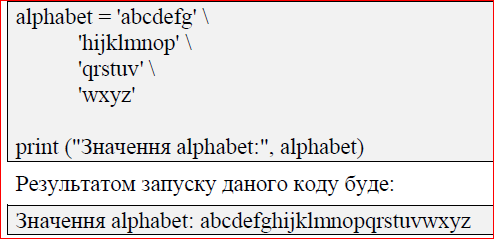
1. **Рядки, операції з рядками.**

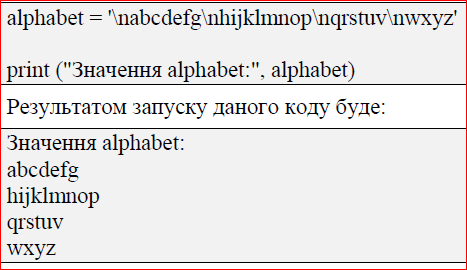
На відміну від інших мов, в Python рядки є незмінними. Не можна змінити сам рядок, але можна скопіювати частини рядків в інший рядок, щоб отримати той же ефект. Рядок в Python створюється заключенням символів в одинарні або подвійні лапки. Два види лапок дозволяють створювати рядки, що містять лапки. Усередині одинарних лапок можна розташувати подвійні і

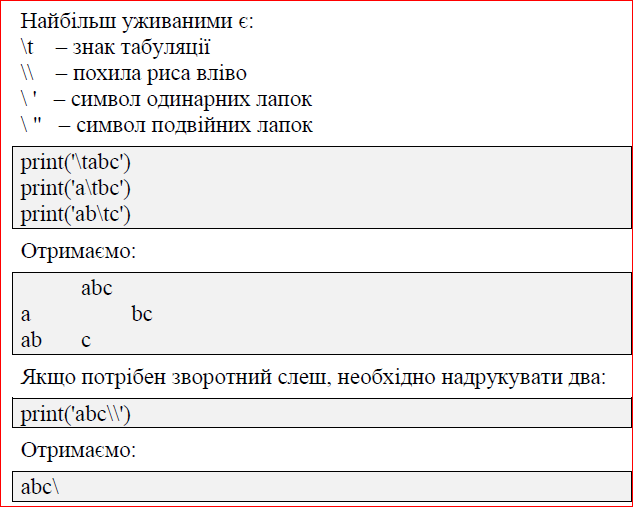
навпаки:

Будь-яка програма стає більш зрозумілою, якщо її рядки відносно короткі. Рекомендована (але не обов’язкова) максимальна довжина рядка дорівнює 80 символам. Якщо є необхідність надрукувати рядок що містить більше 80 символів, використовують символ відновлення \, що розміщюється в кінці рядка, і далі Python буде діяти так, ніби це все той же рядок.



Крім цього зворотний слеш (\) дозволяє створювати керуючі послідовності всередині рядків. Найбільш поширена послідовність \n, яка означає перехід на новий рядок. З її допомогою можна створити багаторядкові рядки з однорядкових:





Числа можуть бути перетворені в рядки за допомогою функції str(). Наприклад, str (123) дасть рядок ‘123’. Якщо рядок є послідовністю знаків-цифр, то він може бути перетворений в *ціле* число за допомогою функції int() (int(‘123’) дасть в результаті число 123), а в *дійсне* – за допомогою функції float() (float(’12.34’) дасть в результаті число 12.34. Для будь-якого символа можна узнати його номер (код символа за допомогою функції ord() (ord(‘s’) дасть результат 115). І навпаки, отримати символ за числовим кодом можна за допомогою функції chr() (chr(100) дасть результат ‘d’).

Правило, за яким відбувається порівняння двох рядків: посимвольно зліва направо порівнюються коди відповідних символів до тих пір, поки не порушиться рівність кодів або не закінчиться один із рядків (або обидва відразу), при цьому відразу робиться висновок про нерівність і визначається, який з рядків менше, а який - більше. Дві величини типу str є рівними, якщо вони рівні по довжині і збігаються посимвольно.

