

Оператори extended

Assignment operators

Оператори присвоїння

x = x + 5

x = 5

Присвоїти значення 5

x += 1

Збільшити x на 1

x -= 1

Зменшити x на 1

x *= 2

Збільшити x у 2 рази

x /=

Зменшити x у 2 рази

x **= 2

Множення і присвоїння

x // = 2

Ділення з остачою і присвоїння

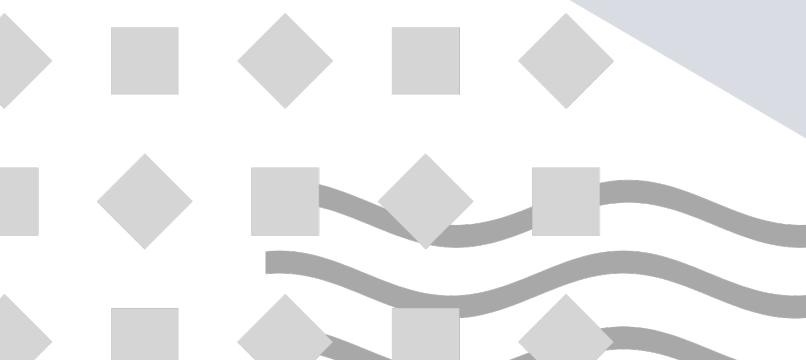
x %= 2

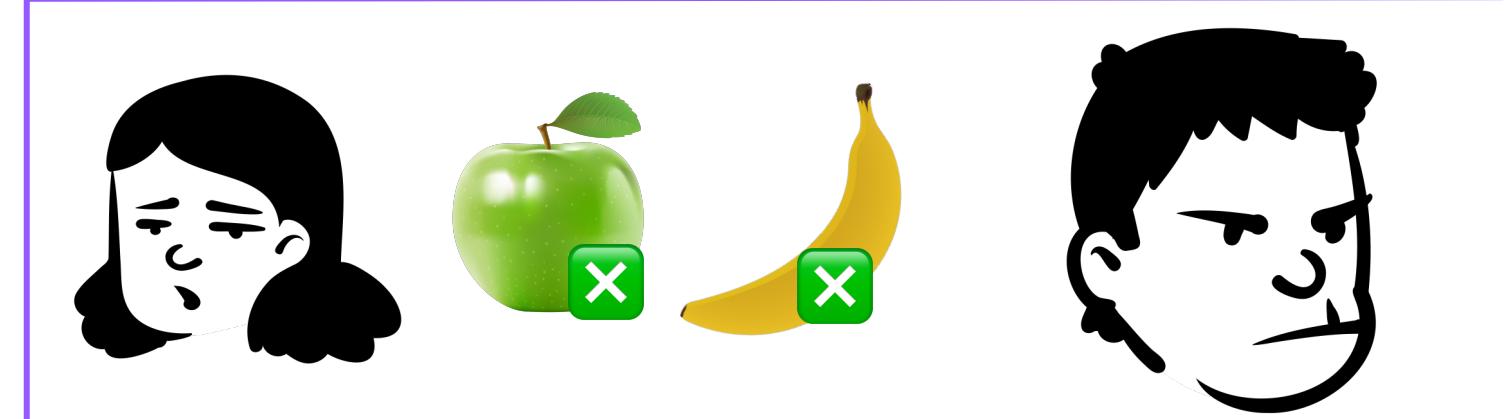
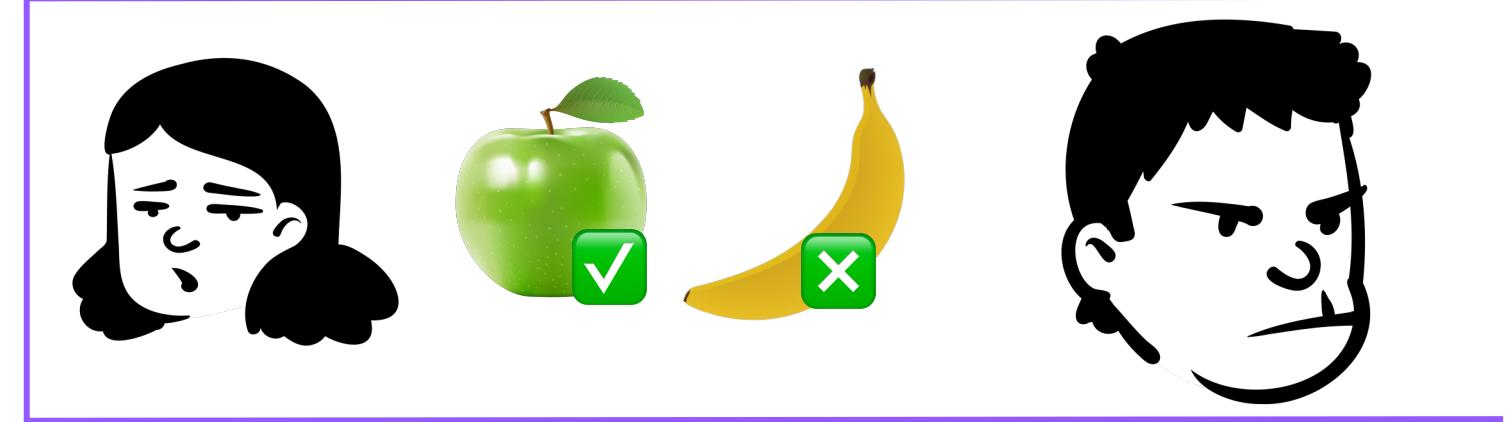
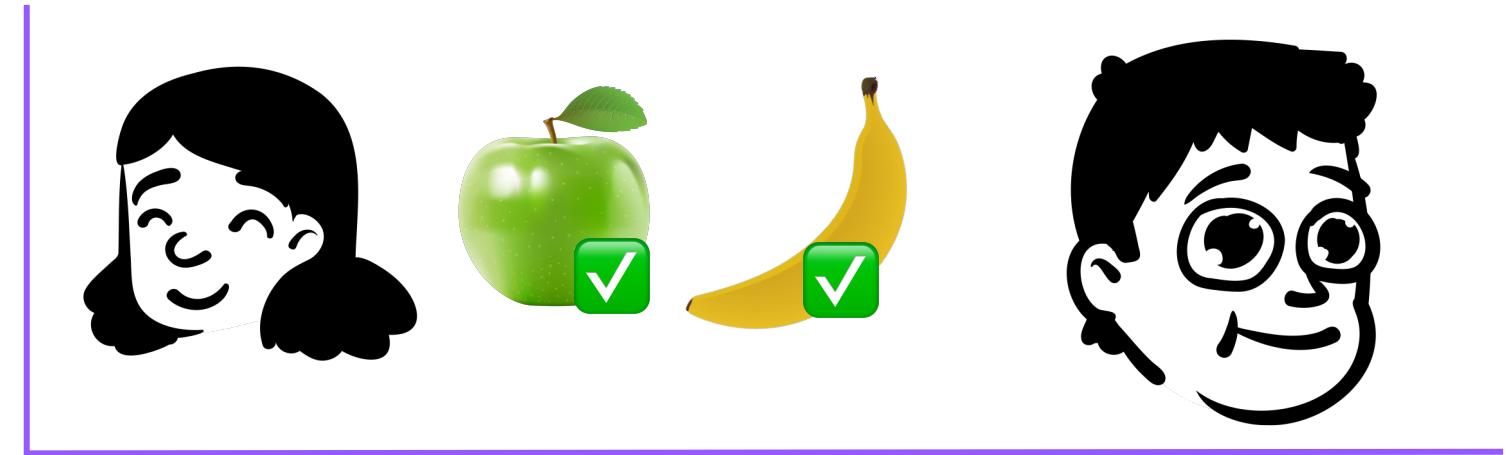
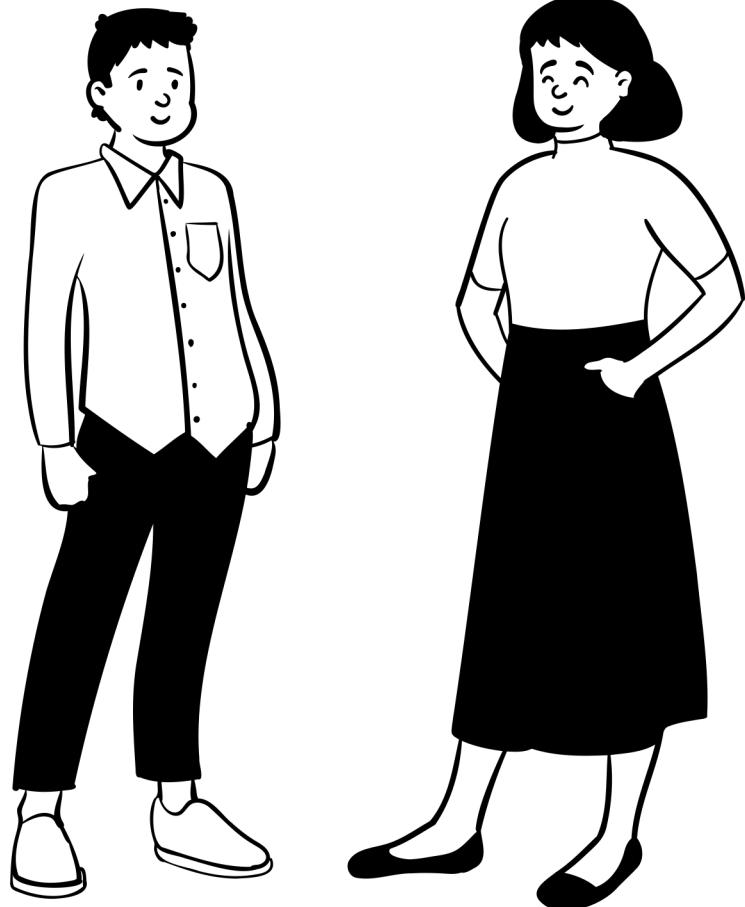
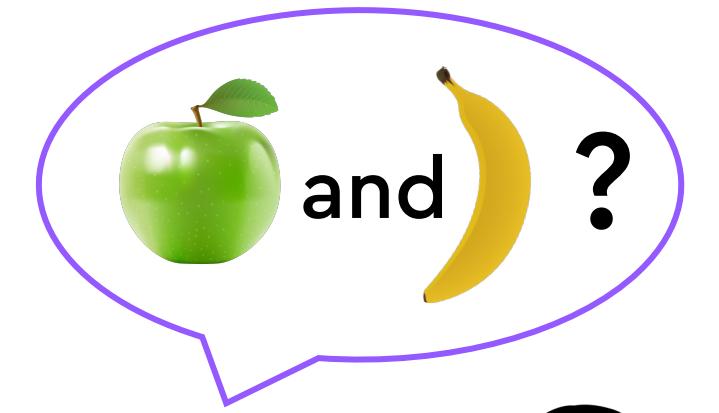
Ділення по модулю і присвоїння

