Зміст

[Встановлення та налаштування оточення](https://textbook.edu.goit.global/python/core-pz9qu8/v1/uk/docs/lesson01/lesson-01#)

[Модуль 1: Типи даних у Python.](https://textbook.edu.goit.global/python/core-pz9qu8/v1/uk/docs/lesson01/lesson-01#) [Теорія](https://textbook.edu.goit.global/python/core-pz9qu8/v1/uk/docs/lesson01/lesson-01)

[Модуль 2: Керуючі конструкції. Винятки](https://textbook.edu.goit.global/python/core-pz9qu8/v1/uk/docs/lesson01/lesson-01#)

[Модуль 3: Функції](https://textbook.edu.goit.global/python/core-pz9qu8/v1/uk/docs/lesson01/lesson-01#)

[Модуль 4: Структури даних](https://textbook.edu.goit.global/python/core-pz9qu8/v1/uk/docs/lesson01/lesson-01#)

[Модуль 5: Просунута робота з рядками](https://textbook.edu.goit.global/python/core-pz9qu8/v1/uk/docs/lesson01/lesson-01#)

[Модуль 6: Робота з файлами.](https://textbook.edu.goit.global/python/core-pz9qu8/v1/uk/docs/lesson01/lesson-01#) [Теорія.](https://textbook.edu.goit.global/python/core-pz9qu8/v1/uk/docs/lesson06/lesson-06) [Домашнє завдання](https://textbook.edu.goit.global/python/core-pz9qu8/v1/uk/docs/lesson06/hw-6)

[Модуль 7: Створення та встановлення власних пакетів, віртуальне оточення](https://textbook.edu.goit.global/python/core-pz9qu8/v1/uk/docs/lesson01/lesson-01#)

[Модуль 8: Можливості деяких вбудованих пакетів Python](https://textbook.edu.goit.global/python/core-pz9qu8/v1/uk/docs/lesson01/lesson-01#)

[Модуль 9: Функції (декоратори, замикання)](https://textbook.edu.goit.global/python/core-pz9qu8/v1/uk/docs/lesson01/lesson-01#)

[Модуль 10: Основи роботи з класами](https://textbook.edu.goit.global/python/core-pz9qu8/v1/uk/docs/lesson01/lesson-01#)

[Модуль 11: 'Магічні' методи](https://textbook.edu.goit.global/python/core-pz9qu8/v1/uk/docs/lesson01/lesson-01#)

[Модуль 12: Серіалізація та копіювання об'єктів](https://textbook.edu.goit.global/python/core-pz9qu8/v1/uk/docs/lesson01/lesson-01#)

[Додаткові матеріали](https://textbook.edu.goit.global/python/core-pz9qu8/v1/uk/docs/lesson01/lesson-01#)

Модуль 1: Типи даних у Python

# Типи даних у Python

## Вступ

Коли ми говоримо про програмування, перше, що спадає на думку — це набір інструкцій у файлі — початковий код.

print("Hello World!")

Наведений вище приклад початкового коду скрипта, який під час запуску виводить у консоль текст "Hello World!".

**Початковий код (source code)** — це набір фраз, слів та спеціальних символів, специфічних для мови програмування, що описують набір інструкцій для комп'ютера.

Але символи, слова та фрази, які складають програму, насправді незрозумілі для машини. Є крок, який виконується після написання програми, який конвертує вихідний код у файлі у набір інструкцій, зрозумілих комп'ютеру. Цим займається спеціальна програма: компілятор або інтерпретатор.

## Системний підхід у програмуванні

Спрощено — у будь-якої програми є:

* Введення (отримання даних)
* Перетворення (обчислення, обробка даних)
* Виведення (виведення даних)

Такі елементи є в будь-якій програмі:

* **Без виведення** програма не має сенсу, тому що ви не дізнаєтесь про те що вона зробила.
* **Немає обчислень** — це не програма, а обчислювальний пристрій (комп'ютер не потрібен).
* **А якщо немає введення** — програма не підключена до реальності. Програми отримують інформацію від користувачів, інших програм, та можуть зчитувати її з файлів на диску.