*Основні операції з рядками*

|  |  |
| --- | --- |
| Функція (операція) | Опис та результат |
| len() | Обчислюється довжина рядка s як число символів |
| s1+s2 | **Конкатенація**. До кінця рядка s1 приєднується рядок s2. Отримуємо новий рядок ‘ви’+’года’ 🡪 ’вигода’  Python не додає пробілів при конкатенації рядків, тому потрібно явно додати пробіли |
| s\*n (n\*s) | n-кратне повторення рядка s. Отримуємо новий рядок (‘кан’\*2 🡪 ’канкан’) |
| s[i] | Вибір із s елемента з номером i, нумерація починається з 0 (перший елемент – номер 0). Результатом є символ. Якщо i<0, відлік йде з кінця (перший символ – номер 0, останній номер -1)  Приклад: s=’дерево’  s[2] 🡪 ‘р’  s[-2] 🡪 ‘в’ |
| s[i:j:k] | Зріз – підрядок, що містить символи рядка s з номерами від i до j з кроком k (елемент з номером i входить до підсумкового рядка, а елемент з номером j вже не входить). Якщо символ k не вказаний (s[i:j]), то символи йдуть підряд (рівносильно s[i:j:1]).  Приклад:  s=’derevo’  s [3:5] 🡪 ‘ev’  s [1:5:2] 🡪’ee’ |
| min(s) | Визначає та виводить символ з найменшим значенням (кодом- номером у кодовій таблиці)  Приклад:  s=’derevo’  min(s) 🡪 ‘d’ |
| max(s) | Визначає та виводить символ з найбільшим значенням  Приклад:  s=’derevo’  max (s) 🡪 ‘v’ |

Оскільки рядки незмінні – не можна вставити символ безпосередньо в рядок або змінити символ за заданим індексом.

З рядка можна вилучати підрядок (частину рядка) за допомогою функції slice. Визначається slice за допомогою квадратних дужок, зміщення початку підрядка start і кінця підрядка end, а також розміру кроку step.

**[start: end: step]**

Деякі з цих параметрів можуть бути відсутні. У підрядок будуть включені символи, розташовані починаючи з точки, на яку вказує зміщення start, і закінчуючи точкою, на яку вказує зміщення end.

– Оператор [:] дозволяє взяти зріз всієї послідовності від початку до кінця.

– Оператор [start:] дозволяє взяти зріз послідовності з точки, на яку вказує зміщення start, до кінця.

– Оператор [: end] дозволяє взяти зріз послідовності від початку до точки, на яку вказує зміщення end - 1.

– Оператор [start: end] дозволяє взяти зріз послідовності з точки, на яку вказує зміщення start, до точки, на яку вказує зміщення end - 1.

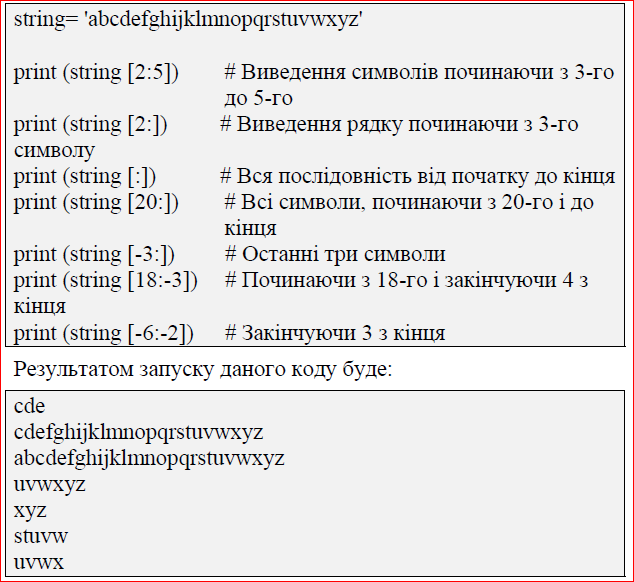
– Оператор [start: end: step] дозволяє взяти зріз послідовності з точки, на яку вказує зміщення start, до точки, на яку вказує зміщення end мінус 1, опускаючи символи, чиє зміщення всередині підрядка кратне step.

Зміщення зліва направо визначається як 0, 1 і т.д., а справа наліво – як -1, -2 і т.д.

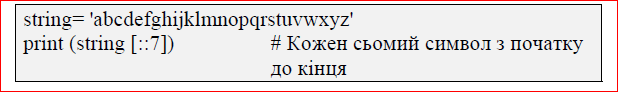
Якщо не вказати start, функція буде використовувати в якості його значення 0 (початок рядку).

