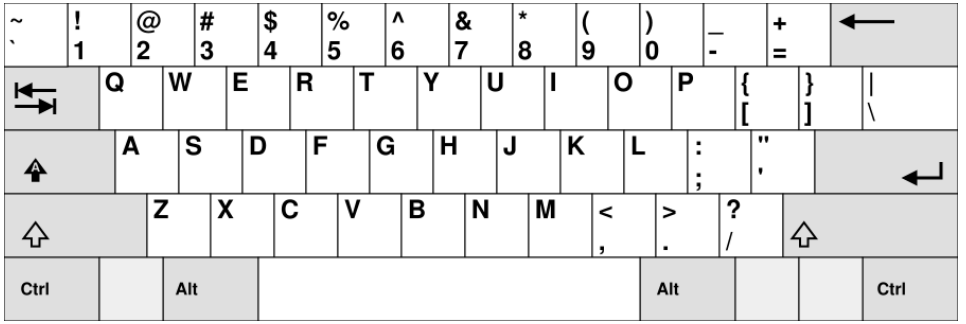
¿Cześć, ça va?

oder

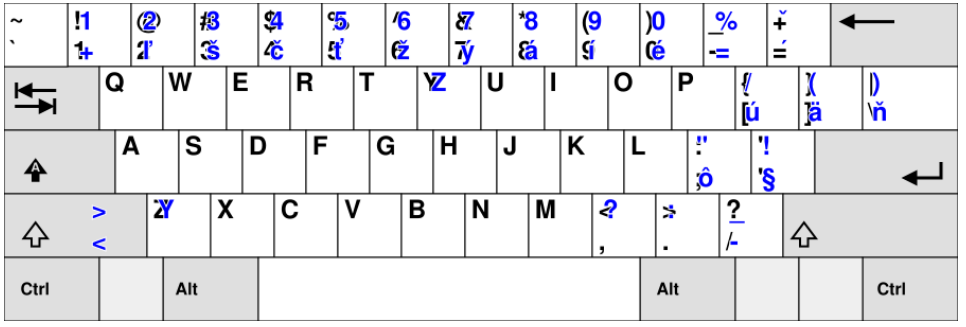
Wie man in der Welt von Unicode überlebt

Miroslav Šedivý

eumiro

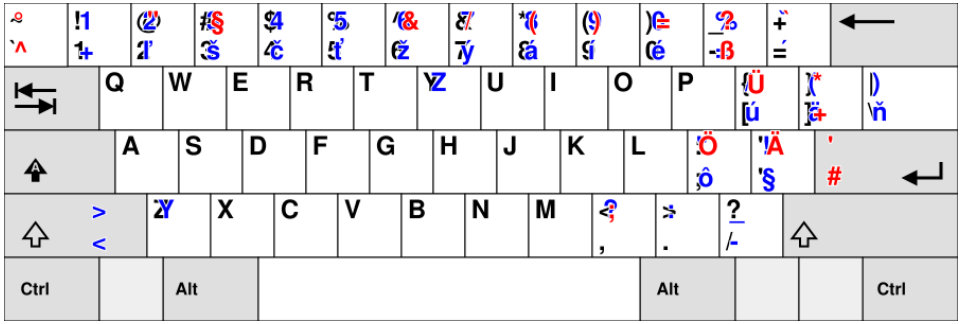

~ `
! 1
@ 2
# 3
$ 4
% 5
^ 6
& 7
* 8
(9
) 0
_ -
+ =
Q
W
E
R
T
Y
U
I
O
P
{ [
}]
| \
A
S
D
F
G
H
J
K
L
: ;
" '
Z
X
C
V
B
N
M
< ,
> .
? /
Ctrl
Alt
Alt
Ctrl

en, sk



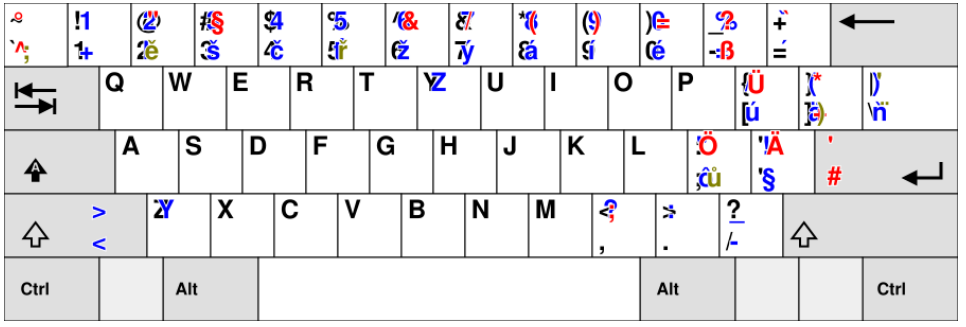
á ä č ď é í ľ Í ň ó ô ŕ š ť ú ý ž

en, sk, **de**



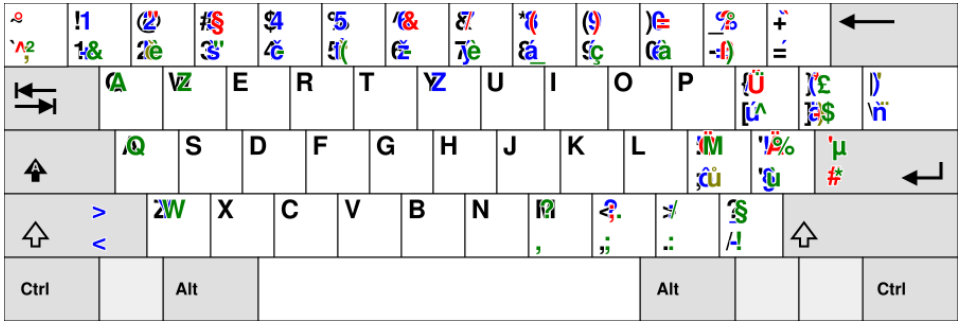
á ä č ď é í ľ Í ň ó ô ŕ š ť ú ý ž **ö ü ß**

