

# C2184 Úvod do programování v Pythonu (2021)

## 3. Řetězce, vstup a výstup

### Znak (*character*)

- Je prvek konkrétní znakové sady
- Python 3 používá znakovou sadu *Unicode*
- Příklady znaků v Unicodu:
  - A b ě 4 ( ) # , - Σ π Ж ѡ □ □ □ ¼
  - Řídící (netisknutelné) znaky (např. nový řádek, zvonek)
- Some more stuff

### Řetězec (*string*)

- Posloupnost znaků
- Datový typ `str`
  - Python nemá speciální datový typ pro samotný znak, jedná se o řetězec délky 1

### Zápis řetězců

- Ohraničujeme je pomocí `'` nebo `"` nebo `'''` nebo `"""`
- Příklad: 4 ekvivalentní zápisy slova Hello

```
'Hello'
"Hello"
'''Hello'''
"""Hello"""
```
- Pozor při kopírování: “sexy” uvozovky nefungují!
  - `"Hello"` `"Hello"` `„Hello“` `'Hello'` `‘Hello’` `,Hello‘` ``Hello``

## Výpis řetězců

```
[1]: message = 'Já jsem řetězec.'
```

- Výpis řetězce funkcí print
  - V normálním i interaktivním módu
  - Uvozovky se nevypisují (nejsou součástí řetězce)

```
[2]: print(message)
```

Já jsem řetězec.

- Výpis řetězce jako hodnoty
  - Pouze v interaktivním módu
  - Vypisuje se tak, jak bychom ho zapsali my v kódu, včetně uvozovek

```
[3]: message
```

```
[3]: 'Já jsem řetězec.'
```

## Víceřádkové řetězce

- Musíme použít ''' nebo """

```
[4]: message2 = "dlouhy retezec  
pres hodne radku"  
print(message2)
```

```
File "<ipython-input-4-0c8c9a82edbf>", line 1  
message2 = "dlouhy retezec  
            ^
```

SyntaxError: EOL while scanning string literal

```
[5]: message2 = """dlouhy retezec  
pres hodne radku"""  
print(message2)
```

dlouhy retezec  
pres hodne radku

## Řetězce s uvozovkami / apostrofy

```
[6]: print('I\'m sorry.')
```

I'm sorry.

```
[7]: print("I'm sorry.")
```

I'm sorry.

```
[8]: print("Say \"hello\".")
```

Say "hello".

```
[9]: print('Say "hello".')
```

Say "hello".

```
[10]: print('\'I can\'t say "hello".\')
```

I can't say "hello".

## Speciální znaky a escapování

- Speciální znaky je možné zapsat pomocí zpětného lomítka (*backslash*) \
- Nejdůležitější speciální znaky:
  - \n nový řádek
  - \t tabulátor
  - \' apostrof
  - \" uvozovky
  - \\ zpětné lomítko

```
[11]: print('A\tB\nC\\nD')
```

A        B  
C\nD

## Raw strings

- r před řetězcem ruší význam zpětného lomítka \

```
[12]: print(r'A\tB\nC\\nD')
```

A\tB\nC\\nD

## Otázky:

Který z těchto řetězců je správně zapsaný?

- A) 'Tento text je hrozně dlouhý'
- B) '''Tento text je ještě o hodně delší'''
- C) 'text\'
- D) 'pštros s pštrosicí'

Který z těchto řetězců je nejdelší (má nejvíc znaků)?

