# Python

— Гарри, ты отлично говоришь на змеином.
— Спс.

# Программирование в научных целях

## Мы не программисты!

Но мы ленивы!

Мы люди и делаем ошибки!

И мы не любим делать однообразные дела! Особенно, если такое дело надо сделать 10000+ раз.

Вывод:

Нужен (простой) способ автоматизировать рутину

# Какие варианты у нас есть?


Нанять работников
за еду

После 1861 года – нельзя

# Какие варианты у нас есть?

Языки общего назначения

- C
- C++
- Python
- D
- Fortran
- Rust
- …

Предметно-ориентированные языки

- Julia
- R
- Matlab
- Mathematica
- …

# Что такое Python?

**Python** (МФА: [ˈpʌɪθ(ə)n]; в русском языке встречаются названия **пито́н**[17] или **па́йтон**[18]) — высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью[19][20], ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ[21]. Язык является полностью объектно-ориентированным в том плане, что всё является объектами[19]. Необычной особенностью языка является выделение блоков кода отступами[22]. Синтаксис ядра языка минималистичен, за счёт чего на практике редко возникает необходимость обращаться к документации[21]. Сам же язык известен как интерпретируемый и используется в том числе для написания скриптов[19]. Недостатками языка являются зачастую более низкая скорость работы и более высокое потребление памяти написанных на нём программ по сравнению с аналогичным кодом, написанным на компилируемых языках, таких как C или C++[19][21].

https://ru.wikipedia.org/wiki/Python

# Что такое Python?

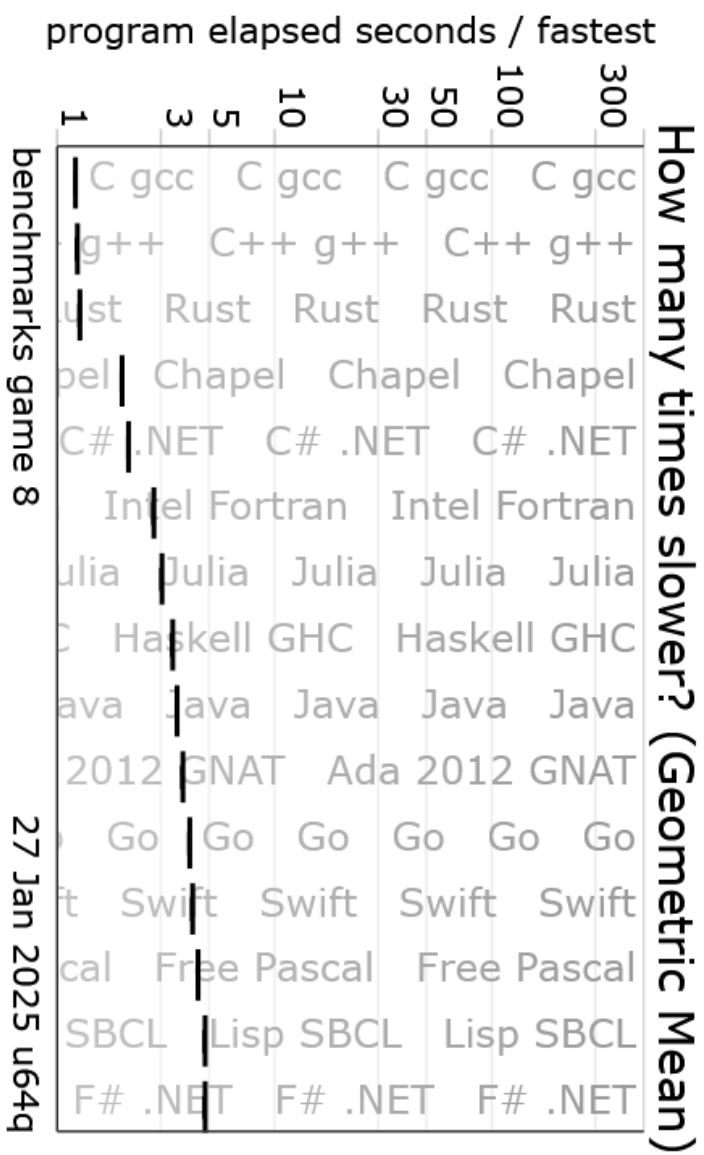
**Python** — высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью,

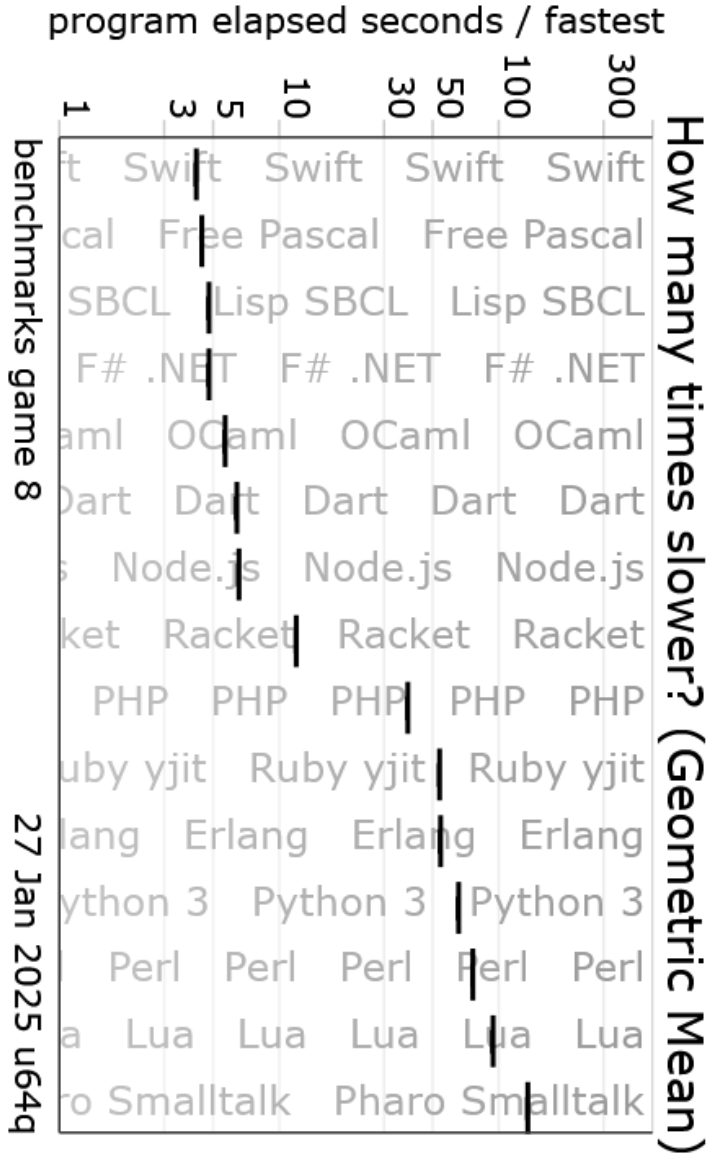
интерпретируемый и используется в том числе для написания скриптов.

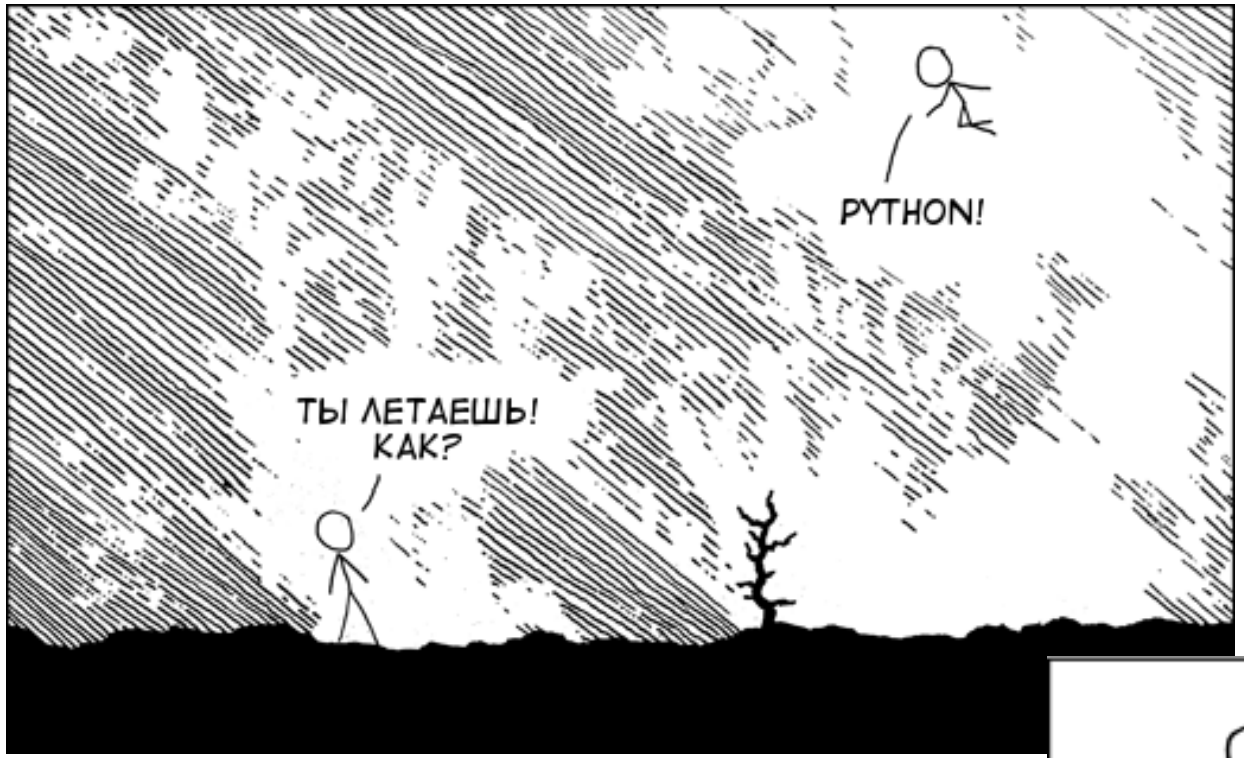
Недостатками языка являются зачастую более низкая скорость работы

https://ru.wikipedia.org/wiki/Python

# Скорость


How many times slower? (Geometric Mean)
program elapsed seconds / fastest
1
3
5
10
30
50
100
300
benchmarks game 8
27 Jan 2025 u64q
C gcc
C++ g++
Rust
Chapel
C# .NET
Intel Fortran
Julia
Haskell GHC
Java
Ada 2012 GNAT
Go
Swift
Free Pascal
Lisp SBCL
F# .NET


How many times slower? (Geometric Mean)
program elapsed seconds / fastest
1
3
5
10
30
50
100
300
benchmarks game 8
27 Jan 2025 u64q
Swift
Free Pascal
Lisp SBCL
F# .NET
OCaml
Dart
Node.js
Racket
PHP
Ruby yjit
Erlang
Python 3
Perl
Lua
Pharo Smalltalk


PYTHON!
ТЫ ЛЕТАЕШЬ! КАК?