en, sk, de, cs



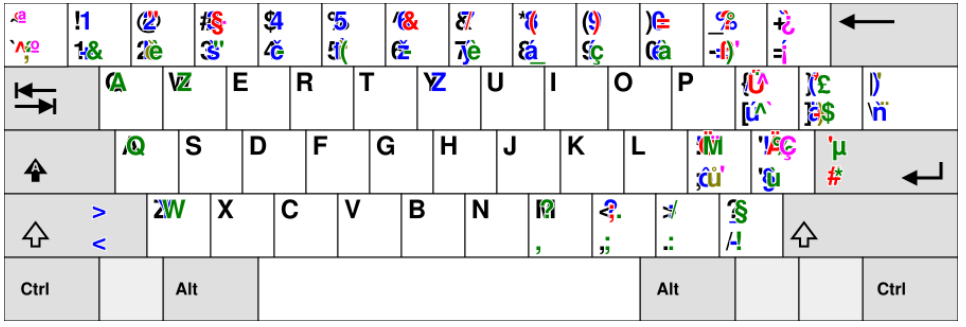
á ä č ď é í ľ ĺ ň ó ô ŕ š ť ú ý ž ö ü ß **ě ř ů**

en, sk, de, cs, **fr**



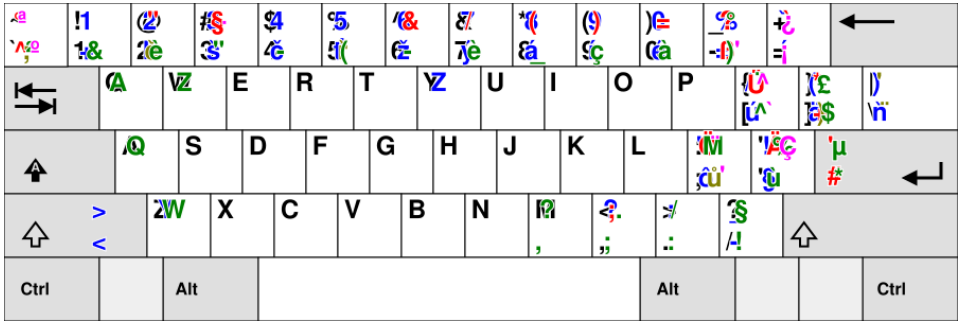
á ä č ď é í ľ ĺ ň ó ô ŕ š ť ú ý ž ö ü ß ě ř ů **à â æ ç è ë ê ï î œ ù û ÿ**

en, sk, de, cs, fr, **es**



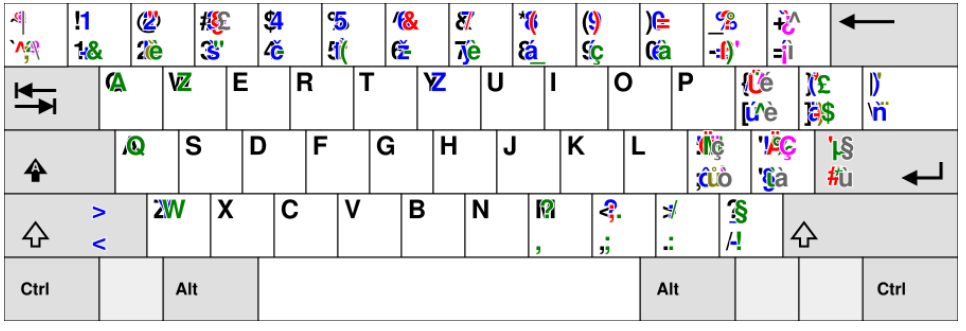
á ä č ď é í ľ ĺ ň ó ô ŕ š ť ú ý ž ö ü ß ě ř ů à â æ ç è ë ê ï î œ ù û ÿ **ñ ¿ ¡**

en, sk, de, cs, fr, es, **pl**



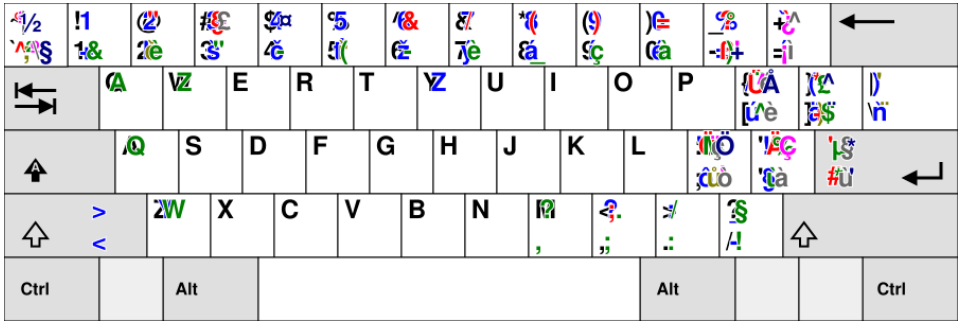
á ä č ď é í ľ ĺ ň ó ô ŕ š ť ú ý ž ö ü ß ě ř ů à â æ ç è ë ê ï î œ ù û ÿ ñ ¿ ¡ **ą ć ę ł ń ś ź ż**

en, sk, de, cs, fr, es, pl, it



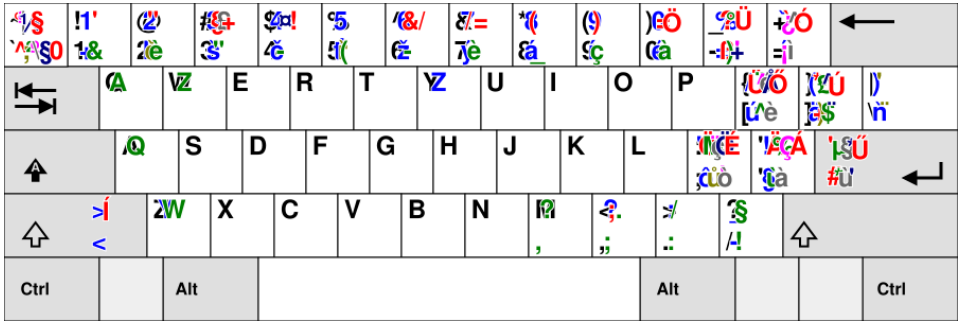
á ä č ď é í ľ Í ň ó ô ŕ š ť ú ý ž ö ü ß ě ř ů à â æ ç è ë ê ï î œ ù û ÿ ñ ¿ ¡ ą ć ę ł ń ś ź ż ò