- A) 'bim bam'
- B) ""pštros""
- C) "pš\t\t\tt"
- D) str(10+10+10)

```
[13]: str(10+10+10)
```

```
[13]: '30'
```

## Operace s řetězci

### Délka řetězce

- Funkce len

```
[14]: len('ahoj')
```

```
[14]: 4
```

```
[15]: len('Dobry den.\n')
```

```
[15]: 11
```

```
[16]: len('')
```

```
[16]: 0
```

### Spojování řetězců (sřetězení, *concatenation*)

- Operátor +

```
[17]: 'dvě' + ' ' + 'slova'
```

```
a = 2
b = 'slova'
str(a) + ' ' + b
```

a + b

## Násobení řetězců

- Operátor \*

```
'ha' * 100
```

## Indexování řetězců

- Pomocí [] a indexu můžeme získat konkrétní znak z řetězce
- Znaky indexujeme zleva, od 0
- Záporné indexy se počítají zprava, od -1

|     |     |     |    |    |    |    |    |    |    |    |    |      |
|-----|-----|-----|----|----|----|----|----|----|----|----|----|------|
| --- | 0   | 1   | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 |      |
|     | H   | e   | l  | l  | o  |    | W  | o  | r  | l  | d  |      |
|     | -11 | -10 | -9 | -8 | -7 | -6 | -5 | -4 | -3 | -2 | -1 | <--- |

```
retezec = 'Hello World'
```

```
[22]: retezec[0]
```

```
[22]: 'H'
```

```
[23]: retezec[1]
```

```
[23]: 'e'
```

```
[24]: retezec[-1]
```

```
[24]: 'd'
```

```
[25]: retezec[-2]
```

```
[25]: 'l'
```

### Podřetězec

- Rozsah zapisujeme [start:stop]
- Index start je zahrnut ve výsledku, index stop nikoliv!
- Prázdné start znamená od začátku
- Prázdné stop znamená do konce

```
[26]: message = 'Hello World'
```

```
[27]: message[2:5]
```

```
[27]: 'llo'
```

```
[28]: message[-4:]
```

```
[28]: 'orld'
```

```
[29]: message[:4]
```

```
[29]: 'Hell'
```

```
[30]: message[:]
```

```
[30]: 'Hello World'
```

- Přeskakování znaků: [start:stop:step]

```
[31]: message = '0123456789'  
message[1:8:2]
```

- Lze vynechat start, stop, nebo oboje

- Obrácení řetězce: nastavíme step na -1

0876543210'

```
[34]: message = 'Hello World'
```

```
[35]: 'W'
```

```

↳ -----
      TypeError                                Traceback (most
↳ recent call last)

      <ipython-input-36-dffff5adfd316> in <module>
----> 1 message[6] = 'X'

      TypeError: 'str' object does not support item assignment

```

```
Hello Xorl'd
Hello World
```

## Hledání podřetězců (*substrings*)

- Operátory `in`, `not in` testují jestli je/není jehla obsažena v kupce sena

```
[38]: '123' in 'ABCDefgh1234'
```

```
[38]: True
```

```
[39]: '456' in 'ABCDefgh1234'
```

```
[39]: False
```

```
[40]: '456' not in 'ABCDefgh1234'
```

```
[40]: True
```

```
[41]: 'ABCDefgh1234' in '123'
```

```
[41]: False
```

## Počítání a hledání

- Pomocí metody `count` počítáme počet výskytů jehel v kupce sena
- Pomocí metody `find` hledáme index prvního výskytu jehly

(Metoda = funkce, kterou voláme přímo na nějakém objektu pomocí tečky.)

```
[42]: message = 'Nesnese se se sestrou.'  
message.count('se')
```

```
[42]: 4
```

```
[43]: 'se'.count(message)
```

```
[43]: 0
```

```
[44]: message.find('se')
```

```
[44]: 5
```

```
[45]: message.find('SE')
```

```
[45]: -1
```

## Hledání pouze na začátku / na konci

- Metody `startswith`, `endswith`



```
[46]: message = 'Nesnese se se sestrou.'  
message.startswith('se')
```

[46]: False

```
[47]: message.startswith('Nes')
```

[47]: True

```
[48]: message.endswith('.')
```

[48]: True

## Nahrazování

- Metoda `replace` nahradí starý podřetězec za nový
- Volitelný třetí parametr `count` nastaví maximální počet nahrazení; pokud není nastavený, nahradí se všechny výskyty

```
[49]: 'Spam, egg, Spam, Spam, bacon and Spam'.replace('Spam', 'banana')
```

