Лекція 9
Функції й методи для роботи з рядками

Функції для роботи з рядками

str ([<Об'єкт>[,<Кодування>[,<Обробка помилок>]]])

– перетворює будь-який об'єкт у рядок. Якщо параметр не зазначений, то повертається порожній рядок. Використовується функцією `print()` для виводу об'єктів.

Приклад 1.

```
>>> str(), str([1,2]), str((3,4)), str({"x": 1})
('', '[1, 2]', '(3, 4)', "{'x': 1}")
>>> print("рядок1\nрядок2")
Рядок1
рядок2
```

"strict"–при невідомому символі Unicodedecodeerror

"replace"–при невідомому символі \ufffd)

"ignore" – невідомі символи ігноруються:

`repr(<Об'єкт>)` – повертає строкове представлення об'єкта.

Приклад 2

```
>>> repr("Рядок"), repr([1,2,3]),repr({"x": 5})
("'Рядок'", '[1, 2, 3]', "{'x': 5}")
```

```
Інтерпретатор:
>>> repr("рядок1\nрядок2")
"'рядок1\\nрядок2'"
```

```
Компілятор:
print(repr("рядок1\nрядок2"))
'рядок1\nрядок2'
```

`ascii(<Об'єкт>)` – повертає строкове представлення об'єкта. У рядку можуть бути символи тільки з кодування ASCII.

Приклад 3

```
>>> ascii([1, 2, 3]), ascii({"x":5})
('[1, 2, 3]', "{'x': 5}")
>>> ascii("рядок")
"'\\u0440\\u044f\\u0434\\u043e\\u043a'"
```

```
print(str("рядок1\nрядок2"))
print(repr("рядок1\nрядок2"))
print(ascii("рядок1\nрядок2"))
```

```
рядок1
рядок2
'рядок1\nрядок2'
'\u0440\u044f\u0434\u043e\u043a1\n\u0440\u044f\u0434\u043e\u043a2'
```

`len(<Рядок>)` – повертає кількість символів у рядку:

Приклад 4

```
>>>len("Python"), len("\r\n\t"),len(r"\r\n\t")
(6, 3, 6)

>>> len("рядок")
5

>>> a=bytes("рядок", "utf-8")
>>> len(a)
10

>>> a=bytearray("рядок", "utf-8")
>>> len(a)
10
```

Методи рядкового типу

`strip([<Символи>])`- видаляє зазначені в параметрі символи на початку й наприкінці рядка. Якщо параметр не заданий, видаляються «пропускові» (пробільні) символи: пробіл, символ переводу рядка (`\n`), символ повернення каретки (`\r`), символи горизонтальної (`\t`) і вертикальної (`\v`) табуляції:

Синтаксис методу: s.strip()

Приклад 5

```
>>> s1, s2 = "   str\n\r\v\t", "strstrstrokstrstrstr"
>>> "'%s' - '%s'" % (s1.strip(), s2.strip("tsr"))
"'str' - 'ok'"
>>> s1, s2 = "   str\n\r\v\t", "strstrstrokstrsmrstr"
>>> "'%s' - '%s'" % (s1.strip(), s2.strip("tsr"))
"'str' - 'okstrsm'"
```

`lstrip([<Символи>])` – видаляє пробільні або зазначені символи на початку рядка:

Приклад 6.

```
>>> s1, s2 = " \tstr ", "strstrstrokstrstrstr"
>>> "'%s' - '%s'" % (s1.lstrip(), s2.lstrip("tsr"))
"'str ' - 'okstrstrstr'"
```

`rstrip([<Символи>])` – видаляє пробільні або зазначені символи наприкінці рядка:

Приклад 7

```
>>> s1, s2 = " str\t ", "strstrstrokstrstrstr"
>>> "'%s' - '%s'" % (s1.rstrip(), s2.rstrip("tsr"))
"' str' - 'strstrstrok'"
```

`split ([<Роздільник>[,<Ліміт>]])` – розділяє рядок на підрядки по зазначеному роздільнику й додає ці підрядки в **список**, який повертається як результат.

1. Якщо перший параметр не зазначений або має значення None, то як роздільник використовується символ пробілу.

