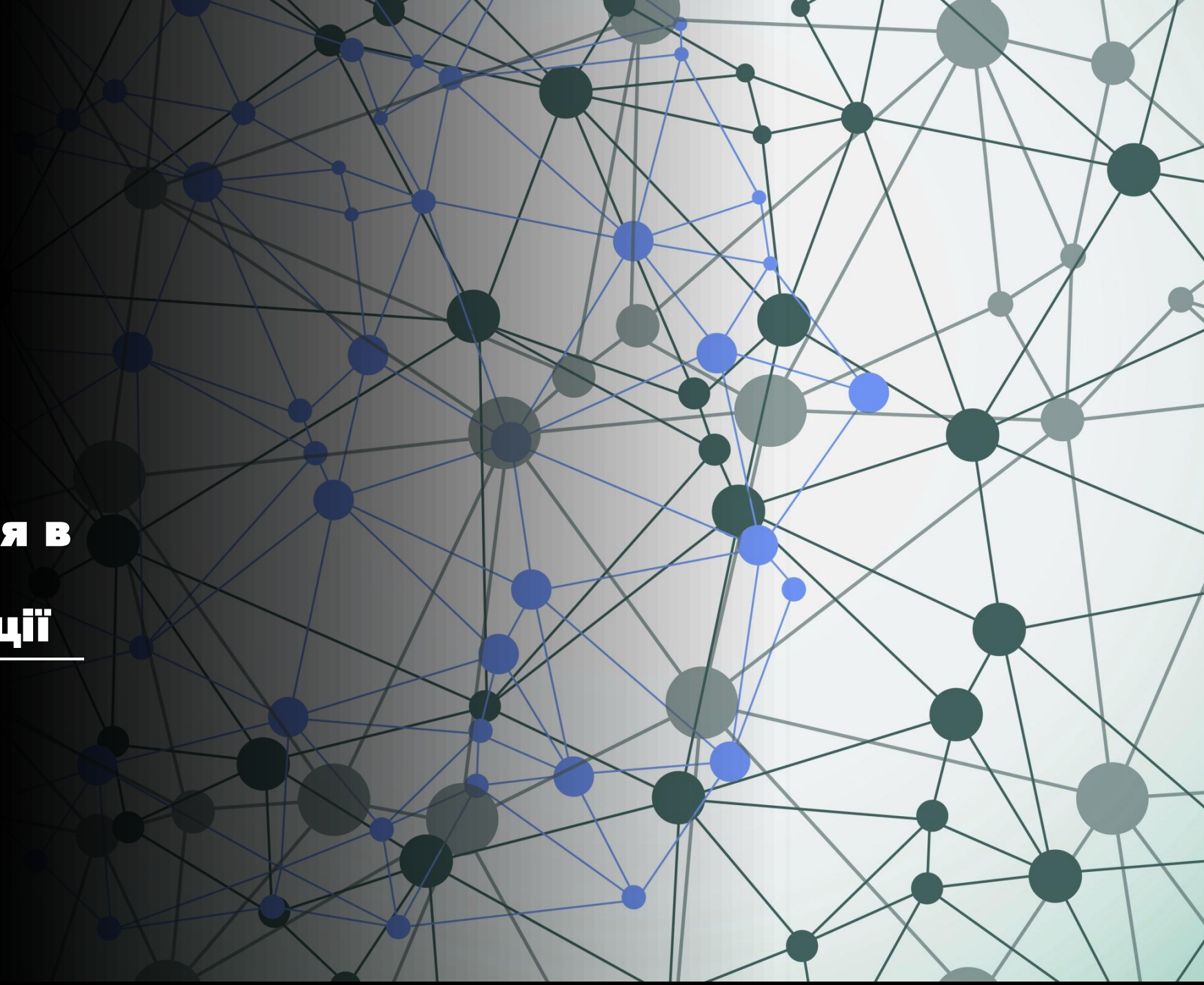
Статична типізація в PYTHON та шляхи її реалізації



Катична типізація в PYTHON.

Качина типізація - різновид динамічної типізації, застосовуваної в деяких мовах програмування, коли межі використання об'єкта визначаються його поточним набором методів і властивостей, на противагу успадкуванню від певного класу. Тобто вважається, що об'єкт реалізує інтерфейс, якщо він містить всі методи цього інтерфейсу, незалежно від зв'язків в ієрархії наслідування та приналежності до якогось конкретного класу.

Система типізації

- **Система типізації в програмуванні визначає множину припустимих значень даних та множину операцій над даними. Тип даних означає множину значень, які мають певне спільне значення або спосіб застосування (хоча, деякі типи, такі як абстрактні або функціональні типи, можуть і не представляти значень в програмі). Системи типізації різних мов програмування істотно різняться одна від одної. Найістотніші відмінності полягають в реалізаціях компіляції та поведінки під час виконання програми.**

Базові відомості

- Базові відомості
- Присвоєння типу даних (типізація) надає значення набору бітів. Як правило, типи надаються або значенням в пам'яті, або об'єктам, таким, як змінні. Оскільки будь-яке значення в комп'ютері складається із множини бітів, апаратне забезпечення не розрізняє навіть адресу, коди операцій, символьні дані, цілі числа. Типи вказують програмам та програмістам на те, як слід обробляти дані.
- Системи типізації виконують наступні функції:
- Безпечність — застосування типів даних дозволяє компілятору знаходити беззмістовний або неправильний код. Наприклад, можна визначити вираз `"Привіт!" + 3` як неправильний, оскільки додавання (в загальному розумінні) рядка до цілого числа не має сенсу. Як зазначено нижче, сильна типізація безпечніша, однак вона не обов'язково гарантує повну безпеку (докладніше, дивіться безпечна типізація).
- Оптимізація — статична перевірка типів може повідомити додаткову інформацію компілятору. Наприклад, якщо тип даних вказує на те, що значення повинні вирівнюватись на границі кратні 4, компілятор зможе використати ефективніші машинні інструкції.
- Документування — у виразніших системах типізації, типи даних можуть служити як вид документації, оскільки вони можуть описувати наміри розробника. Наприклад, довжина може бути підтипом цілих чисел, але, якщо розробник декларує тип результату функції як довжину, а не просто ціле число, це може частково описувати значення функції.
- Абстрагування (або модульність) — типи даних дозволяють розробнику розмірковувати про програми на вищому рівні, не звертаючи увагу на деталі реалізації на нижчому рівні. Наприклад, розробник може вважати рядок значенням, замість простого масиву байт. Або, типи можуть дозволити розробникам виражати інтерфейс між двома підсистемами. Це локалізує необхідні для взаємодії двох підсистем визначення та запобігає появі несумісностей під час взаємодії цих підсистем.