Я ВЫУЧИЛ ЕГО ВЧЕРА ВЕЧЕРОМ! ВСЁ ТАК ПРОСТО!
HELLO WORLD — ЭТО ПРОСТО
print "Hello, world!"

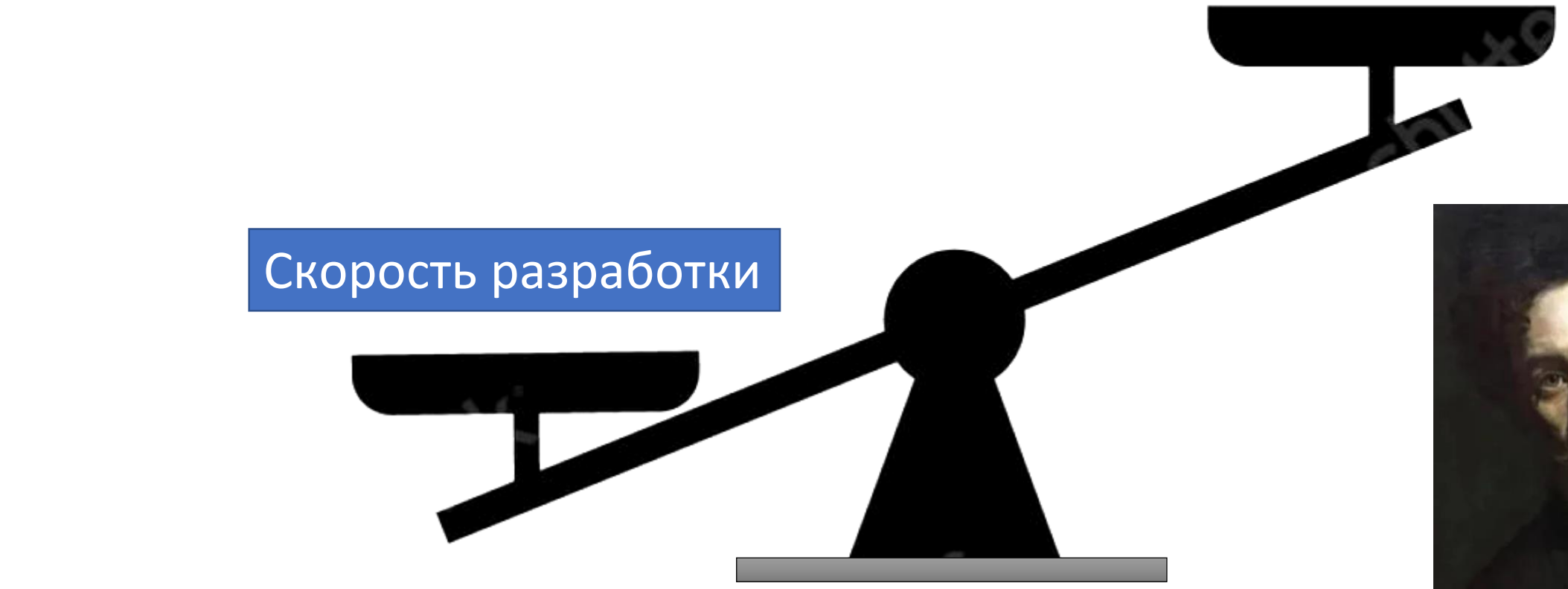

НУ НЕ ЗНАЮ... ДИНАМИЧЕСКАЯ ТИПИЗАЦИЯ? ПРОБЕЛЫ?
ПРИСОЕДИНЯЙСЯ! ПИСАТЬ ПРОГРАММЫ СНОВА СТАЛО ПРИКОЛЬНО. ТАМ ЦЕЛЫЙ НОВЫЙ МИР!
НО КАК ТЫ ЛЕТАЕШЬ?

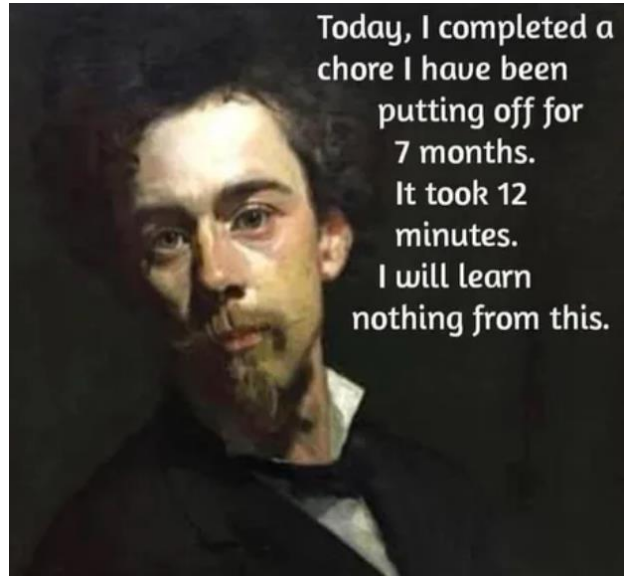

Я ПРОСТО НАПИСАЛ
import antigravity
И ВСЁ?
... НУ ЕЩЁ Я РАДИ СРАВНЕНИЯ ПЕРЕПРОБОВАЛ ВСЁ, ЧТО БЫЛО В АПТЕЧКЕ.
НО Я ДУМАЮ, ЧТО ЭТО PYTHON.

# Почему Python?

Скорость выполнения

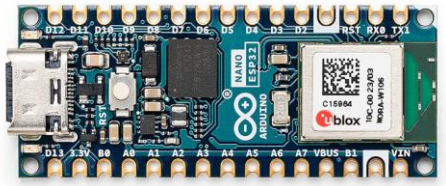
Скорость разработки





# Примеры применения в современной физике

- Обработка (в том числе статистическая) данных, их визуализация
- Численное моделирование – в задачах, менее масштабных, чем high performance computing
  - И даже тут Python хорош для прототипирования
- Символьные вычисления (но не дотягивает до лидеров отрасли)
- Автоматизация эксперимента – например, работа с USB устройствами (датчики etc)
  - MicroPython есть даже для Arduino



# Среды разработки

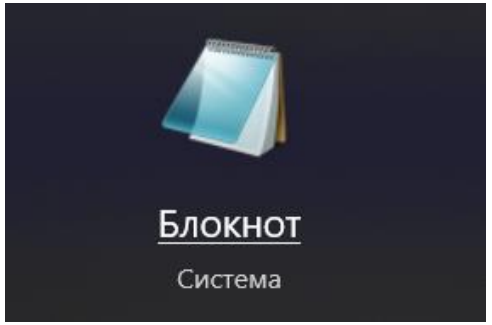

PyCharm

ANACONDA.

Visual Studio Code


python™


spyder




Sublime Text

# С чем будем работать?

Anaconda – большое собрание полезных пакетов

https://www.anaconda.com/download/success

Внутри:

- Python 3.13+
- Numpy, Scipy, Matplotlib, Sympy, …

| | Anaconda Distribution | Miniconda |
|---|---|---|
| Created and published by Anaconda | Yes | Yes |
| Has conda | Yes | Yes |
| Has Anaconda Navigator | Yes | No |
| # of packages | 300+ | < 70 |
| Install space required | ~4.4 GB | ~480 MB |

Среда: Jupyter Notebook внутри VS Code

Нет лишнего места на жестком диске? Miniconda / **Miniforge**

# Ссылки на прикладное ПО

Anaconda https://www.anaconda.com/download/success

Git https://git-scm.com/

VS Code https://code.visualstudio.com/Download

LAMMPS https://github.com/lammps/lammps/releases

OVITO https://www.ovito.org/#download

VESTA https://jp-minerals.org/vesta/en/download.html

# Окружения


conda/mamba + pip

Зачем?

- Dependency hell
  - A[B ⩾ 2.0]
  - C[B<2.0]
- Работа в команде
- Борьба с помойкой в base

# В терминале

conda list

conda install …

conda update …

pip list

pip install …

pip install –U …

```
# conda list
# packages in environment at C:\Mambaforge:
#
# Name                    Version                   Build  Channel
_openmp_mutex             4.5                       2_gnu    conda-forge
_python_abi3_support      1.0                     hd8ed1ab_2    conda-forge
adaptive                  1.4.1                 pyhe01879c_0    conda-forge
anyio                     4.10.0                pyhe01879c_0    conda-forge
archspec                  0.2.5                 pyhd8ed1ab_0    conda-forge
argon2-cffi               25.1.0                pyhd8ed1ab_0    conda-forge
argon2-cffi-bindings      25.1.0              py312he06e257_0    conda-forge
arrow                     1.3.0                 pyhd8ed1ab_1    conda-forge
ase                       3.26.0                pyhd8ed1ab_0    conda-forge
asteval                   1.0.6                 pyhd8ed1ab_0    conda-forge
asttokens                 3.0.0                 pyhd8ed1ab_1    conda-forge
attrs                     25.3.0                pyh71513ae_0    conda-forge
beautifulsoup4            4.13.5                pyha770c72_0    conda-forge
```

…

```
jupyterlab_widgets        3.0.15                pyhd8ed1ab_0    conda-forge
kiwisolver                1.4.9               py312h78d62e6_0    conda-forge
krb5                      1.21.3                  hdf4eb48_0    conda-forge
lark                      1.2.2                 pyhd8ed1ab_1    conda-forge
latex2pydata              0.5.0                       pypi_0    pypi
latexminted               0.6.0                       pypi_0    pypi
latexrestricted           0.6.2                       pypi_0    pypi
lcms2                     2.17                    hbcf6048_0    conda-forge
lerc                      4.0.0                   h6470a55_1    conda-forge
libarchive                3.7.7                   h5343c79_4    conda-forge
libblas                   3.9.0                34_h5709861_mkl    conda-forge
```

# Настройка окружения файлом

conda: environment.yml

conda env create -f environment.yml

conda activate …

```yaml
name: kmc4
channels:
  - conda-forge
dependencies:
  - python=3.8
  - numpy=1.21
  - matplotlib=3.4.2
  - scipy=1.7
  - pandas=1.3.0
  - pyarrow=4.0.1
  - pytables=3.6.1

  - tqdm
  - pyyaml
  - colorama
  - cmaps
  - rich

  - pip
  - pip:
    - plotille

