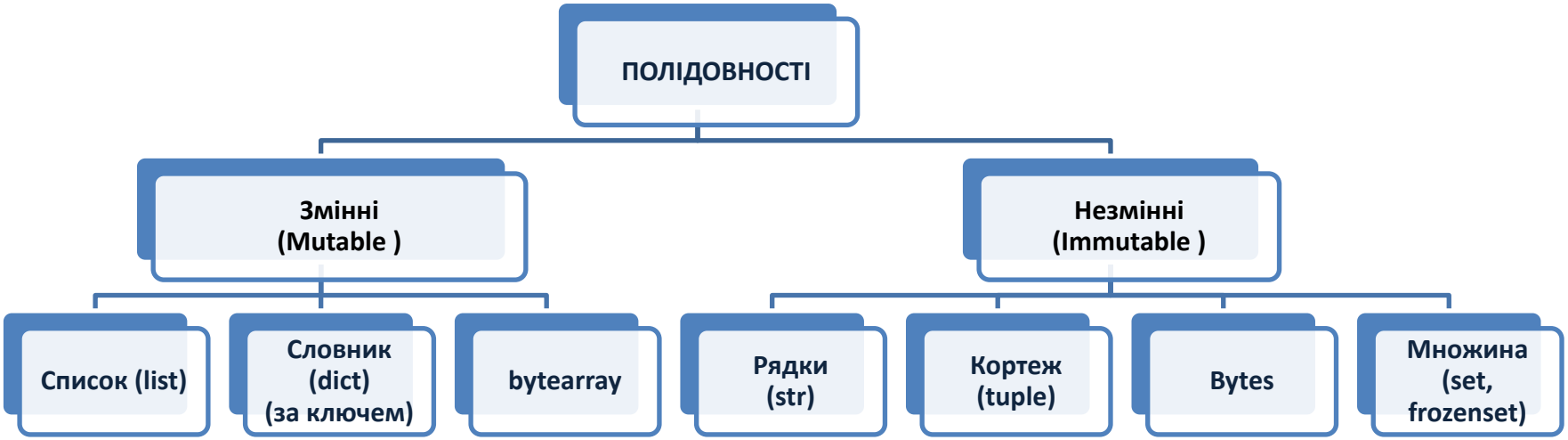
Рядки. Функції та методи для роботи із рядками в Python.
Форматоване виведення даних.

Змінні і незмінні послідовності (колекції) в Python



1. Рядки (str). Функції та методи для роботи із рядками в Python (1/5)

Рядок (class <str>) представляє незмінний (immutable) тип даних , який зберігає послідовність будь-яких символів Unicode (текстову інформацію).

Літерали рядків:

Одинарні лапки: 'allows embedded "double" quotes'

Подвійні лапки: "allows embedded 'single' quotes"

Потрійні лапки: '''Three single quotes'''
"""Three double quotes"""

Рядкові літерали, які є частиною одного виразу і мають лише пробіли між собою, будуть неявно перетворені в один рядковий літерал (аналог явної конкатенації оператор +).
Тобто

"spam " "eggs" == "spam eggs"

еквівалентно:

"spam "+"eggs" == "spam eggs"

1. Рядки (str). Функції та методи для роботи із рядками в Python (2/5)

Службові символи (екрановані послідовності)

```
\\          # \
\'          # '
\"          # "
\0          # символ Null (не є ознакою кінця рядка)
\a          # код дзвінка
\b          # повернення (Backspace)
\f          # ознака кінця сторінки
\n          # ознака кінця рядка
\r          # повернення каретки
\t          # горизонтальна табуляція
\v          # вертикальна табуляція
\N{id}      # ідентифікатор ID бази даних Unicode
\ooo        # символ з 8-им кодом ooo
\uhhhh      # символ з 16-бітовим кодом hhhh Unicode в 16-му поданні
\Uhhhh      # символ з 32-бітовим кодом hhhh Unicode в 16-му поданні
\xhh        # символ з 16-им кодом hh
```

1. Рядки (str). Функції та методи для роботи із рядками в Python (3/5)

S = "s\np\ta\nbbb" — екрановані послідовності;
S = r"C:\temp\new" — неформатований рядок (пригнічене екранування);
S = b"byte" — рядок байтів;