2. У другому параметрі можна задати кількість підрядків у результуючому списку.

3. Якщо другий параметр не зазначений або дорівнює -1, у список потраплять усі підрядки.

4. Якщо підрядків більше зазначеного кількості, то список буде містити ще один елемент - із залишком рядка.

Приклад 8. Дія методу split

```
>>> s = "word1 word2 word3"


>>> s.split(),
['word1','word2','word3']


>>> s.split(None, 1)
['word1','word2 word3']



>>> s = "word1\nword2\nword3"
>>> s.split("\n")
['word1', 'word2', 'word3']
```

Якщо в рядку містяться кілька пробілів підряд і роздільник не зазначений, то порожні елементи не будуть додані в список:

Приклад 9

```
>>> s = "word1       word2             word3  "
>>> s.split()
 ['word1', 'word2', 'word3']
```

При використанні іншого роздільника можуть виникнути порожні елементи:

```
>>> s = ",,word1,,word2,,word3,,"
>>> s.split(",")
['','','word1','','word2','','word3','','']
>>> "1,,2,,3".split(",")
['1', '', '2', '', '3']
```

Якщо роздільник не знайдений у рядку, то список буде складатися з одного елемента, що представляє початковий рядок:

```
>>> "word1 word2 word3".split("\n")
['word1 word2 word3']
```

`rsplit([<Роздільник>[,<Ліміт>]])` - аналогічний методу `split()`, але пошук символу-роздільника проводиться не зліва направо, а справа наліво.

Приклад 10

```
>>> s = "word1 word2 word3"
>>> s.rsplit()
['word1', 'word2', 'word3']

>>> s = "word1 word2 word3"
s.rsplit(None,1)
['word1 word2', 'word3']

>>> "word1\nword2\nword3".rsplit("\n")
['word1', 'word2', 'word3']
```

`splitlines([True])` – розділяє рядок на підрядки по символу переводу рядка (`\n`) і додає їх у список.

1. Символи нового рядка включаються в результат, тільки якщо необов'язковий параметр має значення `True`.
2. Якщо роздільник не знайдений у рядку, то список буде містити тільки один елемент.

Приклад 11

```
>>> "word1\nword2\nword3".splitlines()
[ 'word1', 'word2', 'word3']
# - True – роздільник включено у список
>>> "word1\nword2\nword3".splitlines(True)
['word1\n', 'word2\n', 'word3']
# - False - еквівалентно пустому параметру
>>> "word1\nword2\nword3".splitlines(False)
[ 'word1', 'word2', 'word3']
>>>"word1 word2 word3".splitlines()#Роздільника немає
['word1 word2 word3']
```

`partition(<Роздільник>)` – знаходить перше входження символу-роздільника в рядок і **повертає кортеж** із трьох елементів:

1. Перший елемент буде містити фрагмент, розташований перед роздільником.
2. Другий елемент - сам роздільник,
3. Третій елемент – фрагмент, розташований після роздільника.
4. Пошук проводиться зліва направо.
5. Якщо символ-роздільник не знайдений, то перший елемент кортежу буде містити весь рядок, а інші елементи залишаться порожніми.

Приклад 12

```
>>> "word1 word2 word3".partition(" ")
('word1', ' ', 'word2 word3')
>>> "word1 word2 word3".partition("\n ")
('word1 word2 word3', '', '')
```

`rpartition (<Роздільник>)` – метод аналогічний методу `partition()`, але пошук символу-роздільника проводиться не зліва направо, а справа наліво.

1. Якщо символ-роздільник не знайдений:
- перші два елементи кортежу виявляться порожніми,
- третій елемент буде містити весь рядок.

Приклад 13

```
>>> "word1 word2 word3".rpartition(" ")
( 'word1 word2 ', ' ', 'word3' )

>>> "word1 word2 word3".rpartition("\n" )
('', '', 'word1 word2 word3')
```

`join()` – перетворює послідовність у рядок. Елементи додаються через зазначений роздільник. Формат методу:

```
<Рядок> = <Роздільник>.join(<Послідовність>)
```

Як приклад перетворимо список і кортеж у рядок:

Приклад 14

```
>>> "=>" .join(["word1", "word2", "word3"] )
'word1=>word2=>word3'

>>> " ".join(("word1", "word2", "word3"))
'word1 word2 word3'
```

Елементи послідовностей повинні бути рядками, інакше виконується виключення `Typeerror:`

Приклад 15

```
>>> " ".join(("word1", "word2", 5))
Traceback (most recent call last):
  File "<input>", line 1, in <module>
Typeerror: sequence item 2: expected str
instance, int found
```

Рядки є незмінюваними типами даних. Якщо спробувати змінити символ по індексу, то виникне помилка.