## Python

**Python** — високорівнева мова програмування загального призначення, стандартна бібліотека якого містить великий набір корисних функцій.

Сьогодні **Python** використовують у таких сферах:

* Web-розробка
* Data Science
* Data Mining (отримання даних)
* Machine Learning (машинне навчання)
* Deep Learning (глибинне навчання)

## Змінні

Змінна — це ім'я (псевдонім) для деякої області пам'яті комп'ютера. Передбачається, що в цій пам'яті лежить корисна для розробника інформація, до якої є необхідність звертатися неодноразово. Python, як і будь-яка інша мова програмування, працює з даними в пам'яті, до яких звертається за допомогою імен (змінних).

**Змінна** — це елемент пам'яті, у якого є ім'я і в якому можуть зберігатися данні.

age = 20  
user1\_age = 30  
user2\_age = 30  
ADULT\_THR = 18  
  
\_do\_not\_use\_this = 0

Стосовно іменування змінних у Python є три суворих правила:

* ім'я змінної у Python може складатися лише з цифр, літер та знаків підкреслення \_;
* ім'я змінної не може починатися з цифри, але може зі знака нижнього підкреслення;
* використання зарезервованих слів у якості імені змінної буде призводити до помилки.

Перелік зарезервованих слів: False, None, True, and, as, assert, async, await, break, class, continue, def, del, elif, else, except, finally, for, from, global, if, import, in, is, lambda, nonlocal, not, or, pass, raise, return, try, while, with, yield

## Інструкція

**Інструкція (statement)** — це пов'язаний набір слів і символів з синтаксису мови, які об'єднуються, щоб виразити одну ідею, одну інструкцію для машини.

x = 2  
y = x + 10

Це приклад інструкцій. У Python інструкції розділяються символом нового рядка (починаються з нового рядка). У прикладі вище дві інструкції: x = 2 та y = x + 10.

Зазвичай, один рядок — одна інструкція. Інструкції можна відокремлювати одну від одної символом ;, але такий синтаксис небажаний, хоча і не викликає помилки.

Python виконує інструкції одну за одною по черзі зліва направо та зверху вниз. У нашому прикладі спочатку виконається x = 2, потім y = x + 10.

## Вирази Python

a = 1  
b = 2  
c = a + b + 10

Виразом називається сукупність змінних, операцій, імен функцій, дужок, який може бути обчислений відповідно до синтаксису Python.

Наприклад, a + b + 10 — це вираз, a — вираз, b — вираз.

## Оператори та операнди

Оператори та операнди — це частини виразу. Оператори — визначають дію, операнди — те, з чим ця дія буде вчинена. У виразі 2 + 3 — числа 3 та 2 будуть операндами, а знак + — оператором.

Для Python визначені такі арифметичні оператори:

|  |  |  |
| --- | --- | --- |
| **Оператор** | **Значення** | **Приклади** |
| + | Додавання | 3 + 3 дає результат 6 4 + 5 дає результат 9 |
| - | Віднімання | 3 - 2 дає результат 1 5 - 5 дає результат 0 |
| \* | Множення | 2 \* 2 дає результат 4 6 \* 3 дає результат 18 |
| / | Ділення | 2 / 2 дає результат 1 5 / 2 дає результат 2.5 6 / 3 дає результат 2 |
| % | Залишок від ділення | 5 % 2 дає результат 1 30 % 3 дає результат 0 |
| // | Ділення без остачі | 5 // 2 дає результат 2 26 // 8 дає результат 3 |
| \*\* | Піднесення до степеня | 2 \*\* 6 дає результат 64 5 \*\* 2 дає результат 25 |
| Операндами для арифметичних операторів є числа. |  |  |

### Порядок виконання операцій у виразі

Python виконує операції у виразі у тому ж порядку, в якому виконуються математичні операції. Спочатку виконується вираз у дужках, потім піднесення до степеня, потім множення, потім додавання та віднімання.

