```
#!/usr/bin/python3
#Aufgabe 7-4b
#Autorin: Leonie Weißweiler

verbrechen = open('Verbrechen_Strafe.txt', 'r')
frequenzliste = {}
benutzerwort = input('Geben Sie ein Wort ein\n')

for line in verbrechen:
    for word in line.split(' '):
        word = word.strip()

        if (word in frequenzliste):
            frequenzliste[word] = frequenzliste[word] + 1

        else:
            frequenzliste[word] = 1
wordlower = benutzerwort[0].lower() + benutzerwort[1:]
wordupper = benutzerwort[0].upper() + benutzerwort[1:]
anzahl = frequenzliste.get(wordlower, 0) + frequenzliste.get(wordupper, 0)


print ('Das Wort', benutzerwort, 'kam', anzahl, 'mal in der Datei vor.')
```