Python

03

Оператори порівняння (a = 10, b = 20**)**

==	Якщо значення двох операндів дорівнюють - повертає True	(a == b) – Неправда (False)
!=	Якщо значення двох операндів **НЕ** рівні – повертає True	(a != b) - Правда (True)
>	Якщо значення лівого операнда більше значення правого – повертає True	(a > b) - Неправда (False)
<	Якщо значення правого операнда більше значення лівого – повертає True	(a < b) - Правда (True)
>=	Якщо значення лівого операнда більше, або дорівнює значенню правого – повертає True	(a >= b) - Неправда (False)
<=	Якщо значення правого операнда більше, або рівне значенням лівого - повертає True	(a <= b) - Правда (True)
is	Якщо лівий і правий операнди посилаються на ту саму ділянку пам'яті - повертає True	(a is b) - Неправда (False)
is not	Якщо лівий і правий операнди **НЕ** посилаються на ту саму ділянку пам'яті - повертає True	(a is not b) - Правда (True)
in	Якщо лівий операнд входить до правого складу, то повертається True	a in list(a, b) - Правда (True)

Умовне розгалуження

Загальний синтаксис інструкції умовного розгалуження у мові Python має такий вигляд:

```
if boolean_expression1:
    suite1
elif boolean_expression2:
    suite2
...
elif boolean_expressionN:
    suiteN
else:
    else_suite
```

Загальний синтаксис тернарного виразу:

```
expression1 if boolean_expression else expression2
```

Логічні оператори

Існують три логічні оператори (logical operators): **and**, **or** і **not**. Сенс цих операторів схожий з їх змістом в англійській мові:

x > 0 and x < 10 — це істинно у разі, якщо **x** більше 0 та менше 10.

n%2 == 0 or n%3 == 0 — це істинно, якщо один із виразів є істинним.

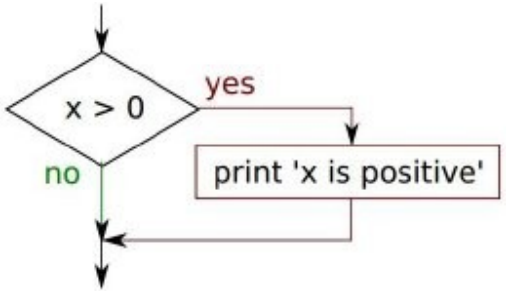
Оператор **not** заперечує логічний вираз, так:

not (x > y) — істинно, якщо **x** > **y** є неправдою, тобто **x** менше чи дорівнює **y**.

not x — істинно, якщо **x** є неправдою (False).

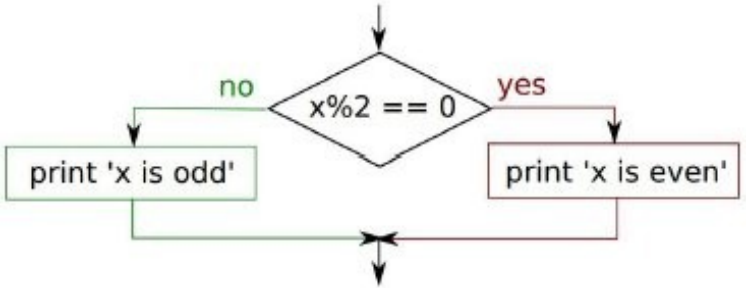
Варіанти виконання умовного розгалуження

Умовне виконання



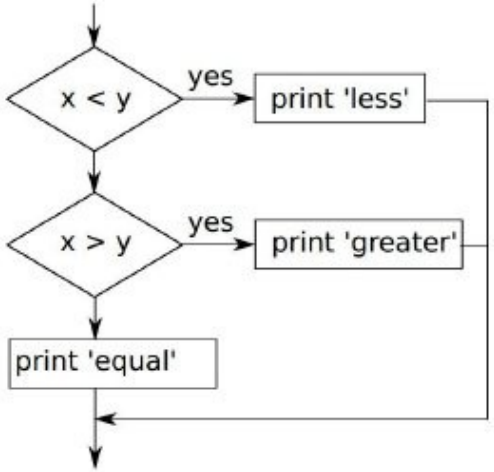

```
if x > 0 :
    print('x is positive')
```

Альтернативне виконання




```
if x%2 == 0 :
        print('x is even')
else :
        print('x is odd')
```

Послідовність умов




```
if x < y:
        print('less')
elif x > y:
        print('greater')
else:
        print('equal')
```

Операції зі списками та деякі методи

>>> list_a = [‘a’, ‘b’, 42]
Додавання
>>> list_a.append(‘p’)

Обращение к элементу списка
>>> list_a[0]

Видалення елементу по значенню
>>> list_a.remove(‘a’)

Видалення елементу за індексом
>>> del list_a[0]
>>> list_a.pop(0)

Зміна елемента списку
>>> list_a[0] = 77

Расширение
>>> a = [1, 2, 3]
>>> b = [7, 8, 9]
>>> b.extend(a)
>>> print(b)
[7, 8, 9, 1, 2, 3]

Довжина списку
>>> len(a)

Вставлення елемента
>>> a.insert(0, 5)

>>> print(a)
[5, 1, 2, 3]

Методи списків

L.append(x)	Додає елемент **x** до кінця списку **L**
L.count(x)	Повертає кількість входжень елемента **x** до списку **L**
L.extend(m) **L += m**	Додає в кінець списку **L** всі елементи, що ітерується. об'єкта **m**; оператор += робить те саме
L.index(**x, start, end)**	Повертає індекс найпершого (ліворуч) входження елемента **x** у список **L** (або в зріз **[start: end]** списку **L**), інакше збуджує виключення **ValueError**
L.insert(i, x)	Вставляє елемент x до списку **L** у позицію **int(i)**
L.pop()	Видаляє останній елемент зі списку **L** і повертає його як результат
L.pop(i)	Видаляє зі списку **L** елемент з індексом **int(i)** і повертає його як результат
L.remove(x)	Видаляє найперший (ліворуч) знайдений елемент **x** з списку **L** або збуджує виключення **ValueError**, якщо елемент **x** не буде знайдено
L.reverse()	Переставляє у пам'яті елементи списку **L** у зворотному порядку
L.sort(...)	Сортує список **L** у пам'яті. Цей метод приймає ті ж необов'язкові аргументи **key** і **reverse**, як і вбудована функція **sorted()**

bool(x) – перетворення **x** на тип **Boolean**

int(x) – перетворення **x** на тип **Integer**

float(x) – перетворення **x** на тип **Float**

str(x) – перетворення **x** на тип **String**

tuple(x) – перетворення **x** на тип **Tuple**

list(x) – перетворення **x** на тип **List**

set(x) – перетворення **x** на тип **Set**

*__dict(x)__ – перетворення **x** на тип **Dictionary**

* - в цьому випадку кожен елемент об’єкта **x** повинен складатися з двох значень, перше з которих стає ключем у словнику, а друге – його значенням