Технологія зміни символів рядка

1. Щоб змінити символ по індексу, можна перетворити рядок у список за допомогою функції `list()`.

2. Виконати зміни символів

3. За допомогою методу `join()` перетворити список знов у рядок.

Приклад 16

```
>>> s = "Python"

>>> arr = list(s); arr # Перетворимо рядок у
список
['P', 'y', 't', 'h', 'o', 'n']

>>> arr[0] = "J"; arr # Змінюємо елемент по
індексу
['J', 'y', 't', 'h', 'o', 'n']

>>> s = "".join(arr); s # Перетворимо список у
рядок
'Jython'
```

Інший спосіб зміни рядка

1. В Python 3 можна також перетворити рядок у тип `bytearray`,
2. Змінити символ по індексу.
3. Виконати зворотне перетворення

Приклад 17

```
>>> s = "python"
>>> b = bytearray(s, "cp1251"); b
bytearray(b'python')
>>> b[0] = ord("j" ); b
bytearray(b'jython')
>>> s = b.decode("cp1251"); s
'jython'
```

`ord (c)` – рядкове представлення одного символу перетворює в ціле число, яке є юнікодом цього символу.
Наприклад: `ord ('a')` повертає ціле число 97. Це функція, зворотна до `chr ()`.

Функції для роботи із символами

Для роботи з окремими символами призначені наступні функції:

`chr(<Код символу>)` – повертає символ по зазначеному коду:

Приклад 24

```
>>> print(chr(1055))
П
```

`ord (<Символ>)` – повертає код зазначеного символу:

Приклад 25

```
>>> print (ord ( "П"))
1055
```

Настроювання локалі

1. Необхідно підключити модуль за допомогою виразу:

```
import locale
```

2. Для установки локалі (сукупності локальних налаштувань системи) слугує функція `setlocale ()` з модуля `locale`.

3.Функція `setlocale ()` має наступний формат:

```
setlocale(<категорія>[,<Локаль>));
```

Параметр `<категорія>` може приймати наступні значення:

Значення параметра <категорія>

`locale. LC_ALL` - установлює локаль для всіх режимів;
`locale.LC_COLLATE` - для порівняння рядків;
`locale. LC_CTYPE` - для переводу символів у нижній або верхній регістр;
`locale.LC_MONETARY` - для відображення грошових одиниць;
`locale. LC_NUMERIC` - для форматування чисел;
`locale.LC_TIME` - для форматування виводу дати й часу.

Одержати поточне значення локалі дозволяє функція `getlocale([<Категорія>])`.

Як приклад налаштуємо локаль під Windows спочатку на кодування Windows-1251, потім на кодування UTF-8, а потім на кодування за замовчуванням.

Далі виведемо поточне значення локалі для всіх категорій і тільки для `locale.LC_COLLATE`.

Приклад 18. Застосування `setlocale() і getlocale()`

```
>>> import locale

>>> # Для кодування windows-1251
>>> locale.setlocale(locale.LC_ALL, "Ukrainian_Ukraine.1251")
' Ukrainian_Ukraine.1251'
>>># встановлюємо локаль за замовчуванням
>>> locale.setlocale(locale.LC_ALL, "")
'Russian Russia.1251'
>>> #отримуємо поточне значення локалі для всіх категорій
>>> locale.getlocale()
( 'Russian_Russia', '1251')
>>> # Одержуємо поточне значення категорії locale. LC COLLATE
>>> locale.getlocale(locale.LC_COLLATE)
( 'Russian_Russia', '1251')
```

Одержання словника з настроюваннями локалі

Одержати налаштування локалі дозволяє функція `localeconv()`. Функція повертає словник з настроюваннями. Результат виконання функції для локалі Russian Russia.1251 виглядає в такий спосіб:

```
>>> locale. localeconv()
{'p_sign_posn': 1, 'n_cs_precedes': 0,
'negative_sign': '-', 'decimal_point': ',',
'int_curr_symbol': 'RUB', 'thousands_sep':
'\xa0', 'mon_thousands_sep': '\xa0',
'mon_decimal_point': ',', 'n_sign_posn': 1,
'positive_sign': '', 'p_cs_precedes': 0,
'currency_symbol': '?', 'p_sep_by_space': 1,
'mon_grouping': [3, 0], 'grouping': [3, 0],
'n_sep_by_space': 1, 'frac_digits': 2,
'int_frac_digits': 2}
```

Зміна регістру символів

Для зміни регістру символів призначені наступні методи:

`upper()` – заміняє всі символи рядка відповідними прописними буквами:

Приклад 19

```
>>> print("рядок".upper())
РЯДОК
```

`lower()` – заміняє всі символи рядка відповідними малими літерами:

Приклад 20

```
>>> print("РЯДОК".lower())
рядок
```

`swapcase()` – заміняє всі малі символи відповідними великими буквами, а всі великі символи – малими:

Приклад 21

```
>>> print("РЯДОК рядок".swapcase())
рядок РЯДОК
```

`capitalize()` – робить першу букву рядка великою:

Приклад 22

```
>>> print("рядок рядок".capitalize())
Рядок рядок
```

`title ()` – робить першу букву кожного слова великою:

Приклад 22

```
>>> s = "перша буква кожного слова стане прописною"
>>> print(s.title())
Перша Буква Кожного Слова Стане Прописною
```

`casefold()` – те ж саме, що й `lower()`, але додатково перетворить усі символи з діакритичними знаками й лігатури в букви стандартної латиниці. Зазвичай застосовується для порівняння рядків:

Приклад 23

```
>>> "Python".casefold()=="python".casefold()
True
>>> "grosse".casefold() == "groβe".casefold()
False
```

Пошук і заміна в рядку. Метод find()

`find()` – шукає підрядок в рядку. Повертає номер позиції, з якої починається входження підрядка в рядок. Якщо підрядок в рядок не входить, то повертається значення `-1`. Метод залежить від регістру символів.

Формат методу:

```
<Рядок>.find(<Підрядок>[,<Початок>[,<Кінець>)))
```

Якщо початкова позиція не зазначена, то пошук буде здійснюватися з початку рядка. Якщо параметри `<Початок>` і `<Кінець>` зазначені, то проводиться операція добування зрізу:

```
<Рядок>[<Початок>:<Кінець>]
```

і пошук підрядка буде виконуватися в цьому фрагменті.

Метод find() продовження

Приклад 26

```
>>> s = "приклад приклад Приклад"
>>> s.find(" при")
7
>>> s.find(" При")
15
>>> s.find("тест")
-1
>>> s.find(" при",9)
-1
s.find(" при",0,6)
-1
s.find(" при",7,12)
7
```

Метод index()

index() – метод аналогічний методу find(), але якщо підрядок в рядок не входить, то виконується виключення Valueerror.

Формат методу:

<Рядок>.**index**(<Підрядок>[,<Початок>[,<Кінець>]])

Приклад 27

```
>>> s = "приклад приклад Приклад"
>>> s.index(" при")
7
s.index(" при", 7, 12)
7
s.index(" При", 1)
15
>>> s.index("тест")
Traceback (most recent call last):
  File "<input>", line 1, in <module>
Valueerror: substring not found
```

Метод rfind()

`rfind()` – шукає підрядок в рядку. Повертає позицію останнього входження підрядка в рядок.

Формат методу:

```
<Рядок>.rfind(<Підрядок>[,<Початок>[,<Кінець>]])
```

- Якщо підрядок в рядок не входить, то повертається значення -1.
- Метод залежить від регістру символів.
- Якщо початкова позиція не зазначена, то пошук буде проводитися з початку рядка.
- Якщо параметри `<Початок>` і `<Кінець>` зазначені, то виконується операція добування зрізу, і пошук підрядка буде проводитися в цьому фрагменті.