[49]: 'banana, egg, banana, banana, bacon and banana'

```
[50]: 'Spam, egg, Spam, Spam, bacon and Spam'.replace('Spam', '')
```

[50]: ', egg, , , bacon and '

```
[51]: 'Spam, egg, Spam, Spam, bacon and Spam'.replace('Spam', 'banana', 2)
```

[51]: 'banana, egg, banana, Spam, bacon and Spam'

## Odstranění bílých znaků na okrajích

- Metoda `strip` odstraní bílé znaky z obou konců řetězce
- Metoda `lstrip` odstraňuje pouze zleva (*left-strip*)
- Metoda `rstrip` odstraňuje pouze zprava (*right-strip*)
- Bílé znaky uvnitř řetězce jsou zachovány
- (Volitelný parametr těchto metod popisuje, jaké znaky se mají z okrajů odstranit.)

```
[52]: message = '    já jsem nepovedený \n řetězec    \t\n '
```

```
[53]: message.strip()
```

[53]: 'já jsem nepovedený \n řetězec'

```
[54]: message.lstrip()
```

```
[54]: 'já jsem nepovedený \n řetězec  \t\n '
```

```
[55]: message.rstrip()
```

```
[55]: '    já jsem nepovedený \n řetězec'
```

```
[56]: message.rstrip('\n\t')
```

```
[56]: '    já jsem nepovedený \n řetězec  \t\n '
```

### Rozdělení řetězce na části

- Metoda `split` rozdělí řetězec dle zadaného separátoru
- Parametr `sep` nastavuje separátor. Defaultní separátor jsou všechny shluky bílých znaků (mezera, `\t`, `\n`...)
- Parametr `maxsplit` omezuje počet dělení. Defaultní `maxsplit` je  $\infty$
- (Tato metoda vrací *seznam* řetězců. O seznamech si víc řekneme později.)

```
[57]: message = 'dvě slova \n\ntři celá slova'
```

```
[58]: message.split()
```

```
[58]: ['dvě', 'slova', 'tři', 'celá', 'slova']
```

```
[59]: message.split(sep='\n')
```

```
[59]: ['dvě slova ', '', 'tři celá slova']
```

```
[60]: message.split(sep=' ', maxsplit=2)
```

```
[60]: ['dvě', 'slova', '\n\ntři celá slova']
```

```
[61]: message.split(sep='lo')
```

```
[61]: ['dvě s', 'va \n\ntři celá s', 'va']
```

- Rozbalení seznamu:

```
[62]: name, surname = 'Jan Novák'.split()  
name
```

```
[62]: 'Jan'
```

```
[63]: surname
```

```
[63]: 'Novák'
```

### **Změna velikosti písma**

```
[64]: message = 'Hello world!'
```

```
[65]: message.upper()
```

```
[65]: 'HELLO WORLD!'
```

```
[66]: message.lower()
```

```
[66]: 'hello world!'
```

```
[67]: message.swapcase()
```

```
[67]: 'hELLO WORLD!'
```

```
[68]: message.capitalize()
```

```
[68]: 'Hello world!'
```

```
[69]: message.title()
```

```
[69]: 'Hello World!'
```

### **Logické operace**

- `isalpha` – obsahuje pouze písmena?
- `isdigit` – obsahuje pouze číslice?
- `isalnum` – obsahuje pouze písmena a číslice?
- `isspace` – obsahuje pouze bílé znaky?
- `isupper`, `islower` – jsou všechna písmena velká/malá?

```
[70]: 'Python3'.isalnum()
```

```
[70]: True
```

```
[71]: 'Python 3'.isalnum()
```

```
[71]: False
```

```
[72]: '\t\n\r'.isspace()
```

[72]: True

```
[73]: 'a \t\n\r'.isspace()
```

[73]: False

```
[74]: 'Mám 5 jablíček.'.islower()
```

[74]: False

```
[75]: 'mám 5 jablíček.'.islower()
```

[75]: True

```
[76]: 'A Je To Tady'.istitle()
```

[76]: True

### Otázky:

text = 'Lorem ipsum dolor sit amet'

Který z těchto výrazů vrátí True?

