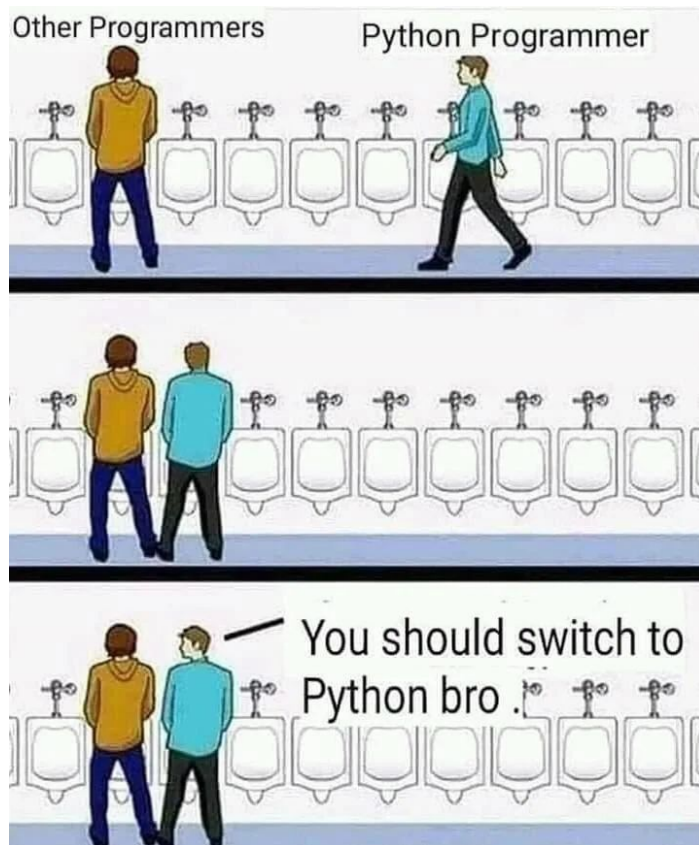

01. Въведение в Python

— 23 февруари 2026 —

Въведение



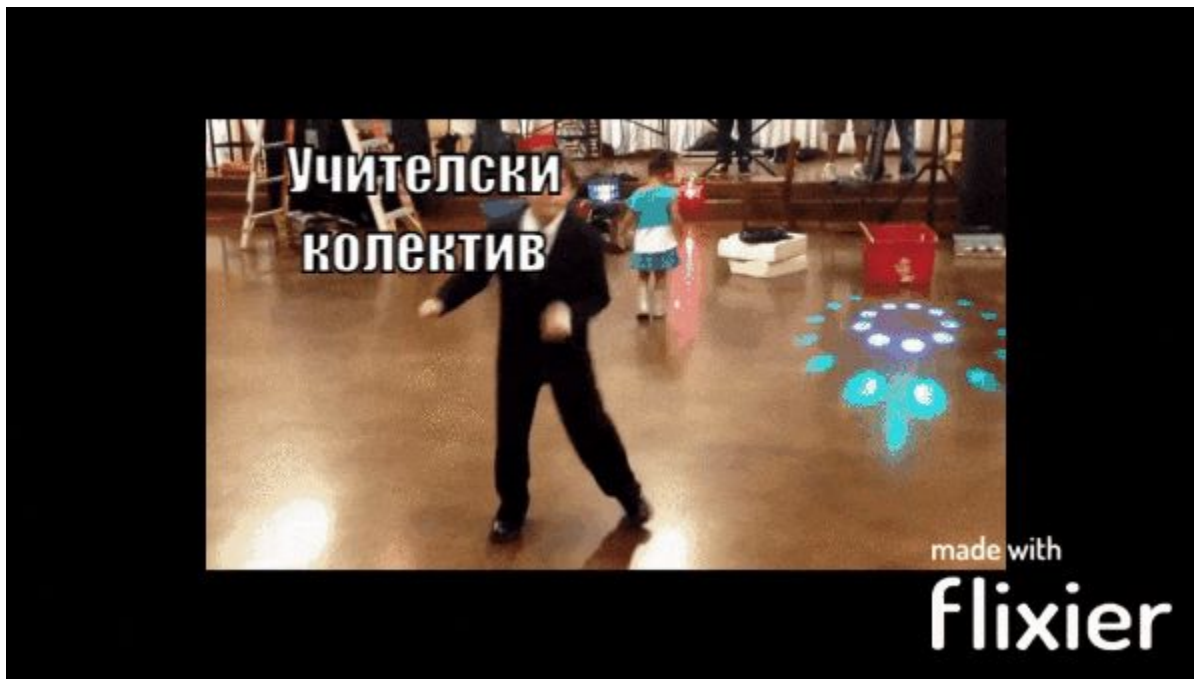
Организационен слайд - какво е това?