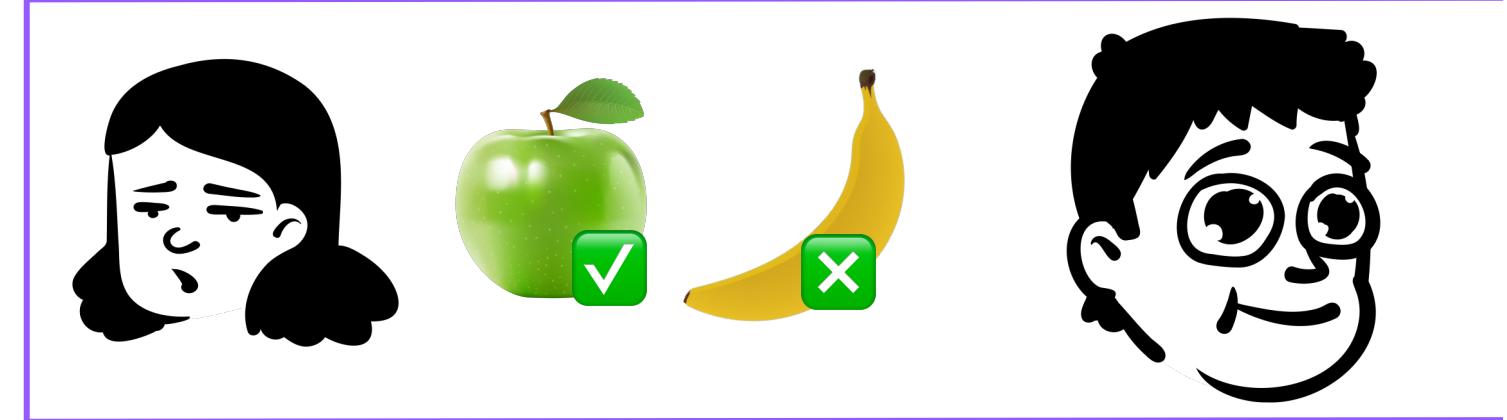
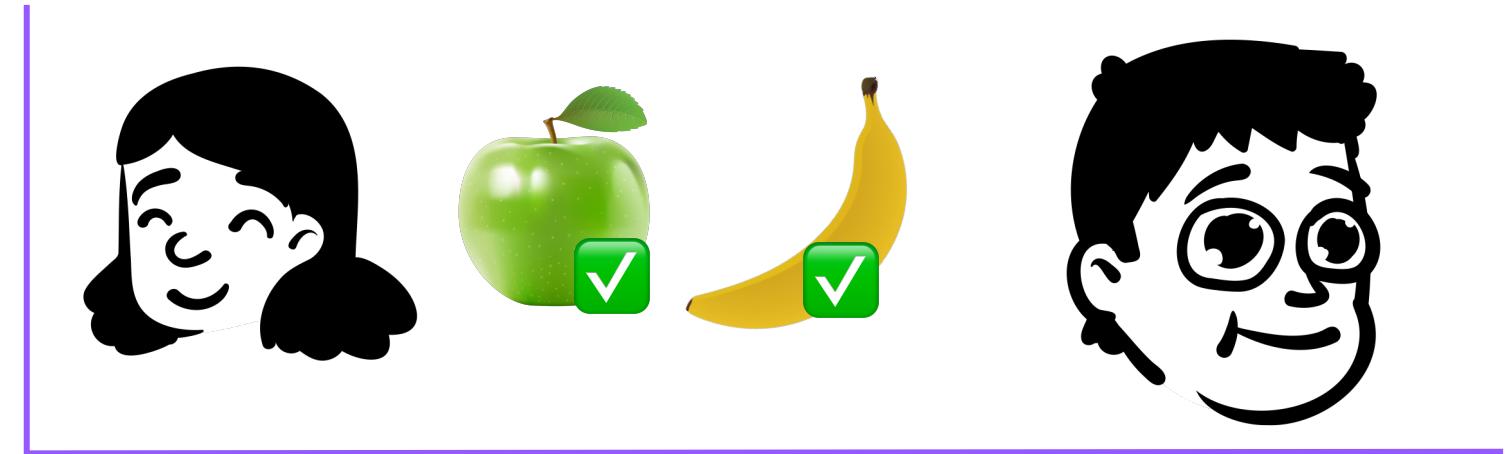
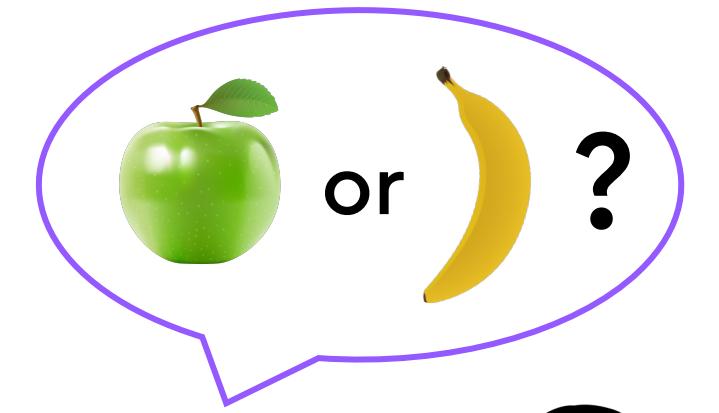
Logical operators

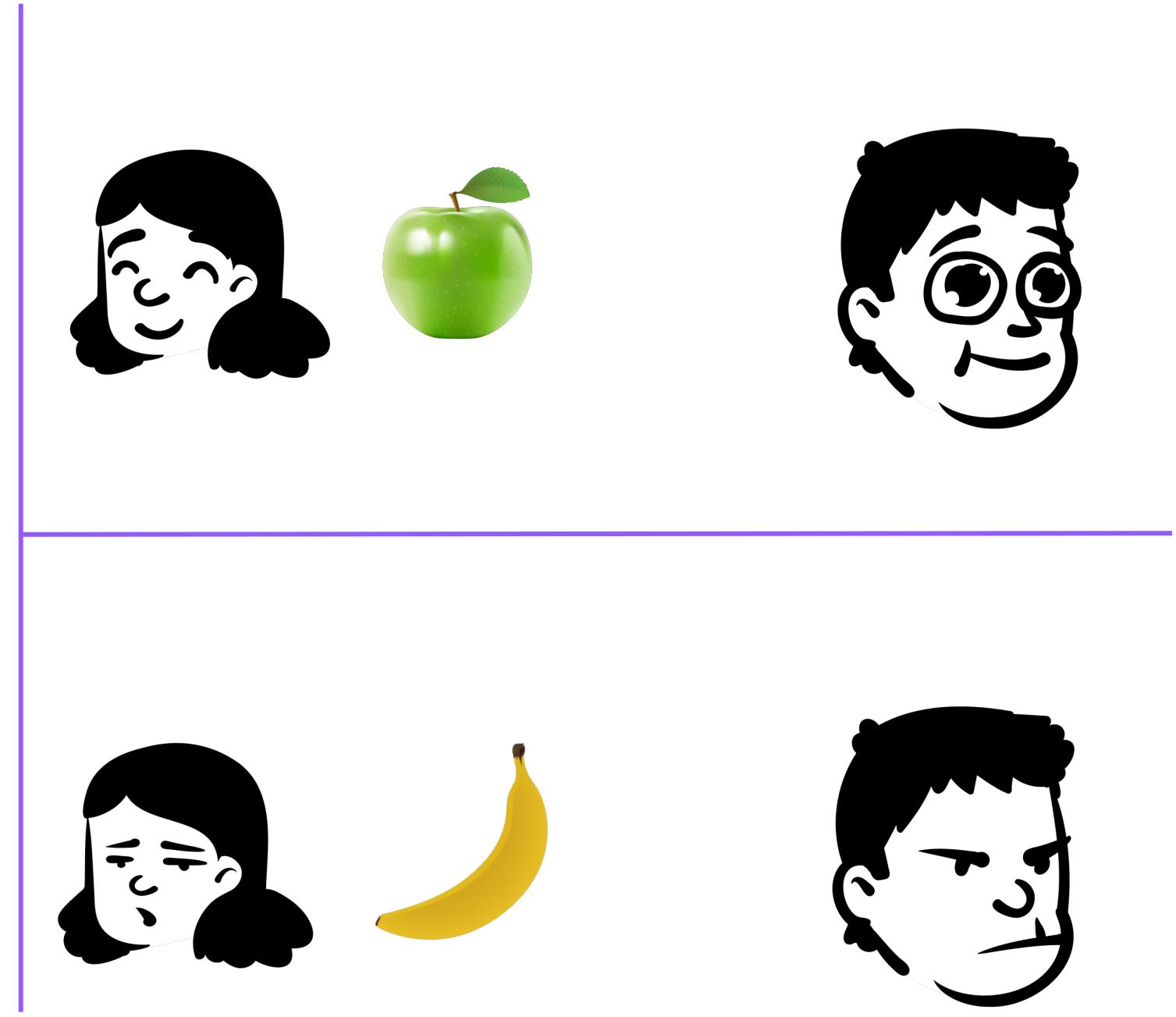
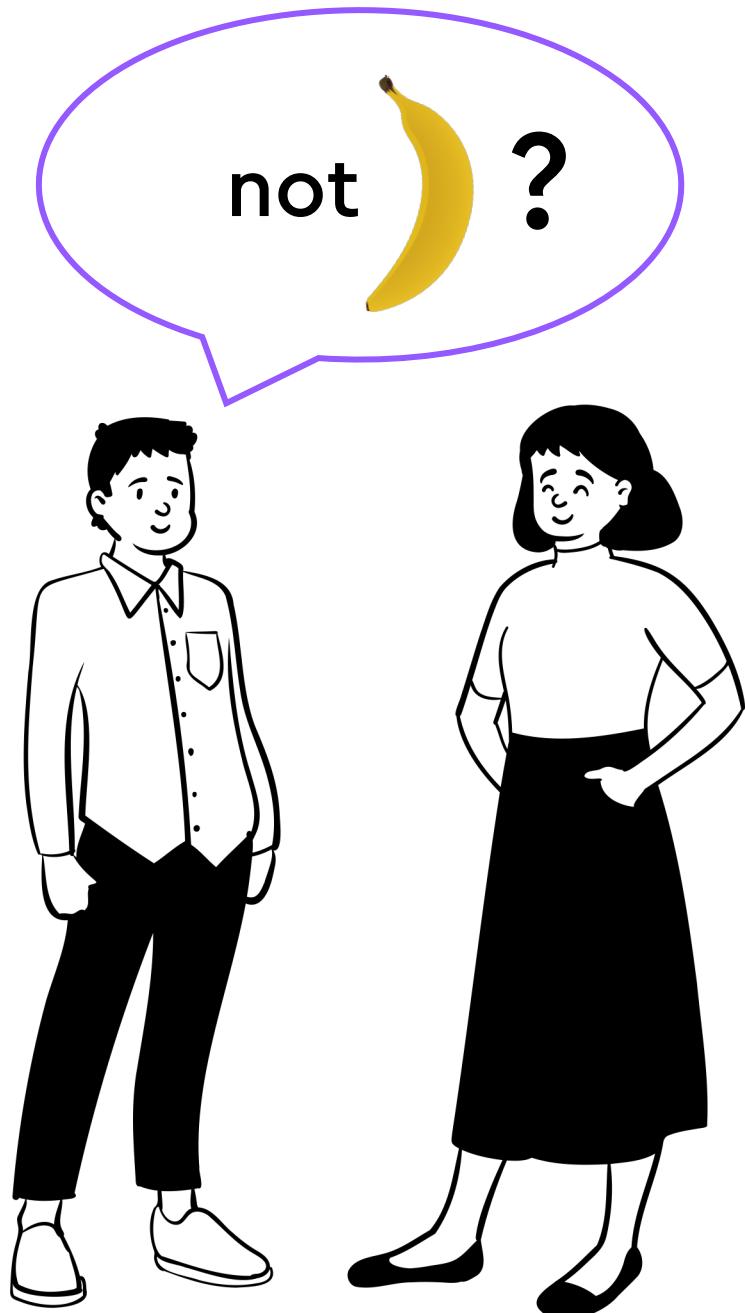
Логичні оператори