- A) text[5] == 'm'
- B) text[1:4] == 'orem'.
- C) ' ' in text
- D) text.isalpha()

Který z těchto výrazů vrátí True?

- A) text.replace('n', 'f') == text
- B) text.strip('Lol') == text
- C) 'abc' + 'def' == 'abc def'
- D) "5" \* 5 == '55555'

Který z těchto výrazů vrátí True?

- A) 'Brrrrr no to je zima'.strip('Br').startswith('no')
- B) 'Brno'.replace('r', 'rrrrr')[-1] == 'n'
- C) 'Toto léto stojí za to'.count('to') <= 4
- D) 'Brno'.find('r') == 'Olomouc'.find('o')

## Formátování řetězce

- **Old style** – pomocí % (zastaralé, nepoužívat)
- **New style** – pomocí metody format
- **f-strings**

```
[77]: name = 'Anička'
      age = 5

      print('Jmenuji se %s a je mi %d let.' % (name, age))      # old style
      print('Jmenuji se {} a je mi {} let.'.format(name, age))  # new style
      print(f'Jmenuji se {name} a je mi {age} let.')            # f-string
```

Jmenuji se Anička a je mi 5 let.  
Jmenuji se Anička a je mi 5 let.  
Jmenuji se Anička a je mi 5 let.

### f-strings

- Nejnovější a nejpraktičtější způsob
- “The best of Python 3.6”
- Těsně před řetězec vložíme f, v řetězci pak můžeme použít značky {}
- Za značku {x} se do f-stringu dosadí str(x)

```
[78]: name = 'Anička'
      age = 5
      what = 'Prasátko Peppa'

      f'Jmenuji se {name}, je mi {age} let, líbí se mi {what}.'
```

[78]: 'Jmenuji se Anička, je mi 5 let, líbí se mi Prasátko Peppa.'

## Typy a formátování

- Ve značce za dvojtečkou můžeme specifikovat:
  - Zarovnání: {x:<}, {x:>} nebo {x:^}
  - Délku: {x:10}
  - Počet desetinných míst: {x:.2}
  - Typ/formát: {x:s} řetězec, {x:n} číslo, {x:f} reálné číslo, {x:e} vědecký formát čísla...
- Zadané pořadí je nutné dodržet

```
[79]: f'{age}' # Defaultní formát
```

```
[79]: '5'
```

```
[80]: f'{age:.3f}' # Reálné číslo se 3 des. místy
```

```
[80]: '5.000'
```

```
[81]: f'{age:04}' # Vědecký formát se 2 des. čísly, roztáhni na 20 znaků a ↵  
      ↪ zarovnej doprava
```

```
[81]: '0005'
```

```
[82]: f'{age:^20.2E}'
```

```
[82]: '          5.00E+00          '
```

```
[83]: f'Jmenuji se {name}, je mi {age:.2f} let, líbí se mi {what:^25}.'
```

```
[83]: 'Jmenuji se Anička, je mi 5.00 let, líbí se mi          Prasátko Peppa ↵  
      ↪ .''
```

### Metoda format

- Umožňuje nám připravit si šablonu se značkami {}
- Značky se nahradí až při volání metody format z její parametrů

```
[84]: template = 'Jmenuji se {name}, je mi {age:.1f} let, líbí se mi {what: ↵  
      ↪ ^25}.' # Toto není f-string, pouze šablona  
template
```

```
[84]: 'Jmenuji se {name}, je mi {age:.1f} let, líbí se mi {what:^25}.'
```

```
[85]: template.format(name='Anička', age=5.123456, what=what)
```

```
[85]: 'Jmenuji se Anička, je mi 5.1 let, líbí se mi          Prasátko Peppa ↵  
      ↪ .''
```

```
[86]: template.format(age=2*50, name='Sigmund', what='Monty Python')
```

```
[86]: 'Jmenuji se Sigmund, je mi 100.0 let, líbí se mi          Monty Python ↵  
      ↪ .''
```