Py-chart на регистрираните потребители към 17:00 днес - 45%

Домашно

Ако не сте се регистрирали в сайта, направете го. Който не го е направил - изпуснали сте една точка, скоро ще имате предизвикателство, а след още няколко дни - домашно.



По темата...



Произход на името



Среда за програмиране

- Който е с Windows - светият граал е наличен на <https://www.python.org/downloads/>
- Или "download python"@Google
- По време на инсталация - "Add Python to PATH"
- Който е с Linux - `sudo apt install python3.13` (или еквивалента на това)
- Инсталирайте **поне** Python 3.12.* (не държим на 3.14, за 99.9% от нещата, които ще ви покажем - няма да има значение)

Новини от света

Download the latest version for Windows

[Download Python install manager](#)

Or get the standalone installer for [Python 3.14.3](#)

Looking for Python with a different OS? Python for [Windows](#),
[Linux/Unix](#), [macOS](#), [Android](#), [other](#)

Want to help test development versions of Python 3.15? [Pre-releases](#),
[Docker images](#)



Active Python releases

Python version	Maintenance status		First released	End of support	Release schedule
3.15	pre-release	Download	2026-10-01 (planned)	2031-10	PEP 790
3.14	bugfix	Download	2025-10-07	2030-10	PEP 745
3.13	bugfix	Download	2024-10-07	2029-10	PEP 719
3.12	security	Download	2023-10-02	2028-10	PEP 693
3.11	security	Download	2022-10-24	2027-10	PEP 664
3.10	security	Download	2021-10-04	2026-10	PEP 619
3.9	end-of-life, last release was 3.9.25	Download	2020-10-05	2025-10-31	PEP 596

Среда за програмиране

- VSCode
- PyCharm
- Sublime
- IDLE
- Notepad (++)
- CMD / Terminal

- *vim (само за смелите)*
- *emacs (само за тези с железни кутрета)*

Къде отива кодът?

- Код се пише в `.py` файлове (например `gameoflife.py`)
- Изпълнява се с `python gameoflife.py`
- Можем да пишем код интерактивно като пуснем `python` без аргументи

Python е предсказуем

Когато не сте сигурни, просто пробвайте.

```
$ python
Python 3.10.7 (tags/v3.10.7:6cc6b13, Sep  5 2022, 14:08:36)
[MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 13 + 10
23
>>> a = 13
>>> b = a + 10
>>> print(b)
23
>>> a * 2
26
>>> a ** 2
169
>>> "hello" + ', ' + "world"
'hello, world'
```

Един ред код

```
>>> my_var = 'spam'.upper()  
>>> print(my_var)  
SPAM
```

- Съдържа един израз
- **никога не завършва с ;**
- Всичко след # е коментар
- Малко уговорки:
 - `print` е функция
 - `нещо_си.upper()` е метод
 - И двете ще обясним малко по-нататък в детайли
 - Все пак трябва да са достатъчно интуитивни

Първа помощ

В интерактивната конзола `help()` показва документацията на всяка функция, клас или тип.

```
>>> help(str)
```

```
>>> help(5)
```

```
>>> help(SomeClass)
```

```
>>> help(some_function)
```

Типове: int

- Цели числа - положителни и отрицателни
- Стандартни операции: +, -, *, /, //, %, ** (степенуване)
- Без максимален размер
- Може да пробваме $2^{**}4^{**}3$

Типове: float

- Числа с плаваща запетая (точка?) - 3.14159
- По всичко друго приличат на целите числа
- $0.1 + 0.2 = ?$

Типове: complex

- Още един вид число - комплексно
- Пишат се така: $(2+3j)$
- Да, j , а не i

Типове: complex

```
>>> a = 1j * 1j  
(-1+0j)
```

Типове: str

```
>>> "hello".upper()
"HELLO"
>>> len("абвгдеж")
7
>>> "hello"[1]
"e"
>>> help(str)
```

- Текстови низове с произволна дължина
- Единични или двойни кавички
- Unicode навсякъде!!!!
- Поддържат `\n`, `\t` и пр.

