Formatowanie napisów

%-formatting, str.format(), f-strings

Adam Chyła
chyla.org

Formatowanie napisów

Formatowanie napisu to operacja przekształcenia napisu wzorcowego, opisującego umiejscowienie i sposób prezentacji danych, w nowy napis złożony z podstawionych w odpowiednie miejsca konkretnych danych o określonym wyglądzie.

W Pythonie najpopularniejszymi metodami formatowania napisów są:

- %-formatting (printf-style String Formatting, stary styl),
- str.format() (nowy styl),
- f-strings (formatted strings, od Pythona 3.6).

%-formatting

Do formatownia napisów można wykorzystać operator % (modulo). Po jego lewej stronie znajduje się wzorzec, a po prawej dane. W wyniku operacji modulo otrzymamy nowy napis.

```
liczba_kotow = 5
nowy_napis = 'Ala ma %d kotow' % liczba_kotow

print(nowy_napis)
```

%-formatting

Każdy fragment pod który zostaną podstawione dane (specyfikator konwersji) musi składać się z co najmniej dwóch znaków,
z których pierwszy to %,
a drugi to identyfikator typu informujący o typie danych do podstawienia (typ konwersji).

```
liczba_kotow = 5
liczba_psow = 6
nowy_napis = 'Ala ma %d kotow i %d psow' %      \
                (liczba_kotow, liczba_psow)

print(nowy_napis)
```

%-formatting

Pełny opis składowych specyfikatora konwersji:

1. *Znak "%", który określa początek definicji.*
2. *Nazwa klucza odwzorowania (opcjonalnie), zawarta w nawiasach okrągłych (n.p.. (nazwa)).*
3. *Flaga konwersji (opcjonalnie), które mają wpływ na niektóre typy danych biorące udział w konwersji.*
4. *Minimalna szerokość pola (opcjonalnie). Jeśli zdefiniowana jako "*" (gwiazdka) wartość pola jest odczytywana z następnego pola w tabeli wartości a obiekt do konwersji następuje po minimalnej szerokości i ew. precyzji.*
5. *Precyzja (opcjonalnie), podana jako "." (kropka) po której następuje wartość precyzji. Jeśli zdefiniowana jako "*" (gwiazdka) wartość pola jest odczytywana z następnego pola w tabeli wartości a obiekt do konwersji następuje po precyzji.*
6. *Modyfikator długości (opcjonalnie).*
7. *Typ do konwersji.*

źródło: https://pl.python.org/docs/lib/typesseq-strings.html

%-formatting

Wybrane flagi konwersji:

`0`	wypełnienie zerami
-	wyrównanie do lewej
+	poprzedzający znak + lub -

źródło:
https://docs.python.org/3/library/stdtypes.html#printf-style-string-formatting

%-formatting

Wybrane identyfikatory typu do konwersji (ostatni znak w specyfikatorze konwersji):

`d`	liczba całkowita
`f`	liczba zmiennoprzecinkowa
`c`	pojedynczy znak
`s`	napis (zamienia dowolny obiekt Pythona na napis)

źródło:
https://docs.python.org/3/library/stdtypes.html#printf-style-string-formatting

%-formatting

Bazując na opisie składowych specyfikatora konwersji (punkty 1, 5 i 7), prezentowany przykład wypisuje na ekranie liczbę zmiennoprzecinkową z dokładnością do trzech miejsc po przecinku.

```
pi = 3.141592653589793

print("%.3f" % pi)
```

%-formatting

Więcej informacji na temat tego sposobu formatowania napisów znajduje się w dokumentacji:

https://docs.python.org/3/library/stdtypes.html#printf-style-string-formatting

```
print('%(lang)s has %(number)03d quote types.' %
      {'lang': "Python", "number": 2})
```

str.format()

Użycie nowego stylu formatowania napisów wymaga wywołania funkcji `format()` na wzorcu. Wzorzec powinien składać się z pól do zastąpienia (replacement fields) otoczonych klamrami.

```
liczba_kotow = 5
nowy_napis = 'Ala ma {} kotow'.format(liczba_kotow)

print(nowy_napis)
```

str.format()

Pola do zastąpienia mogą odnosić się do nazwanych parametrów funkcji `format()`, jednak jest to opcjonalne.

```
liczba_kotow = 5
nowy_napis = '{kto} ma {ile} kotow'.format(
                                    ile=liczba_kotow,
                                    kto="Ala")

print(nowy_napis)
```

str.format()

Podobnie jak w przypadku formatowania napisów z użyciem operatora modulo (%), można podać specyfikator konwersji (*standard format specifier*), należy to jednak zrobić po dwukropku.

```
liczba_kotow = 5
nowy_napis = 'Ala ma {:02d} kotow'.format(
                                        liczba_kotow)

print(nowy_napis)
```

str.format()

Pełny opis składowych specyfikatora konwersji (*standard format specifier*):

```
format_spec     ::=  [[fill]align][sign][#][0][width][grouping_option][.precision][type]
fill            ::=  <any character>
align           ::=  "<" | ">" | "=" | "^"
sign            ::=  "+" | "-" | " "
width           ::=  digit+
grouping_option ::=  "_" | ","
precision       ::=  digit+
type            ::=  "b" | "c" | "d" | "e" | "E" | "f" | "F" | "g" | "G" | "n" | "o" | "s"
| "x" | "X" | "%"
```

źródło: https://docs.python.org/3/library/string.html#format-specification-mini-language

str.format()

Znaczenie opcji `align`:

- `<` - wyrównanie do lewej,
- `>` - wyrównanie do prawej,
- `=` - ważne tylko dla liczb, wymusza wypełnienie po znaku, aż do początku liczby (wyświetlanie liczb w postaci `+000000120`),
- `^` - wyśrodkowanie.

