**Python: опрацювання рядків**

|  |  |
| --- | --- |
| <https://www.e-olymp.com/uk/problems/8921><https://www.e-olymp.com/uk/problems/8922><https://www.e-olymp.com/uk/problems/8923><https://www.e-olymp.com/uk/problems/8924><https://www.e-olymp.com/uk/problems/8925><https://www.e-olymp.com/uk/problems/8926> | <https://www.e-olymp.com/uk/problems/8937><https://www.e-olymp.com/uk/problems/8938><https://www.e-olymp.com/uk/problems/8939><https://www.e-olymp.com/uk/problems/8940><https://www.e-olymp.com/uk/problems/8942> |
| <https://www.e-olymp.com/uk/problems/8977><https://www.e-olymp.com/uk/problems/8978><https://www.e-olymp.com/uk/problems/8979><https://www.e-olymp.com/uk/problems/8980><https://www.e-olymp.com/uk/problems/8981><https://www.e-olymp.com/uk/problems/8982><https://www.e-olymp.com/uk/problems/8983><https://www.e-olymp.com/uk/problems/8984><https://www.e-olymp.com/uk/problems/8985><https://www.e-olymp.com/uk/problems/8986><https://www.e-olymp.com/uk/problems/8987><https://www.e-olymp.com/uk/problems/8988><https://www.e-olymp.com/uk/problems/8989><https://www.e-olymp.com/uk/problems/8990><https://www.e-olymp.com/uk/problems/8991><https://www.e-olymp.com/uk/problems/8992><https://www.e-olymp.com/uk/problems/8993><https://www.e-olymp.com/uk/problems/8994><https://www.e-olymp.com/uk/problems/8995><https://www.e-olymp.com/uk/problems/8996><https://www.e-olymp.com/uk/problems/8997><https://www.e-olymp.com/uk/problems/8998><https://www.e-olymp.com/uk/problems/8999><https://www.e-olymp.com/uk/problems/9000> |

**Рядки** — *упорядковані послідовності символів, що використовують для зберігання і представлення текстової інформації.* За допомогою рядків можна працювати з усім, що може бути подано у текстовій формі.

**Рядки в апострофах і в лапках**. Рівнозначними є обидва записи рядків:

S = 'spam"s'
S = "spam's"

Два варіанти надають можливість дозволяють вставляти в рядок символи лапок або апострофів, не використовуючи екранування.

**Базові дії з рядками**
(у коментарі подано результат і через дефіс - записано назву операції)

'spam'+'eggs' # 'spameggs' - конкатенація (додавання)

'spam'\*3 # 'spamspamspam' - дублювання

len('spam') # 4 - довжина рядка

S = 'spam'

S[0] # 's' - доступ за індексом

S[2] # 'a'

S[-2] # 'a'

Як видно з останнього прикладу, у Python є можливість доступу за від'ємним індексому. При цьому відлік йде від кінця рядка.

*Оператор вилучення зрізу* має такий вигляд: [X:Y:Z], де X — початок зрізу, а Y — закінчення, Z — крок. Символ з номером Y у зріз не входить. Як усталено X дорівнює 0, а Y — довжині рядка, Z — 1.

s = 'spameggs'

s[3:5] # 'me'

s[2:-2] # 'ameg'

s[:6] # 'spameg'

s[1:] # 'pameggs'

s[:] # 'spameggs'

s[::-1] # 'sggemaps'

s[3:5:-1] # ''

s[2::2] # 'aeg'

Викликаючи методи рядків необхідно пам'ятати, що рядки у Python відносять до категорії незмінних послідовностей. Інакше кажучи, всі функції і методи можуть лише створювати новий рядок. Тому всі рядкові методи повертають *новий* рядок, за допомогою якого потрібно надати значення змінній.