en, sk, de, cs, fr, es, pl, it, **sv**



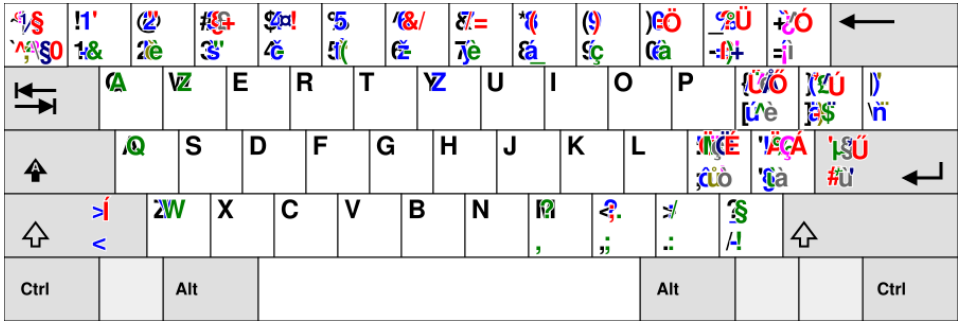
á ä č ď é í ľ Í ň ó ô ŕ š ť ú ý ž ö ü ß ě ř ů à â æ ç è ë ê ï î œ ù û ÿ ñ ¿ ¡ ą ć ę ł ń ś ź ż ò å

en, sk, de, cs, fr, es, pl, it, sv, **hu**



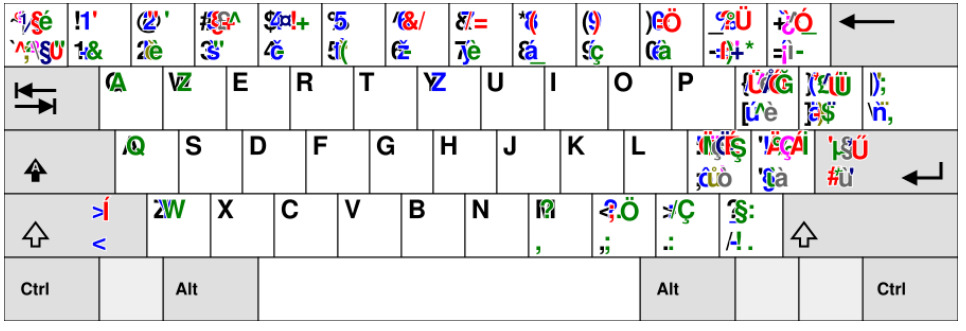
á ä č ď é í ľ Í ň ó ô ŕ š ť ú ý ž ö ü ß ě ř ů à â æ ç è ë ê ï î œ ù û ÿ ñ ¿ ¡ ą ć ę ł ń ś ź ż ò å **ő ű**

en, sk, de, cs, fr, es, pl, it, sv, hu, **eo**



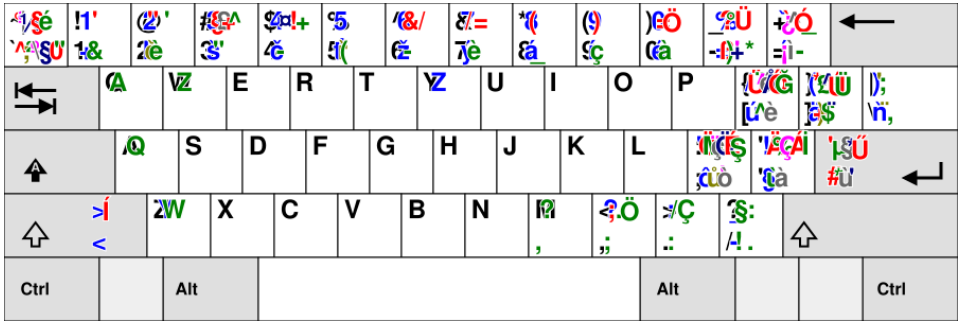
á ä č ď é í ľ Í ň ó ô ŕ š ť ú ý ž ö ü ß ě ř ů à â æ ç è ë ê ï î œ ù û ÿ ñ ¿ ¡ ą ć ę ł ń ś ź ż ò å ő ű **ĉ ĝ ĥ ĵ ŝ ŭ**

en, sk, de, cs, fr, es, pl, it, sv, hu, eo, **tr**



á ä č ď é í ľ Í ň ó ô ŕ š ť ú ý ž ö ü ß ě ř ů à â æ ç è ë ê ï î œ ù û ÿ ñ ¿ ¡ ą ć ę ł ń ś ź ż ò å ő ű ĉ ĝ Ĥ ĵ ŝ ŭ **ğ ı İ ş**

en, sk, de, cs, fr, es, pl, it, sv, hu, eo, tr, …

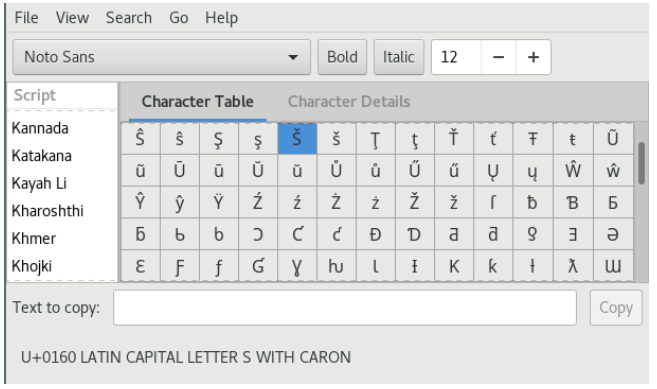


á ä č ď é í ľ ĺ ň ó ô ŕ š ť ú ý ž ö ü ß ě ř ů à â æ ç è ë ê ï î œ ù û ÿ ñ ¿ ¡ ą ć ę ł ń ś ź ż ò å ő ű ĉ ĝ ĥ ĵ ŝ ŭ ğ ı İ ş

þ ð fi fl ø ĸ ə ® © ™ µ § · • ¦ € £ ¥ ¢ … ‹ › « » ‘ ’ “ ” – — ¶ ‰ ∅ √ ± ≤ ≥ ≠ ° ¬ ⓪ ① ② Ⓜ ⓘ ⓡ ⓞ ♯ ♫ ♥

☺ ☹ ← → ⇒ № (1+2) ³ ⁴ ª º ½ ⅓ ⅔ ¼ ¾ ⅕ ⅖ ⅙ ⅚ ⅛ ∞

Zeichentabelle?