Znaczenie opcji `sign`:

- + (plus) - liczby dodatnie otrzymują znak plus, ujemne znak minus przed cyframi
- - (minus) - liczby dodatnie nie otrzymują znaku plus, ujemne otrzymują znak minus przed cyframi,
- spacja - liczby dodatnie poprzedza spacja, ujemne otrzymują znak minus przed cyframi

źródło: https://docs.python.org/3/library/string.html#format-specification-mini-language

str.format()

Znaczenie opcji `#`: alternatywny sposób prezentowania liczb (np. liczby szesnastkowe otrzymają przedrostek `0x`).

Znaczenie opcji `0`: wypełnienie pola zerami.

Znaczenie opcji `width`: szerokość pola.

Znaczenie opcji `grouping_option`:

- `,` (przecinek) - użycie przecinka jako separatora dla tysięcy.
- `_` (podkreślnik/podłoga) - użycie podłogi jako separatora dla tysięcy.

Znaczenie opcji `.precission`:

- dla liczb zmiennoprzecinkowych określa ile liczb po przecinku powinno zostać umieszczonych.
- dla wartości nieliczbowych określa maksymalną długość pola.

źródło: https://docs.python.org/3/library/string.html#format-specification-mini-language

str.format()

Wybrane identyfikatory typu prezentacji (pole `type`):

d	liczba całkowita
f	liczba zmiennoprzecinkowa
c	pojedynczy znak
s	napis (zamienia dowolny obiekt Pythona na napis)

źródło: https://docs.python.org/3/library/string.html#format-specification-mini-language

f-strings

W Pythonie 3.6 wprowadzono nowy mechanizm nazwany f-strings. Jest to bardzo podobny sposób zapisu do `str.format()`, lecz dużo bardziej zwięzły.

F-string wygląda następująco:

```
f '<text> { <expression> <optional !s, !r, or !a> <optional : format specifier> } <text>'
```

Opcja expression to wyrażenie, które ma zostać umieszczone.

Opcja `!s`, `!r`, `!a` wymusza sposób konwersji przekazanego parametru (`str`, `repr`, `ascii`).

Opcja `format specifier` definiuje sposób prezentacji danych, jej składnia jest dokładnie taka sama, jak składnia znanego już specyfikatora konwersji z `str.format()`.

źródło: https://docs.python.org/3/library/string.html#format-specification-mini-language

f-strings

Wprowadzenie f-stringów zmniejszyło ilość potrzebnego kodu do sformatowania napisu.

Przed wzorcem pojawiła się litera `f`, to informacja, że dany napis jest f-napisem i ma zostać sformatowany.

```
liczba_kotow = 5
nowy_napis = f'Ala ma {liczba_kotow} kotow'

print(nowy_napis)
```

f-strings

Część związana
ze specyfikatorem konwersji
jest dokładnie taka sama,
jak w przypadku str.format().

```
lang = 'Python'
number = 3

nowy = f'{lang}s has {number:03d} quote types.'

print(nowy)
```

f-strings

Ważną zmianą w porównaniu do `str.format()` jest możliwość umieszczenia wyrażenia w formatowanym napisie.

Oznacza to możliwość wywołania dowolnej funkcji wewnątrz wzorca.

```
text = 'Ala ma kota'

nowy = f'Napis {text} ma {len(text)} znakow.'

print(nowy)
```

f-strings

Warto również zauważyć, że f-strings pozwala na wielokrotne użycie tej samej zmiennej we wzorcu.

Tej funkcjonalności nie oferują pozostałe metody formatowania napisów.

```
text = 'Ala ma kota'

nowy = f'Napis {text} ma {len(text)} znakow.'

print(nowy)
```

Pytania

1. Na czym polega formatowanie napisu?
2. Jakie znasz sposoby formatowania napisów?

Literatura

1. printf-style String Formatting, https://docs.python.org/3/library/stdtypes.html#printf-style-string-formatting
2. Operacje formatujące napisy, https://pl.python.org/docs/lib/typesseq-strings.html
3. Format String Syntax, https://docs.python.org/3/library/string.html#format-string-syntax
4. PEP 498 -- Literal String Interpolation, https://www.python.org/dev/peps/pep-0498/
5. Python 3's f-Strings: An Improved String Formatting Syntax (Guide), https://realpython.com/python-f-strings/

Tekst podlega ochronie na podstawie ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (tekst jednolity Dz.U. z 2000 r. Nr 80, poz. 904 z późn. zm.). Jakiekolwiek kopiowanie, odtwarzanie, przegrywanie, zwielokrotnianie zawartości jest zabronione. Kopiowanie treści możliwe jest tylko po udzieleniu zgody przez autora.