Програмування

ТЕМА 5. СИМВОЛИ ТА РЯДКИ

Символи та коди

Алфавітом *Ch* назвемо множину символів.

До цієї множини входять латинські букви, арабські цифри, спеціальні математичні і інші символи, розділові знаки, символи національних алфавітів.

Для збереження у пам’яті комп’ютера кожному символу повинно бути співставлене деяке число, яке називають **кодом символа**.

Стандарти кодування символів

Назва	Кіль-кість кодів	Діапа-зон кодів	Символи	Особливості
ASCII	128	0 - 127	латинські літери, цифри, розділові знаки, дужки	Всі цифри йдуть підряд, латинські літери впорядковані за алфавітом
CP866U	256	0 - 255	ASCII + кирилиця	Символи російської абетки впорядковані за алфавітом, але не всі йдуть підряд
ANSI	256	0 - 255	ASCII + символи національних алфавітів	символи національних алфавітів вказані у «кодових сторінках» по 128 символів. Сторінка для кирилиці - 1251
KOI-8	256	0 - 255	ASCII + кирилиця	Символи кирилиці не впорядковані за алфавітом

Стандарт Unicode

Unicode – це універсальний стандарт для кодування всіх символів

UCS (universal character set) та UTF (Unicode transformation format)

Будемо використовувати позначення кодів у системі числення за основою 16 наступним чином:

0xhhhh,

де hhhh – число у системі числення за основою 16.

Спочатку у Unicode було 65 536 (2^{16}) символів.

Потім частину кодів – від 0xD800 до 0xDFFF – виділили для розширення так, що додаткові символи позначаються двохбайтними так званими сурогатними парами.

Стандарт Unicode.2

Зараз вважають, що коди просто можуть мати 6 цифр у системі числення за основою 16 (від 0x000000 до 0x10FFFF).

Є 17 кодових площин (planes) по 65 536 символів (у нульовій площині мінус 2048 символів)

Загальна потенційна кількість символів

65 536 * 17 – 2048 = 1 112 064.

На сьогодні у 10 версії Unicode зайнято біля 137 000 кодових позицій.

Символ у Unicode позначається U+hhhh (або U+hhhhh або U+hhhhhh), де hhhh – код символа у системі числення за основою 16.

UTF-8

Формат представлення зі змінною кількістю байтів на символ.

ASCII – символи кодуються одним байтом так само, як і у ASCII.

Інші символи – від 2 до 4 байтів

- Unicode UTF-8:

Діапазон кодів	Байти
0x00000000 — 0x0000007F:	0xxxxxxx
0x00000080 — 0x000007FF:	110xxxxx 10xxxxxx
0x00000800 — 0x0000FFFF:	1110xxxx 10xxxxxx 10xxxxxx
0x00010000 — 0x001FFFFF:	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

Тут xxxxxxx – це окремі біти, які можуть набувати значення 0 або 1

UTF-16 та UTF-32

UTF-16

- Символи представляються 2 байтами, окрім символів, які представляються сурогатними парами
- UTF-16BE – в цьому представленні старший байт іде спочатку
- UTF-16LE – в цьому представленні молодший байт іде спочатку
- BE та LE означають big-endians та little-endians. У перекладі – «тупокінечники» та «гострокінечники».

UTF-32

- Всі символи представляються 4 байтами
- UTF-32BE та UTF-32LE – аналогічно UTF-16BE та UTF-16LE

UTF-16 та UTF-32 BE та LE

Символи у Python

У Python за угодою символи представлені у форматі UTF-8.

Для перетворення символів з/в інші стандарти кодування передбачені дії кодування (encoding) та декодування (decoding).

Рядки

носій

Визначимо множину слів у алфавіті *Ch*, яку позначимо *W*, наступним чином:

1. Порожнє слово $\Lambda \in W$
2. Якщо $A \in W$, $c \in Ch$, то $Ac \in W$, де *Ac* – результат приписування символа *c* праворуч до слова *A*.

Нехай *len*(*A*) – довжина слова *A*, або кількість символів у слові *A*.