```

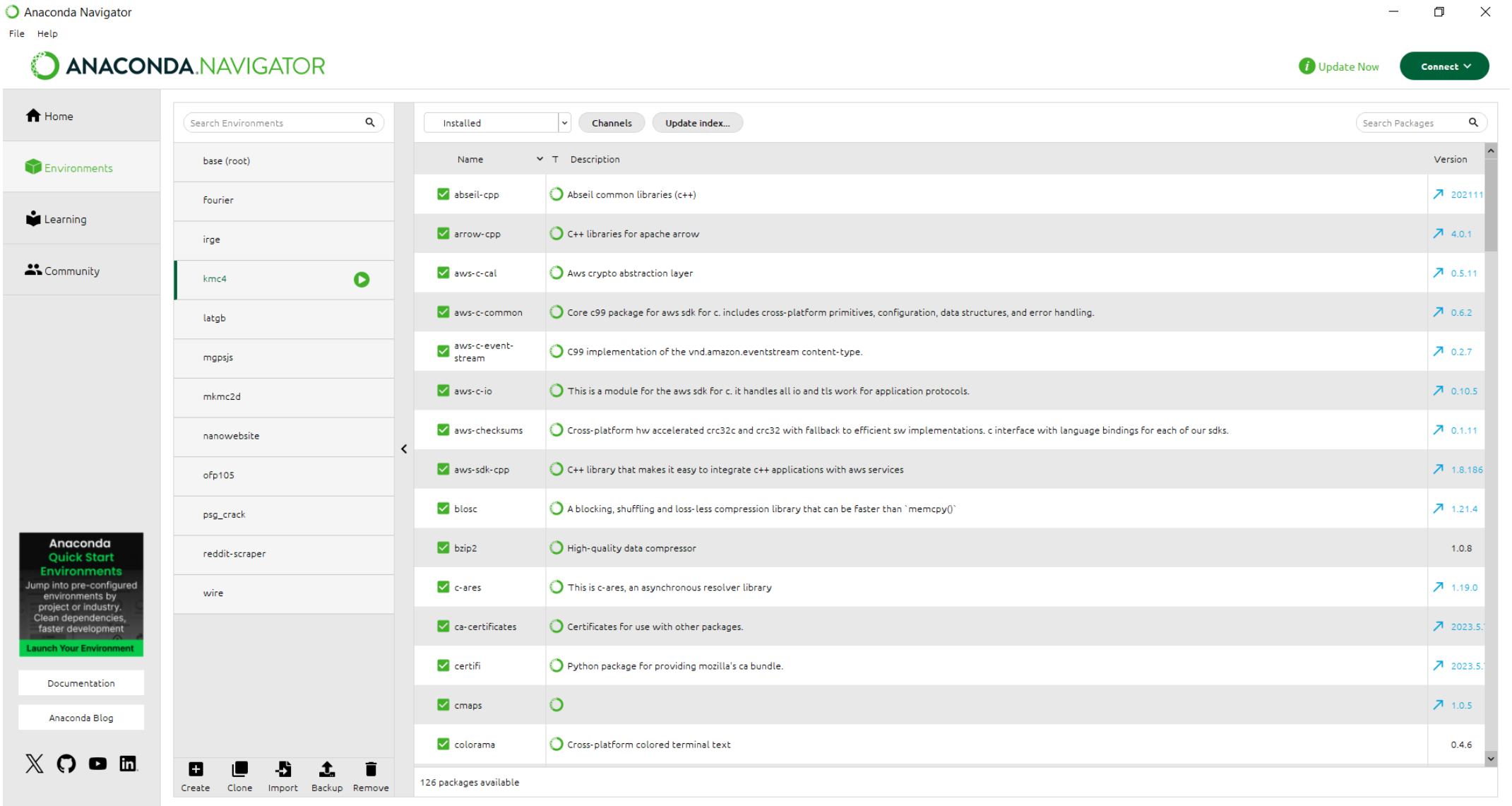
pip: requirements.txt

pip install –r r…txt

```
pyzmq>=20.0.0
numpy>=1.19.4
pillow>=8.1.0
gym
gym[atari]
netaddr>=0.8.0
oauthlib>=3.1.0
```

https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html#

# Anaconda Navigator



# Базовые вещи

Python 2 (устарел в 2012) vs. **Python 3**

```
# комментарии
```

Отступы

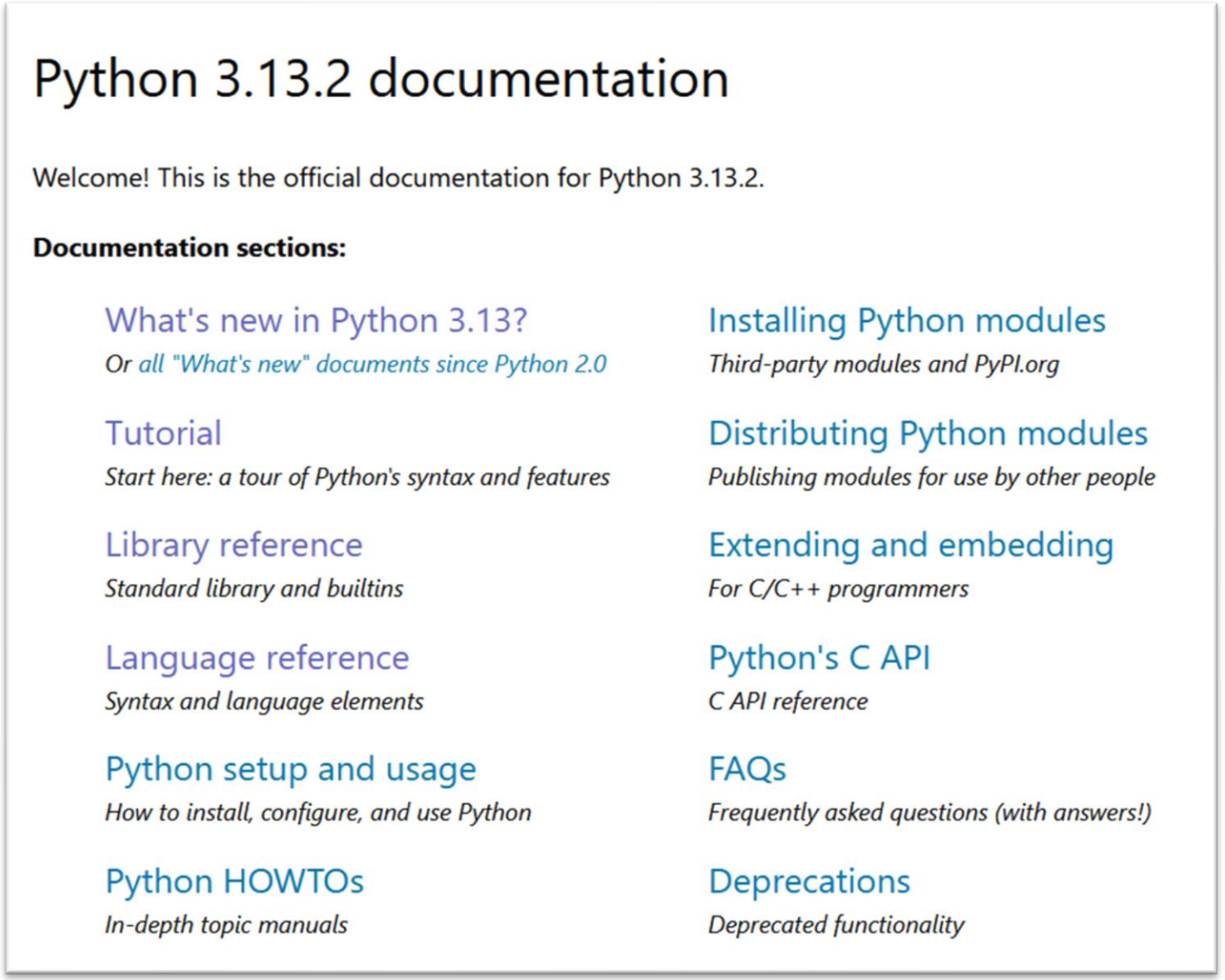
Нет “;”

REPL: `read, eval, print, loop`

# Документация

https://docs.python.org/3/

```
dir()
help()
print()
```



## Python 3.13.2 documentation

Welcome! This is the official documentation for Python 3.13.2.

**Documentation sections:**

**What's new in Python 3.13?**
*Or all "What's new" documents since Python 2.0*

**Tutorial**
*Start here: a tour of Python's syntax and features*

**Library reference**
*Standard library and builtins*

**Language reference**
*Syntax and language elements*

**Python setup and usage**
*How to install, configure, and use Python*

**Python HOWTOs**
*In-depth topic manuals*

**Installing Python modules**
*Third-party modules and PyPI.org*

**Distributing Python modules**
*Publishing modules for use by other people*

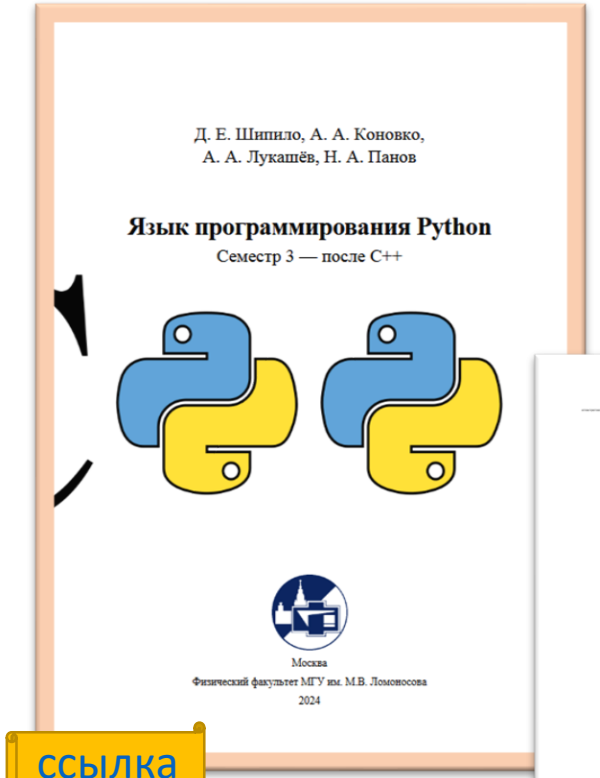
**Extending and embedding**
*For C/C++ programmers*

**Python's C API**
*C API reference*

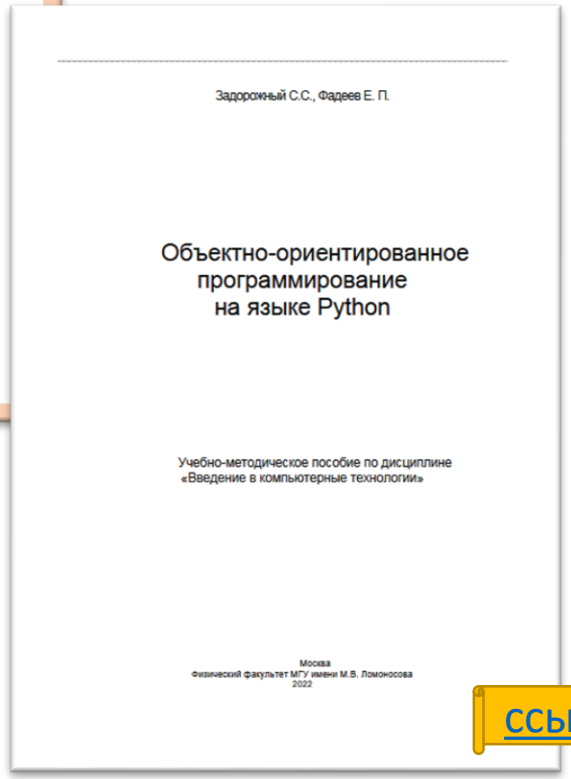
**FAQs**
*Frequently asked questions (with answers!)*

