Зміст

Що таке Python

Python є популярною мовою програмування. Він був створений Гвідо ван Россумом і випущений у 1991 році.

Використання Python

- Веб-розробка (на стороні сервера)
- Розробка програмного забезпечення
- Математика
- Системний сценарій

Що може зробити Python?

- Python можна використовувати на сервері для створення веб-додатків.
- Python можна використовувати разом із програмним забезпеченням для створення робочих процесів.
- Python може підключатися до систем баз даних і читати або змінювати файли.
- Python підходить для обробки великих даних і виконання складної математики.
- Python дозволяє швидко створювати прототипи або розробляти готове програмне забезпечення.

Чому саме Python?

- Python працює на різних платформах (Windows, Mac, Linux, Raspberry Pi тощо).
- Python має простий синтаксис, подібний до англійської мови.
- Синтаксис Python дозволяє писати програми з меншою кількістю рядків, ніж інші мови програмування.
- Python працює в системі інтерпретатора, що дозволяє виконувати код одразу після написання, забезпечуючи швидке прототипування.
- Python підтримує процедурний, об’єктно-орієнтований і функціональний підходи до програмування.

Добре знати

- Найновішою основною версією Python є **Python 3**, яку ми будемо використовувати в цьому підручнику.
- **Python 2** все ще популярний, хоча й отримує лише оновлення безпеки.
- У цьому посібнику Python буде написаний у текстовому редакторі. Можна також використовувати інтегровані середовища розробки, такі як Thonny, PyCharm, NetBeans або Eclipse, які особливо корисні для роботи з великими проєктами.

Синтаксис Python порівняно з іншими мовами програмування

- Python був розроблений для читабельності та має подібності до англійської мови з впливом математики.
- Python використовує нові рядки для завершення команди, на відміну від інших мов, які часто використовують крапку з комою або круглі дужки.
- Python покладається на відступи (пробіли) для визначення області, таких як цикли, функції та класи, тоді як інші мови використовують фігурні дужки.

Приклад

```
print("Hello, World!")
```

Початок роботи з Python

Інсталяція Python

На багатьох комп'ютерах ПК і Mac вже буде встановлено Python.

Щоб перевірити, чи інстальовано Python на комп'ютері з ОС Windows, знайдіть Python на панелі запуску або запустіть у командному рядку (cmd.exe):

```
C:\Users\Your Name>python --version
```

Щоб перевірити, чи встановлено Python на Linux чи Mac, відкрийте командний рядок (Linux) або термінал (Mac) і введіть:

```
python --version
```

Якщо Python не встановлено, ви можете завантажити його безкоштовно з офіційного веб-сайту: https://www.python.org/

Швидкий старт Python

Python є інтерпретованою мовою програмування. Ви пишете файли Python (`.py`) у текстовому редакторі, а потім виконуєте їх за допомогою інтерпретатора Python.

Запуск файлу Python

Щоб запустити файл Python у командному рядку:

```
C:\Users\Your Name>python helloworld.py
```

Де `helloworld.py` — це ім'я вашого файлу Python.

Написання першої програми

Створіть файл `helloworld.py` у будь-якому текстовому редакторі з таким вмістом:

```
print("Hello, World!")
```

Збережіть файл. Відкрийте командний рядок, перейдіть до каталогу, де збережено файл, і виконайте:

```
C:\Users\Your Name>python helloworld.py
```

Вихід має бути таким:

```
Hello, World!
```

Вітаємо, ви написали та виконали свою першу програму на Python!

Редактор Python від W3Schools

У нас є онлайн-редактор Python, де ви можете виконати свій код і побачити результат. Спробуйте:

```
print("Hello, World!")
```

Цей редактор використовуватиметься в усьому посібнику для демонстрації різних аспектів Python.

Версія Python

Щоб перевірити версію Python, імпортуйте модуль `sys` і виконайте:

```
import sys
print(sys.version)
```

Детальніше про імпорт модулів ви дізнаєтеся в розділі «Модулі Python».

Командний рядок Python

Для швидкого тестування невеликого коду Python можна скористатися інтерактивним командним рядком Python.

Введіть у командному рядку Windows, Mac або Linux:

```
C:\Users\Your Name>python
```

Або, якщо команда `python` не працює, спробуйте:

```
C:\Users\Your Name>py
```

У командному рядку Python ви можете виконувати будь-який код, наприклад:

```
>>> print("Hello, World!")
Hello, World!
```

Щоб вийти з командного рядка Python, введіть:

```
exit()
```

Синтаксис Python

Виконати синтаксис Python

Як ми дізналися на попередній сторінці, синтаксис Python можна виконати, написавши безпосередньо в командному рядку:

```
>>> print("Hello, World!")
Hello, World!
```

На цій сторінці

- Вправи

Або створивши файл Python на сервері з розширенням `.py` і запустивши його в командному рядку:

```
C:\Users\Your Name>python myfile.py
```

Відступ Python

Відступ відноситься до пробілів на початку рядка коду.