Compose Key


Return
Shift
Com-
pose
Alt
Graph


Ctrl
Lock
A
S
D
Shift
Z
X
C
Compose
Character

Compose Key Sequenzen

⌦ " a	ä	⌦ a e	æ	⌦ . .	…
⌦ " O	Ö	⌦ s s	ß	⌦ ? ?	¿
⌦ ' e	é	⌦ t h	þ	⌦ ! !	¡
⌦ ` e	è	⌦ / l	ł	⌦ + -	±
⌦ ^ u	ů	⌦ o c	©	⌦ % o	‰
⌦ = o	ő	⌦ o r	®	⌦ / =	≠
⌦ u u	ŭ	⌦ / o	ø	⌦ 1 2	½
⌦ ~ n	ñ	⌦ = e	€	⌦ m u	µ
⌦ c s	š	⌦ - L	£	⌦ o s	§
⌦ , c	ç	⌦ | c	ç	⌦ ^ 2	²
⌦ . z	ż	⌦ m u	µ	⌦ :)	☺

/usr/share/X11/locale/en_US.UTF-8/Compose

```
<Multi_key> <period> <period> : "…" ellipsis   # HORIZONTAL ELLIPSIS
<Multi_key> <plus> <minus>    : "±" plusminus  # PLUS-MINUS SIGN
<Multi_key> <apostrophe> <a>  : "á" aacute     # LATIN SMALL LETTER A WITH ACUTE
<Multi_key> <s> <s>           : "ß" ssharp     # LATIN SMALL LETTER SHARP S
<Multi_key> <comma> <c>       : "ç" ccedilla   # LATIN SMALL LETTER C WITH CEDILLA
<Multi_key> <c> <S>           : "Š" U0160      # LATIN CAPITAL LETTER S WITH CARON
```

+6k weitere Zeilen

/usr/share/X11/locale/en_US.UTF-8/Compose

```
<Multi_key> <period> <period> : "…" ellipsis   # HORIZONTAL ELLIPSIS
<Multi_key> <plus> <minus>    : "±" plusminus  # PLUS-MINUS SIGN
<Multi_key> <apostrophe> <a>  : "á" aacute     # LATIN SMALL LETTER A WITH ACUTE
<Multi_key> <s> <s>           : "ß" ssharp     # LATIN SMALL LETTER SHARP S
<Multi_key> <comma> <c>       : "ç" ccedilla   # LATIN SMALL LETTER C WITH CEDILLA
<Multi_key> <c> <S>           : "Š" U0160      # LATIN CAPITAL LETTER S WITH CARON
```

+6k weitere Zeilen

~/.XCompose

/usr/share/X11/xkb/rules/base

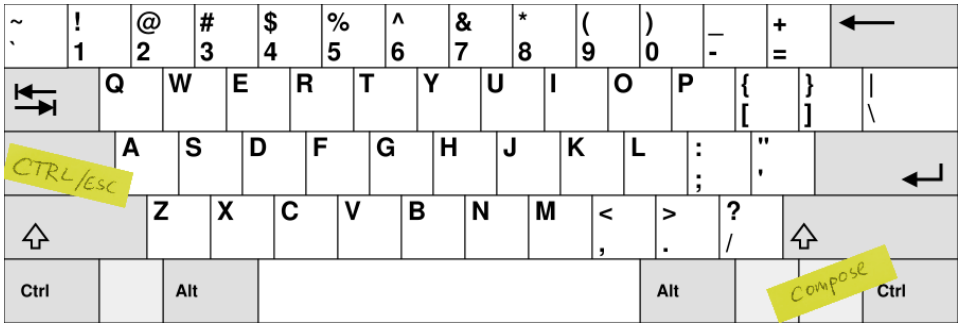
```
compose:ralt        = +compose(ralt)
compose:lwin        = +compose(lwin)
compose:lwin-altgr  = +compose(lwin-altgr)
compose:rwin        = +compose(rwin)
compose:rwin-altgr  = +compose(rwin-altgr)
compose:menu        = +compose(menu)
compose:menu-altgr  = +compose(menu-altgr)
compose:lctrl       = +compose(lctrl)
compose:lctrl-altgr = +compose(lctrl-altgr)
compose:rctrl       = +compose(rctrl)
compose:rctrl-altgr = +compose(rctrl-altgr)
compose:caps        = +compose(caps)
compose:caps-altgr  = +compose(caps-altgr)
compose:102         = +compose(102)
compose:102-altgr   = +compose(102-altgr)
compose:paus        = +compose(paus)
compose:prsc        = +compose(prsc)
compose:sclk        = +compose(sclk)
```

/usr/share/X11/xkb/rules/base

```
compose:ralt        = +compose(ralt)
compose:lwin        = +compose(lwin)
compose:lwin-altgr  = +compose(lwin-altgr)
compose:rwin        = +compose(rwin)
compose:rwin-altgr  = +compose(rwin-altgr)
compose:menu        = +compose(menu)
compose:menu-altgr  = +compose(menu-altgr)
compose:lctrl       = +compose(lctrl)
compose:lctrl-altgr = +compose(lctrl-altgr)
compose:rctrl       = +compose(rctrl)
compose:rctrl-altgr = +compose(rctrl-altgr)
compose:caps        = +compose(caps)
compose:caps-altgr  = +compose(caps-altgr)
compose:102         = +compose(102)
compose:102-altgr   = +compose(102-altgr)
compose:paus        = +compose(paus)
compose:prsc        = +compose(prsc)
compose:sclk        = +compose(sclk)
```

```
setxkbmap us -option 'compose:menu'
```

… et voilà !



