

# Програмування

---

## ТЕМА 6. СПИСКИ

# СПИСКИ

---

**Списки** у Python є послідовностями, що складаються з елементів різних типів.

Списки, на відміну від рядків, є такими, що змінюються (mutable). Тобто кількість елементів та самі елементи можуть змінюватись під час виконання програми.

# Носій типу список

---

Визначимо множину послідовностей з елементів множин  $M_1, \dots, M_k$ , яку позначимо  $Seq$ , наступним чином:

1. Порожня послідовність  $\Lambda \in Seq$
2. Якщо  $A \in Seq$ ,  $c \in M_1 \cup \dots \cup M_k$ , то  $Ac \in Seq$ , де  $Ac$  – результат приписування елемента  $c$  праворуч до послідовності  $A$ .

Нехай  $len(A)$  – довжина послідовності  $A$ , або кількість елементів у послідовності  $A$ .

Тоді позначимо

$$Seq_n = \{A: A \in Seq, len(A) \leq n\}$$

Ця множина  $Seq_n$  і є носієм типу список.

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

# Основні операції для списків

Більша частина операцій повторює аналогічні операції для рядків

Операція	Опис
$[x_1, \dots, x_n]$	Створити список з елементів $x_1, \dots, x_n$
$[]$	Порожній список
$\text{list}(x)$	Перетворення $x$ у список ( $x$ повинно належати типу, що ітерується)
$s + t$	конкатенація $s$ та $t$
$s * n$ або $n * s$	$n$ зчеплених копій $s$
$s[i]$	$i$ -й елемент $s$ , починаючи з 0, якщо $i < 0$ , то повертає $(-i)$ елемент з кінця списку
$s[i:j]$	Вирізка з $s$ від $i$ до $j$ (підсписок, що починається з $i$ -го елементу та закінчується $j-1$ елементом)
$s[i:j:k]$	Вирізка з $s$ від $i$ до $j$ з кроком $k$
$\text{len}(s)$	довжина $s$

# Основні операції для списків.2

---

Операція	Опис
<code>min(s)</code>	Найменший елемент списку <code>s</code>
<code>max(s)</code>	Найбільший елемент списку <code>s</code>
<code>s.index(x[, i[, j]])</code>	Індекс першого входження <code>x</code> до <code>s</code> (починаючи з індекса <code>i</code> та перед індексом <code>j</code> )
<code>s.count(x)</code>	Кількість входжень <code>x</code> до <code>s</code>

# Відношення для списків

---

Для списків визначено відношення 6 стандартних відношень з множини  $Rel = \{==, !=, >, <, >=, <= \}$ .

Відношення  $a == b$  означає попарну рівність всіх елементів двох списків  $a$ ,  $b$ .

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  $a < b \equiv a[0] < b[0]$
4. Якщо  $a != []$ ,  $b != []$ ,  $a[0] == b[0]$  то  $a < b \equiv a[1:] < b[1:]$

Інші відношення з множини  $Rel$  визначається через бульові операції та відношення  $==$  та  $<$ .

Для обчислення відношень з множини  $Rel$  відповідні елементи 2 списків повинні мати порівнювані типи (наприклад, обидва числові або обидва рядки і т.д.)

Обчислення відношень для непорівнюваних типів дає помилку.

# Відношення для списків.2

---

Окрім відношень з множини *Rel*, для списків визначено відношення:

$x \text{ in } a$ ,  $x \text{ not in } a$

- де  $x$  – елемент,  $a$  – список.

$x \text{ in } a == \text{True}$ , коли  $x$  входить у  $a$

$x \text{ not in } a == \text{True}$ , коли  $x$  не входить у  $a$

# Інструкції для списків

---

Для списків визначено присвоєння та виведення.

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

Введення не визначено, тому треба вводити список поелементно.

Визначено також цикл по всіх елементах списку

```
for x in a:
```

*P*

Окрім цього, визначено ще ряд інструкцій



# Інструкції для списків.2

Інструкція	Опис
<code>s[i] = x</code>	<code>i</code> –й елемент <code>s</code> замінюється на <code>x</code>
<code>s[i:j] = t</code>	вирізка <code>s</code> від <code>i</code> до <code>j</code> замінюється значенням <code>t</code> ( <code>t</code> повинно належати типу, що ітерується)
<code>del s[i:j]</code>	Те ж саме, що і <code>s[i:j] = []</code>
<code>s[i:j:k] = t</code>	елементи <code>s[i:j:k]</code> замінюються відповідними елементами <code>t</code> (кількість елементів <code>s[i:j:k]</code> та <code>t</code> повинна бути однаковою)
<code>del s[i:j:k]</code>	Видаляє елементи <code>s[i:j:k]</code> зі списку
<code>s.append(x)</code>	приписує <code>x</code> у кінець списку (те ж саме, що й <code>s[len(s):len(s)] = [x]</code> )
<code>s.sort()</code>	Впорядковує список <code>x</code> за неспаданням

# Реалізація масивів на базі списків

---

Списки можуть бути використані для реалізації масивів.

**Масиви** – це багатовимірні таблиці, які складаються з однотипних елементів.

Елемент масиву доступний за його індексами.

Кількість індексів дорівнює кількості вимірів таблиці.

Одновимірний масив ще називають вектором, а двовимірний – матрицею.