При цьому, будь-який операнд, який є виразом Python, буде виконаний, коли виникне необхідність дізнатися його значення.

x = 8\*\*3 + 4\*(2 + 2)

В результаті виконання цієї інструкції в x буде записано значення 528. Спочатку Python вичислить 8\*\*3, потім (2 + 2), далі результат (2 + 2) помножить на 4 та додасть результат до результату 8\*\*3.

## Коментарі

Навіть ви у своєму коді через місяць мало що згадаєте. Тому, дуже важливим інструментом розробника є коментарі. Також коментарі використовуються у тих випадках, коли виконувати частину програми не потрібно, але видаляти цю частину поки що рано.

Коментарі у Python позначаються символом #, все, що після цього символу, і до початку нового рядка, інтерпретатор просто проігнорує.

*# user\_age = 7*  
*user\_age* = 18  
user\_status = "adult" *# 'adult' for users older 18 and 'child' for younger*

## Типи даних

Змінні можуть бути різного типу (зберігати інформацію у різноманітних форматах):

1. None — порожнє значення і "жодний" тип даних.
2. Числа (Numeric Type)
3. Boolean — бульовий (логічний) тип. Є підтипом цілих чисел.
4. Рядки (Text Sequence Type)

## Тип None

У Python для позначення порожнього значення використовується None

a = None

None використовується тоді, коли потрібно явно повернути якесь значення або створити його (зарезервувати ім'я для чогось), але по суті та за змістом жодної корисної інформації поки що зберегти в цьому значені не можна, навіть якого роду ця інформація (рядок або число, або щось інше) поки неясно. В таких випадках застосовують None.

## Числа

int\_number = 3  
float\_number = 3.3  
complex\_number = 3.3 + 2j

Числа у Python представлені цілими, дробовими і комплексними числами. Python — мова з динамічною типізацією, а значить, при створенні змінної не потрібно визначати її тип.

Інтерпретатор сам визначить, до якого типу вона належить, виходячи з її значення, та визначить, скільки пам'яті для неї виділити.

У прикладі вище інструкція int\_number = 3 створить змінну **int\_number** зі значенням **3** та **int\_number** буде тип int.

Інструкція float\_number = 3.3 створить змінну **float\_number** зі значенням **3.3**, **float\_number** буде мати тип float.

А інструкція complex\_number = 3.3 + 2j створить змінну **complex\_number** зі значенням **3.3 + 2j** — комплексне число.

## Рядки

hello\_string = "Hello"  
world\_string = 'World!'

Рядкові змінні — це впорядковані незмінні набори символів. "Впорядковані" означає, що можна звертатися до символів рядків за індексом, копіювати їх, порівнювати, шукати. "Незмінні" означає, що одного разу створивши рядок, не можна змінити його вміст, можна тільки створити новий.

Для того щоб Python зрозумів, що ви хочете створити рядкову змінну, необхідно взяти символи рядка в лапки. Для цього підійдуть як одинарні лапки ', так і подвійні ".

Наприклад, створимо змінну s, в якій зберігається рядок "Hello, World!":

s = "Hello, World!"

### Операції над рядками

s1 = "Hello,"  
s2 = " World!"  
joined\_string = s1 + s2

Основна операція, яка реалізована для рядків, — це об'єднання рядків (конкатенація). Конкатенація рядків реалізована з використанням оператора додавання + Якщо "додати" два або більше рядків, то в результаті отримаємо об'єднаний рядок. У прикладі вище joined\_string буде дорівнювати "Hello, World!".

Для зручності виведення тексту у Python застосовують спеціальну конструкцію f-рядка.

name = "Oleg"  
hello\_string = f"Hello, {name}!"

f-рядок — це такий шаблон, який дозволяє зручним чином генерувати рядок, підставляючи результат виконання виразів в потрібне місце у шаблоні.