- Značky nemusíme pojmenovávat:

```
[87]: template = 'Jmenuji se {}, je mi {:.1f} let, líbí se mi {}.'  
template.format('Anička', 5, 'Prasátko Peppa')
```

```
[87]: 'Jmenuji se Anička, je mi 5.0 let, líbí se mi Prasátko Peppa.'
```

- Značky můžeme indexovat (od 0, žádné číslo v sekvenci nesmí chybět)

```
[88]: template = 'Jmenuji se {2}, je mi {1:.1f} let, líbí se mi {0}.'  
template.format('Anička', 5, 'Prasátko Peppa')
```

```
[88]: 'Jmenuji se Prasátko Peppa, je mi 5.0 let, líbí se mi Anička.'
```

## Vstup a výstup

Vstup (*input*) / standardní vstup (*stdin*)

- Slouží pro předání informací do běžícího programu
- Funkce `input`

Výstup (*output*) / standardní výstup (*stdout*)

- Slouží pro předání informací ven z běžícího programu
- Funkce `print`

### Funkce `input`

- Uživateli vypíše hlášku (nepovinné)
- Čeká na vstup od uživatele až do stisknutí klávesy Enter
- Výsledkem funkce je řetězec, který zadal uživatel (vždy typu `str`)

Zkuste si spustit tento kód:

```
[89]: name = input('Jak se jmenuješ? ')  
age = input('Kolik ti je let? ')  
what = input('Co se ti líbí? ')  
print(f'Jmenuješ se {name}, je ti {age} let, líbí se ti {what}.')
```

Jmenuješ se Bob, je ti 7 let, líbí se ti programování.

```
[90]: number = input() # input() bez hlášky  
print(2 * int(number))
```

10

### Funkce `print`

- Všechny své parametry přemění na řetězec (pomocí funkce `str`) a vypíše je

```
[91]: print('ahoj', 5, True)
```

ahoj 5 True

### Speciální parametry funkce print

- Parametr sep (default je ' ')

```
[92]: print(1, 2, 3)
```

1 2 3

```
[93]: print(1, 2, 3, sep=', ')
```

1, 2, 3

```
[94]: print(1, 2, 3, sep='\n')
```

1  
2  
3

```
[95]: print(1, 2, 3, sep='')
```

123

- Parametr end (default je '\n')

```
[96]: print(1, 2, 3)  
print(4, 5, 6)
```

1 2 3  
4 5 6

```
[97]: print(1, 2, 3, end='; ')  
print(4, 5, 6)
```

1 2 3; 4 5 6

```
[98]: print(1, 2, 3, sep=',', end='')  
print(4, 5, 6, sep='|', end='|.')
```

1,2,34|5|6.

### Otázky:

Který z těchto příkazů NEVYPÍŠE na výstup True?

- A) print('False' in 'False')



- B) `print('Torture'[0::2])`
- C) `print('Tr', 'U'.lower(), 'e', sep='')`
- D) `print('True'.isupper)`

Který z těchto výrazů se vyhodnotí na řetězec obsahující znak { ?

- A) `f'Number: {123}'`
- B) `'Number: {}' * len('')`
- C) `'Number: {}'.format('{123}')`
- D) `'Number: {}'.format(len(''))`

Napsali jsme si tento skript:

```
text = input('Zadej počet a druh ovoce: ')
a, b = text.split()
print(int(a) * len(b))
```

Co může být zobrazeno na terminále po spuštění skriptu?