Bits of Unicode in Python: ärgerlich, amüsant, zwiespältig

```
>>> a = 'a'
>>> a = a + 'a'
>>> a == 'a'
```

Bits of Unicode in Python: ärgerlich, amüsant, zwiespältig

```
>>> a = 'a'
>>> a = a + 'a'
>>> a == 'a'
```

```
True
```

Bits of Unicode in Python: ärgerlich, amüsant, zwiespältig

```
>>> a = 'a'
>>> a = a + 'a'
>>> a == 'a'
```

```
True
```

```
import unicodedata

def chars(text):
    for i, char in enumerate(text):
        cx = f'\\u{ord(char):04x}'
        cd = f'{ord(char):5}'
        cat = unicodedata.category(char)
        if ord(char) >= 32:
            name = unicodedata.name(char)
        else:
            name = f'^{chr(ord(char) + 64)}'
            char = ' '
        print(f'{i:3}  {char}  {cx}  {cd}  {cat}  {name}')
```

```
>>> code = """a = 'a'
... a = a + 'a'"""
chars(code)
```

```
>>> code = """a = 'a'
... a = a + 'a'"""
chars(code)
```

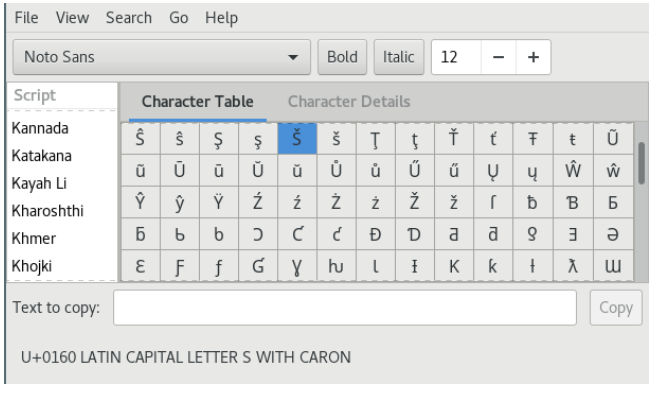
```
  0  a  \u0061     97  Ll  LATIN SMALL LETTER A
  1     \u0020     32  Zs  SPACE
  2  =  \u003d     61  Sm  EQUALS SIGN
  3     \u0020     32  Zs  SPACE
  4  '  \u0027     39  Po  APOSTROPHE
  5  a  \u0061     97  Ll  LATIN SMALL LETTER A
  6  '  \u0027     39  Po  APOSTROPHE
  7     \u000a     10  Cc  ^J
  8  a  \u0430   1072  Ll  CYRILLIC SMALL LETTER A
  9     \u0020     32  Zs  SPACE
 10  =  \u003d     61  Sm  EQUALS SIGN
 11     \u0020     32  Zs  SPACE
 12  a  \u0061     97  Ll  LATIN SMALL LETTER A
 13     \u0020     32  Zs  SPACE
 14  +  \u002b     43  Sm  PLUS SIGN
 15     \u0020     32  Zs  SPACE
 16  '  \u0027     39  Po  APOSTROPHE
 17  a  \u0061     97  Ll  LATIN SMALL LETTER A
 18  '  \u0027     39  Po  APOSTROPHE
```

```
>>> code = """a = 'a'
... a = a + 'a'"""
chars(code)
```

```
  0  a  \u0061     97  Ll  LATIN SMALL LETTER A
  1     \u0020     32  Zs  SPACE
  2  =  \u003d     61  Sm  EQUALS SIGN
  3     \u0020     32  Zs  SPACE
  4  '  \u0027     39  Po  APOSTROPHE
  5  a  \u0061     97  Ll  LATIN SMALL LETTER A
  6  '  \u0027     39  Po  APOSTROPHE
  7     \u000a     10  Cc  ^J
  8  a  \u0430   1072  Ll  CYRILLIC SMALL LETTER A
  9     \u0020     32  Zs  SPACE
 10  =  \u003d     61  Sm  EQUALS SIGN
 11     \u0020     32  Zs  SPACE
 12  a  \u0061     97  Ll  LATIN SMALL LETTER A
 13     \u0020     32  Zs  SPACE
 14  +  \u002b     43  Sm  PLUS SIGN
 15     \u0020     32  Zs  SPACE
 16  '  \u0027     39  Po  APOSTROPHE
 17  a  \u0061     97  Ll  LATIN SMALL LETTER A
 18  '  \u0027     39  Po  APOSTROPHE
```

```
>>> 'ß', '\u1e9e', '\N{LATIN CAPITAL LETTER SHARP S}'
```