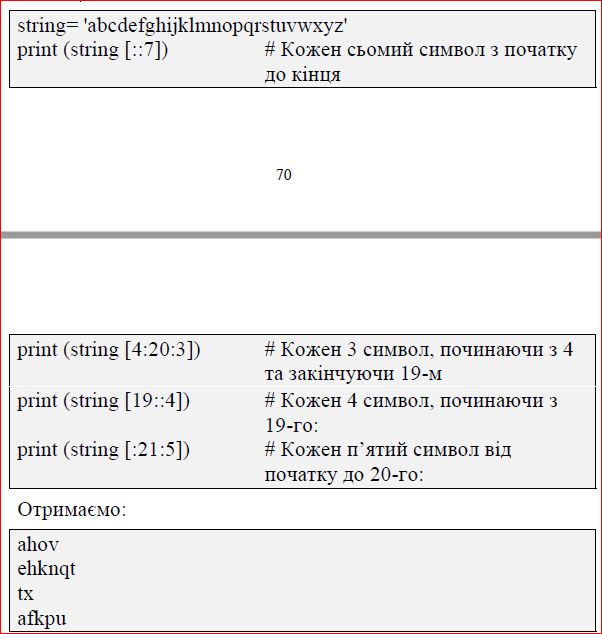
Якщо не вказати end, функція буде використовувати кінець рядка.

Python не включає символ, розташований під номером, який вказаний останнім.



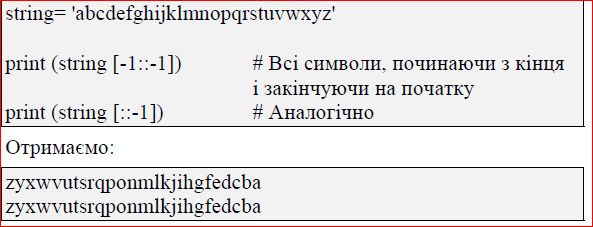
Щоб збільшити крок, необхідно вказати його після другої двокрапки.





Значення end має бути на одиницю більше, ніж реальне зміщення.

Якщо задати від’ємний крок, Python буде рухатися у зворотний бік.



Рядки, як об'єкти Python, володіють методами (тобто функціями, які виконують самі об'єкти). Основні методи перераховані в наступній таблиці. Нехай рядок, до якого застосовуються ці методи, називається s1.

Форматування рядків

|  |  |
| --- | --- |
| Метод | Опис та результат |
| s1. center (n) | Повертається рядок s1, доповнений пробілами праворуч і ліворуч до ширини в n символів. Вихідний рядок не змінюється. Якщо n <= len (s1), пробіли не додаються.  Приклад: s1=’Zoom−Zoom’  s1. center(15) 🡪 ’␣␣␣Zoom−Zoom␣␣␣’ |
| s1. ljust (n) | Рядок s1 вирівнюється по лівому краю (доповнюється пробілами праворуч) в просторі шириною n символів. Якщо n <len (s1), пробіли не добавляються.  Приклад:  s1=’Zoom−Zoom’  s1. ljust (15) 🡪 ’Zoom−Zoom␣␣␣␣␣␣’ |
| s1. rjust (n) | Рядок s1 вирівнюється по правому краю (доповнюється пробілами ліворуч) в просторі шириною n символів. Якщо n <len (s1), пробіли не добавляються.  Приклад:  s1=’Zoom−Zoom’  s1. rjust (15) 🡪 ␣␣␣␣␣␣’Zoom−Zoom’ |

Однією з таких вбудованих функцій є функція split(), що розбиває рядок на список невеликих рядків, спираючись на роздільник.

**рядок.split('роздільник')**

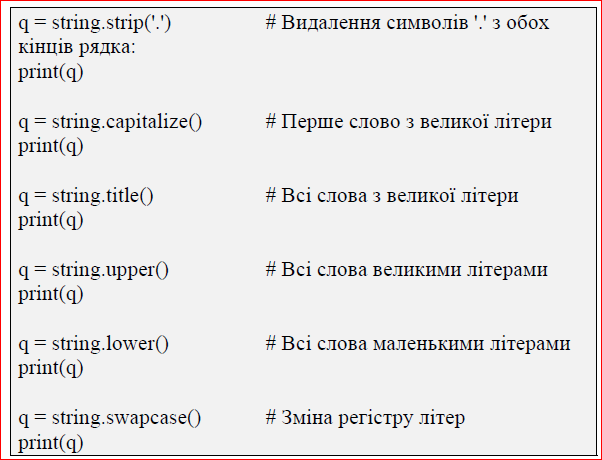
*Об’єднання рядків за допомогою функції join()*

Функція join() є протилежністю функції split(): об’єднує список рядків в один рядок.