and
or
not







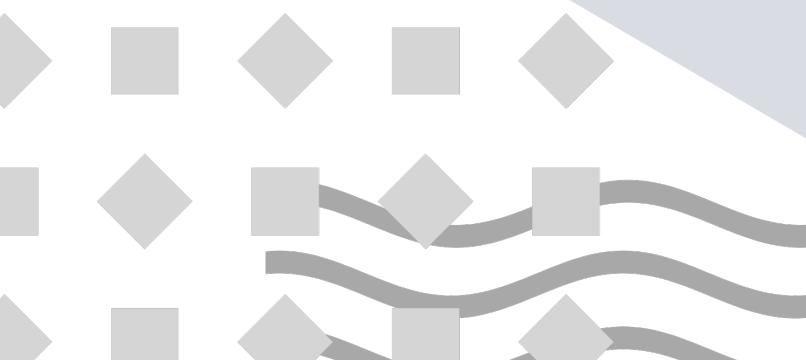


Membership Operators

Оператори входження

in

not in

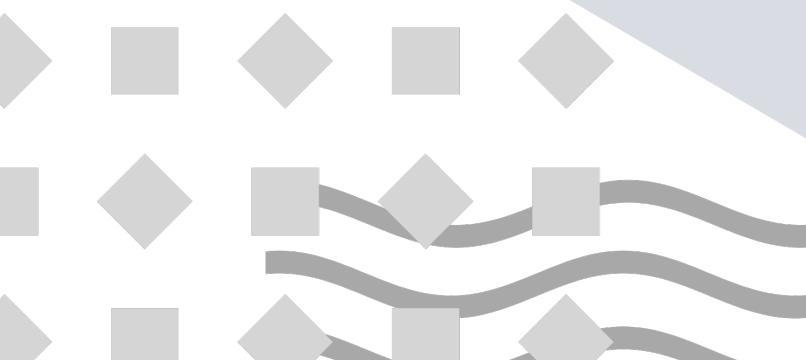


Identity Operators

Оператори тотожності

is

not is



Strings

Рядки

String

«hello»

Рядки потрібні для спілкування
програми з користувачем, аналізу
того, що відбувається в програмі, а
також як частина даних та команд

Название(EN)	Название(UK)	У системі	Приклад
String	Строки	str	"Hey! How are you doing?"

'Hello there'

=

"Hello there"

String Slicing

Зрізи рядків

'Hello there'
0 1 2 3 4 5 6 7 8 9 10

Рядок
Індекс

'My name is Slim Shady' [4]

'My name is Slim Shady' [5:9]

'My name is Slim Shady' [:8]

'My name is Slim Shady' [:-3]

String Format

Форматування

```
name = 'Username'  
print('Hello %s! Nice to meet you!' % name)  
print(f'Hello {name}! Nice to meet you!')  
print('Hello {0}! Nice to meet you!'.format(name))  
print('Hello {name}! Nice to meet you!'.format(name=name))  
print('Hello ' + name + '! Nice to meet you!')
```



Hello Username! Nice to meet you!

Build-in methods

Вбудовані методи

```
text = 'hello'
```

```
text.
```



```
__init__(self, args, kwargs)
m replace(self, __old, __new, __count)
m capitalize(self)
m casefold(self)
m center(self, __width, __fillchar)
m count(self, x, __start, __end)
m encode(self, encoding, errors)
m endswith(self, __suffix, __start, __end)
m expandtabs(self, tabsize)
m format_map(self, map)
m index(self, __sub, __start, __end)
m isalnum(self)
m isalpha(self)
m isascii(self)
m isdecimal(self)
m isdigit(self)
m isidentifier(self)
m islower(self)
m isnumeric(self)
m isprintable(self)
m isspace(self)
m istitle(self)
m isupper(self)
m ljust(self, __width, __fillchar)
m lower(self)
m lstrip(self, __chars)
m maketrans(__x)
m partition(self, __sep)
m removeprefix(self, __prefix)
m removesuffix(self, __suffix)
m rfind(self, __sub, __start, __end)
m rindex(self, __sub, __start, __end)
m rjust(self, __width, __fillchar)
m rpartition(self, __sep)
m rsplit(self, sep, maxsplit)
m rstrip(self, __chars)
m splitlines(self, keepends)
m startswith(self, __prefix, __start, __end)
m strip(self, __chars)
m swapcase(self)
m title(self)
m upper(self)
m zfill(self, __width)
```

Функція чи метод	Призначення
S = 'str'; S = "str"; S = """str"""; S = """"""str""""""	Літерали рядків
S = "s\np\ta\nbbb"	Екрановані послідовності
S = r"C:\temp\new"	Неформатовані рядки (пригнічують екранування)
S = b"byte"	Рядок байтів
S1 + S2	Конкатенація (складання рядків)
S1 * 3	Дублювання рядка
S[i]	Звернення за індексом
S[i:j:step]	Вилучення зрізу
len(S)	Довжина рядка
S.find(str, [start],[end])	Пошук підрядка у рядку. Повертає номер першого входження або -1

S.rfind(str, [start],[end])	Пошук підрядки у рядку. Повертає номер останнього входження або -1
S.index(str, [start],[end])	Пошук підрядки у рядку. Повертає номер першого входження або викликає ValueError
S.rindex(str, [start],[end])	Пошук підрядку у рядку. Повертає номер останнього входження або викликає ValueError
S.replace(шаблон, заміна[, maxcount])	Заміна шаблону на заміну
S.split(символ)	Розбиття рядка по роздільнику
S.isdigit()	Чи складається рядок із цифр
S.isalpha()	Чи складається рядок з літер
S.isalnum()	Чи складається рядок із цифр або букв
S.islower()	Чи складається рядок із символів у нижньому регістрі
S.isupper()	Чи складається рядок із символів у верхньому регістрі

S.isspace()	Чи складається рядок з символів, що не відображаються*
S.istitle()	Чи починаються слова в рядку з великої літери
S.upper()	Перетворення рядка до верхнього регістру
S.lower()	Преобразование строки к нижнему регистру
S.startswith(str)	Чи починається рядок S із шаблону str
S.endswith(str)	Чи закінчується рядок S шаблоном str
S.join(список)	Складання рядка зі списку з роздільником S
ord(символ)	Символ коду ASCII
chr(число)	Код ASCII у символ
S.capitalize()	Перероблює перший символ рядка у верхній регістр, а решта в нижній
S.center(width, [fill])	Повертає відцентрований рядок, по краях якого стоїть символ fill (пробіл за замовчуванням)
S.count(str, [start],[end])	Повертає кількість неперетинних входжень підрядки в діапазоні [початок, кінець] (0 і довжина рядка за замовчуванням)

* пробіл, символ перекладу сторінки ('\f'), "новий рядок" ('\n'), "переклад каретки" ('\r'), "горизонтальна табуляція" ('\t') та "вертикальна табуляція" ('\v')

S.expandtabs([tabsize])	Створює копію строки, в якій всі символи таблиці замінюються одним або кількома пробілами, в залежності від поточного столбця. Якщо TabSize не вказано, розмір табуляції подається рівним 8 пробілам
S.lstrip([chars])	Видалення символів пробілів на початку рядка
S.rstrip([chars])	Видалення символів пробілу в кінці рядка
S.strip([chars])	Видалення символів пробілів на початку і в кінці рядка
S.partition(шаблон)	Повертає кортеж, що містить частину перед першим шаблоном, сам шаблон і частину після шаблону. Якщо шаблон не знайдено, повертається кортеж, що містить самий рядок, а потім два порожні рядки
S.rpartition(sep)	Повертає кортеж, що містить частину перед останнім шаблоном, сам шаблон і частину після шаблону. Якщо шаблон не знайдено, повертається кортеж, що містить два порожні рядки, а потім самий рядок
S.swapcase()	Змінює символи нижнього регістру до верхнього, а верхнього – до нижнього
S.title()	Першу літеру кожного слова змінює у верхній регістр, а решту - у нижній
S.zfill(width)	Робить довжину рядка не меншого width, за необхідності заповнюючи перші символи нулями
S.ljust(width, fillchar=" ")	Робить довжину рядка не меншого width, за необхідності заповнюючи останні символи символом fillchar
S.rjust(width, fillchar=" ")	Робить довжину рядка не меншого width, за необхідності заповнюючи перші символи символом fillchar

Lists

Списки

List

[1,'hi',200]

Це впорядковані за місцем розташування колекції об'єктів довільних типів, розмір яких не обмежений.

