Python for Data Science

Лекція 6. Вступ до алгоритмів

Зміст уроку

- Визначення алгоритму
- Способи представлення алгоритмів: псевдокод, діаграми, блок-схеми
- Поняття складності алгоритмів
- Рекурсія
- Алгоритми пошуку

Визначення алгоритму

Алгоритм – це набір кінцевого числа правил, що задають послідовність виконання операцій для вирішення задачі певного типу.

Дональд Ервін Кнут

Алгоритм – це зрозуміла і точна вказівка для виконавця здійснити послідовність дій, направлених на досягнення вказаної мети або на вирішення поставленого завдання.

А. П. Єршов

Алгоритм – набір інструкцій, які описують порядок дій виконавця, щоб досягти результату розв'язання задачі за скінченну кількість дій; система правил виконання дискретного процесу, яка досягає поставленої мети за скінченний час.

Вікіпедія

Критерії алгоритму

Кінцевість – алгоритм завжди повинен закінчуватись, після виконання кінцевого набору кроків.

Визначеність – дії, які необхідно виконати на кожному кроці повинні бути чітко визначені.

Введення – алгоритм повинен мати перелік вхідних даних, тобто величин, які потрібні для його роботи.

Вивід – те, що ми отримуємо, після того, як ми виконаємо усі кроки алгоритму.

Ефективність – алгоритм повинен бути настільки простим, щоб його кроки можна було б описати словами або зобразити у вигляді схеми.

Способи представлення алгоритму

- Псевдокод
- Діаграма Нассі-Шнейдермана
- Блок-схема

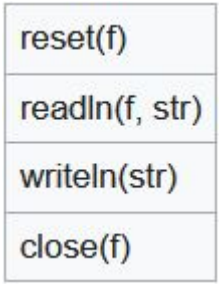
Способи представлення алгоритму – Псевдокод

Псевдокод – це неформальна нотація природною мовою, що описує роботу алгоритму, методу, класу або програми. Це простий запис людською мовою, у вигляді коментарів з поясненнями, що нам потрібно зробити.

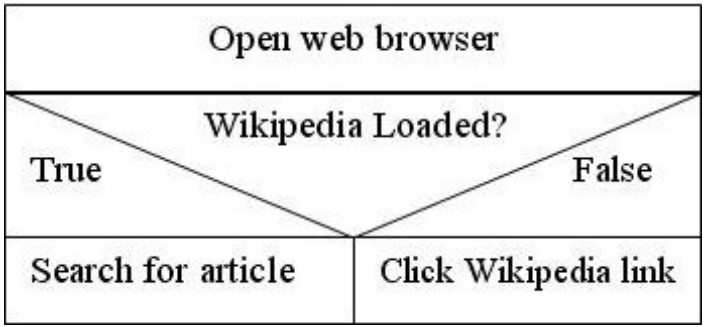
```
# 1. Отримати з консолі введення
   # 1.1. перевірити, чи є отримане введення числом
       # a) якщо ТАК – присвоїти отримане введення змінній,
       #     my num, привівши її до типу integer
       # b) якщо НІ – вивести повідомлення про помилку і
       #     повернутись до пункту 1.
…
```

Способи представлення алгоритму – діаграма Нассі-Шнейдермана

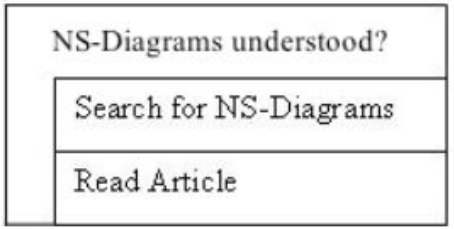
Діаграма Нассі-Шнейдермана – це графічний спосіб представлення структурованих алгоритмів і програм, з використанням пов'язаних блоків і без використання стрілок.



Проста послідовність



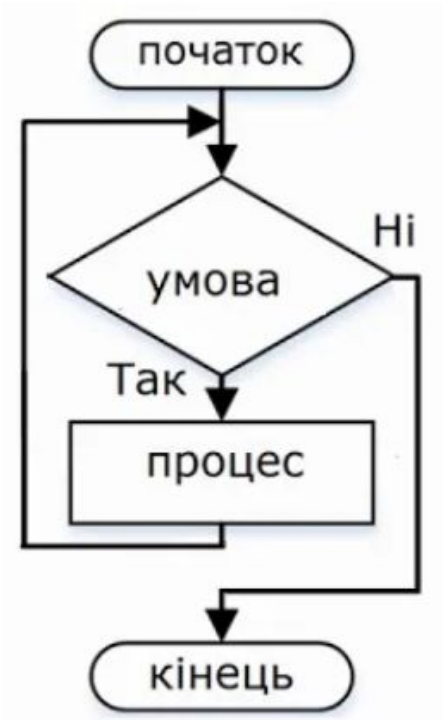
Розгалуження (конструкція if-else)



Повтор з передумовою (цикл while)

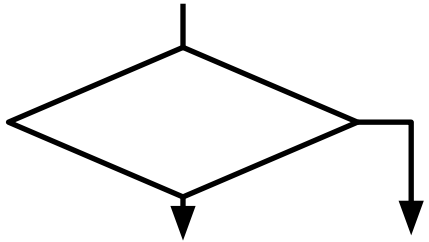
Способи представлення алгоритму – блок-схема

Блок-схема – це спосіб представлення графічних моделей, що описують алгоритми або процеси, в яких окремі кроки зображені у вигляді блоків умовної форми, об’єднаних між собою лініями, що вказують напрямок послідовності виконання.