Для виклику функції спочатку вказується рядок, який об’єднує інші, а потім – список рядків для об’єднання:

**рядок.join(список)**

Для того щоб об’єднати список рядків lines, розділивши їх символами нового рядка, потрібно написати '\n'.join(lines).



Python дозволяє інтерполювати дані в рядки – розмістити значення всередині рядків, – застосовуючи різні формати. Можна використовувати цю можливість, щоб створювати звіти та інші документи, яким необхідно зробити певний зовнішній вигляд.

Python пропонує два способи форматування рядків.

Форматування рядків з використанням символу % має форму:

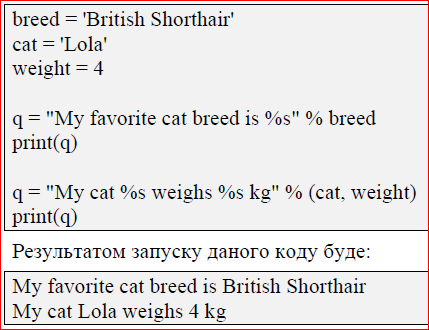
**рядок % дані**

Усередині рядка знаходяться інтерполяційні послідовності. Найпростіша послідовність – це символ %, за яким слідує буква, що представляє тип даних, який повинен бути відформатований

**Типи перетворення.**

|  |  |
| --- | --- |
| %s | Рядок |
| %d | Ціле число в десятковій системі числення |
| %x | Ціле число в шістнадцятковій системі числення |
| %o | Ціле число в вісімковій системі числення |
| %f | Число з плаваючою крапкою в десятковій системі числення |
| %e | Число з плаваючою крапкою в шістнадцятковій системі числення |
| %g | Число з плаваючою крапкою у вісімковій системі числення |
| %% | Символ % |

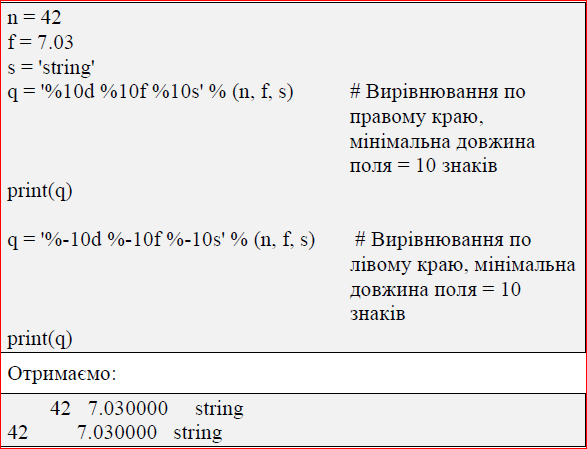
Послідовність %s всередині рядка означає, що в неї потрібно інтерполювати рядок. Кількість використаних символів % повинно збігатися з кількістю об’єктів, які розташовуються після %.



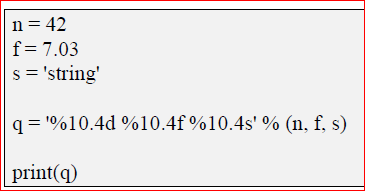
Один об'єкт на зразок breed розташовується відразу після символу %. Якщо таких об’єктів кілька, вони повинні бути згруповані в кортеж (потрібно оточити їх дужками і розділити комами) на зразок (cat, weight).

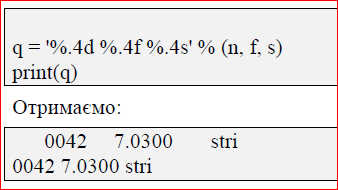
Незважаючи на те що змінна weight цілочисельна, послідовність %s всередині рядка перетворює її в рядок.