Якщо в інших мовах програмування відступи в коді призначені лише для зручності читання, у Python відступи дуже важливі. Python використовує відступи для позначення блоку коду.

Приклад:

```
if 5 > 2:
    print("Five is greater than two!")
```

Python видасть вам помилку, якщо ви пропустите відступ:

Синтаксична помилка:

```
if 5 > 2:
print("Five is greater than two!")
```

Кількість пробілів залежить від вас як програміста, найчастіше використовується чотири, але має бути принаймні один.

Приклад:

```
if 5 > 2:
        print("Five is greater than two!")
if 5 > 2:
                print("Five is greater than two!")
```

Ви повинні використовувати однакову кількість пробілів в одному блоці коду, інакше Python видасть вам помилку:

Синтаксична помилка:

```
if 5 > 2:
        print("Five is greater than two!")
                print("Five is greater than two!")
```

Змінні Python

У Python змінні створюються, коли ви присвоюєте їм значення:

Приклад:

```
# Змінні в Python:
x = 5
y = "Hello, World!"
```

Python не має команди для оголошення змінної.

> Ви дізнаєтеся більше про змінні в розділі «Змінні Python».

Коментарі Python

Python має можливість коментування для документації в коді.

Коментарі починаються з #, і Python відобразить решту рядка як коментар:

Приклад:

```
# This is a comment.
print("Hello, World!")
```

Коментарі в Python

Коментарі використовуються для пояснення коду Python. Вони допомагають зробити код більш читабельним або запобігти виконанню певних частин коду під час тестування.

Створення коментаря

Коментарі починаються з #, і Python їх ігнорує:

```
# Це коментар
print("Hello, World!")
```

Коментарі також можна розміщувати в кінці рядка:

```
print("Hello, World!") # Це коментар
```

Коментарі можуть використовуватися для тимчасового вимкнення коду:

```
# print("Hello, World!")
print("Cheers, Mate!")
```

Багаторядкові коментарі

Python не має спеціального синтаксису для багаторядкових коментарів. Ви можете використовувати # на початку кожного рядка:

```
# Це коментар
# написаний у
# кілька рядків
print("Hello, World!")
```

Або, як альтернативу, можна використовувати багаторядковий рядок (трипл-квоти). Якщо рядок не присвоєно змінній, Python його ігноруватиме:

```
"""
Це коментар,
написаний у
кілька рядків
"""
print("Hello, World!")
```

> **Примітка:** Використання багаторядкових рядків для коментарів є не зовсім стандартним підходом, але це працює, якщо рядок не присвоюється змінній.

Змінні Python

Змінні

Змінні є контейнерами для зберігання значень даних.

Створення змінних

Python не має команди для оголошення змінної. Змінна створюється в той момент, коли ви вперше присвоюєте їй значення.

Приклад:

```
x = 5
y = "John"
print(x)
print(y)
```

Змінні не потрібно оголошувати з певним типом, і вони навіть можуть змінювати тип після того, як їх було встановлено.

Приклад:

```
x = 4       # x is of type int
x = "Sally" # x is now of type str
print(x)
```

Кастинг

Якщо ви хочете вказати тип даних змінної, це можна зробити за допомогою приведення.

Приклад:

```
x = str(3)    # x will be '3'
y = int(3)    # y will be 3
z = float(3)  # z will be 3.0
```

Отримайте тип

Ви можете отримати тип даних змінної за допомогою функції `type()`.

Приклад:

```
x = 5
y = "John"
print(type(x))
print(type(y))
```

Одинарні чи подвійні лапки?

Рядкові змінні можна оголошувати за допомогою одинарних або подвійних лапок.

Приклад:

```
x = "John"
# is the same as
x = 'John'
```

З урахуванням регістру

Імена змінних чутливі до регістру.

Приклад:

```
a = 4
A = "Sally"
# A will not overwrite a
```

Імена змінних

Імена змінних можуть бути короткими (наприклад, `x` і `y`) або більш описовими (`age`, `carname`, `total_volume`).