☺ \N{WHITE SMILING FACE}



```
>>> word1 = 'süß'
>>> word2 = unicodedata.normalize('NFD', word)
```

```
>>> word1 = 'süß'
>>> word2 = unicodedata.normalize('NFD', word)
```

```
>>> print(word1, len(word1)
süß 3

>>> print(word2, len(word2))
süß 4
```

```
>>> word1 = 'süß'
>>> word2 = unicodedata.normalize('NFD', word)
```

```
>>> print(word1, len(word1)
süß 3

>>> print(word2, len(word2))
süß 4
```

```
>>> chars(word1)
  0  S  \u0053     83  Lu  LATIN CAPITAL LETTER S
  1  ü  \u00fc    252  Ll  LATIN SMALL LETTER U WITH DIAERESIS
  2  ß  \u00df    223  Ll  LATIN SMALL LETTER SHARP S

>>> chars(word2)
  0  S  \u0053     83  Lu  LATIN CAPITAL LETTER S
  1  u  \u0075    117  Ll  LATIN SMALL LETTER U
  2  ¨  \u0308    776  Mn  COMBINING DIAERESIS
  3  ß  \u00df    223  Ll  LATIN SMALL LETTER SHARP S
```

```
>>> for letter in ['\N{LATIN SMALL LETTER U}', '\N{LATIN SMALL LETTER SHARP S}']:
>>>     comb = letter + '\N{COMBINING DIAERESIS}'
>>>     norm = unicodedata.normalize('NFC', comb)
>>>     print(chars(comb))
>>>     print(chars(norm))
```

```
>>> for letter in ['\N{LATIN SMALL LETTER U}', '\N{LATIN SMALL LETTER SHARP S}']:
>>>     comb = letter + '\N{COMBINING DIAERESIS}'
>>>     norm = unicodedata.normalize('NFC', comb)
>>>     print(chars(comb))
>>>     print(chars(norm))
```

```
  0  u  \u0075    117  Ll  LATIN SMALL LETTER U
  1  ¨  \u0308    776  Mn  COMBINING DIAERESIS

  0  ü  \u00fc    252  Ll  LATIN SMALL LETTER U WITH DIAERESIS
```

```
>>> for letter in ['\N{LATIN SMALL LETTER U}', '\N{LATIN SMALL LETTER SHARP S}']:
>>>     comb = letter + '\N{COMBINING DIAERESIS}'
>>>     norm = unicodedata.normalize('NFC', comb)
>>>     print(chars(comb))
>>>     print(chars(norm))
```

```
  0  u  \u0075    117  Ll  LATIN SMALL LETTER U
  1  ¨  \u0308    776  Mn  COMBINING DIAERESIS

  0  ü  \u00fc    252  Ll  LATIN SMALL LETTER U WITH DIAERESIS
```

```
  0  ß  \u00df    223  Ll  LATIN SMALL LETTER SHARP S
  1  ¨  \u0308    776  Mn  COMBINING DIAERESIS
```

21 Answers

oldest newest **votes**

1811

You can't parse [X]HTML with regex. Because HTML can't be parsed by regex. Regex is not a tool that can be used to correctly parse HTML. As I have answered in HTML-and-regex questions here so many times before, the use of regex will not allow you to consume HTML. Regular expressions are a tool that is insufficiently sophisticated to understand the constructs employed by HTML. HTML is not a regular language and hence cannot be parsed by regular expressions. Regex queries are not equipped to break down HTML into its meaningful parts. so many times but it is not getting to me. Even enhanced irregular regular expressions as used by Perl are not up to the task of parsing HTML. You will never make me crack. HTML is a language of sufficient complexity that it cannot be parsed by regular expressions. Even Jon Skeet cannot parse HTML using regular expressions. Every time you attempt to parse HTML with regular expressions, the unholy child weeps the blood of virgins, and Russian hackers pwn your webapp. Parsing HTML with regex summons tainted souls into the realm of the living. HTML and regex go together like love, marriage, and ritual infanticide. The <center> cannot hold it is too late. The force of regex and HTML together in the same conceptual space will destroy your mind like so much watery putty. If you parse HTML with regex you are giving in to Them and their blasphemous ways which doom us all to inhuman toil for the One whose Name cannot be expressed in the Basic Multilingual Plane, he comes. HTML-plus-regexp will liquify the nerves of the sentient whilst you observe, your psyche withering in the onslaught of horror. Regex-based HTML parsers are the cancer that is killing StackOverflow *it is too late it is too late we cannot be saved* the trangession of a child ensures regex will consume all living tissue (except for HTML which it cannot, as previously prophesied) *dear lord help us how can anyone survive this scourge* using regex to parse HTML has doomed humanity to an eternity of dread torture and security holes *using regex* as a tool to process HTML establishes a brea*ch between this world* and the dread realm of corrupt entities (like SGML entities, but *more corrupt) a mere glimp*se of the world of reg**ex parsers for HTML will ins**tantly transport a p*rogrammer's consciousness i*nto a world of ceaseless screaming, he comes~~, the pestilent sl~~ithy regex-infection wil**l devour your HT**ML parser, application and existence for all time like Visual Basic only worse *he comes he comes do not fi*ght h**e comes, hi**s unholy radiance de*stroying all enlightenment, HTML tags* ***leaking from your eyes like liq****uid* pain, the song of regular expre~~ssion parsing~~ will exti*nguish the voices of mor****tal man from the sp****here I can see it can you see it it is beautiful t*he f`inal snuf`fing *of the lie****s of Man ALL IS LOST ALL IS L**OST th*e pony he com*es he c~~omes he co~~**mes the ich**or permeates all MY FACE MY FACE *ᵒh god n****o NO NOOOO N**Θ stop t*he an*gles `are` **not real ZALGO IS TONY THE PONY HE COMES**

Have you tried using an XML parser instead?

Zeichenketten alphabetisch sortieren

```
>>> ''.join(sorted('aAoOuUäÄöÖüÜßſ'))
'AOUaouÄÖÜßäöüſs'
```

Zeichenketten alphabetisch sortieren

```
>>> ''.join(sorted('aAoOuUäÄöÖüÜßẞ'))
'AOUaouÄÖÜßäöüẞ'
```

```
>>> import locale

>>> locale.setlocale(locale.LC_ALL, 'de_DE.UTF-8')
>>> ''.join(sorted('abABäÄ', key=locale.strxfrm))
'aAäÄbB'
```

Zeichenketten alphabetisch sortieren

```
>>> ''.join(sorted('aAoOuUäÄöÖüÜßẞ'))
'AOUaouÄÖÜßäöüẞ'
```

```
>>> import locale

>>> locale.setlocale(locale.LC_ALL, 'de_DE.UTF-8')
>>> ''.join(sorted('abABäÄ', key=locale.strxfrm))
'aAäÄbB'
```

```
>>> locale.setlocale(locale.LC_ALL, 'sv_SE.UTF-8')
>>> ''.join(sorted('abABäÄ', key=locale.strxfrm))
'aAbBäÄ'
```

Zeichenketten alphabetisch sortieren

```
>>> ''.join(sorted('aAoOuUäÄöÖüÜßẞ'))
'AOUaouÄÖÜßäöüẞ'
```

```
>>> import locale

>>> locale.setlocale(locale.LC_ALL, 'de_DE.UTF-8')
>>> ''.join(sorted('abABäÄ', key=locale.strxfrm))
'aAäÄbB'
```

```
>>> locale.setlocale(locale.LC_ALL, 'sv_SE.UTF-8')
>>> ''.join(sorted('abABäÄ', key=locale.strxfrm))
'aAbBäÄ'
```