Оператори та функції	Результат виконання
S1 + S2	додавання рядків S1 і S2;
S * n	дублювання рядка S n-у кількість разів
S[i]	звернення за індексом;
S[i:j:step]	витяг зрізу від i-го до j-1 символу з кроком step ;
len(S)	повертає довжину рядка;
ord(символ)	повертає ASCII код символу;
chr(число)	повертає символ із вказаним кодом ASCII;

Рядок	P	y	t	h	o	n	
Індексація символів в рядку	0	1	2	3	4	5	6
	-6	-5	-4	-3	-2	-1	

```
s = ‘Python'
s[3:5]    # 'ho'
s[2:-2]   # 'th'
s[:6]     # Python'
s[1:]     # 'ython'
s[:]      # ‘Python‘ – копіювання рядка
s[::-1]   # 'nohtyP''
s[3:5:-1] # ''
s[2::2]   # ‘to'
```

1. Рядки (str). Функції та методи для роботи із рядками в Python (4/5)

Методи	Результат використання
S.find(str, [start],[end])	пошук підрядка в рядку (в діапазоні з start до end), повертає номер першого входження або -1;
S.rfind(str, [start],[end])	пошук підрядка str в рядку (в діапазоні з start до end), повертає номер останнього входження або -1;
S.index(str, [start],[end])	пошук підрядка str в рядку ((в діапазоні з start до end)), повертає номер першого входження або викликає ValueError;
S.rindex(str, [start],[end])	пошук підрядка str в рядку (в діапазоні з start до end), повертає номер останнього входження або викликає ValueError;
S.replace(old, new [, num]))	заміна в рядку підрядка old на новий підрядок new разів num;
S.split([delimeter [, num]]))	розбиття рядка на підрядки в залежності від delimeter за роздільником;
S.isdigit()	повертає True, якщо всі символи рядка – цифри;
S.isnumeric()	повертає True, якщо рядок являє собою число
S.isalpha()	повертає True, якщо рядок складається тільки з алфавітних символів;
S.isalnum()	повертає True, якщо рядок складається з цифр або літер;
S.islower()	повертає True, якщо рядок складається із символів у нижньому регістрі;
S.isupper()	повертає True, якщо рядок складається із символів у верхньому регістрі;
S.isspace()	повертає True, якщо рядок складається з невідображуваних символів (пробіл, ознаки кінця сторінки '\f' і рядка '\n', переведення каретки '\r', горизонтальна табуляція '\t' і вертикальна табуляція '\v');
S.istitle()	повертає True, якщо слова в рядку починаються слова в рядку з великої літери;
S.upper()	перетворення символів рядка до верхнього регістру;
S.lower()	перетворення символів рядка до нижнього регістру;
S.startswith(str)	повертає True, якщо рядок S починається з підрядка str;
S.endswith(str)	повертає True, якщо рядок S закінчується підрядком str;
S.join(список)	об'єднує рядки в один рядок, вставляючи між ними роздільник S;
S.capitalize()	переводить перший символ рядка у верхній регістр, а всі інші — в нижній;
S.center(width, [fill])	якщо довжина рядка менша параметра width, то ліворуч і праворуч від рядка рівномірно додається символ fill (пробіл за замовчуванням), щоб доповнити значення width, а сам рядок вирівнюється по центру

1. Рядки (str). Функції та методи для роботи із рядками в Python (5/5)