Правила для змінних Python:

1. Ім'я змінної має починатися з літери або символу підкреслення.
2. Ім'я змінної не може починатися з числа.
3. Ім’я змінної може містити лише буквено-цифрові символи та підкреслення (`A-z`, `0-9`, `_`).
4. Назви змінних чутливі до регістру (`age`, `Age`, і `AGE` — це різні змінні).
5. Ім'я змінної не може бути жодним із ключових слів Python.

Правильні назви змінних:

```
myvar = "John"
my_var = "John"
_my_var = "John"
myVar = "John"
MYVAR = "John"
myvar2 = "John"
```

Неправильні назви змінних:

```
2myvar = "John"
my-var = "John"
my var = "John"
```

Багатослівні імена змінних

Для підвищення читабельності багатослівних імен змінних можна використовувати кілька стилів:

- **Верблюжа справа:** `myVariableName = "John"`
- **Кейс Паскаль:** `MyVariableName = "John"`
- **Зміїна справа:** `my_variable_name = "John"`

Призначення значень змінним

Багато значень для кількох змінних

Python дозволяє призначати значення декільком змінним в одному рядку.

Приклад:

```
x, y, z = "Orange", "Banana", "Cherry"
print(x)
print(y)
print(z)
```

Одне значення для кількох змінних

Можна призначити те саме значення кільком змінним в одному рядку.

Приклад:

```
x = y = z = "Orange"
print(x)
print(y)
print(z)
```

Розпакування колекції

Python дозволяє видобувати значення зі списків, кортежів тощо у змінні.

Приклад:

```
fruits = ["apple", "banana", "cherry"]
x, y, z = fruits
print(x)
print(y)
print(z)
```

Вихідні змінні

Функція `print()` використовується для виведення змінних.

Приклад:

```
x = "Python is awesome"
print(x)
```

Виведення кількох змінних

Приклад:

```
x = "Python"
y = "is"
z = "awesome"
print(x, y, z)
```

Використання оператора `+`

Приклад:

```
x = "Python "
y = "is "
z = "awesome"
print(x + y + z)
```

Для чисел `+` працює як математичний оператор:

```
x = 5
y = 10
print(x + y)
```

Глобальні змінні

Змінні, створені поза функцією, є глобальними і можуть використовуватися як усередині функцій, так і поза ними.

Приклад:

```
x = "awesome"

def myfunc():
    print("Python is " + x)
```

```
myfunc()
```

Локальні змінні

Якщо змінна створена всередині функції, вона є локальною.

Приклад:

```
x = "awesome"

def myfunc():
    x = "fantastic"
    print("Python is " + x)

myfunc()

print("Python is " + x)
```

Ключове слово `global`

Щоб створити або змінити глобальну змінну всередині функції, використовуйте ключове слово `global`.

Приклад:

```
def myfunc():
    global x
    x = "fantastic"

myfunc()

print("Python is " + x)
```

Типи даних Python

Вбудовані типи даних

У програмуванні тип даних є важливим поняттям. Змінні можуть зберігати дані різних типів, і різні типи можуть виконувати різні дії.

За замовчуванням Python має вбудовані такі типи даних у цих категоріях:

- **Тип тексту**: `str`
- **Числові типи**: `int`, `float`, `complex`
- **Типи послідовностей**: `list`, `tuple`, `range`

- **Тип відображення**: `dict`
- **Типи наборів**: `set`, `frozenset`
- **Логічний тип**: `bool`
- **Двійкові типи**: `bytes`, `bytearray`, `memoryview`
- **Немає Тип**: `NoneType`

Отримання типу даних

Ви можете отримати тип даних будь-якого об’єкта за допомогою функції `type()`:

Приклад

Отримайте тип даних змінної `x`:

```
x = 5
print(type(x))
```

Встановлення типу даних

У Python тип даних встановлюється, коли ви присвоюєте значення змінній:

Example	Data Type
`x = "Hello World"`	`str`
`x = 20`	`int`
`x = 20.5`	`float`
`x = 1j`	`complex`
`x = ["apple", "banana", "cherry"]`	`list`
`x = ("apple", "banana", "cherry")`	`tuple`
`x = range(6)`	`range`
`x = {"name" : "John", "age" : 36}`	`dict`
`x = {"apple", "banana", "cherry"}`	`set`
`x = frozenset({"apple", "banana", "cherry"})`	`frozenset`
`x = True`	`bool`
`x = b"Hello"`	`bytes`
`x = bytearray(5)`	`bytearray`
`x = memoryview(bytes(5))`	`memoryview`
`x = None`	`NoneType`

Встановлення конкретного типу даних

Якщо ви хочете вказати тип даних, ви можете використовувати наступні функції конструктора:

Example	Data Type
`x = str("Hello World")`	`str`
`x = int(20)`	`int`
`x = float(20.5)`	`float`
`x = complex(1j)`	`complex`
`x = list(("apple", "banana", "cherry"))`	`list`
`x = tuple(("apple", "banana", "cherry"))`	`tuple`
`x = range(6)`	`range`
`x = dict(name="John", age=36)`	`dict`
`x = set(("apple", "banana", "cherry"))`	`set`
`x = frozenset(("apple", "banana", "cherry"))`	`frozenset`
`x = bool(5)`	`bool`
`x = bytes(5)`	`bytes`
`x = bytearray(5)`	`bytearray`
`x = memoryview(bytes(5))`	`memoryview`

Числа в Python

У Python існує три типи чисел:

- `int`
- `float`
- `complex`

Змінні числового типу створюються, коли ви присвоюєте їм значення:

```
# Приклад
x = 1    # int
y = 2.8  # float
z = 1j   # complex
```

Щоб перевірити тип будь-якого об’єкта в Python, використовуйте функцію `type()`:

```
# Приклад
print(type(x))
print(type(y))
print(type(z))
```

Int

`Int`, або ціле число, є цілим числом, додатним або від'ємним, без десяткових знаків, необмеженої довжини.

```
# Приклад: Цілі числа
x = 1
y = 35656222554887711
z = -3255522

print(type(x))
print(type(y))
print(type(z))
```

Float

Число з плаваючою комою або «float» — це число, додатне чи від'ємне, що містить один або більше десяткових знаків.

```
# Приклад: Float
x = 1.10
y = 1.0
z = -35.59

print(type(x))
print(type(y))
print(type(z))
```

Число з плаваючою точкою також може бути науковим числом із літерою `e`, яка вказує ступінь числа 10.

```
# Приклад: Float у науковій нотації
x = 35e3
y = 12E4
z = -87.7e100

print(type(x))
print(type(y))
print(type(z))
```

Complex

Комплексні числа записуються через `j` як уявну частину:

```
# Приклад: Комплексні числа
x = 3+5j
y = 5j
z = -5j

print(type(x))
print(type(y))
print(type(z))
```

Перетворення типу

Ви можете конвертувати з одного типу в інший за допомогою методів `int()`, `float()` та `complex()`:

```
# Приклад: Перетворення з одного типу в інший
x = 1    # int
y = 2.8  # float
z = 1j   # complex

# Convert from int to float:
a = float(x)

# Convert from float to int:
b = int(y)

# Convert from int to complex:
c = complex(x)

print(a)
print(b)
print(c)

print(type(a))
print(type(b))
print(type(c))
```

> **Примітка:** Ви не можете перетворити комплексні числа на інший тип чисел.

Випадкове число

Python не має функції `random()` для створення випадкових чисел, але Python має вбудований модуль під назвою `random`, який можна використовувати для створення випадкових чисел:

```
# Приклад: Імпортуйте модуль random і відобразіть випадкове число від 1 до
9
```

```
import random

print(random.randrange(1, 10))
```

Кастинг у Python

Указання типу змінної

Іноді може знадобитися вказати тип змінної. Це можна зробити за допомогою кастингу. Python є об'єктно-орієнтованою мовою, і як така вона використовує класи для визначення типів даних, включаючи примітивні типи.

Тому кастинг у Python виконується за допомогою функцій-конструкторів:

- `int()` — створює ціле число з:
  - літералу цілого числа,
  - літералу з плаваючою точкою (шляхом видалення всіх десяткових знаків),
  - літералу рядка (за умови, що рядок представляє ціле число).
- `float()` — створює число з плаваючою точкою з:
  - цілого літералу,
  - літералу з плаваючою точкою,
  - рядкового літералу (за умови, що рядок представляє число з плаваючою точкою або ціле число).
- `str()` — створює рядок із різноманітних типів даних, включаючи:
  - рядки,
  - цілі літерали,
  - літерали з плаваючою точкою.

Приклади

Цілі числа:

```
x = int(1)   # x буде 1
y = int(2.8) # y буде 2
z = int("3") # z буде 3
```

Числа з плаваючою точкою:

```
x = float(1)     # x буде 1.0
y = float(2.8)   # y буде 2.8
```

```
z = float("3")   # z буде 3.0
w = float("4.2") # w буде 4.2
```

Рядки:

```
x = str("s1") # x буде 's1'
y = str(2)    # y буде '2'
z = str(3.0)  # z буде '3.0'
```