**Deprecations**
*Deprecated functionality*

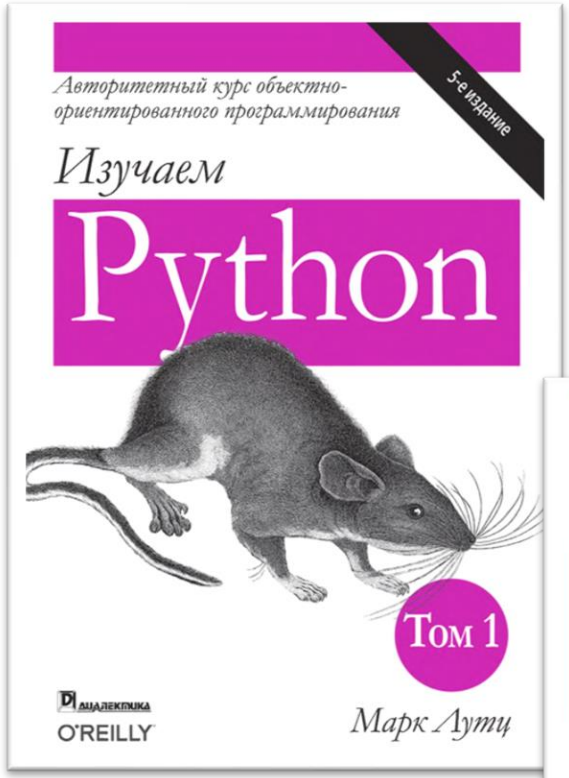
# Литература

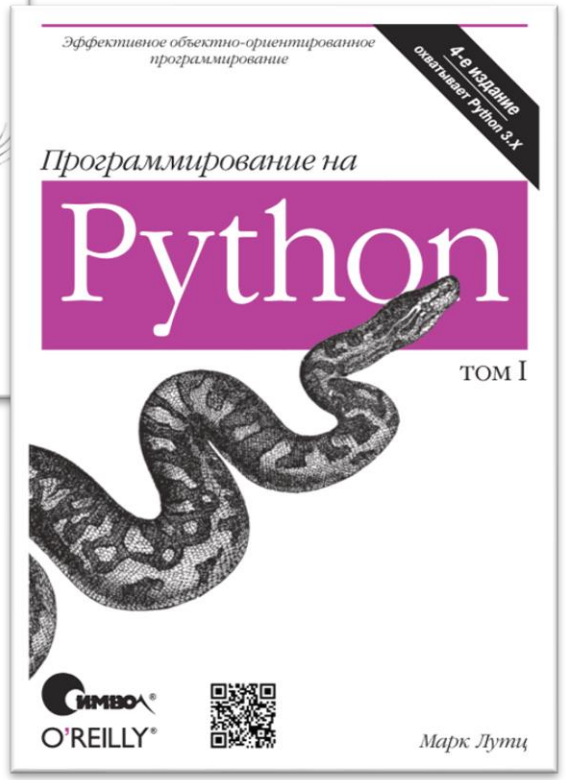



ссылка

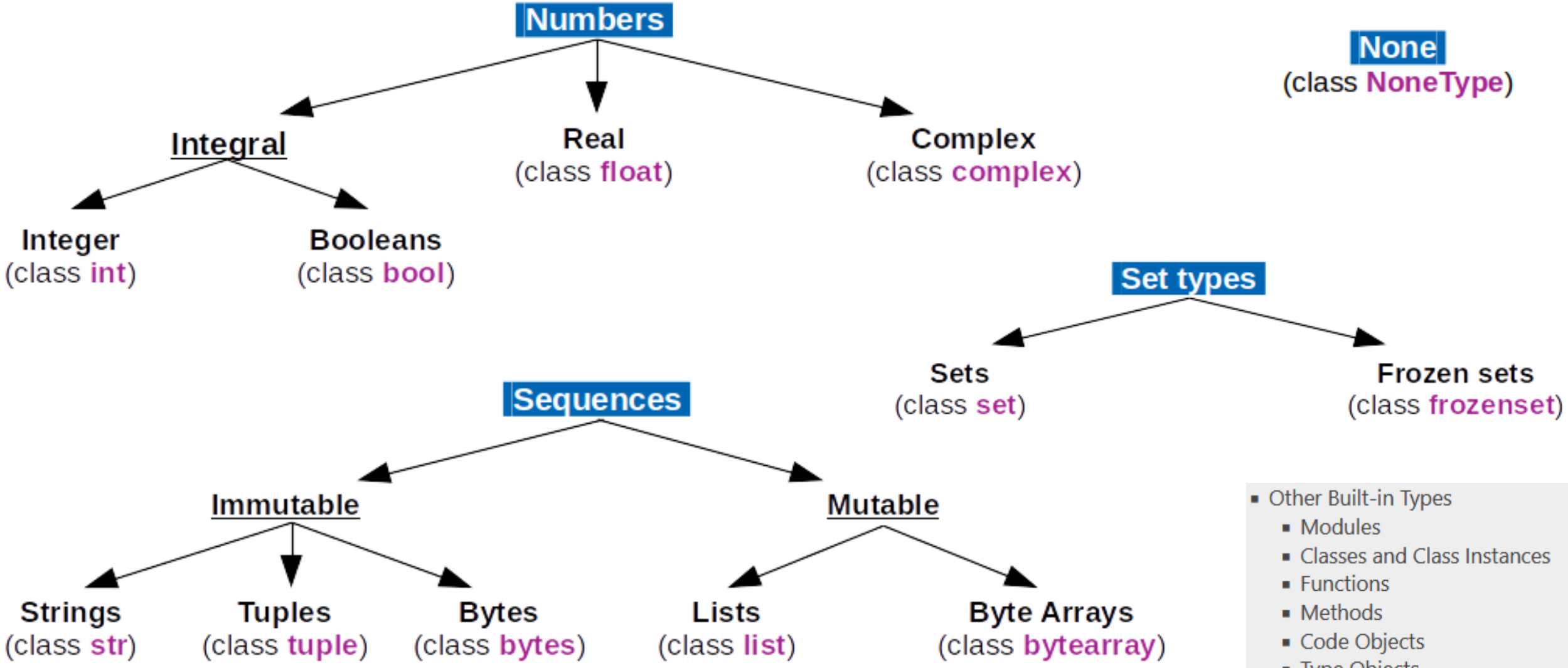



ссылка



# Типы данных




- Other Built-in Types
  - Modules
  - Classes and Class Instances
  - Functions
  - Methods
  - Code Objects
  - Type Objects
  - The Null Object

# Главные кирпичики: числа и строки

Всё – это объект

| PyObject | |
|---|---|
| type | float |
| value | 20.125 |
| refcount | 1 |

```
>>> isinstance(20.125, float)
True
```

```
>>> type(20.125)                    # узнаем тип данных
<class 'float'>

>>> 20.125.real                     # поле (атрибут)
20.125

>>> 20.125.imag
0.0

>>> 20.125.as_integer_ratio()       # метод (вызывается)
(161, 8)

>>> 20.125.hex()                    # 16-ричное представление
'0x1.4200000000000p+4'

>>> 20.125.is_integer()
False

>>> 20.125.__sizeof__()             # размер объекта в байтах
24
```

# Просмотр всех атрибутов

```
>>> dir(20.125)
['__abs__', '__add__', '__bool__', '__ceil__', '__class__', '__delattr__', '
    __dir__', '__divmod__', '__doc__', '__eq__', '__float__', '__floor__', '
    __floordiv__', '__format__', '__ge__', '__getattribute__', '
    __getformat__', '__getnewargs__', '__getstate__', '__gt__', '__hash__',
    '__init__', '__init_subclass__', '__int__', '__le__', '__lt__', '__mod__
    ', '__mul__', '__ne__', '__neg__', '__new__', '__pos__', '__pow__', '
    __radd__', '__rdivmod__', '__reduce__', '__reduce_ex__', '__repr__', '
    __rfloordiv__', '__rmod__', '__rmul__', '__round__', '__rpow__', '
    __rsub__', '__rtruediv__', '__setattr__', '__sizeof__', '__str__', '
    __sub__', '__subclasshook__', '__truediv__', '__trunc__', '
    as_integer_ratio', 'conjugate', 'fromhex', 'hex', 'imag', 'is_integer',
    'real']
```

Сюда попали:

- Методы/атрибуты класса `float`
- Методы/атрибуты класса `object` – родителя `float`

# Dunder методы

```
>>> dir(20.125)
['__abs__', '__add__', '__bool__', '__ceil__', '__class__', '__delattr__', '
    __dir__', '__divmod__', '__doc__', '__eq__', '__float__', '__floor__', '
    __floordiv__', '__format__', '__ge__', '__getattribute__', '
    __getformat__', '__getnewargs__', '__getstate__', '__gt__', '__hash__',
    '__init__', '__init_subclass__', '__int__', '__le__', '__lt__', '__mod__
    ', '__mul__', '__ne__', '__neg__', '__new__', '__pos__', '__pow__', '
    __radd__', '__rdivmod__', '__reduce__', '__reduce_ex__', '__repr__', '
    __rfloordiv__', '__rmod__', '__rmul__', '__round__', '__rpow__', '
    __rsub__', '__rtruediv__', '__setattr__', '__sizeof__', '__str__', '
    __sub__', '__subclasshook__', '__truediv__', '__trunc__', '
    as_integer_ratio', 'conjugate', 'fromhex', 'hex', 'imag', 'is_integer',
    'real']
```

```
>>> -20.125.__abs__()
-20.125
>>> (-20.125).__abs__()
20.125
>>> abs(20.125)
20.125

>>> 15.__mul__(8)
120
>>> 15 * 8
120
```

Операторы – это тоже функции

Надо учитывать приоритет операторов

# Целые числа `int`

```
>>> 15**125
10266144715765749474109597622615111533630688210188001861145930871028588075174184080051744262713642920082637507994682
```

«длинная» арифметика «из коробки»

`+` `-` `*` `/` `//` `**` `%` `<<` `>>` `&` `|` `^` `~`

# Комплексные числа `complex`

```
>>> 1j**2
(-1+0j)

>>> 12.4+3.5j.conjugate()
(12.4-3.5j)
```

# Логический тип bool

```
True / False

>>> 1 > 2
False
```

`<` `>` `<=` `>=` `==` `!=`

Операции сравнения всегда возвращают булево значение

Нельзя сравнивать напрямую числа с плавающей запятой!

```
a = (0.3 * 3) + 0.1
print(a == 1.0)
```

```
Last executed at 2025-02-16

False
```

**math.isclose**(*a*, *b*, *, *rel_tol=1e-09*, *abs_tol=0.0*)

Return `True` if the values *a* and *b* are close to each other and `False` otherwise.

