C2184 Úvod do programování v Pythonu

3. Řetězce, vstup a výstup

Znak (*character*)

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

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 textu: “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)
```

```
  Input In [4]
    message2 = "dlouhy retezec
               ^
SyntaxError: unterminated string literal (detected at line 1)
```

[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('Say "hello".')
```

```
Say "hello".
```

[8]:
```
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

[9]:
```
print('A\tB\nC\\nD')
```

```
A       B
C\nD
```

[10]:
```
print('I\'m sorry')
```

```
I'm sorry
```

Raw strings

- `r` před řetězcem ruší význam zpětného lomítka `\`

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

```
A\tB\nC\\nD
```

Otázky:

Který z těchto řetězců je správne 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)

Operace s řetězci

Délka řetězce

- Funkce len

[12]:
```
len('ahoj')
```

[12]: 4

[13]:
```
len('Dobrý den.\n')
```

[13]: 11

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

[14]: 0

Spojování řetězců (sřetězení, *concatenation*)

- Operátor +

[15]:
```
'dvě' + ' ' + 'slova'
```

[15]: 'dvě slova'

[16]:
```
a = 2
b = 'slova'
str(a) + ' ' + b
```

[16]: '2 slova'

[17]:
```
a + b
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call
  ↪last)
/home/adam/School/Praca/Python/2022/Python/cviko_03/03_Retezce.ipynb␣
  ↪Cell 34 in <cell line: 1>()
----> <a href='vscode-notebook-cell:/home/adam/School/Praca/Python/
  ↪2022/Python/cviko_03/03_Retezce.ipynb#X45sZmlsZQ%3D%3D?line=0'>1</
  ↪a> a + b
```

```
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Násobení řetězců

- Operátor *

[18]:
```
'ha' * 10
```

[18]:
```
'hahahahahahahahahaha'
```

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  <---
```

[19]:
```
retezec = 'Hello World'
```

[20]:
```
retezec[0]
```

[20]:
```
'H'
```

[21]:
```
retezec[1]
```

[21]:
```
'e'
```

[22]:
```
retezec[-1]
```

[22]:
```
'd'
```

[23]:
```
retezec[-2]
```

[23]:
```
'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

[24]:
```
message = 'Hello World'
```

[25]:
```
message[2:5]
```

[25]:
```
'llo'
```

[26]:
```
message[-4:]
```

[26]:
```
'orld'
```

[27]:
```
message[:4]
```

[27]:
```
'Hell'
```

[28]:
```
message[:]
```

[28]:
```
'Hello World'
```

- Přeskakování znaků: `[start:stop:step]`

[29]:
```
message = '0123456789'
message[1:8:2]
```

[29]:
```
'1357'
```

- Lze vynechat `start`, `stop`, nebo oboje

[30]:
```
message[::3]
```

[30]:
```
'0369'
```

- Obrácení řetězce: nastavíme `step` na -1

[31]:
```
message[::-1]
```

[31]:
```
'9876543210'
```

Řetězce nelze upravovat (*strings are immutable*)

[32]:
```
message = 'Hello World'
```

[33]:
```
message[6]
```

[33]:
```
'W'
```

[34]:
```
message[6] = 'X'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call
  ↪last)
/home/adam/School/Praca/Python/2022/Python/cviko_03/03_Retezce.ipynb␣
  ↪Cell 60 in <cell line: 1>()
----> <a href='vscode-notebook-cell:/home/adam/School/Praca/Python/
  ↪2022/Python/cviko_03/03_Retezce.ipynb#Y113sZmlsZQ%3D%3D?line=0'>1</
  ↪a> message[6] = 'X'

TypeError: 'str' object does not support item assignment
```

[35]:
```
message2 = message[:6] + 'X' + message[7:]
print(message)
print(message2)
```

```
Hello World
Hello Xorld
```

Hledání podřetězců (*substrings*)

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

[36]:
```
'123' in 'ABCDefgh1234'
```

[36]:
```
True
```

[37]:
```
'456' in 'ABCDefgh1234'
```

[37]:
```
False
```

[38]:
```
'456' not in 'ABCDefgh1234'
```

[38]:
```
True
```

[39]:
```
'ABCDefgh1234' in '123'
```

[39]:
```
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.)

[40]:
```
message = 'Nesnese se se sestrou.'
message.count('se')
```

[40]: 4

[41]:
```
'se'.count(message)
```

[41]: 0

[42]:
```
message.find('se')
```

[42]: 5

[43]:
```
message.find('SE')
```

[43]: -1

Hledání pouze na začátku / na konci

- Metody `startswith`, `endswith`

[44]:
```
message = 'Nesnese se se sestrou.'
message.startswith('se')
```

[44]: False

[45]:
```
message.startswith('Nes')
```

[45]: True

[46]:
```
message.endswith('.')
```

[46]: True

Nahrazování

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

[47]:
```
'Spam, egg, Spam, Spam, bacon and Spam'.replace('Spam', 'banana')
```

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

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

```
[48]: 'banana, egg, banana, Spam, bacon and Spam'
```

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

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

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

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

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

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

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

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

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

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

- Volitelný parametr těchto metod popisuje výčet znaků, které se mají z okrajů odstranit

```
[54]: message.strip('\n\t')
```

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

- Naproti tomu, metody `removeprefix` / `removesuffix` odstraňují přesnou předponu/příponu řetězce

```
[55]: message.removesuffix('\n\t')
```

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

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

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

Rozdělění 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 ∞
- (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']
```

- Bez nastaveného `sep` se ignorují prázdné řetězce mezi separátory
- S nastaveným `sep` se vracejí i prázdné řetězce

[62]:
```
' Hello   World '.split()
```

[62]:
```
['Hello', 'World']
```

[63]:
```
' Hello   World '.split(' ')
```

[63]:
```
['', 'Hello', '', '', 'World', '']
```

- Rozbalení seznamu:

[64]:
```
name, surname = 'Jan Novák'.split()
name
```

[64]:
```
'Jan'
```

[65]:
```
surname
```

[65]: 'Novák'

Změna velikosti písma

[66]:
```
message = 'Hello world!'
```

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

[67]: 'HELLO WORLD!'

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

[68]: 'hello world!'

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

[69]: 'hELLO WORLD!'

[70]:
```
message.capitalize()
```

[70]: 'Hello world!'

[71]:
```
message.title()
```

[71]: '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?

[72]:
```
'Python3'.isalnum()
```

[72]: True

[73]:
```
'Python 3'.isalnum()
```

[73]: False

[74]:
```
' \t\n\r'.isspace()
```

```
[74]: True
```

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

```
[75]: False
```

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

```
[76]: False
```

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

```
[77]: True
```

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

```
[78]: 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**

[79]:
```
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)`

[80]:
```
name = 'Anička'
age = 5
what = 'Prasátko Peppa'

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

[80]: `'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}` / `{x:E}` vědecký formát čísla…
- Zadané pořadí je nutné dodržet

[81]:
```
f'{age}'  # Defaultní formát
```

[81]: `'5'`

[82]:
```
f'{age:.3f}'  # Reálné číslo se 3 des. místy
```

[82]:
```
'5.000'
```

[83]:
```
f'{age:>20.2E}'  # Vědecký formát se 2 des. čísly, roztáhni na 20 znaků a zarovnej doprava
```

[83]:
```
'            5.00E+00'
```

[84]:
```
f'{age:.1%}'
```

[84]:
```
'500.0%'
```

[85]:
```
f'{age:04}'
```

[85]:
```
'0005'
```

[86]:
```
f'Jmenuji se {name}, je mi {age:.2f} let, líbí se mi {what:^25}.'
```

[86]:
```
'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

[87]:
```
template = 'Jmenuji se {name}, je mi {age:.1f} let, líbí se mi {what:^25}.'  # Toto není f-string, pouze šablona
template
```

[87]:
```
'Jmenuji se {name}, je mi {age:.1f} let, líbí se mi {what:^25}.'
```

[88]:
```
template.format(name='Anička', age=5.123456, what=what)
```

[88]:
```
'Jmenuji se Anička, je mi 5.1 let, líbí se mi      Prasátko Peppa      .'
```

[89]:
```
template.format(age=2*50, name='Sigmund', what='Monty Python')
```

[89]:
```
'Jmenuji se Sigmund, je mi 100.0 let, líbí se mi       Monty Python       .'
```

- Značky nemusíme pojmenovávat:

[90]:
```
template = 'Jmenuji se {}, je mi {:.1f} let, líbí se mi {}.'
template.format('Anička', 5, 'Prasátko Peppa')
```

[90]:
```
'Jmenuji se Anička, je mi 5.0 let, líbí se mi Prasátko Peppa.'
```

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

[91]:
```
template = 'Jmenuji se {2}, je mi {1:.1f} let, líbí se mi {0}.'
template.format('Anička', 5, 'Prasátko Peppa')
```

[91]:
```
'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:

[92]:
```
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 Sigmund, je ti 100 let, líbí se ti Monty Python.
```

[93]:
```
number = input()  # input() bez hlášky
print(2 * int(number))
```

```
50
```

Funkce `print`

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

[94]:
```
print('ahoj', 5, True)
```

```
ahoj 5 True
```

Speciální parametry funkce `print`

- Parametr `sep` (default je `' '`)

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