Методи	Результат використання
S.count(str, [start],[end])	повертає кількість непересічних входжень підрядка в діапазоні [start, end] (0 і довжина рядка як усталено);
S.expandtabs([tabsize])	повертає копію рядка, в якому всі символи табуляції замінено одним або кількома пропусками залежно від поточного стовпчика. Якщо TabSize не вказано, розмір табуляції — 8 пробілів;
S.lstrip([chars])	видалення пробілів (або chars) на початку рядка;
S.rstrip([chars])	видалення пробілів (або chars) в кінці рядка;
S.strip([chars])	видалення пробілів (або chars)на початку і в кінці рядка;
S.partition(str)	повертає кортеж, що містить частину перед першим str, сам str, і частину після str. Якщо str не знайдено, повертається кортеж, що містить сам рядок, а далі два порожніх рядки;
S.rpartition(str)	повертає кортеж, що містить частину перед останнім str, сам str, і частина після str, якщо str не знайдений, повертається кортеж, що містить два порожні рядки, а потім сам рядок;
S.swapcase()	повертає рядок, у якому символи нижнього регістру замінюються на верхній, а верхнього — на нижній;
S.title()	початкові символи всіх слів в рядку переводяться у верхній регістр
S.zfill(width)	робить довжину рядка не меншою width, в разі потреби заповнює перші символи нулями;
S.ljust(width, fillchar)	робить довжину рядка не меншою width, в разі потреби заповнює останні символи символом fillchar (за замовчуванням пропуск);
S.rjust(width, fillchar)	робить довжину рядки не меншою width, в разі потреби заповнює перші символи символом fillchar (за замовчуванням пропуск);

2. Форматоване виведення даних (1/3)

рядок.format(*args, **kwargs) - форматування рядка.

Синтаксис команди:

```
replacement_field ::=  "{" [field_name] ["!" conversion] [":" format_spec] "}"
field_name        ::=  arg_name ("." attribute_name | "[" element_index "]")*
arg_name          ::=  [identifier | digit+]
attribute_name    ::=  identifier
element_index     ::=  digit+ | index_string
index_string      ::=  <any source character except "]"> +
conversion        ::=  "r" | "s" | "a"
format_spec       ::=  <described in the next slide>
```

2. Форматоване виведення даних (2/3)

```
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" | "%"
```

2. Форматоване виведення даних (3/3)

```
>>> '{0}, {1}, {2}'.format('a', 'b', 'c')
'a, b, c'
>>> '{}, {}, {}'.format('a', 'b', 'c')
'a, b, c'
>>> '{2}, {1}, {0}'.format('a', 'b', 'c')
'c, b, a'
>>> '{2}, {1}, {0}'.format(*'abc')
'c, b, a'
>>> '{0}{1}{0}'.format('abra', 'cad')
'abracadabra'
>>> 'Coordinates: {latitude}, {longitude}'.format(latitude='37.24N', longitude='-115.81W')
'Coordinates: 37.24N, -115.81W'
>>> coord = {'latitude': '37.24N', 'longitude': '-115.81W'}
>>> 'Coordinates: {latitude}, {longitude}'.format(**coord)
'Coordinates: 37.24N, -115.81W'
```

```
>>> '{0}, {1}, {2}'.format('a', 'b', 'c')
'a, b, c'
>>> '{}, {}, {}'.format('a', 'b', 'c')  # 3.1+ only
'a, b, c'
>>> '{2}, {1}, {0}'.format('a', 'b', 'c')
'c, b, a'
>>> '{2}, {1}, {0}'.format(*'abc')      # unpacking argument sequence
'c, b, a'
>>> '{0}{1}{0}'.format('abra', 'cad')   # arguments' indices can be repeated
'abracadabra'
```

```
>>> '{:<30}'.format('left aligned')
'left aligned                  '
>>> '{:>30}'.format('right aligned')
'                 right aligned'
>>> '{:^30}'.format('centered')
'           centered           '
>>> '{:*^30}'.format('centered')  # use '*' as a fill char
'***********centered***********'
```

```
>>> '{:+f}; {:+f}'.format(3.14, -3.14)  # show it always
'+3.140000; -3.140000'
>>> '{: f}; {: f}'.format(3.14, -3.14)  # show a space for positive numbers
' 3.140000; -3.140000'
>>> '{:-f}; {:-f}'.format(3.14, -3.14)  # show only the minus -- same as '{:f}; {:f}'
'3.140000; -3.140000'

>>> # format also supports binary numbers
>>> "int: {0:d};  hex: {0:x};  oct: {0:o};  bin: {0:b}".format(42)
'int: 42;  hex: 2a;  oct: 52;  bin: 101010'
>>> # with 0x, 0o, or 0b as prefix:
>>> "int: {0:d};  hex: {0:#x};  oct: {0:#o};  bin: {0:#b}".format(42)
'int: 42;  hex: 0x2a;  oct: 0o52;  bin: 0b101010‘
```