Можна встановити мінімальну довжину поля для кожної змінної і вирівняти їх по правому (лівому) краю, заповнюючи невикористане місце пробілами:



Можна вказати довжину поля та максимальну кількість символів (вирівнювання по правому краю). Таке налаштування обрізає рядок і обмежує число з плаваючою точкою чотирма цифрами після десяткової коми:





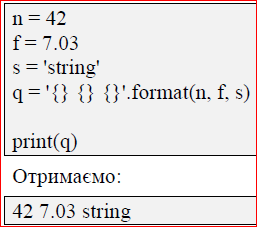
Новий стиль форматування за допомогою методу format() має синтаксис:

**рядок\_спеціального\_формату.format(\*args, \*\*kwargs)**

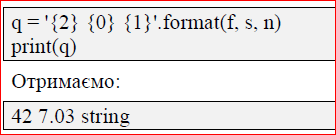
У параметрі рядок\_спеціального\_формату всередині символів {} можуть бути вказані деякі специфікатори.

Всі символи, розташовані поза фігурних дужок, виводяться без перетворень. Якщо всередині рядка необхідно використовувати символи {}, то ці символи слід подвоїти, інакше збуджується виняток ValueError.

Аргументи старого стилю потрібно надавати в порядку появи їх наповнювачів з символами % в оригінальній рядку. За допомогою нового стилю можна вказувати будь-який порядок:

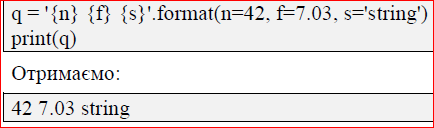




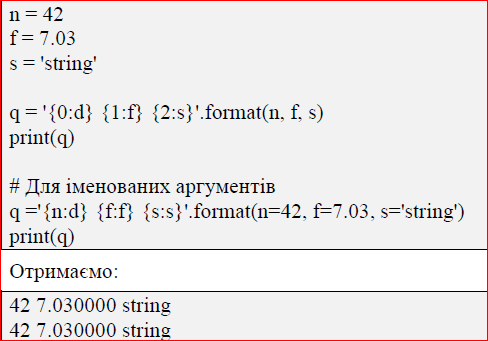


Значення 0 відноситься до першого аргументу, f, 1 відноситься до рядка s, а 2 – до останнього аргументу, цілого числа n.

Аргументи можуть бути словником або іменованими аргументами, а специфікатори можуть включати їх імена:

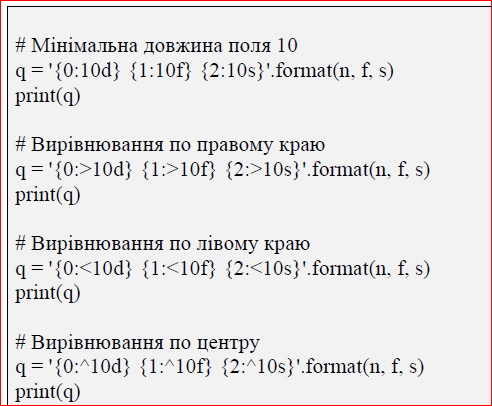


Старий стиль дозволяє вказати специфікатор типу після символу %, а новий стиль – після :.

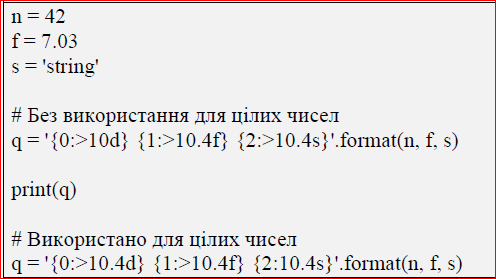


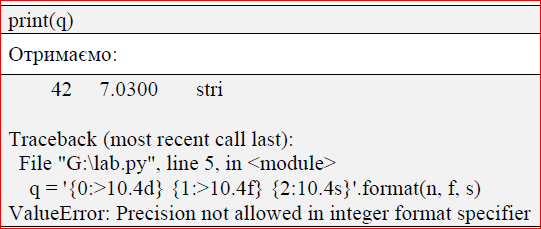
Інші можливості (мінімальна довжина поля, максимальна ширина символів, зміщення і т.д.) також підтримуються.





Значення точності (після десяткової коми) означає кількість цифр після десяткової коми для дробових чисел і максимальне число символів рядка, але не можна використовувати його для цілих чисел:





Якщо необхідно заповнити поле виведення чимось крім пробілів, можна розмістити необхідний символ відразу після двокрапки, але перед символами вирівнювання (<,>, ^) або специфікатором ширини:

