Formátování řetězců I

Novější způsob formátování výstupu je pomocí metody .format() v sestavě "formátovaný_řetězec_s_výměnnými_poli" **.** format().

Výměnná pole jsou instrukce, uzavřené ve složených závorkách { }. Vše, co je mimo těchto závorek, je považováno za text, který je v nezměněném stavu kopírován do výstupu.

Skladbu výměnného pole tedy vyjádříme schematicky takto:

```
{ [field_name] [!conversion] [:format_spec] }
```

hranatými závorkami jsou označeny údaje, které mohou chybět.

Nejjednodušší možná forma výměnného pole jsou prázdné složené závorky { }.

Sektor **field_name**
je buď číslo, nebo klíčové slovo. Číslo odkazuje na poziční argument, klíčové slovo na pojmenovaný argument metody .format(). Tvoří-li čísla řadu 0, 1, 2, ..., lze je vynechat.

```
print("Od {} do {}" .format("rána", 22.00))
                    'Od rána do 22.0'  # totéž jako "Od {0} do {1}"
print("Jmenuji se {name}" .format(name = "Pavel"))
                    'Jmenuji se Pavel'
print('{0} i {1}'.format('ano', 'ne'))
                    'ano i ne'
print('{1} i {0}'.format('ano', 'ne'))
                    'ne i ano'
```

Sektor **!conversion**
způsobí změnu typu před formátováním. Používá se značení
!s, které volá funkci str(), jež vrací objekt coby řetězec
!r, které volá funkci repr(), jež vrací řetězec obsahující tisknutelnou prezentaci objektu
!a, které volá funkci ascii(), jež vrací řetězec, jehož non-ASCII znaky jsou nahrazeny escape sekvencí.

```
print("Harold je chytrý {0!s}" .format("chlapeček"))
                            'Harold je chytrý chlapeček'
print("Harold je chytrý {0!r}" .format("chlapeček"))
                            "Harold je chytrý 'chlapeček'"
print("Harold je chytrý {0!a}" .format("chlapeček"))
                            "Harold je chytrý 'chlape\\u010dek'"
                                    # ošetřené non-ASCII
```

Sektor **:format_spec**
upřesňuje, jak má být hodnota prezentována, to jest určuje:
šířku - pole pro zadávanou hodnotu

výplň - libovolný znak kromě závorek { }; následuje pokyn pro zarovnání
zarovnání - vlevo (<), vpravo (>), na střed (^) a mezi(=) signum a číslici
signum - +, -, " ", (také # a 0)
přesnost - počet desetinných míst čísla: . údaj
typ - určuje způsob prezentace dat, například b je pro binární formát, d je pro decimální celé číslo, f pro formát float

```
print("{:<25}" .format("zarovnáno vlevo"))
print("{:>25}" .format("zarovnáno vpravo"))
print("{:˘^25}" .format("hulín"))
```

Další zajímavé příklady namátkou:

```
import math
print("Hodnota Pí je asi {0:.3f}" .format(math.pi))


telef = {"Jan":4127, "Dana":4098, "Ota":863678}
for name, phone in telef.items():
    print("{0:10} ==> {1:10d}" .format(name, phone))

for x in range(7,11):
    print("{0:2} {1:3} {2:4}" .format(x, x*x, x*x*x))
```

Formátování řetězců II

Nejnovější způsob formátování řetězců byl zaveden ve verzi Python 3.6. Označuje se jako **formátovaný literál** řetězce (formatted string literal), stručně f-string. Literál f-stringu se uvozuje písmenem f nebo F a lze použít obdobné konverze (!s, !r, !a) jako u předchozího způsobu.
Obdobně jako u předchozího způsobu může tento literál obsahovat výměnná pole, ohraničená složenými závorkami { }. Zatímco u předchozího způsobu odkazoval obsah těchto závorek pouze na konstantní hodnotu, u f-stringu může odkazovat také na výraz, funkci, metodu aj.
V následující ukázce vidíme volání funkce a metody:

```
name = "IDLE"
def to_lowercase(input):
    return input.lower()

print(f"{to_lowercase(name)} je zábavné.")          volání funkce
                            'idle je zábavné.'

print(f"{name.lower()} je zábavné.")                volání metody
                            'idle je zábavné.'
```

V další ukázce je odkaz na slovník:

```
postava = {'name': 'Petr Parléř', 'age': 67}
print(f"Stavitel {postava['name']} se dožil {postava['age']} let."
                     'Stavitel Petr Parléř se dožil 67 let.'
```

Další příklady:

```
name = 'Fred'
age = 42
print(f'Řekl, že se jmenuje {name} a je mu {age} let.')

name = 'Fred'
seven = 7
print(f'Řekl, že se jmenuje {name.upper()}a je mu {6 * seven} let.')

import math
print(f'Velikost Pí je přibližně {math.pi:.3f}.')

table = {'Karel': 4127, 'Jan': 4098, 'David': 7678}
for name, phone in table.items():
    print(f'{name:10} ==> {phone:10d}')
```