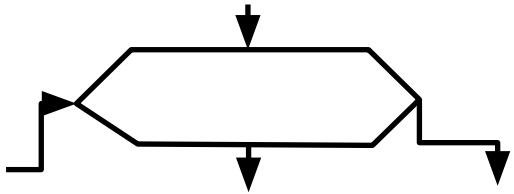


Елементи блок-схеми алгоритму

Початок і кінець програми, або вхід і вихід у підпрограмах

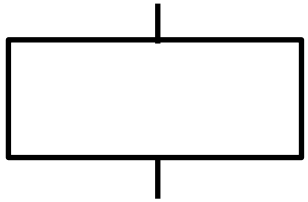


Вирішення – перевірка умови:
вибір одного або декількох напрямків виконання алгоритму



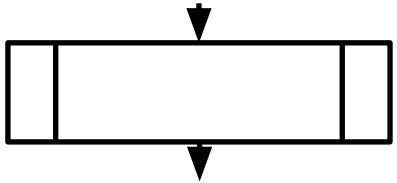
Модифікація – організація циклічних конструкцій

Елементи блок-схеми алгоритму (продовження)



Процес – формування нових значень, виконання арифметичних або логічних операцій, результат яких зберігається у оперативні пам’яті

Дані – елементи введення або виводу



Попередньо визначений процес – підпрограма, процедура або функція

Поняття складності алгоритмів

Складність алгоритму – це кількісна характеристика, яка говорить про те, скільки часу, або який об’єм пам'яті потрібно для виконання алгоритму.

Складність алгоритму вимірюється у **елементарних кроках** – кількості дій, необхідних для його виконання. Будь-який алгоритм включає в себе певну кількість кроків, необхідних для свого виконання, незалежно від пристрою, на якому цей алгоритм виконується.

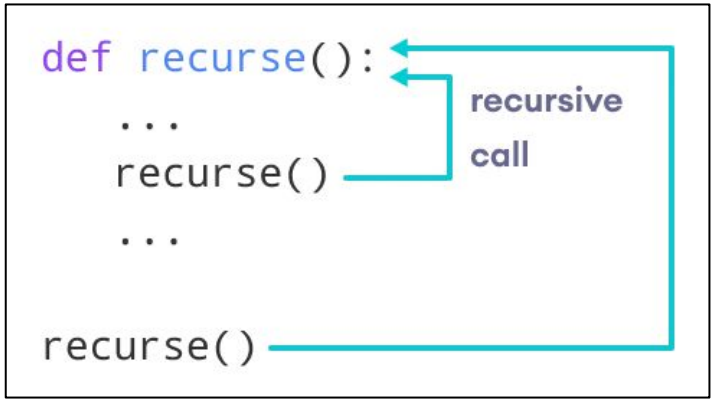
Таку характеристику представляють у вигляді показника **Big O**, який демонструє верхню межу залежності між вхідними параметрами функції і кількістю операцій, які виконуватиме процесор.

Поширені види складності алгоритму

Назва	Індекс	Визначення
Константна	O(1)	Обчислювальна складність не залежить від вхідних даних
Лінійна	O(n)	Складність алгоритму лінійно зростає зі збільшенням вхідних даних
Логарифмічна	O(log n)	Складність алгоритму зростає логарифмічно зі збільшенням вхідних даних
Лінеарифметична	O(n * log n)	Подвоєння розміру вхідних даних збільшить час виконання трохи більше, ніж у двічі
Квадратична	O(n2), O(n**2)	Подвоєння розміру вхідних даних збільшить час виконання у 4 пази

Рекурсія

Рекурсія — це фундаментальна концепція математики та комп'ютерних наук. В мовах програмування рекурсивна функція — це функція, яка здатна викликати сама себе. Рекурсивна функція не може визивати себе до нескінченності.
Отже, друга важлива особливість рекурсивної функції — це наявність умови завершення, яка дозволяє припинити виклики.



Алгоритми пошуку

- Оператори членства
- Лінійний пошук (L)
- Бінарний пошук (B)
- Пошук стрибка (B)
- Пошук Фібоначі (B)
- Експоненціальний пошук (B)
- Інтерполяційний пошук (B)

Завдання

- Написати рекурсивну функцію, яка б вертала ряд Фібоначі у вигляді списку цифр. Функція повинна мати вхідний цифровий параметр максимального значення у списку.

- Написати функцію, яка б вираховувала кількість від’ємних чисел у списку [20, -33, 16, 21, -5, -66, 74, -3, 27, 87, -4].

- Написати функцію визначення максимального елементу списку.

Завдання

- Написати функцію, яка б здійснювала реверсування цілого числа. Наприклад: число 298276534 перетворювалося б на 435672892.

- Написати рекурсивну функцію, яка вертає суму чисел, що є елементами списку.