Синтаксично f-рядок відрізняється від звичайного тим, що на початку рядка стоїть символ f. Інтерпретатор зрозуміє, що якщо в такому рядку він зустріне символи фігурних дужок {}, то всередині них міститься вираз, який потрібно виконати і результат підставити в рядок.

У прикладі hello\_string буде дорівнювати "Hello, Oleg!".

## Boolean

Boolean тип даних — підтип цілих чисел, який може приймати тільки два значення, True або False.

### Створення змінних типу bool

Є 2 прості способи створити змінну з типом bool:

* Присвоїти змінній значення True або False
* a = True  
  b = False
* Присвоїти змінній результат виконання логічного виразу, наприклад порівняння:
* age = 18  
  adult1 = age >= 18 *# True*  
    
  *age* = 15  
  adult2 = age >= 18 *# False*

### Вирази порівняння

Для порівняння у Python є оператори < (менше), <= (менше або дорівнює), > (більше), >= (більше або дорівнює), == (дорівнює), != (не дорівнює).

a = 3  
b = 5  
c = a < b *# True*  
*d* = a > b *# False*  
*a* == b *# False*  
*a* != b *# True*

## Вбудовані функції

Стандартна бібліотека Python реалізує ряд функцій, які одразу доступні розробнику без необхідності їх якось імпортувати. Повний перелік можна знайти на [сторінці офіційної документації](https://docs.python.org/3/library/functions.html).

У міру освоєння мови ви знайомитиметеся з цими функціями і вивчатимете їх можливості.

Інструкція виклику функції у Python (не важливо вбудованої або власної) — ім'я функції і круглі дужки, в яких перераховані через кому аргументи, якщо такі є.

print("Hello World!")

В цьому прикладі ми викликали функцію print, яка виводить в консоль те, що їй передадуть як аргумент. В такому випадку аргумент — рядок "Hello World!".

Функція print приймає довільну кількість аргументів і все намагається перетворити в рядок і вивести в консоль. print зручно використовувати, щоб дізнатися вміст тієї або іншої змінної у міру виконання інструкцій.

## Введення даних

Для введення даних з консолі використовується функція input Значення аргументу функції (рядок) буде виводитися в консоль, а далі — буде активовано спеціальний курсор, який означає, що програма чекає введення даних.



Інтерпретатор буде чекати до тих пір, доки не зустріне символ нового рядка (Enter). Після цього весь введений з клавіатури текст буде повернений як результат роботи функції input.

a = input("Рядок запрошення: ")  
# На екрані ви побачите: Рядок запрошення:

Змінна a отримає те значення, яке ввів користувач.

## Приведення типів

Python — це мова з динамічною суворою типізацією. Це означає, що одна й та сама змінна може змінювати свій тип у міру виконання нових інструкцій, але автоматично інтерпретатор не змінюватиме тип даних.

age = input("How old are you? ")

Функція input повертає str — рядок, та порівняти значення age з числом 18 — не можна, бо незрозуміло, яким чином повинно відбуватися таке порівняння.

Але можна перетворити тип змінної age у int за допомогою вбудованої функції int (функція називається так само, як і тип):

age = input("How old are you? ")  
age = int(age)

Для перетворення рядків в числа з дробовою частиною можна використати функцію float:

pi = float('3.14')

Також можна перетворити практично будь-який Python об'єкт в рядок функцією str:

pi\_str = str(3.14)  
age\_str = str(29)

## Запуск програми

Для того щоб запустити програму мовою програмування Python на локальному комп'ютері, необхідно:

* Завантажити та встановити інтерпретатор [Python](https://www.python.org/downloads/);
* Створити текстовий файл з кодом скрипта, наприклад:
* print("Hello World!")  
  input()
* Зберегти створений файл з ім'ям hello.py;
* Запустити скрипт hello.py за допомогою python інтерпретатора.

Скрипт hello.py виводить результат (текст "Hello, World!") у консоль.