Тоді позначимо

$W_n = \{A: A \in W, len(A) \leq n\}$

Ця множина W_n і є носієм типу рядок.

У Python обмеження *n*, як і для цілих чисел, залежить тільки від об'єму доступної пам'яті.

Рядки-константи

Константи-рядки, як і було вже сказано, беруться у

- апострофи ‘ ’
- подвійні лапки “ ”
- потрійні апострофи ‘’’ ’’’
- потрійні подвійні лапки “””” “””“

Порожній рядок позначається ‘’ (або “”).

Основні операції над рядками

Операція	Опис
ord(c)	Код символа c
chr(n)	Символ з кодом n (точніше, - рядок з 1 символа з кодом n)
str(x)	Перетворення x у рядок
s + t	конкатенація s та t
s * n або n * s	n зчеплених копій s
s[i]	i-й символ s, починаючи з 0 (точніше, - рядок з 1 i-го символа), якщо i < 0, то повертає (-i) символ з кінця рядка
s[i:j]	Вирізка з s від i до j (підрядок, що починається з i –го символа та закінчується j -1 символом)
s[i:j:k]	Вирізка з s від i до j з кроком k
len(s)	довжина s
min(s)	Найменший символ рядка s
max(s)	Найбільший символ рядка s
s.index(x[, i[, j]])	Індекс першого входження x до s (починаючи з індекса i та перед індексом j)
s.count(x)	Кількість входжень x до s

Відношення для символів

Для символів визначені 6 стандартних відношень з множини

$Rel = \{==, !=, >, <, >=, <=\}$

При цьому, якщо c_1, c_2 – символи, $r \in Rel$, то

$c_1\ r\ c_2 \equiv \mathrm{ord}(c_1)\ r\ \mathrm{ord}(c_2)$

Відношення для рядків

Визначено 6 стандартних відношень з множини *Rel*.

- Відношення a == b означає попарну рівність всіх символів з двох рядків a, b.
- Відношення a < b визначається рекурсивно:
  1. Якщо a == ‘’, b != ‘’, то a < b == True
  2. Якщо b == ‘’, то a < b == False
  3. Якщо a != ‘’, b != ‘’, a[0] != b[0] то a < b ≡ a[0] < b[0]
  4. Якщо a != ‘’, b != ‘’, a[0] == b[0] то a < b ≡ a[1:] < b[1:]
- Інші відношення з множини *Rel* визначається через бульові операції та відношення == та <.

Окрім відношень з множини *Rel*, для рядків визначено ще 2 відношення:

x in a, x not in a

- де x – символ (чи рядок), a – рядок.

x in a == True, коли x входить у a

x not in a == True, коли x не входить у a

Інструкції для рядків

Визначено присвоєння, введення та виведення

```
a = e, a = input(S), print(a)
```

Визначено також цикл по всіх символах рядка з лічильником - символом

```
for x in a:
    P
```

Рядки є такими, що не змінюються (immutable). Це означає, що вже існуючий рядок змінити не можна.

Так, s = s + t створює новий рядок, який є конкатенацією s та t.

Приклади

Показати всі символи у діапазоні від *a* до *b* разом з їх кодами

Обчислити кількість входжень символа *a* у рядок *s*

Рядки як послідовності

Рядки у Python є одним з типів послідовностей.

Послідовності складаються з елементів. Для рядків цими елементами є символи.

До послідовностей відносяться також раніше розглянуті діапазони (range) та списки і кортежі, які будуть розглянуті пізніше.

Визначені вище операції для рядків (окрім ord та chr) а також відношення in та not in є спільними для всіх типів послідовностей.

Спільним також є цикл

```
for x in a:
    P
```

Вирізки

Вирізки (slices) також визначені для всіх типів послідовностей

Вирізки задають частину послідовності.

Повний формат

s[i:j:k], що означає елементи від i-го до (j-1) з кроком k

Наприклад, якщо s == ‘abcd’,

s[1:3:1] == ‘bc’, s[1:4:2] == ‘bd’

Якщо опущено k, то вважається, що k == 1. Якщо k опущено, то не вказують також другу ‘:’.