**Приклади використання функцій і методів рядків**

* S = 'str'; S = "str"; S = '''str'''; S = """str""" — літерали рядків;
* S = "s\np\ta\nbbb" — екрановані послідовності;
* S = r"C:\temp\new" — пригнічення екранування;
* S = b"byte" — рядок байтів;
* S1 + S2 — додавання;
* S1 \* 3 — повторення;
* S[i] — звернення за індексом;
* S[i:j:step] — витяг зрізу;
* len(S) — довжина рядка;
* S.find(str, [start],[end]) — пошук підрядка в рядку, повертає номер першого входження або -1;
* S.rfind(str, [start],[end]) — пошук підрядка в рядку, повертає номер останнього входження або -1;
* S.index(str, [start],[end]) — пошук підрядка в рядку, повертає номер першого входження або викликає ValueError;
* S.rindex(str, [start],[end]) — пошук підрядка в рядку, повертає номер останнього входження або викликає ValueError;
* S.replace(*шаблон, заміна*) — заміна шаблону;
* S.split(*символ*) — розбиття рядка за роздільником;
* S.isdigit() — чи складається рядок з цифр;
* S.isalpha() — чи складається рядок з літер;
* S.isalnum() — чи складається рядок з цифр або літер;
* S.islower() — чи складається рядок із символів у нижньому регістрі;
* S.isupper() — чи складається рядок із символів у верхньому регістрі;
* S.isspace() — чи складається рядок з невідображуваних символів (пробіл, ознаки кінця сторінки '\f' і рядка '\n', переведення каретки '\r', горизонтальна табуляція '\t' і вертикальна табуляція '\v');
* S.istitle() — чи починаються слова в рядку з великої літери;
* S.upper() — перетворення рядка до верхнього регістру;
* S.lower() — перетворення рядка до нижнього регістру;
* S.startswith(str) — чи починається рядок S з шаблону str;
* S.endswith(str) — чи закінчується рядок S шаблоном str;
* S.join(*список*) — збірка рядка зі списку з роздільником S;
* ord(*символ*) — ASCII код символу;
* chr(*число*) — символ з вказаним кодом ASCII;
* S.capitalize() — переводить перший символ рядка у верхній регістр, а всі інші — в нижній;
* S.center(width, [fill]) — повертає відцентрований рядок, по краях якої стоїть символ fill (пробіл за замовчуванням);
* S.count(str, [start],[end]) — повертає кількість непересічних входжень підрядка в діапазоні [початок, кінець] (0 і довжина рядка як усталено);
* S.expandtabs([tabsize]) — повертає копію рядка, в якому всі символи табуляції замінtено одним або кількома пропусками залежно від поточного стовпчика. Якщо TabSize не вказано, розмір табуляції — 8 пробілів;
* S.lstrip([chars]) — видалення символів пробілів на початку рядка;
* S.rstrip([chars]) — видалення символів пробілів в кінці рядка;
* S.strip([chars]) — видалення символів пробілів на початку і в кінці рядка;
* S.partition(*шаблон*) — повертає кортеж, що містить частину перед першим шаблоном, сам шаблон, і частина після шаблону. Якщо шаблон не знайдено, повертається кортеж, що містить самий рядок, а потім два порожніх рядки;
* S.rpartition(sep) — повертає кортеж, що містить частину перед останнім шаблоном, сам шаблон, і частина після шаблону, якщо шаблон не знайдений, повертається кортеж, що містить два порожні рядки, а потім самий рядок;
* S.swapcase() — перекладає символи нижнього регістра в верхній, а верхнього — в нижній;
* S.title() — першу букву кожного слова переводить в верхній регістр, а всі інші в нижній;
* 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;
* S.rjust(width, fillchar=" ") — робить довжину рядки не меншою width, в разі потреби заповнює перші символи символом fillchar;
* S.format(\*args, \*\*kwargs) — [форматування](https://pythonworld.ru/osnovy/formatirovanie-strok-metod-format.html) рядка.

**Завдання 1.** Скласти програму для підрахунку кількості слів у введеному з клавіатури рядку. Словом вважати послідовність символів, що відокремлену символами пропуску.