- A)  
Zadej počet a druh ovoce: 10 jablek  
60
- B)  
Zadej počet a druh ovoce: 10 granátových jablek  
180
- C)  
Zadej počet a druh ovoce: '3 melouny'  
21
- D)  
Zadej počet a druh ovoce:  
5 švestek  
35

## Reprezentace znaků v počítači

- Každý znak v znakové sadě je reprezentován svým ordinálním číslem
- Funkce `ord` zjišťuje ordinální číslo znaku
- Opakem je funkce `chr`, která vrací znak pro zadané ordinální číslo

## Znaková sada ASCII = prvních 128 znaků sady Unicode

**Dec** = ord. č. v desítkové soustavě, **Hex** = ord. č. v šestnáctkové soustavě, **Char** = znak

| Dec       | Hex | Char                      | Dec       | Hex | Char         | Dec       | Hex | Char | Dec        | Hex | Char          |
|-----------|-----|---------------------------|-----------|-----|--------------|-----------|-----|------|------------|-----|---------------|
| <b>0</b>  | 00  | <i>Null</i>               | <b>32</b> | 20  | <i>Space</i> | <b>64</b> | 40  | @    | <b>96</b>  | 60  | `             |
| <b>1</b>  | 01  | <i>Start of heading</i>   | <b>33</b> | 21  | !            | <b>65</b> | 41  | A    | <b>97</b>  | 61  | a             |
| <b>2</b>  | 02  | <i>Start of text</i>      | <b>34</b> | 22  | "            | <b>66</b> | 42  | B    | <b>98</b>  | 62  | b             |
| <b>3</b>  | 03  | <i>End of text</i>        | <b>35</b> | 23  | #            | <b>67</b> | 43  | C    | <b>99</b>  | 63  | c             |
| <b>4</b>  | 04  | <i>End of transmit</i>    | <b>36</b> | 24  | \$           | <b>68</b> | 44  | D    | <b>100</b> | 64  | d             |
| <b>5</b>  | 05  | <i>Enquiry</i>            | <b>37</b> | 25  | %            | <b>69</b> | 45  | E    | <b>101</b> | 65  | e             |
| <b>6</b>  | 06  | <i>Acknowledge</i>        | <b>38</b> | 26  | &            | <b>70</b> | 46  | F    | <b>102</b> | 66  | f             |
| <b>7</b>  | 07  | <i>Bell \a</i>            | <b>39</b> | 27  | '            | <b>71</b> | 47  | G    | <b>103</b> | 67  | g             |
| <b>8</b>  | 08  | <i>Backspace \b</i>       | <b>40</b> | 28  | (            | <b>72</b> | 48  | H    | <b>104</b> | 68  | h             |
| <b>9</b>  | 09  | <i>Tab \t</i>             | <b>41</b> | 29  | )            | <b>73</b> | 49  | I    | <b>105</b> | 69  | i             |
| <b>10</b> | 0a  | <i>Line feed \n</i>       | <b>42</b> | 2a  | *            | <b>74</b> | 4a  | J    | <b>106</b> | 6a  | j             |
| <b>11</b> | 0b  | <i>Vertical tab \v</i>    | <b>43</b> | 2b  | +            | <b>75</b> | 4b  | K    | <b>107</b> | 6b  | k             |
| <b>12</b> | 0c  | <i>Form feed \f</i>       | <b>44</b> | 2c  | ,            | <b>76</b> | 4c  | L    | <b>108</b> | 6c  | l             |
| <b>13</b> | 0d  | <i>Carriage return \r</i> | <b>45</b> | 2d  | -            | <b>77</b> | 4d  | M    | <b>109</b> | 6d  | m             |
| <b>14</b> | 0e  | <i>Shift out</i>          | <b>46</b> | 2e  | .            | <b>78</b> | 4e  | N    | <b>110</b> | 6e  | n             |