# Строки `str`

```
'string'
"also string"
'''yet another string'''

>>> 'a' + 'b'
'ab'

>>> 'abc' * 3
'abcabcabc'

>>> len("string")
6

>>> "A" in 'ABC'    # 'ABC'.__contains__("A")
True

>>> 'ourstring'[0]  # индексация с 0
'o'
```

```
>>> 'this long string'.count('t')
2

>>> 'this long string'.replace('t', 'a')
'ahis long saring'
```

Форматированный вывод:

- C-style
- Метод format()
- F-строки

# Форматированный вывод

```
print('%d %s ate %f mice' % (1, 'cat', 2.5))
print('%3d %s ate %.2f mice' % (1, 'cat', 2.5))
print()

print('{} {} ate {} mice'.format(1, 'cat', 2.5))
print('{:3d} {} ate {:.2f} mice'.format(1, 'cat', 2.5))
print('{0} {1} ate {2} mice'.format(1, 'cat', 2.5))
print('{2} {0} ate {1} {1} {1} mice'.format(1, 'cat', 2.5))
print()

data = (1, 'cat', 2.5)
print('{} {} ate {} mice'.format(*data))
print()

a = 1
b = 'cat'
c = 2.5
print(f'{a} {b} ate {c} mice')
print(f'{a=} {b=} ate {c=} mice')
```

```
1 cat ate 2.500000 mice
  1 cat ate 2.50 mice

1 cat ate 2.5 mice
  1 cat ate 2.50 mice
1 cat ate 2.5 mice
2.5 1 ate cat cat cat mice

1 cat ate 2.5 mice

1 cat ate 2.5 mice
a=1 b='cat' ate c=2.5 mice
```

# Переменные

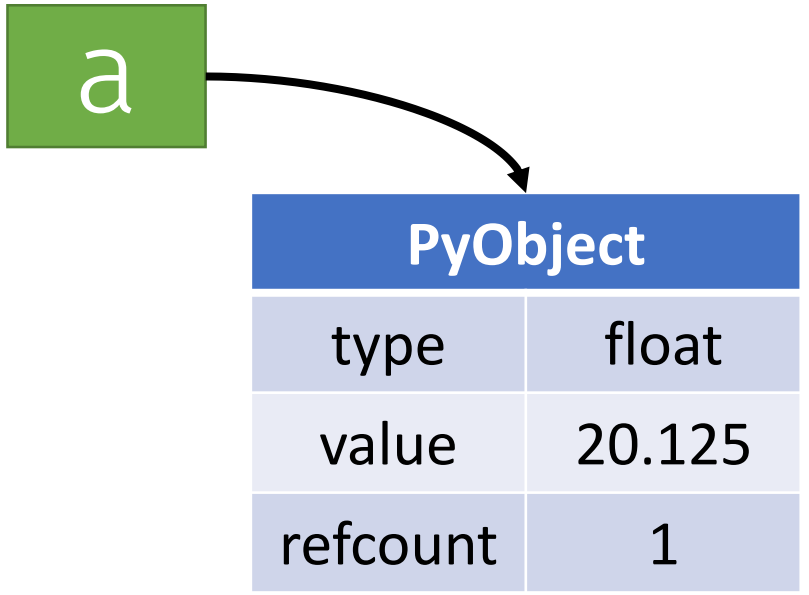
Важно: имя переменной – это только ссылка на объект данных!

Присваивание не создает объект, а только назначает ссылку!

Есть «сборщик мусора» – garbage collector. Удаляет те объекты, счетчик ссылок на которые равен нулю.

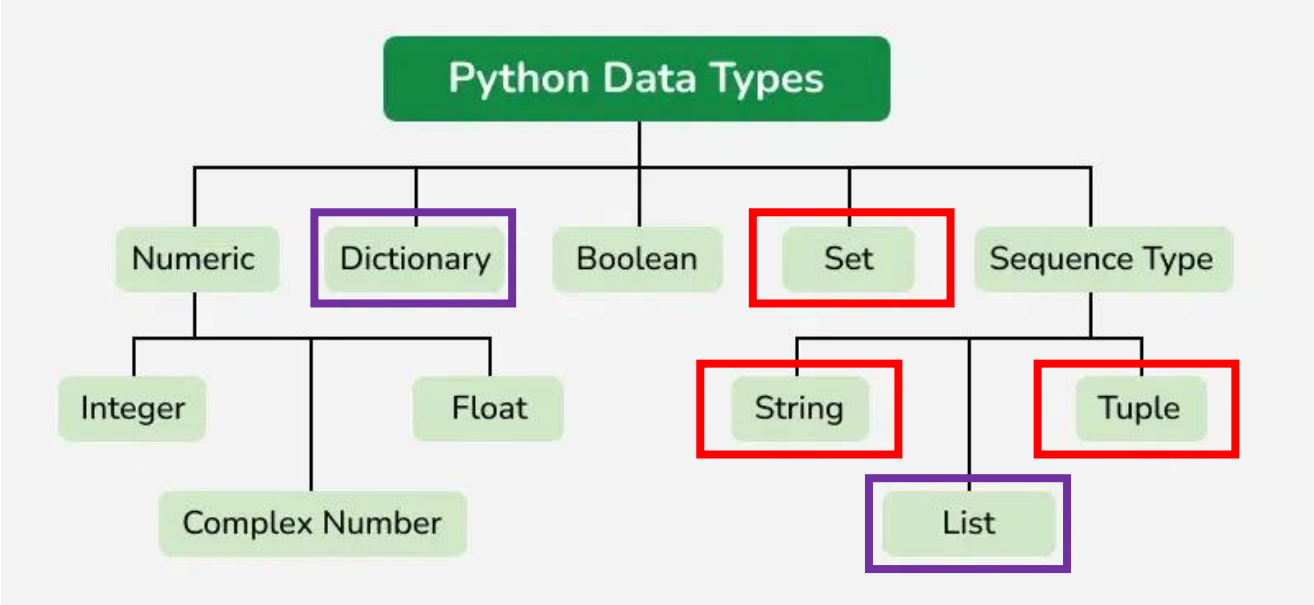
Спойлер:
распаковка

```
1   a = 20.125
2
3   del a
4
5   c = b = a
```



# Коллекции. Списки

```
>>> a = [1, 2.0, '3', 0-4j, [5, '5']]
>>> a[2]
'3'

>>> 2 in a
False

>>> b = list()
>>> b = []
>>> b.append(1)
[1]

>>> a.reverse()
>>> print(a)
[[5, '5'], -4j, '3', 2.0, 1]

>>> a.pop(2)
'3'      # осталось [[5, '5'], -4j, 2.0, 1]

>>> del a[2]
[[5, '5'], -4j, 1]
```




Списки: квадратные скобки

Изменяемые

# Ветвление

```
1   if a == 0:
2       do1()
3   elif a == 1:
4       do2()
5   else:
6       do3()
7
8
9   0, '', [], None # False
10
11
12  and or not      # Логические операторы
13
14  >>> bool([0, 0, 0])
15  True
16
17  >>> bool(any([0, 0, 0]))
18  False
19
20  # Краткая запись
21  x = a if condition else b
22
23  x = condition? a : b    # C, C++
```

Законы де Моргана

не (a и b) = (не a) или (не b)
не (a или b) = (не a) и (не b)

# Циклы

```
1 a = [1, 2, 3]
2 for x in a:
3     print(x)
4 else:
5     print('End')
```

```
1
2
3
End
```

```
1 a = 8
2 while a:
3     if a % 2:
4         a -= 1
5         continue
6     print(a)
7     a-=1
8 else:
9     print('End')
```

```
8
6
4
2
End
```

2 сорта: `while` и `for`

Внутри возможны:
`continue`
`break`
`else`

# range и slice. Срезы последовательностей

## Диапазоны

```
1 a = range(5)
2 print(a)
3 print(list(a))
4
5 print(list(range(1, 10, 2)))
6 print(list(range(10, 1, -2)))
```

```
range(0, 5)
[0, 1, 2, 3, 4]
[1, 3, 5, 7, 9]
[10, 8, 6, 4, 2]
```

## Срезы

```
1 b = list(range(1, 30, 3))
2 print(b)
3 print(b[2:])
4 print(b[:-3])
5 print(b[2:4])
6 print(b[::-1])
```

```
[1, 4, 7, 10, 13, 16, 19, 22, 25, 28]
[7, 10, 13, 16, 19, 22, 25, 28]
[1, 4, 7, 10, 13, 16, 19]
[7, 10]
[28, 25, 22, 19, 16, 13, 10, 7, 4, 1]
```

```
1 b = list(range(1, 30, 3))
2 s = slice(2, 4)
3 print(s)
4 print(b[s])
```

```
slice(2, 4, None)
[7, 10]
```

```
1 b = list(range(1, 30, 3))
2 print(b[8:100])
3 print(b[100])
```

```
[25, 28]
---------------------------------------
IndexError
Cell In[92], line 3
      1 b = list(range(1, 30, 3))
      2 print(b[8:100])
----> 3 print(b[100])

IndexError: list index out of range
```

# Функции

```
def f(x):
    if isinstance(x, (int, float, complex)):
        return x**2
    else:
        return None

print(f(2))
print(f(2+4j))
print(f('str'))
```

```
4
(-12+16j)
None
```

```
def g(x, p=2):
    return x**p

print(g(2, 4))
print(g(2))
```

```
16
4
```

Аргументы:

- Позиционные
- По умолчанию

## Безымянные функции

```
h = lambda x,a: x*a
print(h)
print(h(4,5))
```

```
<function <lambda> at 0x0000023A8BC08E00>
20
```

```
a = [(3,7), (6,9), (0,1), (2,8)]

print(sorted(a, key=lambda x:x[1]))
```

```
[(0, 1), (3, 7), (2, 8), (6, 9)]
```

# Коллекции. Кортежи, множества, словари

```
1 a = tuple([1, 2, 3])
2 a = (1, 2, 3)
3 a[1] = 4
```

```
----------------------
TypeError
```

Кортеж (`tuple`): неизменяем сам, но не его содержимое!