- Ungarisch: `cékla, cvikli, csípős`

Zeichenketten alphabetisch sortieren

```
>>> ''.join(sorted('aAoOuUäÄöÖüÜßẞ'))
'AOUaouÄÖÜßäöüẞ'
```

```
>>> import locale

>>> locale.setlocale(locale.LC_ALL, 'de_DE.UTF-8')
>>> ''.join(sorted('abABäÄ', key=locale.strxfrm))
'aAäÄbB'
```

```
>>> locale.setlocale(locale.LC_ALL, 'sv_SE.UTF-8')
>>> ''.join(sorted('abABäÄ', key=locale.strxfrm))
'aAbBäÄ'
```

- Ungarisch: `cékla, cvikli, csípős`
- Französisch: `cote, côte, coté, côté`

Zeichenketten alphabetisch sortieren

```
>>> ''.join(sorted('aAoOuUäÄöÖüÜßẞ'))
'AOUaouÄÖÜßäöüẞ'
```

```
>>> import locale

>>> locale.setlocale(locale.LC_ALL, 'de_DE.UTF-8')
>>> ''.join(sorted('abABäÄ', key=locale.strxfrm))
'aAäÄbB'
```

```
>>> locale.setlocale(locale.LC_ALL, 'sv_SE.UTF-8')
>>> ''.join(sorted('abABäÄ', key=locale.strxfrm))
'aAbBäÄ'
```

- Ungarisch: `cékla, cvikli, csípős`
- Französisch: `cote, côte, coté, côté`
- Tschechisch & Slowakisch: `c, č, d, …, h, ch, i`

Locale ist mit dem Prozess verbunden ☹

```
import locale

locale.setlocale(locale.LC_ALL, 'de_DE.UTF-8')
sorted(words, key=locale.strxfrm))
```

Locale ist mit dem Prozess verbunden ☹

```
import locale

locale.setlocale(locale.LC_ALL, 'de_DE.UTF-8')
sorted(words, key=locale.strxfrm))
```

- ICU: International Components for Unicode

```
import icu

collator = icu.Collator.createInstance(icu.Locale('de_DE.UTF-8'))
sorted(words, key=collator.getSortKey)
```

Python 3 und Unicode

- `str` 1M+ Unicode code points
- `bytes`: 256 verschiedene Bytes

Python 3 und Unicode

- `str` 1M+ Unicode code points
- `bytes`: 256 verschiedene Bytes
- encodings: 1 Buchstabe = 1/2/4/n Bytes

```
>>> 'hello world'.encode('ascii')
b'hello world'

>>> b'hello world'.decode('ascii')
'hello world'
```

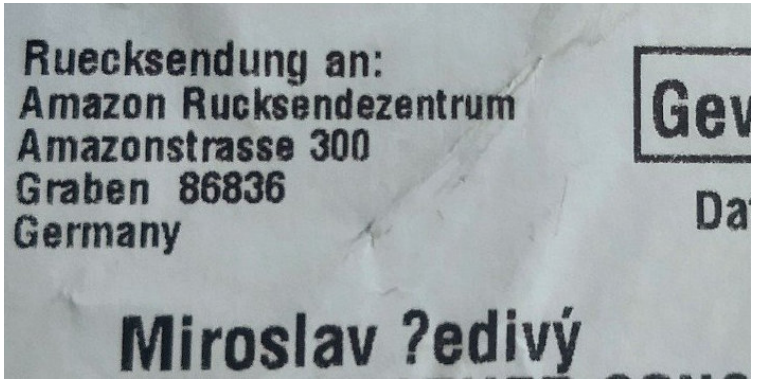
```
>>> 'hello world'.encode('ascii')
b'hello world'

>>> b'hello world'.decode('ascii')
'hello world'
```

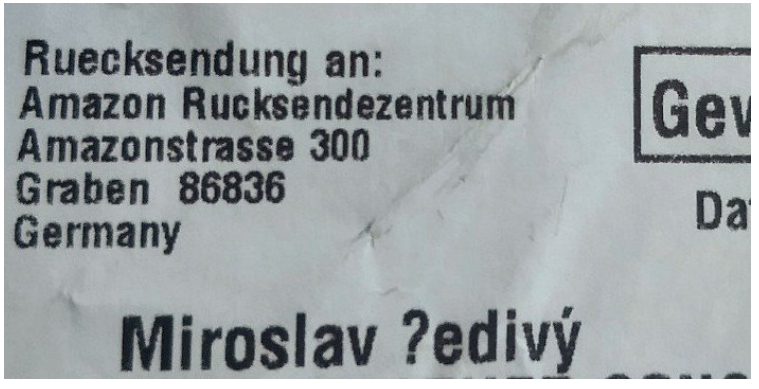
```
>>> 'Genève'.encode('latin1')
b'Gen\xe8ve'

>>> b'Gen\xe8ve'.decode('latin1')
'Genève'
```

```
>>> 'hello world'.encode('ascii')
b'hello world'

>>> b'hello world'.decode('ascii')
'hello world'
```

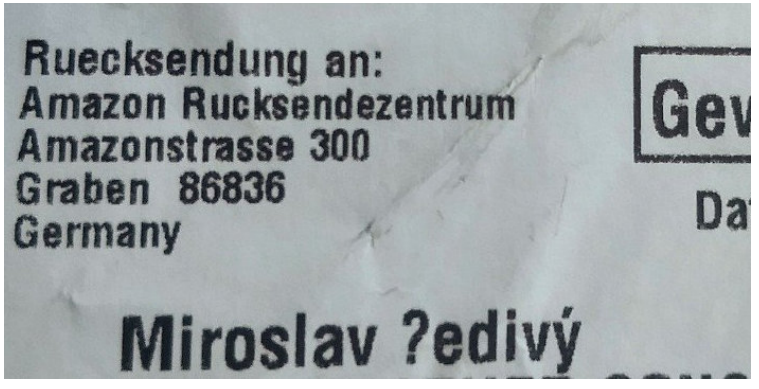
```
>>> 'Genève'.encode('latin1')
b'Gen\xe8ve'

>>> b'Gen\xe8ve'.decode('latin1')
'Genève'
```

```
>>> 'strč prst skrz krk'.encode('latin2')
b'str\xe8 prst skrz krk'

>>> b'str\xe8 prst skrz krk'.decode('latin2')
'strč prst skrz krk'
```