Типове: bool

- True и False
- **NB!** главните букви

Типове: None

- Като `null` в другите езици
- Когато една функция не върне нищо, тя връща `None`
- Използвайте го, за да кажете "нищо" или "няма"

Типове

- Всяка стойност има тип

```
>>> type(5.5)
```

```
<class 'float'>
```

```
>>> type("Питона искаш ли да ти покажа?")
```

```
<class 'str'>
```

- Включително и функциите

```
>>> type(len)
```

```
<class 'builtin_function_or_method'>
```

Типове

- Може би се питате “тогава къде ми е `int`-а, преди `a = 13`”?
- Всяка стойност е обект и има клас (включително функциите)
- Всъщност **ВСИЧКО** в Python е обект (включително функциите **и типовете!**)
- Можем да проверим типа на един обект с функцията `type()`

Типове

type е функция

⇒ type е обект

⇒ type си има тип

```
>>> type(type)
<class 'type'>
```

```
>>> type(type(type(type)))
<class 'type'>
```

It's turtles all the way down...



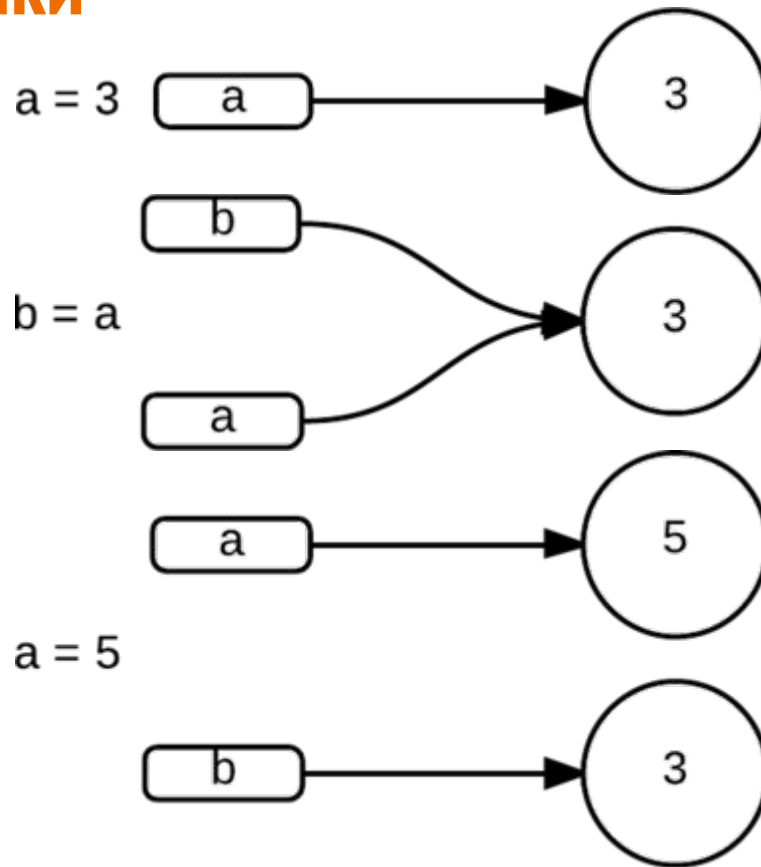
Имена

Можем да присвоим стойност на име. Така създаваме променлива.

Python е динамичен език - стойностите имат тип, но не и имената.

```
>>> a = 5
>>> type(a)
<class 'int'>
>>> a = 'test'
>>> type(a)
<class 'str'>
```

Имена в картинке



Динамично типизиране ** 2

```
>>> a = 2
>>> b = 2.1
>>> type(a)
<class 'int'>
>>> type(b)
<class 'float'>
>>> b > a
True
```

```
>>> message = 'python goes b' + 'r' * 10
>>> message
'python goes brrrrrrrrrrr'
```

Python е умен

C++: Can not compare
float and int

Python:



Структури от данни

- Списък - `list`
- Tuple - `tuple` а.к.а кортеж
- Речник - `dict`
- Множество - `set`
- `help` е ваш пръв приятел!