Наприклад,

$v = [1, 3, 7, 11]$

- $v$  - вектор з 4 цілими елементами.  $v[1] == 3$

$a = [[0.5, 2.0, -3.2], [2.1, 4.6, 0.1]]$

- $a$  – матриця  $2 \times 3$  з дійсних чисел.  $a[1][2] == 0.1$

# Приклади

---

Обчислити скалярний добуток векторів  $a$ ,  $b$ , що складаються з  $n$  дійсних компонент.

Обчислити значення мінімального елемента матриці  $m \times n$

# Взаємозв'язок рядків та списків

---

Python має потужні засоби, які зв'язують рядки та списки.

Використовуючи ці засоби, можна легко розв'язувати задачі, пов'язані з аналізом текстів.

Розглянемо декілька операцій над рядками, в яких використовуються списки.

# Взаємозв'язок рядків та списків.2

Операція	Опис
<code>s.join(t)</code>	Повертає рядок з усіма елементами <code>t</code> , між якими в якості розділювача стоїть рядок <code>s</code>
<code>s.split(sep=None, maxsplit=-1)</code>	Повертає список, у якому рядок <code>s</code> розбито на підрядки рядками-розділювачами <code>sep</code> . Якщо <code>sep</code> не вказано, то мається на увазі рядок з довільної кількості пропусків. Якщо вказано <code>maxsplit!= -1</code> , то виконується рівно <code>maxsplit</code> розбиттів.
<code>s.splitlines([keepends])</code>	Повертає список рядків, який є розбиттям <code>s</code> на підрядки, обмежені символами кінця рядка ( <code>\n</code> , <code>\r</code> або <code>\r\n</code> ).
<code>s.rsplit(sep=None, maxsplit=-1)</code>	Діє як <code>split</code> , але при вказанні <code>maxsplit!= -1</code> рахує розбиття не від початку, а від кінця рядка <code>s</code>

# Приклади

---

Словами будемо називати послідовності символів рядка, які не містять пропусків всередині та розділені пропусками. Обчислити кількість слів у рядку.

Видалити повторні входження слів у рядку та показати всі різні слова рядка у алфавітному порядку.

# Додаткові інструкції для списків

Інструкція	Опис
<code>s.clear()</code>	Вилучає всі елементи з <code>s</code> (те ж саме, що й <code>del s[:]</code> )
<code>s.copy()</code>	Робить копію <code>s</code> (те ж саме, що й <code>s[:]</code> )
<code>s.extend(t)</code>	розширює <code>s</code> значенням <code>t</code> (те ж саме, що й <code>s[len(s):len(s)] = t</code> )
<code>s.insert(i, x)</code>	вставляє <code>x</code> у <code>s</code> у позицію, задану <code>i</code> (те ж саме, що й <code>s[i:i] = [x]</code> )
<code>s.pop([i])</code>	Повертає елемент з позиції <code>i</code> а також вилучає його зі списку <code>s</code> (за угодою <code>i == len(s)-1</code> , тобто буде вилучений останній елемент)
<code>s.remove(x)</code>	Видаляє перший такий елемент <code>s</code> , для якого <code>s[i] == x</code>
<code>s.reverse()</code>	Переставляє елементи <code>s</code> у зворотному порядку

# Спискоутворення

---

Спискоутворення (list comprehension) – це вираз, результатом якого є список.

Вираз має такий синтаксис:

`[ $e(x)$  for  $x$  in  $t$  if  $F$ ]`

- де  $e(x)$  – вираз, який залежить від  $x$ ;  $t$  – вираз типу, що ітерується,  $F$  – умова.

Python вибирає всі  $x$  з  $t$ , які задовольняють умову  $F$ , застосовує до кожного  $x$  вираз  $e(x)$  та повертає отриманий список.

Якщо умова  $F$  відсутня, то `if  $F$`  опускають.



# Функція enumerate

---

Функція `enumerate(s)` застосовується для того, щоб окрім елементів списку `s` можна було у циклі `for` аналізувати індекси цих елементів.

Ця функція повертає пари (індекс, елемент).

Наприклад:

```
for i, c in enumerate(s):  
    if i == c:  
        print (i)
```

# Приклади

---

Обчислити значення мінімального елемента матриці  $m \times n$  (версія 2)

Видалити повторні входження слів у рядку та показати всі різні слова рядка у алфавітному порядку (версія 2)

# Резюме

---

Ми розглянули:

1. Списки: носій, операції, відношення та інструкції для списків.
2. Реалізацію масивів на базі списків
3. Взаємозв'язок між рядками та списками
4. Спискоутворення. Функцію `enumerate`

# Де прочитати

---

1. Обвінцев О.В. Інформатика та програмування. Курс на основі Python. Матеріали лекцій. – К., Основа, 2017
2. A Byte of Python (Russian) Версія 2.01 Swaroop С Н (Translated by Vladimir Smolyar),  
<http://wombat.org.ua/AByteOfPython/AByteofPythonRussian-2.01.pdf>
3. Марк Лутц, Изучаем Python, 4-е издание, 2010, Символ-Плюс
4. Python 3.4.3 documentation
5. Бублик В.В., Личман В.В., Обвінцев О.В.. Інформатика та програмування. Електронний конспект лекцій, 2003 р.,