Множество (`set`): неупорядоченный набор уникальных элементов

```
1 a = set([1,2,3,4,4,4,4,5,1,2,3])
2 print(a)
3 print(dir(a))
```

```
{1, 2, 3, 4, 5}
```

```
[..., 'add', 'clear', 'copy', 'difference', 'difference_update',
    'discard', 'intersection', 'intersection_update',
    'isdisjoint', 'issubset', 'issuperset', 'pop',
    'remove', 'symmetric_difference',
    'symmetric_difference_update', 'union', 'update']
```

```
1 a = set([1,2,3,6])
2 b = set([3,4,5])
3 print(a & b) # a.intersection(b)
4 print(a | b) # a.union(b)
5 print(a ^ b) # a.symmetric_difference(b)
6 print(a - b) # a.difference(b)
7 print(len(a))
```

```
{3}
{1, 2, 3, 4, 5, 6}
{1, 2, 4, 5, 6}
{1, 2, 6}
4
```

# Словари

```python
d = dict(key='value')
d = {'key': 'value', 'k2': 'v2', 'kkk': 'vvv'}
print(d['k2'])

print(d.keys())  # имеет свойства set
print(d.values())
print(d.items())

Na = ('Sodium', 11, 22, +1)
Na = {'name': 'Sodium', 'num': 11, 'mass': 22, 'ion': +1}

print('name' in Na)

for k,v in Na.items():
    print(f'{k=} {v=}')
```

```
v2
dict_keys(['key', 'k2', 'kkk'])
dict_values(['value', 'v2', 'vvv'])
dict_items([('key', 'value'), ('k2', 'v2'), ('kkk', 'vvv')])
True
k='name' v='Sodium'
k='num' v=11
k='mass' v=22
k='ion' v=1
```

Словари (`dict`) – они же хэш-таблицы, коллекции для хранения данных с доступом по ключу вместо индекса.
Изменяемы.

```python
a = dict(a=1, b=2, c=3)
a
print(a.get('d', 99))
print(a['d'])
```

```
99
---------------------------------------
KeyError
Cell In[54], line 4
      2 a
      3 print(a.get('d', 99))
----> 4 print(a['d'])

KeyError: 'd'
```

Значение по умолчанию

# Чем плохо, что переменные – это ссылки

```
1 a = [1,2,3]
2 b = a
3 print(a, b)
4 a[1] = 4
5 print(a, b)
```

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

```
1 a = [1,2,3]
2 b = a.copy()
3 print(a, b)
4 a[1] = 4
5 print(a, b)
```

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

Как обойти?

Скопировать объект
(для вложенных коллекций
нужна глубокое копирование)

```
1 import copy
2 b = copy.deepcopy(a)
```

Борьба:

- Копирование
- Использование операций, создающих новые объекты

Нежелательные эффекты:

```
1 def f(x, y):
2     x += y
3     return x
4 L = [1, 2]
5 M = [3, 4]
6
7 print(f(L, M), ';', L, M)
8 print(f(L, M) is L)
```

```
[1, 2, 3, 4] ; [1, 2, 3, 4] [3, 4]
True
```

```
1 def f(x, y):
2     return x + y
3 L = [1, 2]
4 M = [3, 4]
5
6 print(f(L, M), ';', L, M)
7 print(f(L, M) is L)
```

```
[1, 2, 3, 4] ; [1, 2] [3, 4]
False
```

# Распаковка

```
1 a, b, c = 1, 2, 3
2 print(f'{a=} {b=} {c=}')
3
4 a, *b, c = 1,2,3,4,5
5 print(f'{a=} {b=} {c=}')
6
7 a, *b, c = 1,2
8 print(f'{a=} {b=} {c=}')
```

На самом деле справа формируется кортеж!

```
a=1 b=2 c=3
a=1 b=[2, 3, 4] c=5
a=1 b=[] c=2
```

```
1 a, b, c = range(5), range(5,10), range(7,7+5)
2 for x in zip(a,b,c):
3     print(x)
```

Last executed at 2025-02-15 23:26:42 in 4ms

```
(0, 5, 7)
(1, 6, 8)
(2, 7, 9)
(3, 8, 10)
(4, 9, 11)
```

```
1 a, b, c = range(5), range(5,10), range(7,7+5)
2 for x,y,z in zip(a,b,c):
3     print(f'{x=} {y=} {z=}')
```

Last executed at 2025-02-15 23:27:37 in 5ms

```
x=0 y=5 z=7
x=1 y=6 z=8
x=2 y=7 z=9
x=3 y=8 z=10
x=4 y=9 z=11
```

## Работает и с функциями

```
1 def f(x, y):
2     return x+y, x-y
3
4 a,b = f(1,2)
5 print(a)
```

```
1 s = 'string'
2 for i,v in enumerate(s):
3     print(f'{i=} {v=}')
```

Last executed at 2025-02-15 23:28:30

```
i=0 v='s'
i=1 v='t'
i=2 v='r'
i=3 v='i'
i=4 v='n'
i=5 v='g'
```

# Распаковка аргументов

## Распаковка кортежа/списка

```
1 def f1(x, y):
2     return x*y
3 a = (3, 4)
4 # print(f1(a)) # TypeError
5 print(f1(*a))
```

```
12
```

## Распаковка словаря

```
1 def f1(x, y):
2     return x*y
3 d = dict(x=3, y=4)
4 print(f1(**d))
```

```
12
```

Одна звездочка передаст только ключи.
Две – значения в аргументы, соответствующие ключу.

## Работает и в обратную сторону

```
1 def f2(*args, **kwargs):
2     print(f'{args=} {kwargs=}')
3
4 f2(1, 2, x=3, y=4)
```

```
args=(1, 2) kwargs={'x': 3, 'y': 4}
```

# Исключения

```
1 int('a')
```

Last executed at 2025-02-15 23:37:08 in 15ms

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[104], line 1
----> 1 int('a')

ValueError: invalid literal for int() with base 10: 'a'
```

```
1 'a'*'b'
```

Last executed at 2025-02-15 23:37:20 in 15ms

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[105], line 1
----> 1 'a'*'b'

TypeError: can't multiply sequence by non-int of type 'str'
```

```
1 print(var)
```

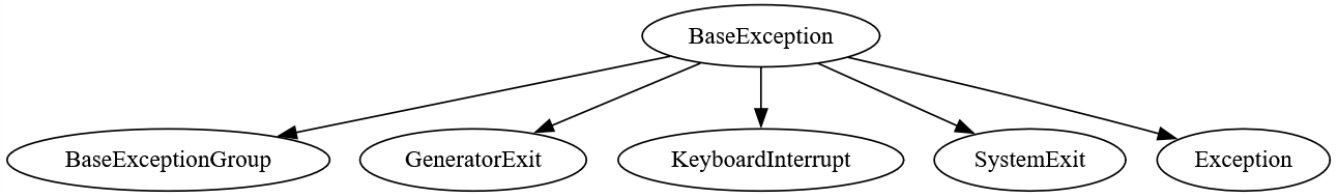
Last executed at 2025-02-15 23:38:06 in 15ms

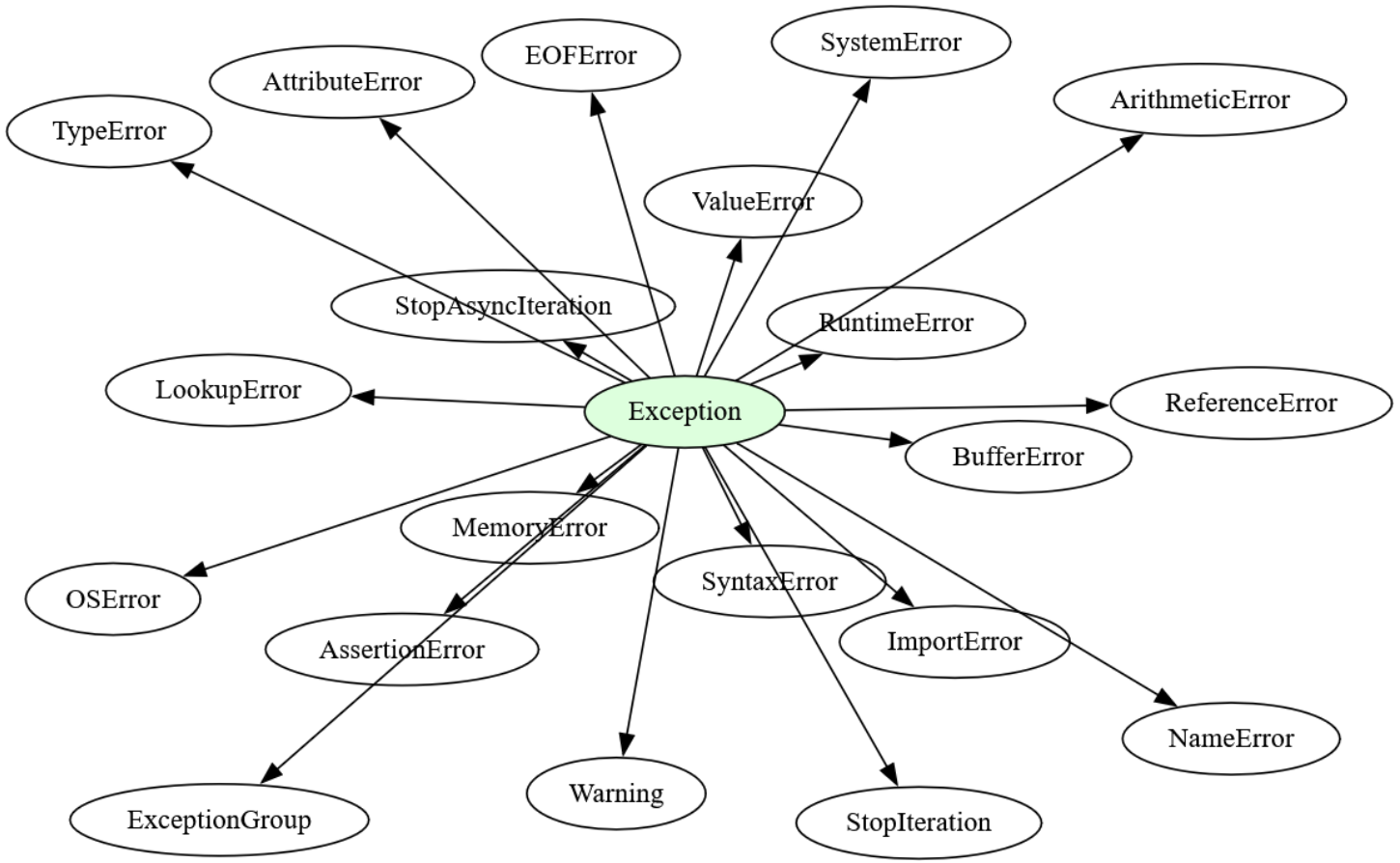
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[107], line 1
----> 1 print(var)