Списъци

```
>>> my_list = []          # препоръчително!  
>>> my_list = list()     # иначе може и така
```

- Списък = `list` = масив = `array`
- Mutable и без фиксирана дължина
- Бързи за търсене по индекс, бавни за търсене по стойност
- Гарантиран ред
- Не е нужно елементите да са от еднакъв тип (т.е. списъците са хетерогенни)

Списъци

```
>>> my_list = []  
>>> my_list.append('word')  
>>> my_list.append(5)  
>>> my_list.append(False)
```

```
>>> my_list[1] == 5  
True
```

```
>>> my_list[0]  
'word'
```

Списъци

```
>>> my_other_list = ['foo', 'bar', 'spam']
```

```
>>> len(my_other_list)
```

```
3
```

```
>>> del my_other_list[1]
```

```
>>> my_other_list
```

```
['foo', 'spam']
```

```
>>> 'foo' in my_other_list
```

```
True
```

```
>>> False in my_list
```

```
True
```

```
>>> 'spam' in my_list
```

```
False
```

Списъци - slicing

```
cute_animals = ['cat', 'raccoon', 'panda', 'red panda', 'marmot']
```

```
cute_animals[1:3] # ['raccoon', 'panda']
```

```
cute_animals[-1] # 'marmot'
```

```
cute_animals[1:-1] # ['raccoon', 'panda', 'red panda']
```

```
cute_animals[::-1] # ['marmot', 'red panda', 'panda', 'raccoon', 'cat']
```

```
cute_animals[-1:0:-1] # ['marmot', 'red panda', 'panda', 'raccoon']
```

```
cute_animals[-1:0:-2] # ['marmot', 'panda']
```

tuple

- tuple = кортеж = n-торка
- Immutable
- Гарантиран ред
- Хетерогенни

tuple

```
>>> args = (9.8, 3.14, 2.71)
>>> args[2]
2.71
>>> args[1:]
(3.14, 2.71)
```

Wait, what?

Това подозрително много прилича на списък...

Mutable vs immutable

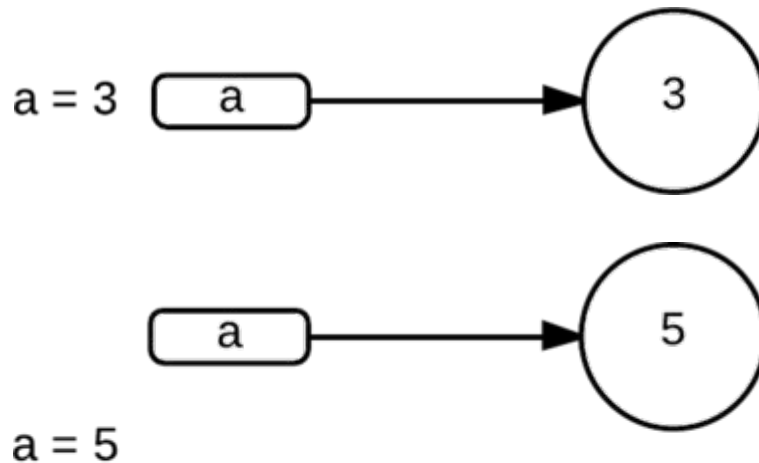
```
a = 3
```

```
a += 2
```

```
a # 5
```

- Immutable са стойностите, които не могат да бъдат променени.
- Този код не променя стойността на 3, а кара `a` да сочи към друга стойност - 5.
- Immutable са числата, низовете, `tuple`-ите, `True`, `False`, `None` etc.

Имена в картинке (пак)



(Или в нашай случай - $a += 2$)

Mutable vs immutable (2)

```
a = [1, 2, 3]
```

```
a.append(4)
```

```
a # [1, 2, 3, 4]
```

```
a[2] = 5
```

```
a # [1, 2, 5, 4]
```

- Този код променя списъка, към който сочи **a**.
- Списъците са mutable.

Mutable vs immutable (3)

```
a = (1, 2, 3)
a[2] = 5
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
```

```
    TypeError: 'tuple' object does not support item assignment
```

- Този код гърми.
- tuple-ите са immutable.
- But... Why?

tuple (cont.)

- “Описват” логически свързани стойности
- Нека вземем за пример вектор:
 - Имаме вектора (4, 2)
 - Искаме да променим стойността на първи индекс от 2 на 3
 - Грешка, вече нямаме същия вектор, (4, 3) е изцяло различен такъв, а не просто лека модификация на предния
- Намеква за намерението тези данни да не се променят, един вид “константи”
- Често се използват се, за да подадете или върнете няколко стойности от функция, когато специален клас би бил твърде много
- Tuple от един елемент - със запетайка на края:
(`'This is the tale of captain Jack Sparrow.'`,)
- Може и без скобите

Въпроси?