Название(EN)	Название(UK)	Название(RU)	У системе	Приклад
List	Список	Список	list	[1,2,3,4,5,'six','seven']

List slices

Зрізи списків

Список

[1,2,3,4,5,'six','seven']

0 1 2 3 4 5

6

Індекс

[1,2,3,4,5,'six','seven']

[2]

[1,2,3,4,5,'six','seven']

[1:3]

[1,2,3,4,5,'six','seven']

[::5]

[1,2,3,4,5,'six','seven']

[::-3]

Додати у список

Один об'єкт

[1,2,3].append([4,5]) → [1,2,3,[4,5]]

Декілька об'єктів

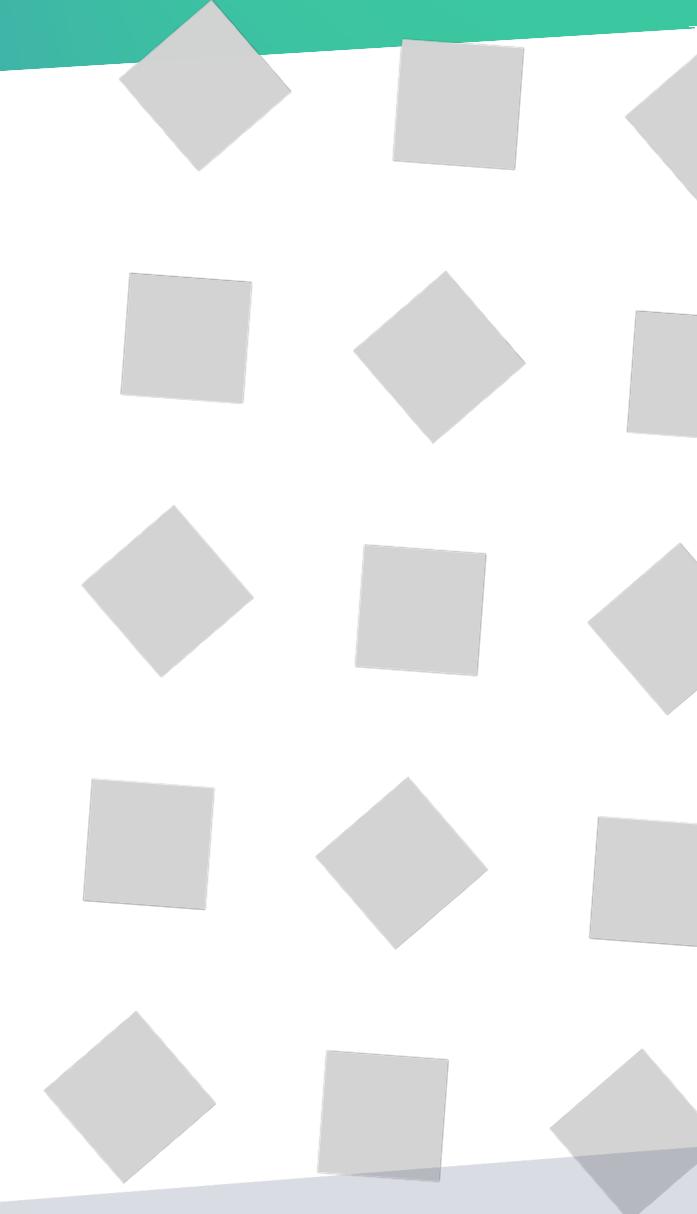
[1,2,3].extend([4,5]) → [1,2,3,4,5]

Отримати значення

[1,2,3][1] → **2**

[1,2,3].pop(1) → **2**

[1,3]



Nested lists

Вложені списки

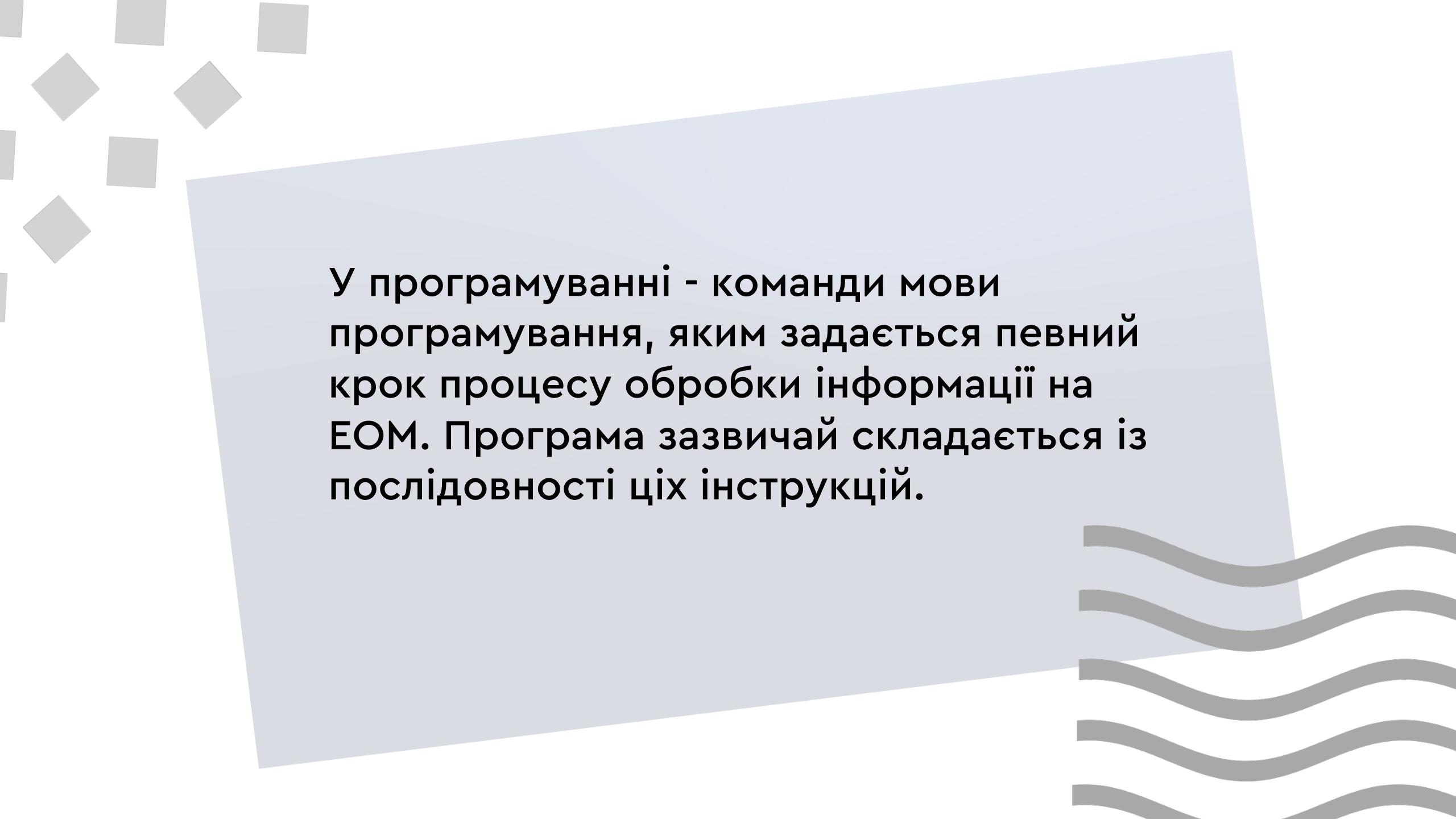
a = [1,2,3,[4,5,[2,3,4],6,7],6,7]



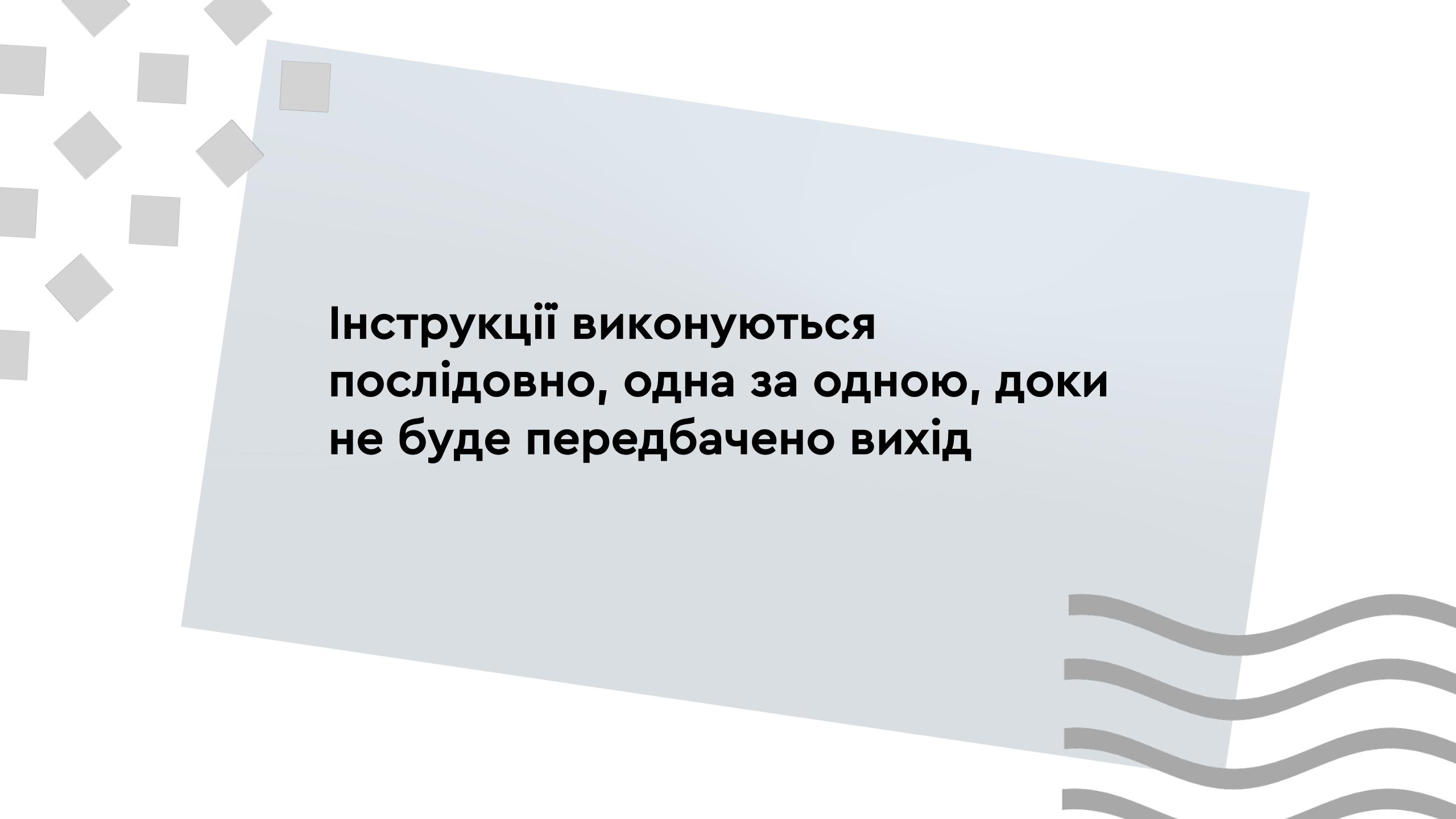
Метод	Что делает
list.append(x)	Додає елемент у кінець списку
list.extend(L)	Розширює список list, додаючи до кінця всі елементи списку L
list.insert(i, x)	Розширює список list, додаючи до кінця всі елементи списку L
list.remove(x)	Видаляє перший елемент у списку, що має значення x
list.pop([i])	Видаляє перший елемент «i» і повертає його. Якщо індекс не вказано, видаляється останній елемент
list.index(x, [start [, end]])	Повертає положення первого елемента від start до end із значенням x
list.count(x)	Повертає кількість елементів зі значенням x
list.sort([key = функція])	Сортує список на основі функції
list.reverse()	Розгортаває список
list.copy()	Поверхнева копія списку
list.clear()	Очищає список

Conditions

Керуючі конструкції

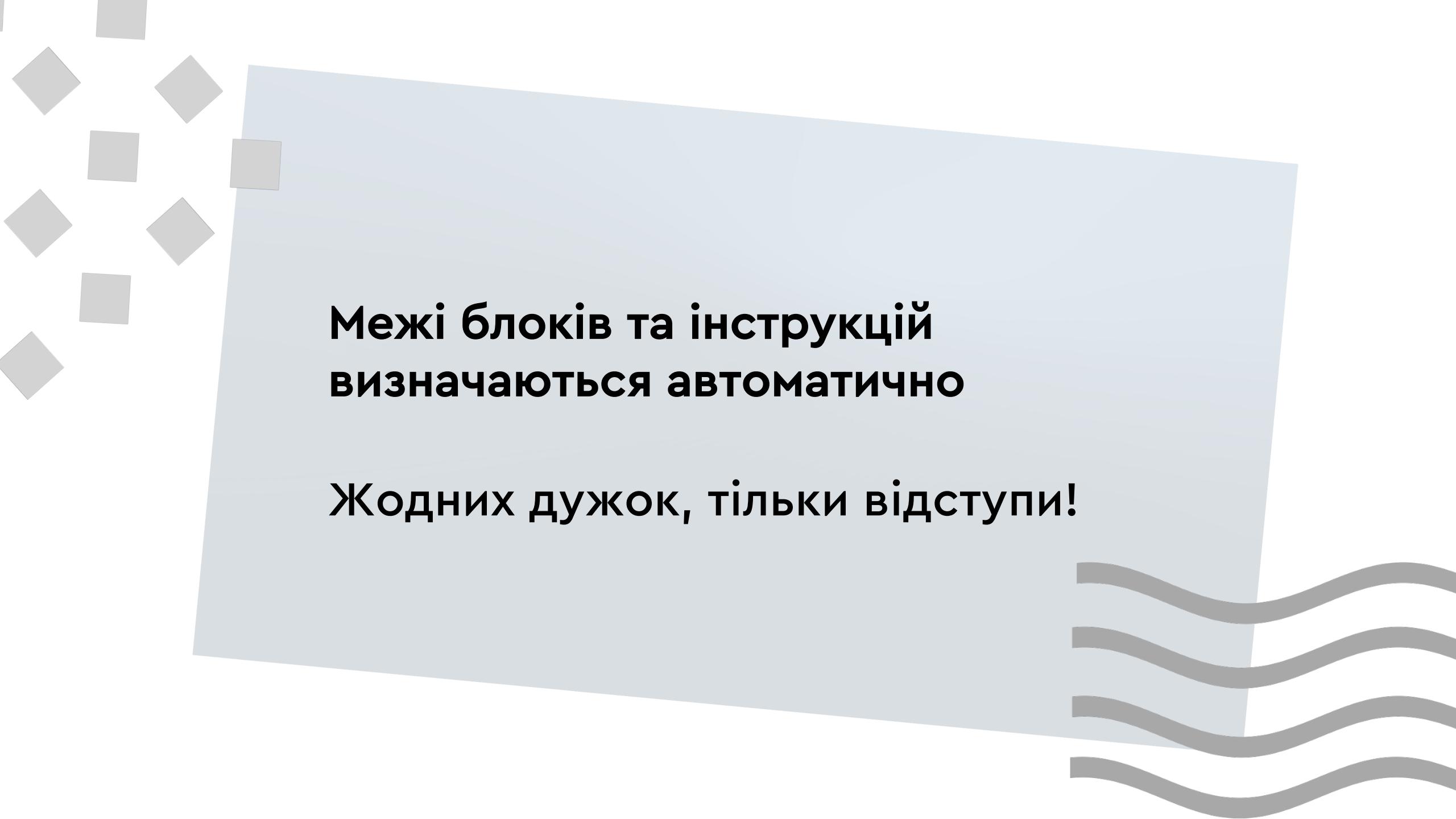


У програмуванні - команди мови
програмування, яким задається певний
крок процесу обробки інформації на
ЕОМ. Програма зазвичай складається із
послідовності ціх інструкцій.



**Інструкції виконуються
послідовно, одна за одною, доки
не буде передбачено вихід**

Зазвичай інтерпретатор виконує інструкції у файлі чи блоці від початку до його кінця, але такі інструкції як `if` і цикли змушують інтерпретатор виконувати переходи всередині програмного коду. Оскільки шлях інтерпретатора Python через текст програми називається потоком управління, такі інструкції, як `if`, часто називаються інструкціями управління потоком виконання.



Межі блоків та інструкцій
визначаються автоматично

Жодних дужок, тільки відступи!

Правило написання будь-якої інструкції

**Умова:
Що робимо**

if statements



основний інструмент вибору Python. Простіше кажучи, вона вибирає, яку дію слід виконати, залежно від значення змінних під час перевірки умови.

Починаємо блок - умова

if <условие 1>:

Что делаем при этом условии

elif <условие 2>:

Что делаем при этом условии

else:

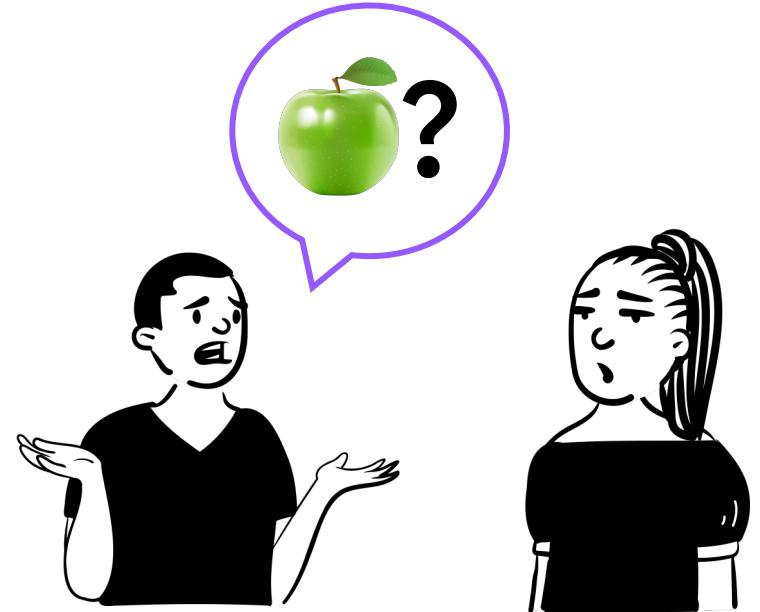
Что делаем при этом условии

На випадок якщо

умов багато -

Альтернативна дія

Що робимо, якщо інші умови
не трапилися – за умовчанням



if

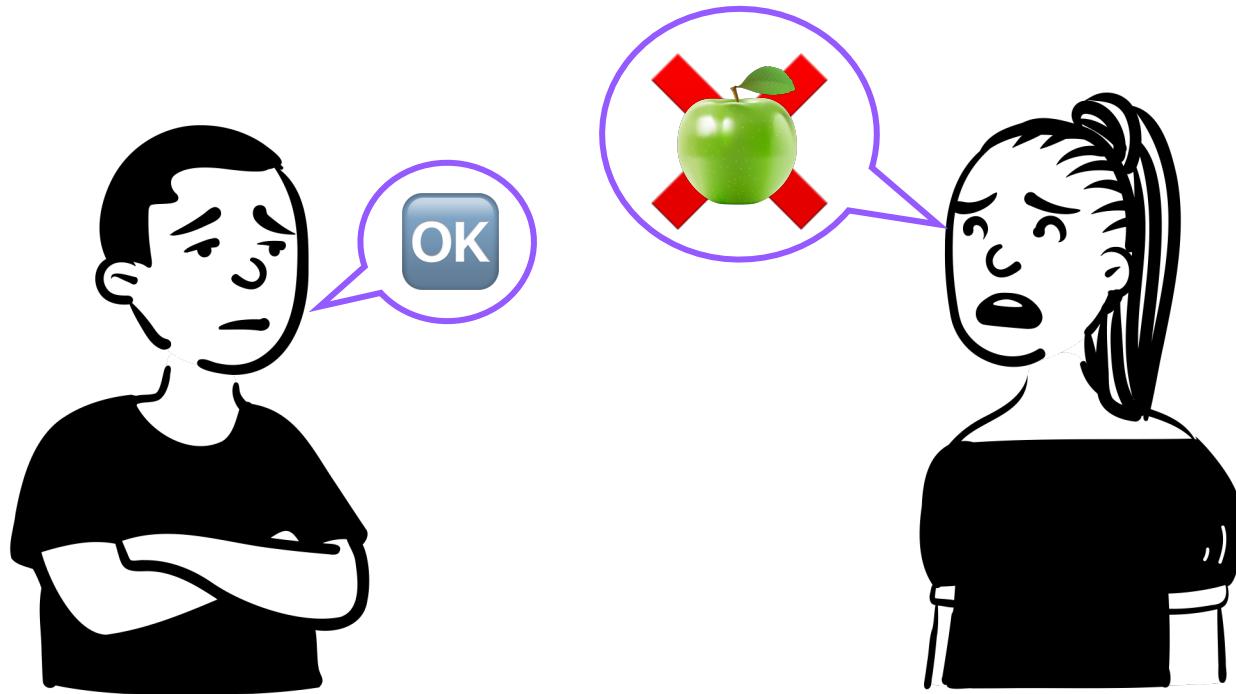


elif



elif

else



Pass

Блок не може бути порожнім – це помилка системи!

◆ Pass

На випадок, якщо ми не хочемо поки що нічого писати,
або ще не знаємо що.



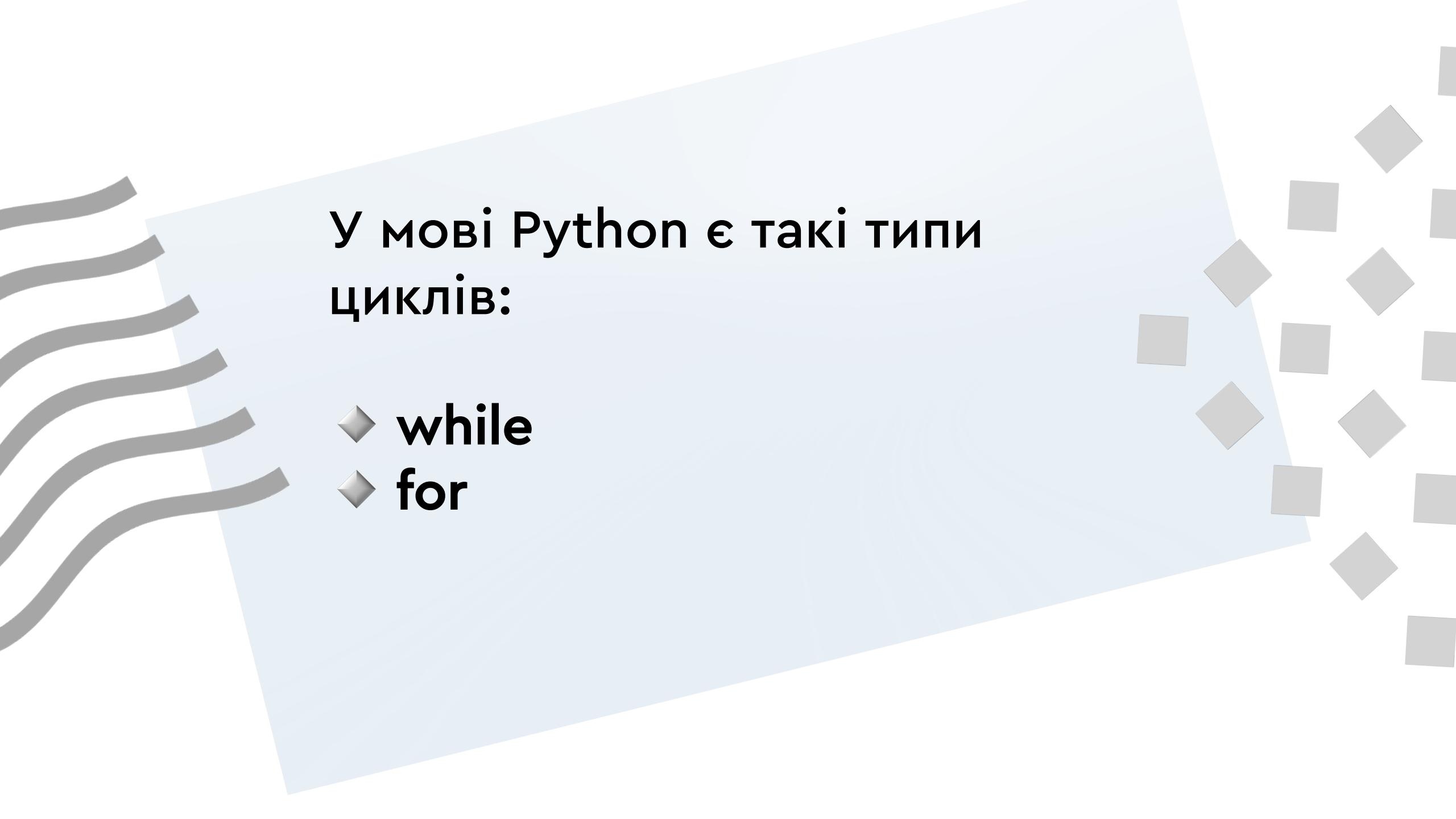
1. Take an integer number as input and print if it's "Even" or "Odd".
2. Take as inputs sides of rectangular and print "Big" if the area is larger than 100, otherwise "small".
3. Введіть назву фігури та відповідні параметри для того, щоб дізнатись її площа. Виведіть напис «Великий», якщо площа більше 100, «Маленький», якщо менше. І якщо відповідь від'ємна – «Помилка»
4. Take as input two real numbers and an operation ("+" or "-" or "/" or "*") and calculate the result of the operation. Watch for division by zero errors!



Цикли



Цикли використовуються в тих випадках, коли нам потрібно зробити щось багато разів. Нерідко вам доведеться виконати якусь операцію даних знову і знову. При цьому ми можемо змінювати умову у процесі роботи



У мові Python є такі типи циклів:

- ◆ **while**
- ◆ **for**

Ітерація

Ітерація в програмуванні - організація обробки даних, при якій дії повторюються багаторазово, не призводячи до викликів самих себе (на відміну від рекурсії)

◆ Інкремент

$a+=$

◆ Декремент

$a-=$

While

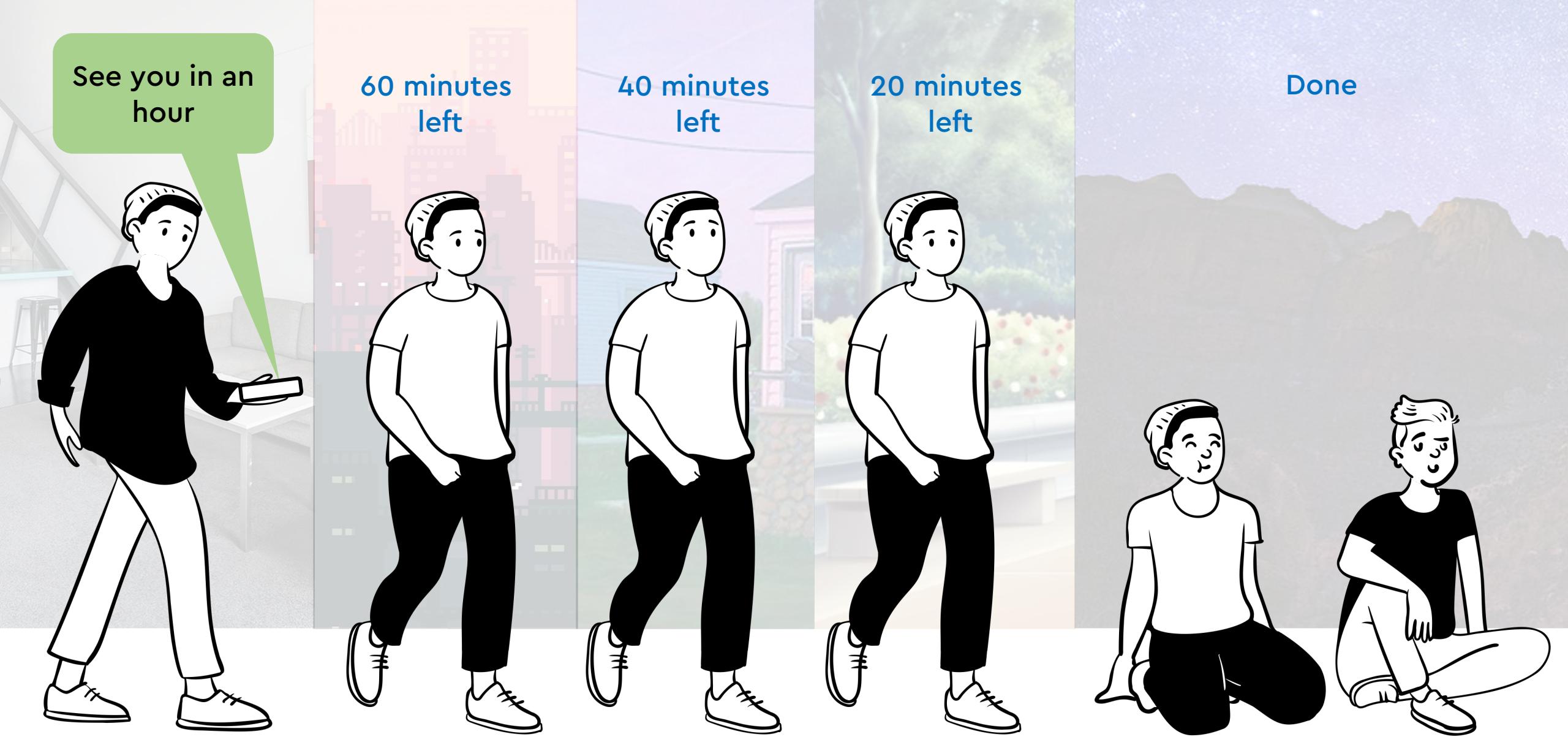
Цикл `while` перевіряє істинність деякої умови, і якщо умова істинна, виконує інструкції циклу.

Починаємо блок

`while <умова>:
 інструкції`

Te, що робимо

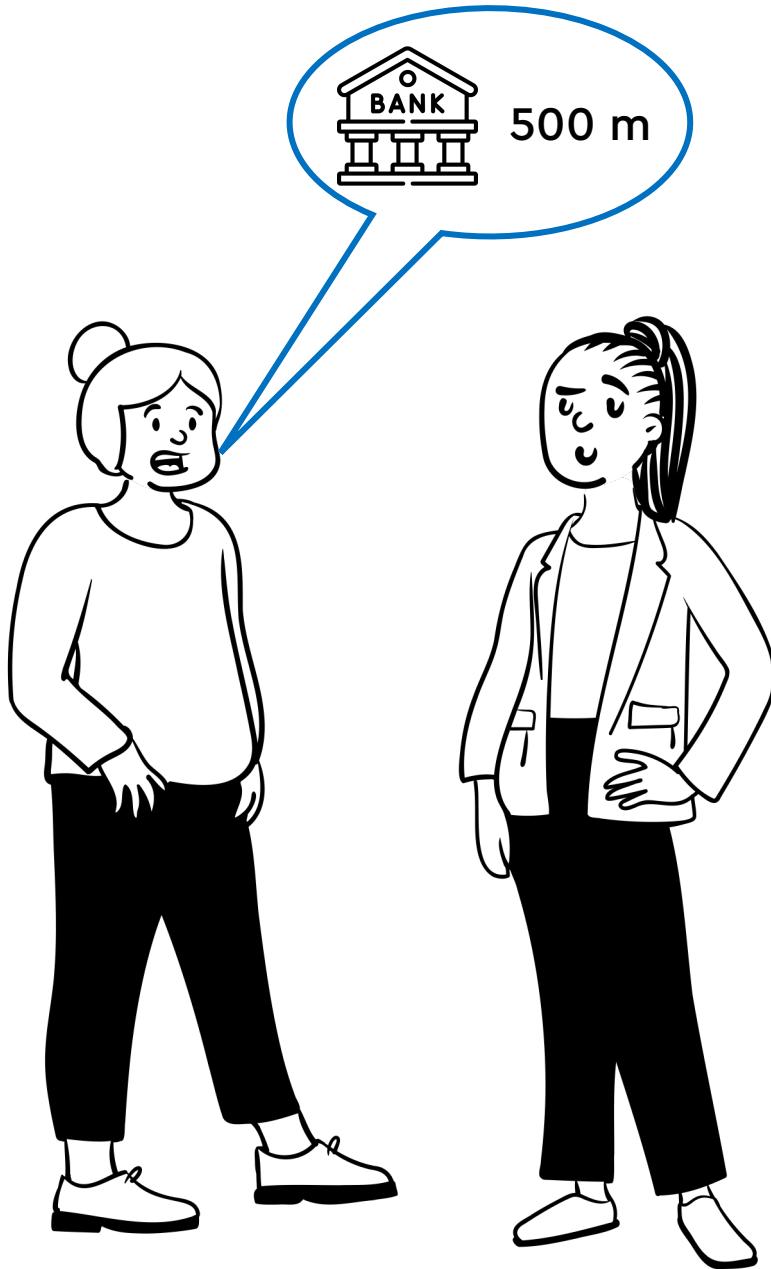
Після ключового слова `while` вказується умовний вираз, і поки цей блок повертає значення `True` - виконуватиметься блок інструкцій, що йде далі.



Типи циклів

- ◆ Передумова
- ◆ Післяумова
- ◆ Нескінчений цикл

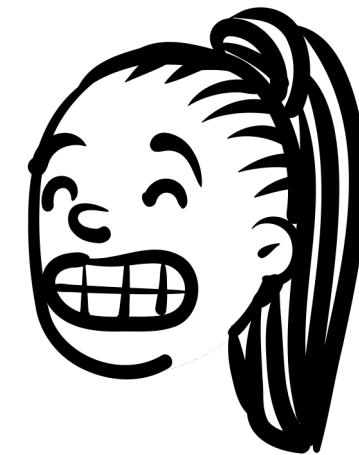




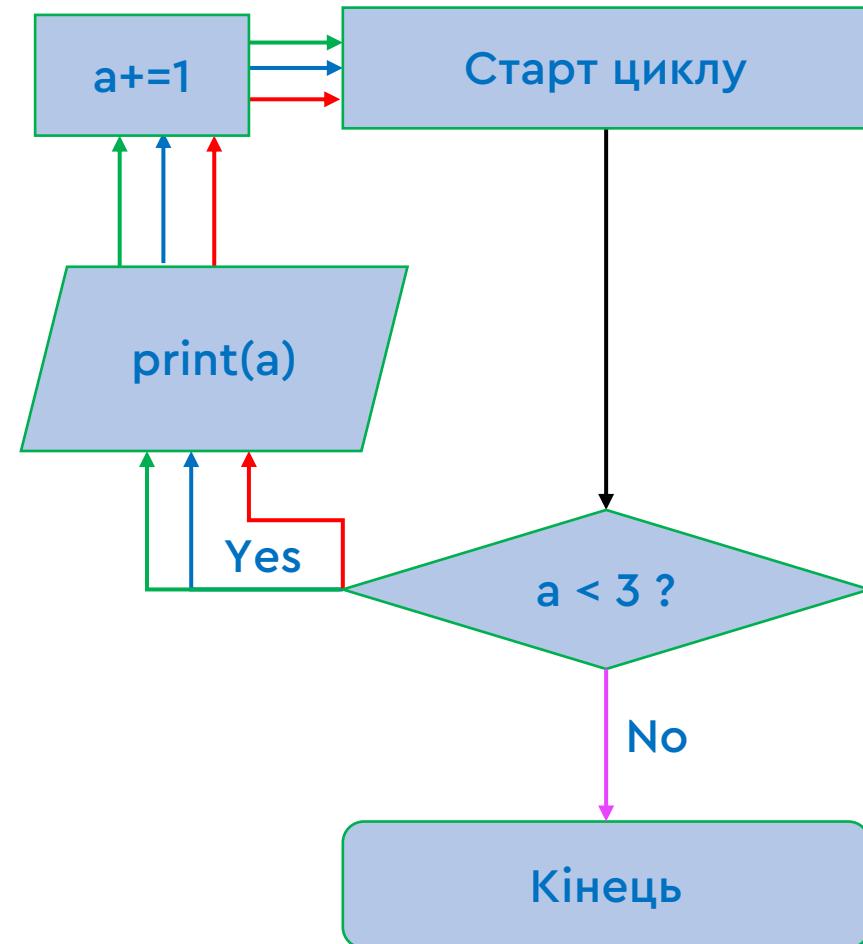
Передумова



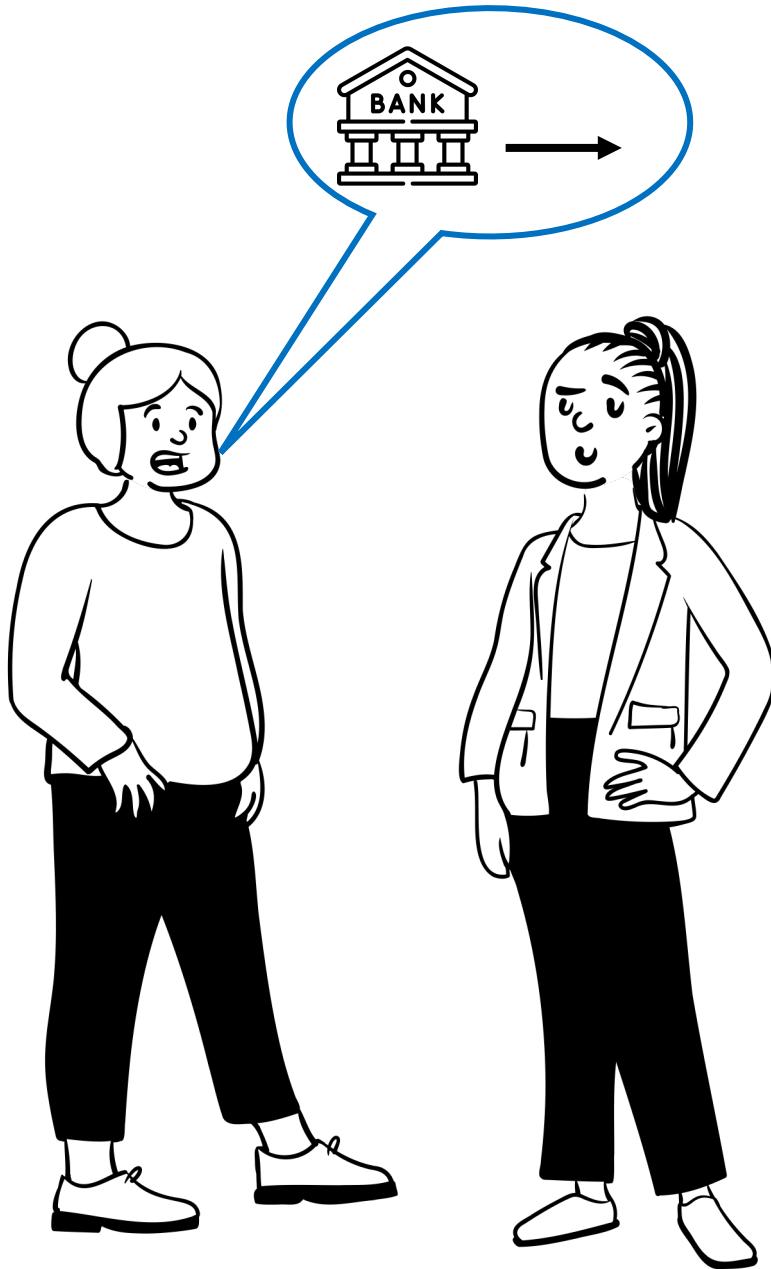
500 м
500 м?