Метод rfind() продовження

Приклад 28

```
>>> s = "приклад приклад Приклад Приклад"

>>> s.rfind("при")
8
s.rfind("При")
24
s.rfind("тест")
-1
>>> s.find(" при",0,61)
7
s.find(" При",10,20)
15
```

Метод rindex()

`rindex()` - метод аналогічний методу `rfind()`,

Формат методу:

`<Рядок>.rindex(<Підрядок>[,<Початок>[,<Кінець>]])`

Якщо підрядок в рядок не входить, то виконується виключення `Valueerror`.

Приклад 29

```
>>> s = "приклад приклад Приклад Приклад"
>>> s.rindex(" при")
7
>>> s.rindex(" при",0,11)
7
>>> s.rindex("тест")
Traceback (most recent call last):
  File "<input>", line 1, in <module>
Valueerror: substring not found
```

Метод count()

`count()` – повертає число входжень підрядка в рядок.

Формат методу:

```
<Рядок>.count(<Підрядок>[,<Початок>[,<Кінець>]])
```

Якщо підрядок в рядок не входить, то повертається значення 0. Метод залежить від регістру символів.

Приклад 30

```
>>> s = "приклад приклад Приклад Приклад"
>>> s.count(" при")
1
>>> s.count(" при", 6)
1
>>> s.count(" При")
2
>>> s.count("тест")
0
```

Перевірка типу вмісту рядка

(Букви і цифри)`isalnum()` - повертає `True`, якщо рядок містить тільки букви й (або) цифри, а якщо ні, то - `False`. Якщо рядок порожній, то вертається значення `False`.

Приклад 31

```
>>> "0123".isalnum()
True
>>> "123abc".isalnum()
True
>>> "рядок".isalnum()
True
>>> "".isalnum()
False
>>> "123 abc".isalnum()
False
>>> "abc, 123.".isalnum()
False
```

(Тільки букви) `isalpha()` – повертає `True`, якщо рядок містить тільки букви, а якщо ні, то – `False`.
Якщо рядок порожній, то повертається значення `False`.

Приклад 32

```
>>> "string".isalpha()
True
>>> "рядок".isalpha()
True
>>> "".isalpha()
False
>>> "123abc".isalpha()
False
>>> "str str".isalpha()
False
>>> "st,st". isalpha()
False
```

(Тільки цифри) `isdigit()` – повертає `True`, якщо рядок містить тільки цифри, а якщо ні, то– `False`:

Приклад 33

```
>>> "0123".isdigit()
True

>>> "1.3".isdigit()
False

>>> "123abc".isdigit()
False

>>> "abc123".isdigit()
False
```

(Тільки десяткові символи) `isdecimal()` – повертає `True`, якщо рядок містить тільки десяткові символи, а якщо ні, то – `False`.

1. Десятковими символами є не лише десяткові цифри в кодуванні ASCII, а також десяткові цифри в інших мовах.

Приклад 34

```
>>> "123".isdecimal()
True

>>> "123стр".isdecimal()
False

>>> "⅔".isdecimal()
False
```

Приклад аналізу рядка з використанням методу split() та isdecimal()

```
a = input("Ведіть число")
c = a.split(".")
print(c)
if len(c)==2:
    if c[0].isdecimal() and c[1].isdecimal():
        result = float(a)
        print("float:", result)
    else: print("str:", a)
elif len(c)==1:
    if c[0].isdecimal():
        result = int(a)
        print("int:", result)
    else: print("str:", a)
```

(Тільки числові символи) `isnumeric()` – повертає `True`, якщо рядок містить тільки числові символи, а якщо ні, то – `False`.

1. Числовими символами є не тільки десяткові цифри в кодуванні ASCII, але символи римських чисел, дробові числа й ін.