Якщо опущено i, то вважається, що i == 0.

Якщо опущено j, то вважається, що j == len(s).

Так, у попередньому прикладі s[:3] == ‘abc’, s[2:] == ‘cd’. S[:] == ‘abcd’.

k може набувати також від’ємних значень. Це означає вибір елементів послідовності справа наліво.

Наприклад, s[::-1] == ‘dcba’

Escape-послідовності

Escape-послідовності призначені для завдання спеціальних символів у рядках.

Escape-послідовність	Значення
\<новий рядок>	Ігнорується (продовження рядка на наступний)
\\	Обернена коса риска(зберігає \)
\'	Апостроф (зберігає ')
\"	Подвійні лапки (зберігає ")
\a	Дзвінок
\b	Крок назад
\f	Завершення форми
\n	Кінець рядка
\r	Повернення каретки
\t	Табуляція
\v	Вертикальна табуляція

Escape-послідовності.2

Escape-послідовність	Значення
\xhh	Символ зі значенням hh (рівно 2 цифри) у системі числення за основою 16
\ooo	Символ зі значенням ooo (до 3 цифр) у вісімковій системі числення
\0	Null: двйковий 0-символ (не завершує рядок)
\uhhhh	Символ Unicode з 16-бітним значенням у системі числення за основою 16
\Uhhhhhhhh	Символ Unicode з 32-бітним значенням у системі числення за основою 16
\<інше>	Не є escape-послідовністю (зберігає \ та <інше>)

Приклади

Перевірити, чи є рядок симетричним

Замінити всі входження у перший рядок s другого рядка с третім рядком s1

Додаткові функції для рядків

Функція	Опис
s.capitalize()	Повертає копію рядка s, у якій перший символ – велика літера, а інші – маленькі.
s.center(width[, fillchar])	Повертає s, центрований у рядку довжини width. Початок та кінець рядка заповнюються символом fillchar (за угодою - пропуск).
s.endswith(suffix[, start[, end]])	Повертає True, якщо рядок s завершується суфіксом suffix. Якщо вказано start, end, то перевіряється s[start:end]
s.expandtabs(tabsize=8)	Повертає копію рядка s, у якій всі символи табуляції (‘\t’) замінюються визначеною кількістю пропусків, в залежності від поточної позиції.
s.find(sub[, start[, end]])	Повертає найменший індекс входження sub у s. Якщо вказано start, end, то перевіряється s[start:end] Повертає -1 якщо sub не знайдено.
s.format(*args, **kwargs)	Виконує форматування рядка. Замість полів підстановки ‘{ }’ вставляються аргументи.

Додаткові функції для рядків.2

Функція	Опис
s.isalnum()	Повертає True, якщо всі символи рядка s є літерами або цифрами.
s.isalpha()	Повертає True, якщо всі символи рядка s є літерами.
s.isdigit()	Повертає True, якщо всі символи рядка s є цифрами.
s.isidentifier()	Повертає True, якщо рядок s є ідентифікатором.
s.islower()	Повертає True, якщо всі літери рядка s у нижньому регістрі
s.isnumeric()	Повертає True, якщо всі символи рядка s є числовими.
s.isprintable()	Повертає True, якщо всі символи рядка s є друкованими.
s.isspace()	Повертає True, якщо всі символи рядка s є пропусками.
s.istitle()	Повертає True, якщо рядок s є заголовком (усі слова починаються з великої літери).
s.isupper()	Повертає True, якщо всі літери рядка s у верхньому регістрі
s.ljust(width[, fillchar])	Повертає s, вирівняний по лівому краю у рядку довжини width. Кінець рядка заповнюються символом fillchar (за угодою - пропуск).
s.lower()	Повертає копію рядка s, у якій всі літери рядка s переведені до нижнього регістру

Додаткові функції для рядків.3

Функція	Опис
s.lstrip([chars])	Повертає копію рядка s, в якій ліворуч видалено символи, що входять у chars (за угодою – пропуски)
s.replace(old, new[, count])	Повертає копію рядка s, в якій всі входження рядка old замінюються new. Якщо задано count, то замінюється не більше count входжень
s.rfind(sub[, start[, end]])	Повертає найбільший індекс входження sub у s. Якщо вказано start, end, то перевіряється s[start:end] Повертає -1 якщо sub не знайдено.
s.rindex(sub[, start[, end]])	Те ж саме, що rfind(), однак дає помилку, якщо sub не знайдено.
s.rjust(width[, fillchar])	Повертає s, вирівняний по правому краю у рядку довжини width. Початок рядка заповнюються символом fillchar (за угодою - пропуск).
s.rstrip([chars])	Повертає копію рядка s, в якій праворуч видалено символи, що входять у chars (за угодою – пропуски)

Додаткові функції для рядків.4

Функція	Опис
s.strip([chars])	Повертає копію рядка s, в якій ліворуч та праворуч видалено символи, що входять у chars (за угодою – пропуски)
s.swapcase()	Повертає копію рядка s, в якій маленькі літери змінені на великі та навпаки.
s.title()	Повертає копію рядка s у форматі заголовку (усі слова починаються з великої літери).
s.upper()	Повертає копію рядка s у форматі з усіма великими літерами.
s.zfill(width)	Повертає копію рядка s, в якій зліва вставлені символи ‘0’ так, щоб загальна довжина рядка дорівнювала width. Якщо присутній знак + або -, він зберігається на початку рядка.

Приклад

Перевірити, чи є рядок присвоєнням вигляду

<змінна> = <ціле_число>

Резюме

Ми розглянули:

1. Символи. Стандарти кодування символів.
2. Unicode. Кодування символів у Unicode та формати представлення символів.
3. Рядки: носій, операції, відношення та інструкції.
4. Рядки як послідовності
5. Вирізки та їх використання.
6. Додаткові функції для рядків.

Де прочитати

1. Обвінцев О.В. Інформатика та програмування. Курс на основі Python. Матеріали лекцій. – К., Основа, 2017
2. A Byte of Python (Russian) Версия 2.01 Swaroop C H (Translated by Vladimir Smolyar), http://wombat.org.ua/AByteOfPython/AByteofPythonRussian-2.01.pdf
3. Бублик В.В., Личман В.В., Обвінцев О.В.. Інформатика та програмування. Електронний конспект лекцій, 2003 р.,
4. Марк Лутц, Изучаем Python, 4-е издание, 2010, Символ-Плюс
5. Python 3.4.3 documentation
6. https://docs.python.org/3/howto/unicode.html
7. http://unicode-table.com/ru/
8. http://www.joelonsoftware.com/articles/Unicode.html (переклад російською http://local.joelonsoftware.com/wiki/%D0%90%D0%B1%D1%81%D0%BE%D0%BB%D1%8E%D1%82%D0%BD%D1%8B%D0%B9_%D0%9C%D0%B8%D0%BD%D0%B8%D0%BC%D1%83%D0%BC,_%D0%BA%D0%BE%D1%82%D0%BE%D1%80%D1%8B%D0%B9_%D0%9A%D0%B0%D0%B6%D0%B4%D1%8B%D0%B9_%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D1%87%D0%B8%D0%BA_%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%9E%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F_%D0%9E%D0%B1%D1%8F%D0%B7%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE_%D0%94%D0%BE%D0%BB%D0%B6%D0%B5%D0%BD_%D0%97%D0%BD%D0%B0%D1%82%D1%8C_%D0%BE_Unicode_%D0%B8_%D0%9D%D0%B0%D0%BE%D1%80%D0%B0%D1%85_%D0%A1%D0%B8%D0%BC%D0%B2%D0%BE%D0%BB%D0%BE%D0%B2)
9. https://ru.wikipedia.org/wiki/%D0%AE%D0%BD%D0%B8%D0%BA%D0%BE%D0%B4