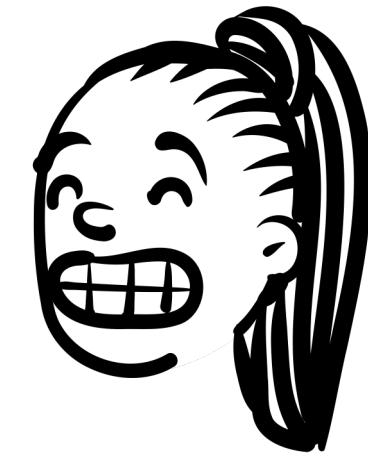
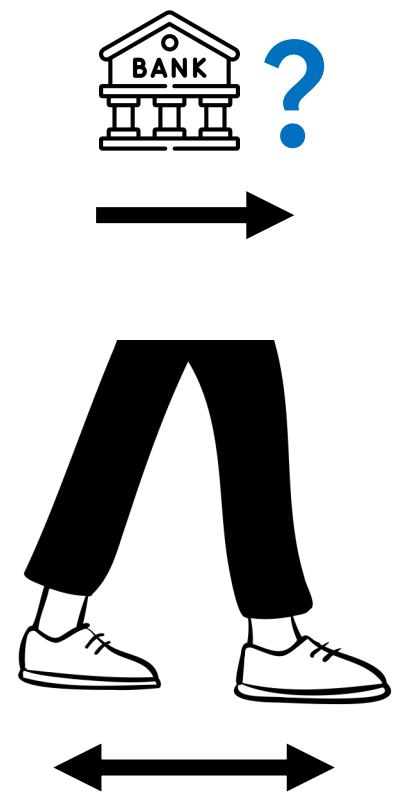
```
a = 0  
while a<3:  
    print(a)  
    a+=1
```



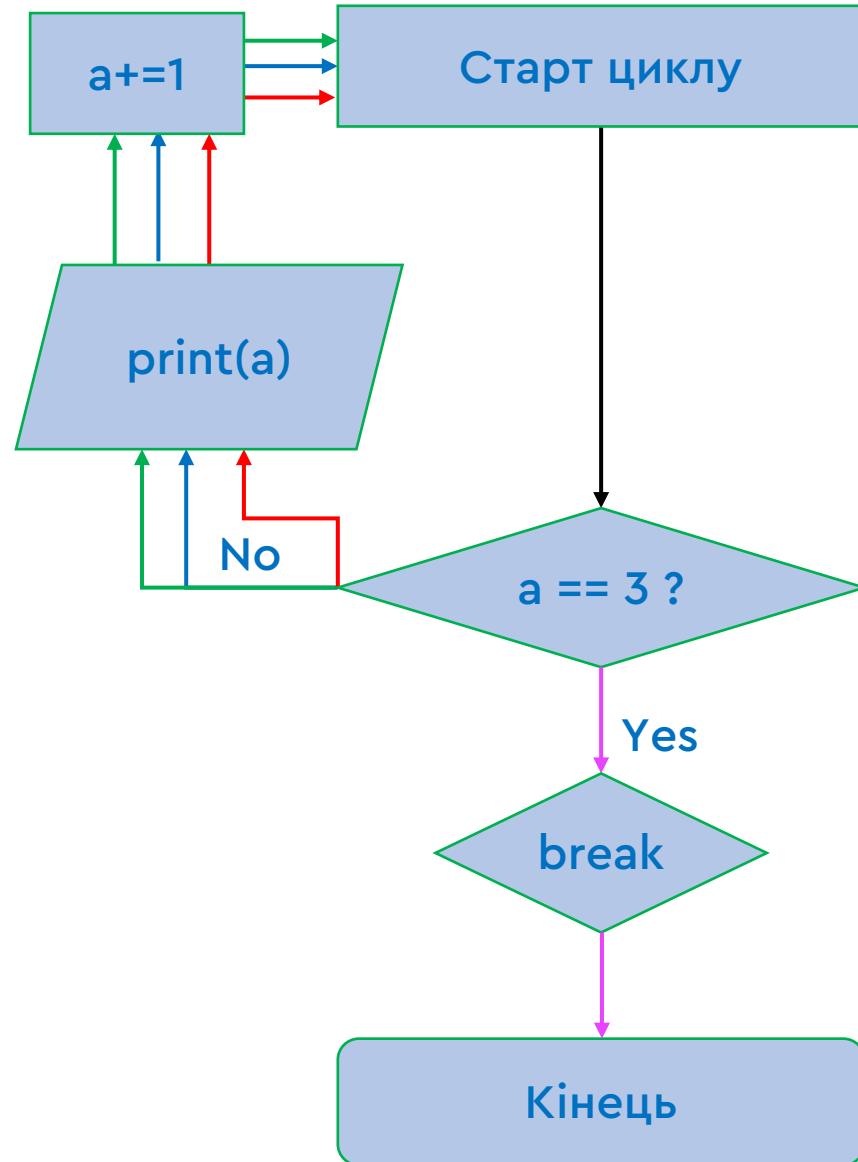
a = 0
a = 1
a = 2
a = 3



Післяумова



```
a = 0
while True:
    if a == 3:
        break
    else:
        print(a)
    a+=1
```



a = 0
a = 1
a = 2
a = 3

Команди керування потоком

◆ **break**

Здійснює перехід за межі об'ємного циклу (усієї інструкції циклу)

◆ **continue**

Здійснює перехід на початок циклу (у рядок заголовка)

◆ **else**

Виконується тільки якщо цикл завершився звичайним чином



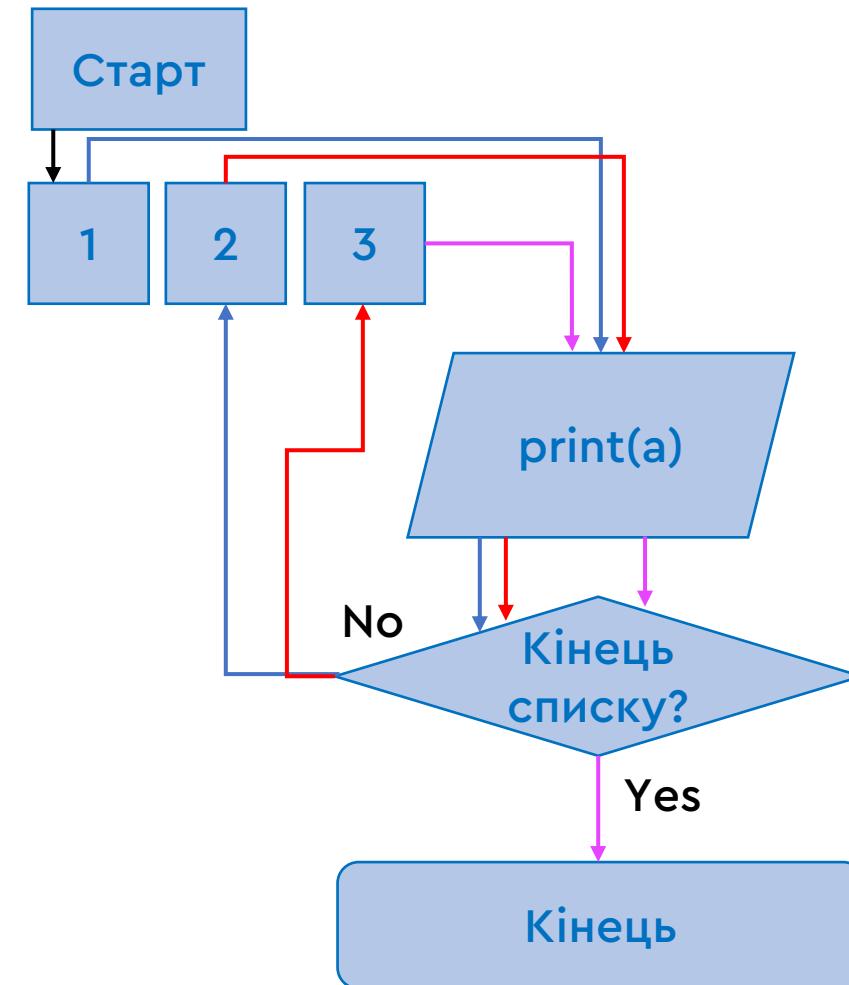
For

Цей цикл пробігається по набору значень, що поміщає кожне значення у змінну, і потім у циклі ми можемо її використовувати

Починаємо блок
Змінна для підстановки
Де будемо проходити
↑
for змінна in набір_значень:
інструкції
Te, що робитимемо



```
for x in [1,2,3]:  
    print(x)
```



a = 1
a = 2
a = 3