Приклад 35

```
>>> "\u2155".isnumeric() # символ ⅕
True
>>> "\u2155".isdigit ()
False
>>> print("\u2155")
⅕
>>> "⅕".isnumeric()
True
>>> "⅕".isdigit()
False
```

(Тільки верхній регістр) `isupper()` – повертає `True`, якщо рядок містить букви тільки верхнього регістру, а якщо ні, то – `False`:

Приклад 36

```
>>> "STRING".isupper()
True
>>> "РЯДОК".isupper()
True
>>> "".isupper()
False
>>> "String1". isupper()
False
>>> "РЯДОК,123".isupper()
True
>>> "123".isupper()
False
>>> "string".isupper(), "String".isupper()
(False, False)
```

(Тільки нижній регістр)`islower()`- повертає `True`, якщо рядок містить букви тільки нижнього регістру, а якщо ні, то - `False`:

Приклад 37

```
>>> "srting".islower()
True
>>> "рядок".islower()
True
>>> "".islower()
False
>>> "string1".islower()
True
>>> "s tr, 123".islower()
True
"123".islower()
False
>>> "STRING".islower(), "Рядок".islower()
(False, False)
```

(Все з великої букви)`istitie()` – повертає `True`, якщо всі слова в рядку починаються з великої букви, а якщо ні, то – `False`. Якщо рядок порожній, також повертається `False`.

Приклад 38

```
>>> "Str Str".istitle()
True
>>> "Стр Стр".istitle()
True
>>> "Str Str 123".istitle()
True
>>> "Стр Стр 123".istitle()
True

>>> "Str str".istitle(), "Стр стр".istitle()
(False, False)

>>> "".istitle(), "123".istitle()
(False, False)
```

(Тільки символи для друку) `isprintable()` – повертає `True`, якщо рядок містить тільки символи, що друкуються, а якщо ні, то – `False`. Відзначимо, що пробіл є символом, що друкується.

Приклад 39

```
>>>"123".isprintable()
True
>>> "PHP Python".isprintable()
True
>>> "\n".isprintable()
False
```

`isspace()` – повертає `True`, якщо рядок містить тільки «пропускові» (пробільні) символи, а якщо ні, то – `False`:

Приклад 40

```
>>> "".isspace()," \n\r\t".isspace(),"str
str".isspace()
(False, True, False)
```

`isidentifier()` – повертає `True`, якщо рядок є припустимим з погляду Python ім'ям змінної, функції або класу, а якщо ні, то – `False`:

Приклад 41

```
>>> "s".isidentifier()
True
>>> "func".isidentifier()
True
>>> "123func".isidentifier()
False
```

Метод `isidentifier ()` лише перевіряє, чи задовольняє задане ім'я правилам мови.

Він не перевіряє, чи збігається це ім'я з ключовим словом Python.

Для перевірки на ключове слово слід застосовувати функцію `iskeyword ()`, оголошену в модулі `keyword`, `iskeyword ()` повертає `True`, якщо переданий їй рядок збігається з одним із ключових слів:

Приклад 42

```
>>> import keyword

>>> keyword. iskeyword ("else")
True

>>> keyword.iskeyword("elsewhere")
False
```

Програма додавання довільної кількості цілих чисел, введених користувачем

При введенні рядка замість числа програма виводить повідомлення.
Передбачимо можливість введення від'ємних цілих чисел.

Приклад 43. Додавання довільної кількості чисел

```
print("Введіть слово 'stop' для отримання
результату")
suma = 0
while True:
    x = input("Введіть число: ")
    if x == "stop":
        break  # вихід з циклу
    if x == "":
        print("Ви не ввели значення!")
```

```
            continue
        if x[0] == "-": # Якщо першим символом є мінус
            if not x[1:].isdigit(): # Якщо фрагмент не складається з цифр
                print("Необхідно ввести число, а не рядок!")
                continue
        else: # Якщо мінуса немає, то перевіряємо весь рядок
            if not x.isdigit(): #  Якщо рядок не складається з цифр
                print("Необхідно ввести число, а не рядок!")
                continue
        x = int(x) # Перетворюємо рядок в число
        suma += x
print("Сума чисел дорівнює:", suma)
```

Процес введення значень і одержання результату має такий вигляд (значення, введені користувачем, виділені напівжирним шрифтом):

```
Введіть слово 'stop' для одержання результату
Введіть число: 10
Введіть число:
Ви не ввели значення!
Введіть число: str
Необхідно ввести число, а не рядок!
Введіть число: -5
Введіть число: -str
Необхідно ввести число, а не рядок!
Введіть число: stop
Сума чисел дорівнює: 5
```