```
1 2 3
```

[96]:
```
print(1, 2, 3, sep=', ')
```

```
1, 2, 3
```

[97]:
```
print(1, 2, 3, sep='\n')
```

```
1
2
3
```

[98]:
```
print(1, 2, 3, sep='')
```

```
123
```

- Parametr `end` (default je `'\n'`)

[99]:
```
print(1, 2, 3)
print(4, 5, 6)
```

```
1 2 3
4 5 6
```

[100]:
```
print(1, 2, 3, end='; ')
print(4, 5, 6)
```

```
1 2 3; 4 5 6
```

[101]:
```
print(1, 2, 3, sep=',', end='')
print(4, 5, 6, sep='|', end='.')
```

```
1,2,34|5|6.
```

Otázky:

Který z těchto príkazů NEVYPÍŠE na výstup `True`?

```
(A)  print('False' in 'False')
(B)  print('Torture'[0::2])
(C)  print('Tr', 'U'.lower(), 'e', sep='')
(D)  print('true'.capitalize)
```

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
0	00	*Null*	**32**	20	*Space*	**64**	40	@	**96**	60	`
1	01	*Start of heading*	**33**	21	!	**65**	41	A	**97**	61	a
2	02	*Start of text*	**34**	22	"	**66**	42	B	**98**	62	b
3	03	*End of text*	**35**	23	#	**67**	43	C	**99**	63	c
4	04	*End of transmit*	**36**	24	$	**68**	44	D	**100**	64	d
5	05	*Enquiry*	**37**	25	%	**69**	45	E	**101**	65	e
6	06	*Acknowledge*	**38**	26	&	**70**	46	F	**102**	66	f
7	07	*Bell* \a	**39**	27	'	**71**	47	G	**103**	67	g
8	08	*Backspace* \b	**40**	28	(	**72**	48	H	**104**	68	h
9	09	*Tab* \t	**41**	29	)	**73**	49	I	**105**	69	i

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
10	0a	*Line feed* \n	**42**	2a	*	**74**	4a	J	**106**	6a	j
11	0b	*Vertical tab* \v	**43**	2b	+	**75**	4b	K	**107**	6b	k
12	0c	*Form feed* \f	**44**	2c	,	**76**	4c	L	**108**	6c	l
13	0d	*Carriage return* \r	**45**	2d	-	**77**	4d	M	**109**	6d	m
14	0e	*Shift out*	**46**	2e	.	**78**	4e	N	**110**	6e	n
15	0f	*Shift in*	**47**	2f	/	**79**	4f	O	**111**	6f	o
16	10	*Data link escape*	**48**	30	0	**80**	50	P	**112**	70	p
17	11	*Device control 1*	**49**	31	1	**81**	51	Q	**113**	71	q
18	12	*Device control 2*	**50**	32	2	**82**	52	R	**114**	72	r
19	13	*Device control 3*	**51**	33	3	**83**	53	S	**115**	73	s
20	14	*Device control 4*	**52**	34	4	**84**	54	T	**116**	74	t
21	15	*Neg. acknowledge*	**53**	35	5	**85**	55	U	**117**	75	u
22	16	*Synchronous idle*	**54**	36	6	**86**	56	V	**118**	76	v
23	17	*End trans. block*	**55**	37	7	**87**	57	W	**119**	77	w
24	18	*Cancel*	**56**	38	8	**88**	58	X	**120**	78	x
25	19	*End of medium*	**57**	39	9	**89**	59	Y	**121**	79	y
26	1a	*Substitution*	**58**	3a	:	**90**	5a	Z	**122**	7a	z
27	1b	*Escape*	**59**	3b	;	**91**	5b	[	**123**	7b	{
28	1c	*File separator*	**60**	3c	<	**92**	5c	\	**124**	7c	|
29	1d	*Group separator*	**61**	3d	=	**93**	5d	]	**125**	7d	}
30	1e	*Record separator*	**62**	3e	>	**94**	5e	^	**126**	7e	~
31	1f	*Unit separator*	**63**	3f	?	**95**	5f	_	**127**	7f	*Delete*

[102]:
```
ord('A')
```

[102]:
```
65
```

[103]:
```
ord('č')
```

[103]:
```
269
```

[104]:
```
chr(65)
```

[104]:
```
'A'
```

[105]:
```
chr(269)
```

[105]:
```
'č'
```

[106]:
```
chr(127159)
```

[106]:
```
'🂷'
```

- 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

[107]:
```
print('\x7A \u007A \U0000007A \U0001F0B9')
```

```
z z z 🂹
```

[108]:
```
print('\N{pound sign} \N{playing card seven of spades}')
```

```
£ 🂷
```