| <b>15</b> | 0f  | <i>Shift in</i>           | <b>47</b> | 2f  | /            | <b>79</b> | 4f  | O    | <b>111</b> | 6f  | o             |
| <b>16</b> | 10  | <i>Data link escape</i>   | <b>48</b> | 30  | 0            | <b>80</b> | 50  | P    | <b>112</b> | 70  | p             |
| <b>17</b> | 11  | <i>Device control 1</i>   | <b>49</b> | 31  | 1            | <b>81</b> | 51  | Q    | <b>113</b> | 71  | q             |
| <b>18</b> | 12  | <i>Device control 2</i>   | <b>50</b> | 32  | 2            | <b>82</b> | 52  | R    | <b>114</b> | 72  | r             |
| <b>19</b> | 13  | <i>Device control 3</i>   | <b>51</b> | 33  | 3            | <b>83</b> | 53  | S    | <b>115</b> | 73  | s             |
| <b>20</b> | 14  | <i>Device control 4</i>   | <b>52</b> | 34  | 4            | <b>84</b> | 54  | T    | <b>116</b> | 74  | t             |
| <b>21</b> | 15  | <i>Neg. acknowledge</i>   | <b>53</b> | 35  | 5            | <b>85</b> | 55  | U    | <b>117</b> | 75  | u             |
| <b>22</b> | 16  | <i>Synchronous idle</i>   | <b>54</b> | 36  | 6            | <b>86</b> | 56  | V    | <b>118</b> | 76  | v             |
| <b>23</b> | 17  | <i>End trans. block</i>   | <b>55</b> | 37  | 7            | <b>87</b> | 57  | W    | <b>119</b> | 77  | w             |
| <b>24</b> | 18  | <i>Cancel</i>             | <b>56</b> | 38  | 8            | <b>88</b> | 58  | X    | <b>120</b> | 78  | x             |
| <b>25</b> | 19  | <i>End of medium</i>      | <b>57</b> | 39  | 9            | <b>89</b> | 59  | Y    | <b>121</b> | 79  | y             |
| <b>26</b> | 1a  | <i>Substitution</i>       | <b>58</b> | 3a  | :            | <b>90</b> | 5a  | Z    | <b>122</b> | 7a  | z             |
| <b>27</b> | 1b  | <i>Escape</i>             | <b>59</b> | 3b  | ;            | <b>91</b> | 5b  | [    | <b>123</b> | 7b  | {             |
| <b>28</b> | 1c  | <i>File separator</i>     | <b>60</b> | 3c  | <            | <b>92</b> | 5c  | \    | <b>124</b> | 7c  |               |
| <b>29</b> | 1d  | <i>Group separator</i>    | <b>61</b> | 3d  | =            | <b>93</b> | 5d  | ]    | <b>125</b> | 7d  | }             |
| <b>30</b> | 1e  | <i>Record separator</i>   | <b>62</b> | 3e  | >            | <b>94</b> | 5e  | ^    | <b>126</b> | 7e  | ~             |
| <b>31</b> | 1f  | <i>Unit separator</i>     | <b>63</b> | 3f  | ?            | <b>95</b> | 5f  | _    | <b>127</b> | 7f  | <i>Delete</i> |

```
[99]: ord('A')
```

```
[99]: 65
```

```
[100]: ord('č')
```

```
[100]: 269
```

```
[101]: chr(65)
```

```
[101]: 'A'
```

```
[102]: chr(269)
```

```
[102]: 'č'
```

```
[103]: chr(127159)
```

```
[103]: '☐'
```

- Escapování pomocí ordinálních čísel:
  - \x??, kde ?? je ordinální číslo znaku v šestnáctkové soustavě
  - \u????, kde ???? je ordinální číslo znaku šestnáctkové soustavě
  - \U????????, kde ???????? je ordinální číslo znaku v šestnáctkové soustavě
  - \N{name}, kde name je název Unicode znaku

```
[104]: print('\x7A \u007A \U0000007A \U0001F0B9')
```

```
z z z ☐
```

```
[105]: print('\N{pound sign} \N{playing card seven of spades}')
```

```
£ ☐
```