ЖИТОМИРСЬКА ПОЛІТЕХНІКА
ДЕРЖАВНИЙ УНІВЕРСИТЕТ

@ZTUEDUUA

АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРИЗОВАНИХ ІНФОРМАЦІЙНО-ВИМІРЮВАЛЬНИХ СИСТЕМ

Лекція 3
Тема: Математичні обчислення на мові Python

1. Основні математичні операції зі змінними.
2. Математичні функції.
3. Логічні оператори в Python.

python™

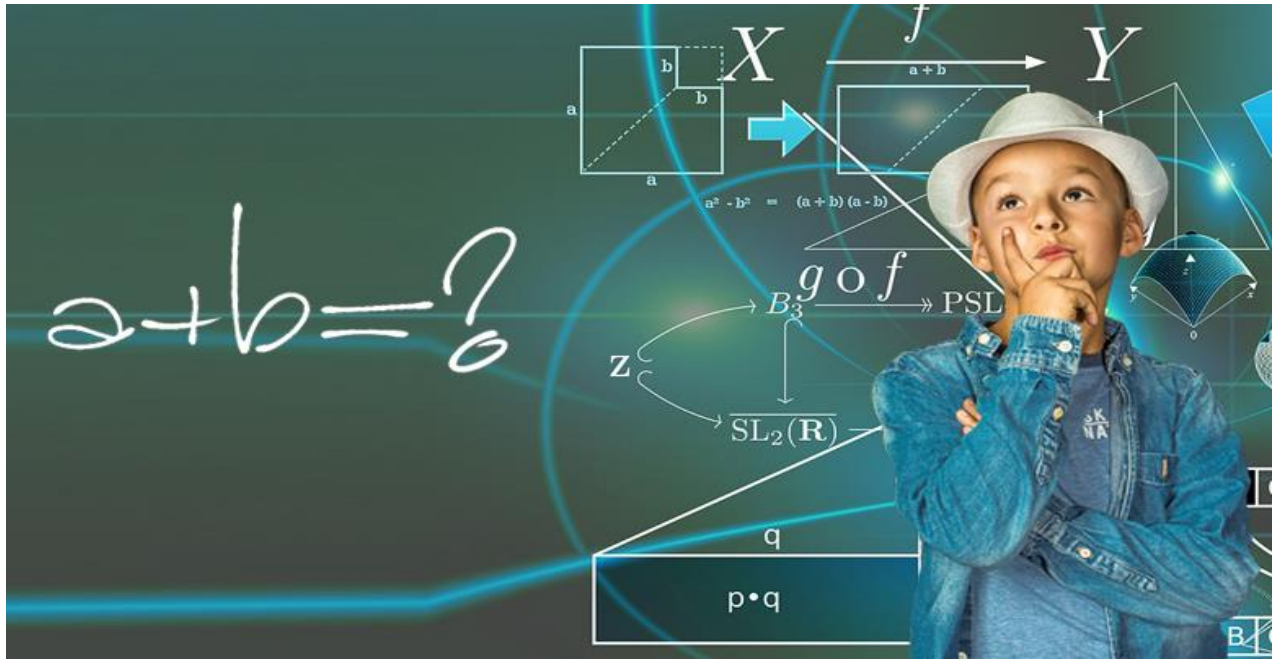
1. Основні математичні операції зі змінними

У цьому пункті краще дізнаємося про числа та різні математичні функції, які доступні в Python.

Говорячи про числа, пройдемося по найчастіше використовуваних математичних операторів, які потрібні для виконання простих арифметичних операцій з числами в Python. Також пізніше в розділі познайомимося з функціями, які будуть дуже корисні при обчисленні складних математичних виразів: зведення у ступінь, синус/косинус, факторіали тощо.

У Python є 6 основних «математичних» операторів:

- додавання;
- віднімання;
- множення;
- ділення;
- зведення в ступінь;
- залишок від поділу (mod);
- цілісний поділ (div).



Більшість має бути знайома з усіма вищепереліченими операторами, крім оператора взяття залишку від поділу та цілого чисельного поділу.

Додавання

- Неважко здогадатися, що робить цей оператор: він складає числа. Щоб перевірити, як це працює, просто перейдіть в IDLE і введіть число, потім знак додавання + (плюс), а потім ще одне число, яке потрібно додати до першого числа. Натисніть клавішу Enter. Це має виглядати так, як показано нижче.

Наприклад: складемо 8 та 19.

```
>>> 8+19
27
```

Після натискання нами клавіші Enter, можемо побачити, що трохи нижче рядка коду з'явиться відповідь. Висновок так і відображатиметься щоразу — трохи нижче за код. Натискаєте Enter – отримуєте результат.

Віднімання

- У віднімання такий самий синтаксис, як і у додавання. Просто змініть оператор на знак віднімання - (мінус). Також виберіть випадкові числа і спробуйте відняти одне з іншого.

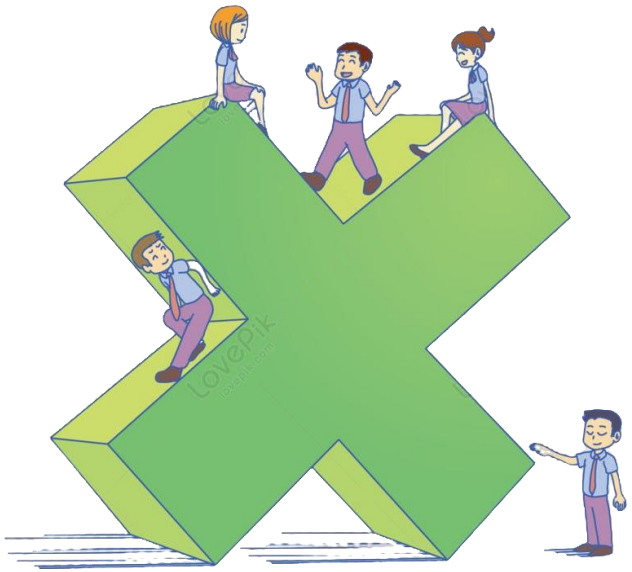
Наприклад: віднімемо від 89.33 число 23.67.

```
>>> 89.33-23.67
65.66
```

Можемо побачити, що знову відповідь була написана після самого виразу, а саме 65.66

Множення

- Знову те саме! Просто змініть оператор на * (зірочку). Адже ви знаєте, що він використовується для множення, вірно? Спробуйте використовувати оператор IDLE.

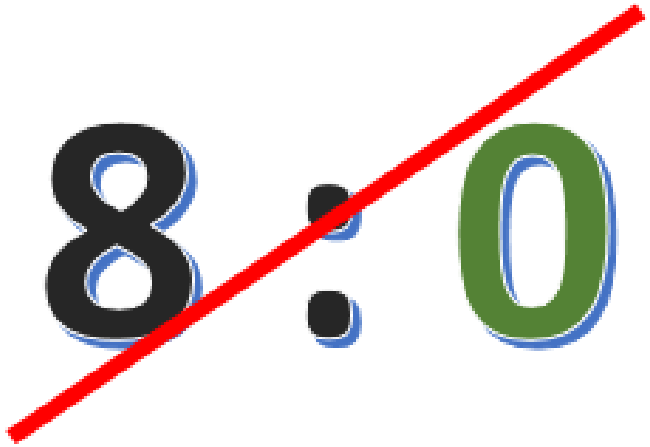


Наприклад: Візьміть будь-які два числа і помножте їх за допомогою оператора множення, як показано нижче.

```
>>> 56.92*23.23
1322.2516
```

Ділення

- Цього разу нам знадобиться знак / (слеш). Спробуйте використати оператор із випадковими числами.



Візьмемо цілі числа (числа без десяткового дробу), наприклад 16 та 2, і розділимо одне на інше.

```
>>> 16/2
8.0
```

Зведення в ступінь

- **Цього математичного оператора зазвичай мовами програмування немає. В інших мовах для зведення у ступінь використовують інші оператори. У Python для цього достатньо поставити між двома числами ** (дві зірочки). Праве число зведеться в ступінь, що дорівнює лівому числу. Наприклад, щоб знайти 10 ступінь числа 2, потрібно написати:**


```
>>> 2**10
1024
```

Залишок від ділення (mod);

Оператор взяття залишку від ділення в Python позначається % (знак відсотка). Якщо вам знайомі інші мови програмування, швидше за все ви знаєте, що таке взяття залишку від ділення. В інших мовах цей оператор часто називають mod.

У IDLE так само:

```
>>> 12%2
0
>>> 13%2
1
>>> 19%5
4
```

Адже ви знаєте оператор ділення, так? Тоді ви знаєте, який буде залишок від цього ділення,правда? Цей оператор якраз і повертає цей залишок як відповідь. Ціла частина хіба що відкидається.

Ось кілька прикладів:

- **12%2 = 0, оскільки 12 повністю поділяється на 2.**
- **13%2 = 1, оскільки залишок від розподілу 13 на 2 дорівнює 1.**
- **19%5 = 4, оскільки, знову ж таки, залишок від розподілу 19 на 5 дорівнює 4.**

Цілісне ділення (div)

Цей оператор Python позначається знаком // (двома слешами), в інших мовах його називають div. Оператор відкидає залишок і залишає цілу частину. Ось як це працює:

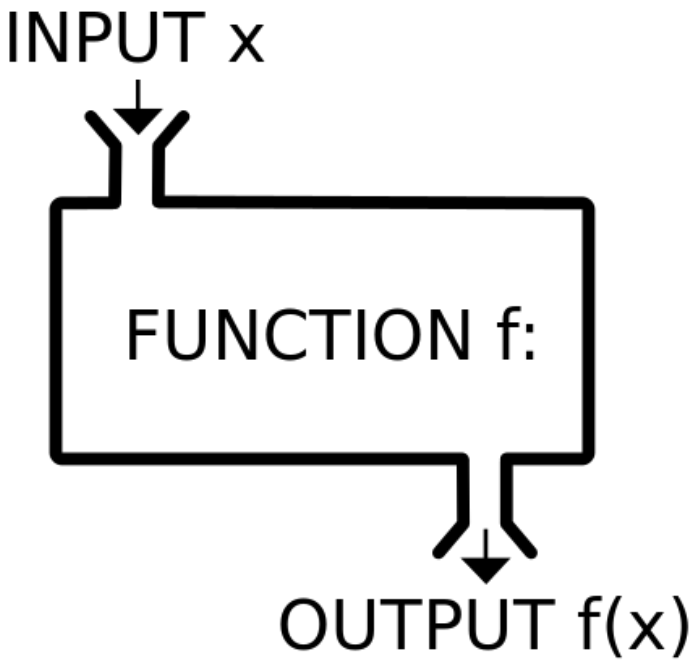
- **13//2 = 6, оскільки 13/2 = 6.5**
- **19//5 = 3, оскільки 19/5 = 3.8**

2. Математичні функції

Коли дізнаємося про Python більше, можливо, захочеться створити «науковий калькулятор» як проект. Для цього крім простих математичних операцій вам доведеться реалізувати математичні функції: тригонометричні, логарифмічні і так далі. Але навіть якщо забути про створення калькулятора, в житті програміста достатньо ситуацій, коли вам можуть знадобитися ці функції.

Python вже написаний код практично для всіх математичних функцій і додано до бібліотеки. Так що вручну писати функції не доведеться, можна без сором'язливості використовувати готові рішення.

Функція - це фрагмент коду, який приймає (або не приймає) як вхідні дані деякі значення, обробляє їх, а потім повертає (або не повертає) певне значення як вихідні дані.

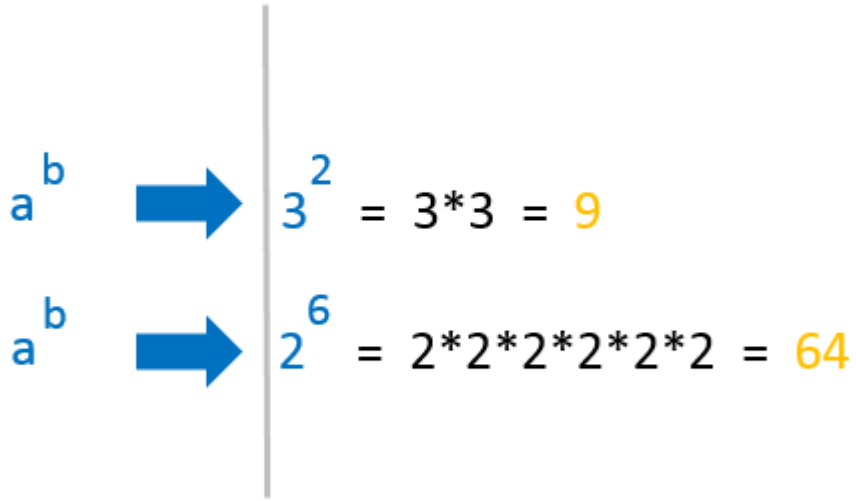


Як можна побачити на малюнку, на вхід функції подається **x**, а на виході функція **f** виводить значення **f(x)**. Загалом функції не обов'язково приймати щось на вхід або виводити щось. Але для математичної функції важливо і те, й інше. Наприклад, щоб обчислити **sin(x)** обов'язково потрібне значення **x**.

- У Python є два типи функцій.
- **Вбудовані функції** — це функції, для яких не потрібні жодні сторонні файли коду (вони ж модулі або бібліотеки). Вони є частиною Python і генеруються в компіляторі Python, тому нічого імпортувати для їх використання не потрібно.
- **Функції користувача** вимагають зовнішніх файлів, їх називають модулями. Використання цих зовнішніх файлів у коді називається імпортом. Все, що вам потрібно зробити, щоб використовувати функції з бібліотеки, це імпортувати їх у ваш код.

Зведення в ступінь

- **Оскільки це вбудована функція, імпортувати бібліотек не потрібно. На вхід функція pow(x, y) приймає два числа - основу та показник ступеня.**



Тут проаналізуємо, що написано та що зробить Python. Спочатку введено pow – це просто ім'я функції, яку намагаємось викликати. Це вкаже компілятор Python знайти вбудовану функцію з ім'ям pow і визначити, що вона може робити. Далі в дужках написано два числа через кому: 3 і 2. Перше число - 3 - основа, а друге - 2 - показник ступінь. Інакше кажучи, намагаємося звести 3 до другого ступеня

```
>>> pow(3,2)
9
```

Модуль

- **Функція модуля повертає невід'ємне значення аргументу. Інакше висловлюючись, вона змінює невід'ємні значення, а невід'ємні значення робить додатніми.**

Наприклад: модуль -3 дорівнює 3 модуль -8.74 дорівнює 8.74 і так далі.

```
>>> abs(-3)
3
>>> abs(-8.74)
8.74
```

Тригонометричні функції

- Однією з характерних переваг модуля **math()** є те, що він підтримує тригонометричні функції. Підтримуються всі існуючі зараз – синус, косинус, тангенс, арксинус, арктангенс тощо.
- Найпоширеніші тригонометричні функції всі приймають лише один аргумент - кут, для якого необхідно знайти синус, косинус і таке інше.

- Отримання синусу радіана. І тому використовується функція **sin()**. Приймає один аргумент – **радіан.**
- Отримання косинуса. Щоб досягти цієї мети, застосовується функція **cos()**. Як аргумент використовується радіан.
- Отримання тангенсу. Використовується функція **tan**.

Функція	Значення
Sin	Приймає радіан і повертає його синус
Cos	Приймає радіан і повертає його косинус
Tan	Приймає радіан і повертає його тангенс
Asin	Приймає один параметр і повертає арксинус
Acos	Приймає один параметр і повертає арккосинус
Atan	Приймає один параметр і повертає арктангенс
SinH	Приймає один параметр і повертає гіперболічний синус
cosH	Приймає один параметр і повертає гіперболічний косинус
TanH	Приймає один параметр і повертає гіперболічний тангенс
AsinH	Приймає один параметр і повертає зворотній синус
AcosH	Приймає один параметр і повертає зворотній косинус
AtanH	Приймає один параметр і повертає зворотній тангенс

Функції зведення в ступінь та логарифма

- **exp() - Функція приймає ціле число або число з точкою, що плаває, а повертає e у відповідній мірі. Наприклад:**

```
print(«e у ступені 5 «, math.exp(5))
print(«e у ступені 2.5», math.exp(2.5))
```

- **expm1() - Есть еще одна причина: точность результатов. Если значение x меньше 10, то эта функция дает лучше результат по сравнению с exp()-1. Приклад функції:**

```
print(math.exp(5)-1)
print(math.expm1(5))
```

- **log() - логарифм числа - Це ступінь, у якому треба звести основу, щоб отримати аргумент, тобто. функція від двох змінних.**

- **log1p()** - Загалом ця функція працює так само, як і попередня. Її принцип такий самий. Єдиний виняток – вона додає x одиницю. Для використання цієї функції, так само, як і у всіх попередніх прикладах, необхідно спочатку імпортувати модуль math, а потім використовувати як аргумент число і базу.
  Приклад функції: `import math`
  `print(math.log1p(2))`
- **log10()** - Эта функция, как и все остальные, используется, как метод модуля math. Соответственно, его нужно вызывать с этим объектом, как положено, через точку. Ця функція, як і решта, використовується, як метод модуля math. Відповідно, його потрібно викликати з цим об'єктом, як ведеться, через точку. Приклад функції:
  `print(math.log10(1000))`

sqrt() — квадратний корінь числа

- **Math.sqrt() – це єдина функція з допомогою якої можна зробити операцію зведення ступінь. Щоб глибше зрозуміти принципи її роботи, треба порівняти з іншими способами отримання квадратного кореня з числа.**
- **За допомогою math.sqrt() можна отримати квадратний корінь із нуля та позитивних чисел. Для цього необхідно як аргумент використовувати число, для якого потрібно отримати квадратний корінь.**
- **Недолік цього у тому, що не підтримує вилучення квадратного кореня з негативних чисел. Щоб така можливість з'явилася, необхідно підключити іншу бібліотеку cmath() і використовувати метод cmath.sqrt() з числом як аргумент.**

Математичні константи

- **Існують дві основні математичні константи. Про одну з них ми вже сьогодні говорили. Це число Ейлер. Воно позначається літерою e. Щоб отримати його, необхідно використовувати функцію math.e (без аргументів).**
- **Для отримання числа Пі, яка є другою відомою константою, використовується функція math.pi, яка також використовується без аргументів. Приклад функції:**

```
# виведення значення PI
print(«значення PI», math.pi)
# виведення значення e
print(«значення e», math.e)
```

3. Логічні оператори в Python.

Логічні оператори	
Назва	**Запис в Python**
Кон'юнкція	and
Диз'юнкція	or
Заперечення	not
Імплікація	<=
Еквівалентність	==

- **Кон'юнкція (логічне «і»)** – даний оператор перевіряє, щоб усі функції у виразі були дійсними.
- Він працює за принципом множення: якщо хоч один елемент дорівнює нулю, то весь твір дорівнює нулю.

На картинці зображений торт, про який ми можемо сказати:"Торт прикрашений полуницею" і "на торті 3 свічки" - такий вислів буде істинним.

Але якщо скажемо: «Торт прикрашений полуницею» та «на торті 6 свічок» – такий вираз буде хибним.

- **Диз'юнкція (логічне «або»)** – логічний оператор, який перевіряє, що хоча б один елемент із виразу істинний
- Він за способом роботи схожий на додавання: якщо серед елементів є одиниця, то вираз буде істинним.

- Тут можна буде скласти такий вираз:"Торт прикрашений полуницею" або "на торті всі свічки сині" - такий вислів буде істинним, тому що серед елементів є вірний.Із запереченням все працює ще простіше.

- **Заперечення** – логічний оператор, який змінює значення елемента протилежне, тобто змінює істину на брехню, і навпаки.

- У прикладі з тортом можна сказати так:«На торті не 6 свічок» - це вираз істинно, адже ми заперечуємо хибне висловлювання.

- **Імплікація (слідування)** – логічний оператор, який перетворює вираз на хибне тільки у випадку, коли з істини випливає брехня.

- **Еквівалентність** – логічний оператор, який перевіряє, щоб обидва елементи були або хибними, або істинними.(Тобто якщо перший елемент дорівнює другому, то вираз буде дійсним)

Як дані оператори працюють в Python

- **завжди спочатку виконується заперечення, потім кон'юнкція, диз'юнкція, імплікація та еквівалентність. У програмі він буде наступним:**

1) **Еквіваленція та імплікація матимуть рівний пріоритет та виконуватимуться в порядку черги. Але щодо логічних операторів їхній пріоритет стане найвищим.**
2) **Інверсія, кон'юнкція та диз'юнкція матимуть свій законний пріоритет.**

- Як розставляти **пріоритети**?
- Як знаєте, вміння правильно розставляти пріоритети просто необхідне у звичайному житті, але, виявляється, воно також необхідне і в інформатиці, зокрема в алгебрі логіки.Щоб отримати правильну відповідь при вирішенні логічного виразу, **потрібно розставляти дужки. Це обов'язково пам'ятати, особливо при використанні математичних операторів в Python.**

Але є одне **«але»**, пов'язана з останніми двома операторами. Вона полягає в тому, що для **імплікації** та **еквіваленції** немає спеціальних логічних операторів, але для них можна використовувати математичні:

- Математичне порівняння на рівність працює так само, як логічна еквіваленція: поверне True, якщо значення будуть однакові і False інакше.

- Математичне «менше або одно» повністю відповідає логічному дотриманню: False буде повернено лише в тому випадку, якщо значення зліва буде більше значення праворуч. А якщо пригадати аналогію логічних змінних і цілих чисел, це станеться лише в ситуації 1 <= 0. У решті випадків буде істина.

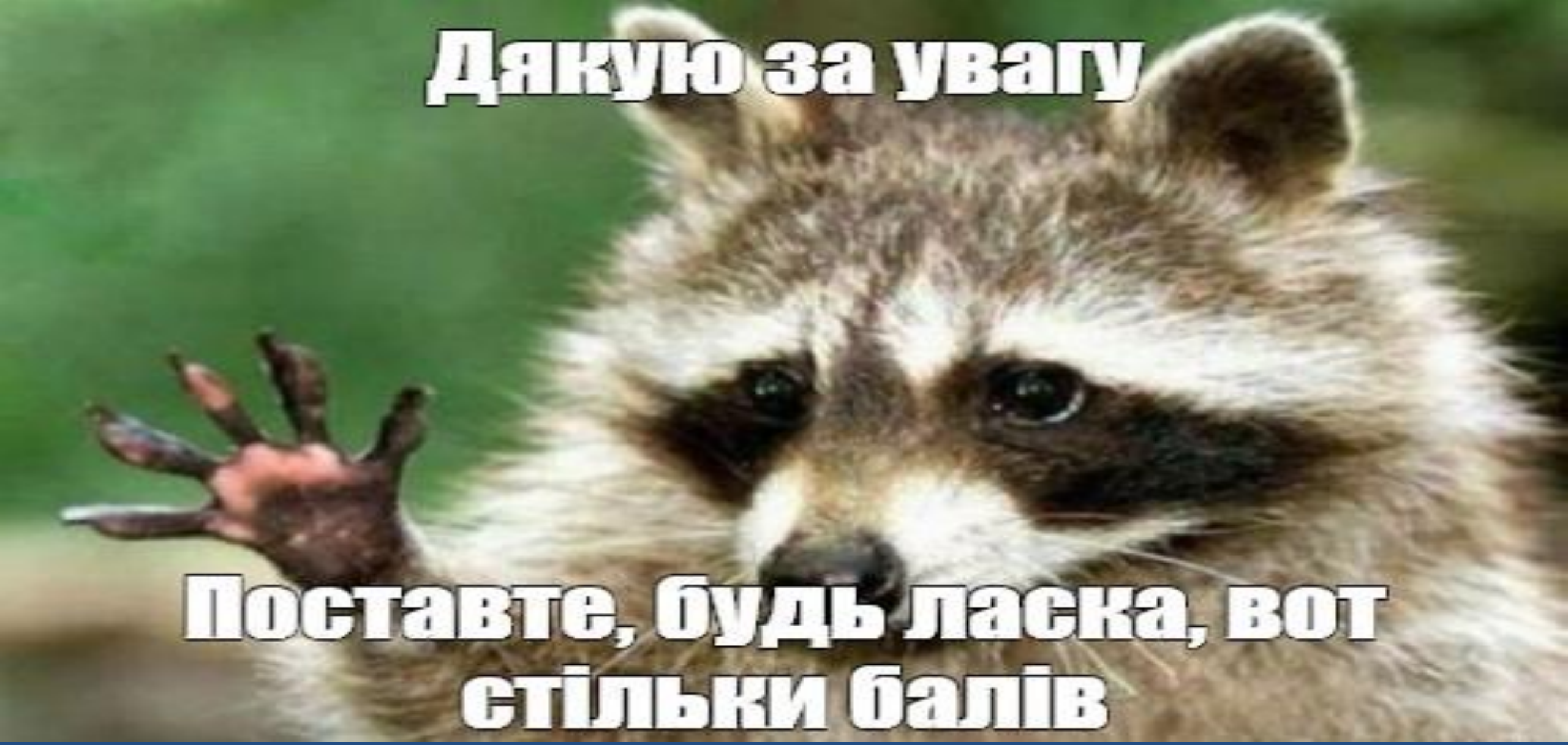
Приклади

- **просте логічне рівняння тільки з кон'юнкції, диз'юнкції та інверсії зайвих дужок не потребує (крім тих, звичайно, що вже є в рівнянні):**

$A \overset{4}{\vee} \overset{2}{\neg}(B \overset{1}{\wedge} C) \overset{3}{\wedge} A$	A or not(B and C) and A

- **при появі імплікації та еквіваленції підключаємо дужки, щоб зберегти пріоритет і цих, та інших логічних операторів:**

$A \overset{3}{\equiv} B \overset{1}{\wedge} C \overset{2}{\Rightarrow} A$	A == ((B and C) <= A)



ЖИТОМИРСЬКА ПОЛІТЕХНІКА
ДЕРЖАВНИЙ УНІВЕРСИТЕТ

@ZTUEDUUA

ЖИТОМИРСЬКА ПОЛІТЕХНІКА

ДЕРЖАВНИЙ УНІВЕРСИТЕТ

- **Розвиваємо лідерів**
- **Створюємо інновації**
- **Змінюємо світ на краще**