NameError: name 'var' is not defined
```

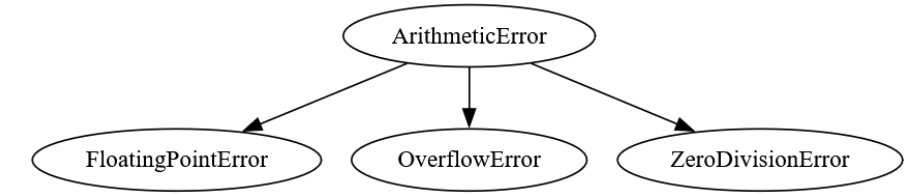
```
1 try:
2     'a'*'b'
3 except Exception as e:
4     print(e)
5     print('still normal flow here')
6     raise e
```

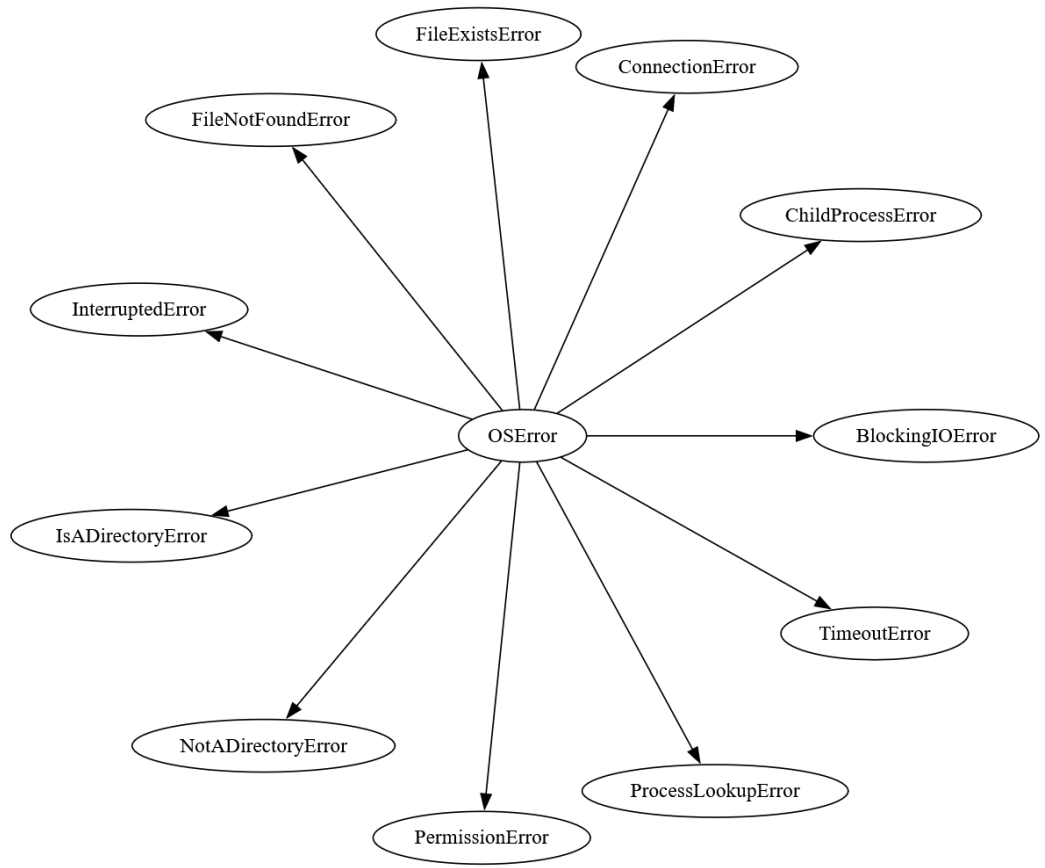
Last executed at 2025-02-15 23:40:49 in 22ms

```
can't multiply sequence by non-int of type 'str'
still normal flow here
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[112], line 6
      4 print(e)
      5 print('still normal flow here')
----> 6 raise e

Cell In[112], line 2
      1 try:
----> 2     'a'*'b'
      3 except Exception as e:
      4     print(e)

TypeError: can't multiply sequence by non-int of type 'str'
```




https://docs.python.org/3/library/exceptions.html#exception-hierarchy

# Прочее

- Итерационный протокол. Итераторы и генераторы
- Области видимости

# Выражения-генераторы

```
a = range(10)
b = list()
for x in a:
    if not x % 3:
        b.append(100*x**2)
    else:
        b.append(x)
print(b)
```

```
b = [x if x % 3 else 100*x**2 for x in a]
print(b)
```

```
[0, 1, 2, 900, 4, 5, 3600, 7, 8, 8100]
```

```
[0, 1, 2, 900, 4, 5, 3600, 7, 8, 8100]
```

```
a = {x:str(x)*3 for x in range(5)}
print(a)
```

```
{0: '000', 1: '111', 2: '222', 3: '333', 4: '444'}
```

Скобки имеют значение!

- () – генератор
- [] – `list`
- {} – `dict` или `set`

# Классы

Для хранения структурированных данных

```python
class Vector:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __str__(self):
        return f'<{self.x}, {self.y}>'

    def __add__(self, x):
        if isinstance(x, Vector):
            return Vector(self.x+x.x, self.y+x.y)

    def len(self):
        return (self.x**2 + self.y**2)**0.5

    @property
    def len2(self):
        return (self.x**2 + self.y**2)**0.5
```

```python
a = Vector(1,2)
b = Vector(3,4)
print(a, b)
print(a.len())
print(a.len2)
print(a+b)
```

```
<1, 2> <3, 4>
2.23606797749979
2.23606797749979
<4, 6>
```

# Работа с файлами

```
1  f = open('160224.csv')
2  print(f)
3  f = open('160224.csv', encoding='utf8')
4  print(f)
5
6  # print(len(f)) # TypeError: object of type '_io.TextIOWrapper' has no len()
7  bulk = f.read()
8  print(len(bulk))
9  lines = f.readlines()
10 print(len(lines))
11 f.close()
12
13 f = open('160224.csv', encoding='utf8')
14 lines = f.readlines()
15 print(len(lines))
16 f.close()
```

```
<_io.TextIOWrapper name='160224.csv' mode='r' encoding='cp1251'>
<_io.TextIOWrapper name='160224.csv' mode='r' encoding='utf8'>
259998
0
2
```

В виде менеджера контекста

```
1 with open('160224.csv', encoding='utf8') as f:
2     lines = f.readlines()
3     print(len(lines))
```

```
2
```

# Встроенные функции

**Built-in Functions**

**A**
abs()
aiter()
all()
anext()
any()
ascii()

**B**
bin()
bool()
breakpoint()
bytearray()
bytes()

**C**
callable()
chr()
classmethod()
compile()
complex()

**D**
delattr()
dict()
dir()
divmod()

**E**
enumerate()
eval()
exec()

**F**
filter()
float()
format()
frozenset()

**G**
getattr()
globals()

**H**
hasattr()
hash()
help()
hex()

**I**
id()
input()
int()
isinstance()
issubclass()
iter()

**L**
len()
list()
locals()

**M**
map()
max()
memoryview()
min()

**N**
next()

**O**
object()
oct()
open()
ord()

**P**
pow()
print()
property()

**R**
range()
repr()
reversed()
round()

**S**
set()
setattr()
slice()
sorted()
staticmethod()
str()
sum()
super()

**T**
tuple()
type()

**V**
vars()

**Z**
zip()

**_**
__import__()

# Скучное задание

Написать функцию, которая будет заполнять прямоугольный массив размера $x \times y$ целыми числами зигзагом.

$f(x, y)$ :

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 4 | 7 | 11 | 16 | 22 | 29 | 37 |
| 3 | 5 | 8 | 12 | 17 | 23 | 30 | 38 | 47 |
| 6 | 9 | 13 | 18 | 24 | 31 | 39 | 48 | 58 |
| 10 | 14 | 19 | 25 | 32 | 40 | 49 | 59 | 70 |
| 15 | 20 | 26 | 33 | 41 | 50 | 60 | 71 | 83 |
| 21 | 27 | 34 | 42 | 51 | 61 | 72 | 84 | 97 |
| 28 | 35 | 43 | 52 | 62 | 73 | 85 | 98 | 112 |
| 36 | 44 | 53 | 63 | 74 | 86 | 99 | 113 | 128 |
| 45 | 54 | 64 | 75 | 87 | 100 | 114 | 129 | 145 |

# Синтетические задачи

https://projecteuler.net/archives

https://adventofcode.com/

# Стандартная библиотека

https://docs.python.org/3/library/index.html

Половина сделанного дела – это

знание того, КАК это дело делать.

```
import что-то

import что-то as нечто
```

# pprint

Красивый вывод на экран

```
>>> import pprint
>>> stuff = ['spam', 'eggs', 'lumberjack', 'knights', 'ni']
>>> stuff.insert(0, stuff)
>>> pprint.pp(stuff)
[<Recursion on list with id=...>,
 'spam',
 'eggs',
 'lumberjack',
 'knights',
 'ni']
```


Более навороченная сторонняя библиотека:

rich

https://docs.python.org/3/library/pprint.html

# random

## Генерация случайных чисел

```
>>> random()                            # Random float:  0.0 <= x < 1.0
0.37444887175646646

>>> uniform(2.5, 10.0)                  # Random float:  2.5 <= x <= 10.0
3.1800146073117523

>>> expovariate(1 / 5)                  # Interval between arrivals averaging 5 seconds
5.148957571865031

>>> randrange(10)                       # Integer from 0 to 9 inclusive
7

>>> randrange(0, 101, 2)                # Even integer from 0 to 100 inclusive
26

>>> choice(['win', 'lose', 'draw'])     # Single random element from a sequence
'draw'

>>> deck = 'ace two three four'.split()
>>> shuffle(deck)                       # Shuffle a list
>>> deck
['four', 'two', 'ace', 'three']

>>> sample([10, 20, 30, 40, 50], k=4) # Four samples without replacement
[40, 10, 50, 30]
```

random.**randrange**(*stop*)

random.**randrange**(*start*, *stop*[, *step*])

random.**randint**(*a*, *b*)

random.**choice**(*seq*)

random.**shuffle**(*x*)

random.**uniform**(*a*, *b*)

random.**random**()

random.**choices**(*population*, *weights=None*, *, *cum_weights=None*, *k=1*)

https://docs.python.org/3/library/random.html

numpy

# math

```python
import math

a = math.sin(3.141592653589793)
print(a)

b = math.tan(math.pi)
print(b)

print(math.hypot(3, 4))
```

Last executed at 2025-02-16 00:21:21 in 5ms

```
1.2246467991473532e-16
-1.2246467991473532e-16
5.0
```

<module 'math' (built-in)>

```
This module provides access to the mathematical functions
defined by the C standard.