Ruecksendung an:
Amazon Rucksendezentrum
Amazonstrasse 300
Graben 86836
Germany
Miroslav ?edivý

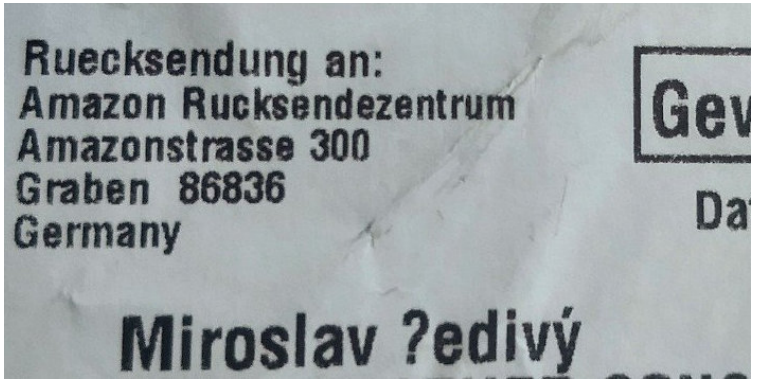



```
>>> 'Šedivý'.encode('latin2')
b'\xa9ediv\xfd'
```




```
>>> 'Šedivý'.encode('latin2')
b'\xa9ediv\xfd'
```

```
>>> 'Šedivý'.encode('latin1')
UnicodeEncodeError: 'latin-1' codec can't encode character '\u0160' in position 0: ordinal not in range(256)
```




```
>>> 'Šedivý'.encode('latin2')
b'\xa9ediv\xfd'
```

```
>>> 'Šedivý'.encode('latin1')
UnicodeEncodeError: 'latin-1' codec can't encode character '\u0160' in position 0: ordinal not in range(256)
```

```
>>> 'Šedivý'.encode('latin1', errors='replace')
>>> b'?ediv\xfd'
```

UTF-8 is cool

UTF-8 is cool

```
>>> 'Šedivý'.encode('utf8')
b'\xc5\xa0ediv\xc3\xbd'
```

UTF-8 is cool

```
>>> 'Šedivý'.encode('utf8')
b'\xc5\xa0ediv\xc3\xbd'
```

```
>>> list('Šedivý'.encode('utf8'))
[197, 160, 101, 100, 105, 118, 195, 189]
```

UTF-8 is cool

```
>>> 'Šedivý'.encode('utf8')
b'\xc5\xa0ediv\xc3\xbd'
```

```
>>> list('Šedivý'.encode('utf8'))
[197, 160, 101, 100, 105, 118, 195, 189]
```

```
>>>'\N{LATIN CAPITAL LETTER S WITH CARON}ediv\N{LATIN SMALL LETTER Y WITH ACUTE}'
'Šedivý'
```

Unicode regular expressions

```
>>> text = 'Zürich_42 - Genève_07'
```

Unicode regular expressions

```
>>> text = 'Zürich_42 - Genève_07'
```

```
>>> import re
>>> re.findall(r'[a-zA-Z]+', text)

['Z', 'rich', 'Gen', 've']
```

Unicode regular expressions

```
>>> text = 'Zürich_42 - Genève_07'
```

```
>>> import re
>>> re.findall(r'[a-zA-Z]+', text)

['Z', 'rich', 'Gen', 've']
```

```
>>> re.findall(r'\w+', text)

['Zürich_42', 'Genève_07']
```

Unicode regular expressions

```
>>> text = 'Zürich_42 - Genève_07'
```

```
>>> import re
>>> re.findall(r'[a-zA-Z]+', text)

['Z', 'rich', 'Gen', 've']
```

```
>>> re.findall(r'\w+', text)

['Zürich_42', 'Genève_07']
```

```
>>> import regex
>>> regex.findall(r'\p{L}+', text)

['Zürich', 'Genève']
```

Ça va ?

Ça va ?

- eine Tastaturbelegung (US QWERTY mit Compose)

Ça va ?

- eine Tastaturbelegung (US QWERTY mit Compose)
- die Nutzers locale nicht kaputt machen: `icu`

Ça va ?

- eine Tastaturbelegung (US QWERTY mit Compose)
- die Nutzers locale nicht kaputt machen: `icu`
- bytes → str (so bald wie möglich)

Ça va ?

- eine Tastaturbelegung (US QWERTY mit Compose)
- die Nutzers locale nicht kaputt machen: `icu`
- bytes → str (so bald wie möglich)
- str → bytes (so spät wie möglich)

Ça va ?

- eine Tastaturbelegung (US QWERTY mit Compose)
- die Nutzers locale nicht kaputt machen: `icu`
- bytes → str (so bald wie möglich)
- str → bytes (so spät wie möglich)
- UTF-8 ist toll, Python 3 ist toll, sei auch toll: nutze Python 3 und UTF-8

Ça va ?

- eine Tastaturbelegung (US QWERTY mit Compose)
- die Nutzers locale nicht kaputt machen: `icu`
- bytes → str (so bald wie möglich)
- str → bytes (so spät wie möglich)
- UTF-8 ist toll, Python 3 ist toll, sei auch toll: nutze Python 3 und UTF-8
- dem Nutzer deiner App zu sagen „Dein Name ist ungültig” nur weil er irgendwas eingibt, was du wegen Faulheit, Ignoranz oder Bosheit nicht behandelst, ist nicht toll

“The enjoyment of one's tools

is an essential ingredient of successful work.”

Donald E. Knuth

Miroslav Šedivý

['mɪrɔslaʊ 'ʃɛɟɪviː]

eumiro eumiro šedivý