S = input()

l = S.split()

print(len(l))

**Завдання 2**. Скласти програму для підрахунку кількості входжень символа a у рядок S.

S = input()

a = input()

n = 0

for ch in S:

 if ch == a:

 n += 1

print(n)

Розглянемо деякі базові операції — найосновніші базові функції і методи класу str в Python 3 на простих прикладах. Повністю переписувати документацію немає сенсу — вона є на офіційних джерелах завжди. А ось наочно розглянути прості практичні приклади може бути корисно для початкових програмістів на Python.

Слід запам'ятати новачкам, що рядки в Python є незмінними послідовностями, тому робота методів і функцій відбувається зі створенням нового рядка. Для цієї мети і необхідно створення змінної.

**Конкатенація** — приєднання букв один до одного без пробілів. По синтаксису дана проста функція реалізується через ***плюс*** (***+***, знак складання). Приклад:

a="prisoe"

b="dinenie"

print(a+b)

**Дублювання** рядка здійснюється за допомогою ***зірочки \**** після самого елемента дублювання із зазначенням кількості потрібних копій. У прикладі спеціально після слова перед закриттям лапок прописаний пробіл, щоб копії були візуально розділені, інакше вони будуть зліплені в одне слово. Приклад:

print("Example " \* 4)

**Довжина** рядка реалізується за допомогою функції *len()*. Повертає кількість елементів у рядку. Синтаксис для прикладу наступний. Виведе в результаті число 12 (довжину повну разом з пропуском).

s=len('dlina stroki')

print(s)

**Доступ за індексом** до певного елемента. Можна використовувати також і негативний індекс, в такому випадку відлік почнеться з кінця рядка і не з нуля, а з одиниці (з -1). Приклад доступу за звичайним і негативним індексом нижче.

p = 'primer'

print(p[2])

print(p[-1])

print(p[-3])

**Витяг зрізу** елементів рядка відбувається за формулою ***[a:b:c]***, де a — початок зрізу, b — його закінчення (сам b в зріз не входить), c — крок зрізу. Перший індекс за замовчуванням відраховується з нуля, а другий дорівнює довжині рядка. Мінус в даному випадку потрібен, якщо необхідно робити зріз в зворотну сторону (з кінця). На прикладах нижче все зрозуміло стане навіть, якщо в теорії неясно.

i = "indeksirovanie"

print (i [1: 6])

# Два елементи через двокрапку. Зріз списку з 1 по 6 індекс (крок не вказано, якщо в зрізі всього 2 значення), значить будуть виведені всі елементи за вказаними параметрами.

i = "indeksirovanie"

print (i [3: -3])

# Два елементи через двокрапку, другий елемент має негативний індекс. Зріз списку з 3 елементи (спочатку) і до 3 елемента з кінця списку (-3 означає що відлік буде проведений в зворотну сторону).

i = "indeksirovanie"

print (i [: 4])

# Якщо вказати двокрапка, а потім змінну, зріз буде проведений з першого елемента за індексом (з 0), число, що йде після двокрапки, вказує кількість елементів в зрізі.

i = "indeksirovanie"

print (i [1:])

# Якщо вказати спочатку число, а потім двокрапка, зріз буде виконаний з усього списку елементів, починаючи з зазначеного індексу.

i = "indeksirovanie"

print (i [0])

# Зазначено просто одне число (індекс) Зріз списку з одного елемента (першого).

i = "indeksirovanie"

print (i [:])

# Зазначено просто двокрапка. В даному випадку наявність порожніх всіх змінних з формули показує, що буде виведений весь список елементів з самого початку і до кінця.

i = "indeksirovanie"

print (i [:: - 3])

# Дві двокрапки і значення. Крок мінус три при наявності двох порожніх змінних у формулі буде виробляти зріз із зворотного боку (кінця списку) і виводити кожен третій елемент (починаючи з індексу 0).

i = "indeksirovanie"

print (i [1: 4: 1])

# Вказані всі елементи зрізу за формулою. Виведено елементи з 1 по 4 з кроком в 1.

i = "indeksirovanie"

print (i [6 :: 6])