       e = 2.718281828459045
     inf = inf
     nan = nan
      pi = 3.141592653589793
     tau = 6.283185307179586
    acos = def acos(x, /): Return the arc cosine (measured in radians) of x.
   acosh = def acosh(x, /): Return the inverse hyperbolic cosine of x.
    asin = def asin(x, /): Return the arc sine (measured in radians) of x.
   asinh = def asinh(x, /): Return the inverse hyperbolic sine of x.
    atan = def atan(x, /): Return the arc tangent (measured in radians) of x.
   atan2 = def atan2(y, x, /): Return the arc tangent (measured in radians) of y/x.
   atanh = def atanh(x, /): Return the inverse hyperbolic tangent of x.
    ceil = def ceil(x, /): Return the ceiling of x as an Integral.
    comb = def comb(n, k, /): Number of ways to choose k items from n items without repetition and without order.
copysign = def copysign(x, y, /): Return a float with the magnitude (absolute value) of x but the sign of y.
     cos = def cos(x, /): Return the cosine of x (measured in radians).
    cosh = def cosh(x, /): Return the hyperbolic cosine of x.
 degrees = def degrees(x, /): Convert angle x from radians to degrees.
    dist = def dist(p, q, /): Return the Euclidean distance between two points p and q.
     erf = def erf(x, /): Error function at x.
    erfc = def erfc(x, /): Complementary error function at x.
     exp = def exp(x, /): Return e raised to the power of x.
   expm1 = def expm1(x, /): Return exp(x)-1.
    fabs = def fabs(x, /): Return the absolute value of the float x.
factorial = def factorial(x, /): Find x!.
   floor = def floor(x, /): Return the floor of x as an Integral.
    fmod = def fmod(x, y, /): Return fmod(x, y), according to platform C.
   frexp = def frexp(x, /): Return the mantissa and exponent of x, as pair (m, e).
    fsum = def fsum(seq, /): Return an accurate floating point sum of values in the iterable seq.
   gamma = def gamma(x, /): Gamma function at x.
     gcd = def gcd(*integers): Greatest Common Divisor.
   hypot = def hypot(...) hypot(*coordinates) -> value
 isclose = def isclose(a, b, *, rel_tol=1e-09, abs_tol=0.0): Determine whether two floating point numbers are close in value.
isfinite = def isfinite(x, /): Return True if x is neither an infinity nor a NaN, and False otherwise.
   isinf = def isinf(x, /): Return True if x is a positive or negative infinity, and False otherwise.
   isnan = def isnan(x, /): Return True if x is a NaN (not a number), and False otherwise.
```

# statistics

## Функции для работы со статистическими данными

| | |
|---|---|
| mean() | Arithmetic mean ("average") of data. |
| fmean() | Fast, floating-point arithmetic mean, with optional weighting. |
| geometric_mean() | Geometric mean of data. |
| harmonic_mean() | Harmonic mean of data. |
| kde() | Estimate the probability density distribution of the data. |
| kde_random() | Random sampling from the PDF generated by kde(). |
| median() | Median (middle value) of data. |
| median_low() | Low median of data. |
| median_high() | High median of data. |
| median_grouped() | Median (50th percentile) of grouped data. |
| mode() | Single mode (most common value) of discrete or nominal data. |
| multimode() | List of modes (most common values) of discrete or nominal data. |
| quantiles() | Divide data into intervals with equal probability. |

| | |
|---|---|
| covariance() | Sample covariance for two variables. |
| correlation() | Pearson and Spearman's correlation coefficients. |
| linear_regression() | Slope and intercept for simple linear regression. |

| | |
|---|---|
| pstdev() | Population standard deviation of data. |
| pvariance() | Population variance of data. |
| stdev() | Sample standard deviation of data. |
| variance() | Sample variance of data. |

https://docs.python.org/3/library/statistics.html

numpy

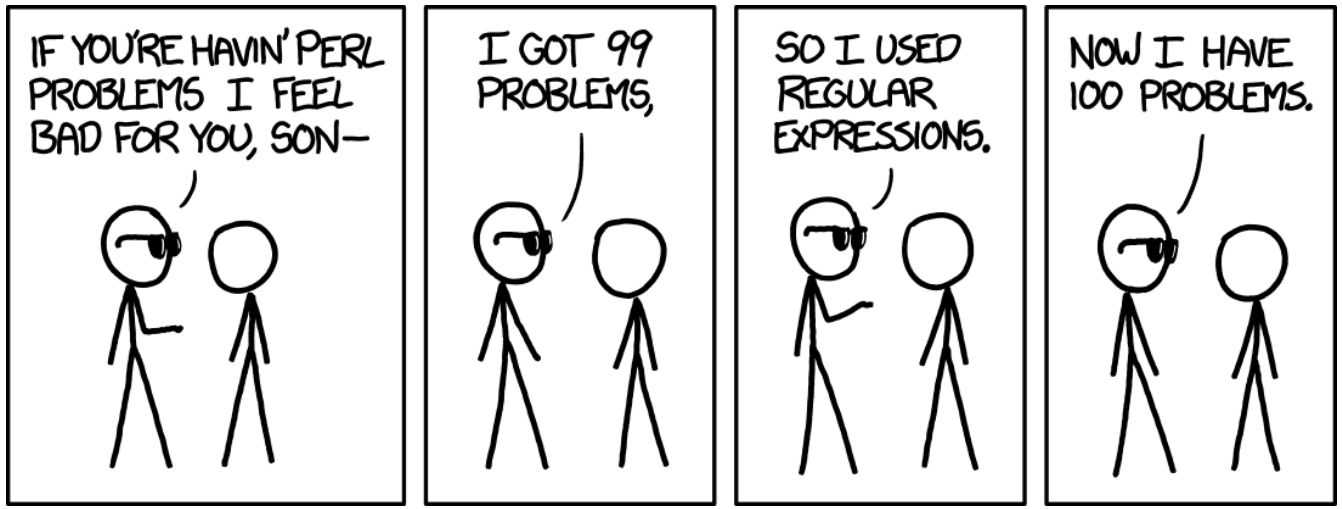
# re

Регулярные выражения

Wisywig: https://regex101.com/

https://docs.python.org/3/howto/regex.html

https://docs.python.org/3/library/re.html



# Что это такое?

| Regular Expression Basics | |
|---|---|
| . | Any character except newline |
| a | The character a |
| ab | The string ab |
| a\|b | a or b |
| a* | 0 or more a's |
| \ | Escapes a special character |

| Regular Expression Quantifiers | |
|---|---|
| * | 0 or more |
| + | 1 or more |
| ? | 0 or 1 |
| {2} | Exactly 2 |
| {2, 5} | Between 2 and 5 |
| {2,} | 2 or more |
| {,5} | Up to 5 |

Default is greedy. Append ? for reluctant.

| Regular Expression Assertions | |
|---|---|
| ^ | Start of string |
| \A | Start of string, ignores m flag |
| $ | End of string |
| \Z | End of string, ignores m flag |
| \b | Word boundary |
| \B | Non-word boundary |
| (?=...) | Positive lookahead |
| (?!...) | Negative lookahead |
| (?<=...) | Positive lookbehind |
| (?<!...) | Negative lookbehind |
| (?()\|) | Conditional |

| Regular Expression Character Classes | |
|---|---|
| [ab-d] | One character of: a, b, c, d |
| [^ab-d] | One character except: a, b, c, d |
| [\b] | Backspace character |
| \d | One digit |
| \D | One non-digit |
| \s | One whitespace |
| \S | One non-whitespace |
| \w | One word character |
| \W | One non-word character |

| Regular Expression Special Characters | |
|---|---|
| \n | Newline |
| \r | Carriage return |
| \t | Tab |
| \YYY | Octal character YYY |
| \xYY | Hexadecimal character YY |

https://www.debuggex.com/cheatsheet/regex/python

# Пример

При покрытии $D = 1.3$~МС рост структур кобальта на медной подложке происходит главным образом у ступеней. Цепочки, образованные таким путем, не распределены равномерно, в силу неоднородного распределения самих ступеней на поверхности~\cite{Figuera1995} (см. рис.~\ref{fig:CoRTdeposition}e).

При значении $D = 1.54$~МС на террасах начинают формироваться отдельные острова больших размеров. Острова у краев террас растут и увеличиваются в сторону террасы~\cite{JApplPhys.100.084304} (см. рис.~\ref{fig:CoRTdeposition}ж).

Далее, при значениях покрытия $D \le 2$~МС двухслойные острова кобальта продолжают расти, и они не коалесцируют --- подложка все еще видима. При покрытии $D$, равном 2~МС, сверху двухслойных кристаллитов нарастает третий слой. Преимущественно он вырастает на островах, декорирующих край террасы, что делает его положение все еще различимым. С другой стороны, третий слой, образующийся на двухслойных островах, более не демонстрирует треугольной формы; он показывает тенденцию к росту в форме шестиугольников, покрывающих некоторую часть двухслойных кристаллитов, которые, предположительно, срослись~\cite{Figuera1994,PhysRevB.62.2126} (см. рис.~\ref{fig:CoRTdeposition}з).

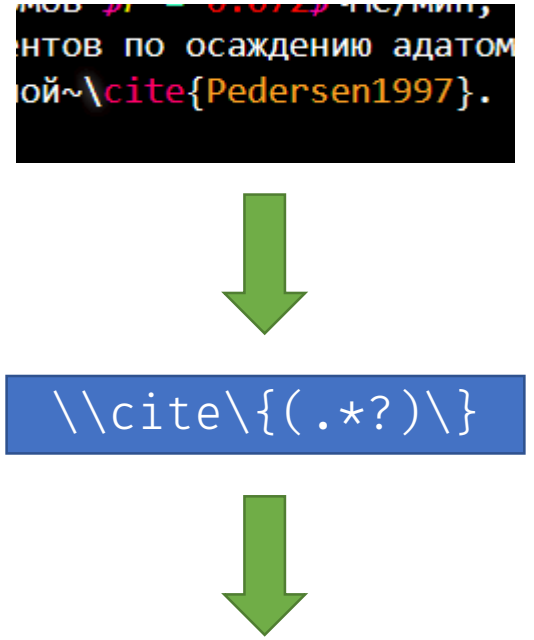
Стоит отметить, что и при гомоэпитаксиальном росте Cu/Cu(111)~\cite{Giesen2003} образующиеся острова имеют компактную форму.
Анализ СТМ изображений показал, что рост островков Cu на широких террасах ступенчатой вицинальной поверхности Cu(111) начинается в ГПУ-узлах, в то время как на узких они возникают в ГЦК-узлах. Эти различия в процессе нуклеации во время гомоэпитаксиального роста приписываются влиянию деформаций, создаваемых наборами узких террас между широкими. Предполагается, что энергия связи в центрах роста изменяется таким образом, что ГПУ-узлы становятся предпочтительными. Наблюдения ясно демонстрируют, что экспериментальные результаты в исследованиях гомо- и гетероэпитаксиального роста могут существенно зависеть от кристаллографической природы субстрата, и что интерпретация экспериментальных результатов должна выполняться с известной долей осторожности. Таким образом демонстрируется важность высокоточной ориентации образцов в плоскости кристалла.

При значениях покрытия $D \sim 5$~МС морфология роста островов сохраняется: наблюдаются многослойные острова. Пленка состоит из плоских кристаллитов, чей размер практически не отличается от размера островов при значении $D \sim 0.6$~МС. Рост островов не происходит слой за слоем: фактически, первый слой не покрывает всю поверхность. Острова растут в высоту и формируют сначала двухслойные, а потом и многослойные острова, т.е. хотя эпитаксиальная пленка и образуется, она не ровная и однородная, а гранулированная, зернистая~\cite{PhysRevB.62.2126,PhysRevB.47.13043} (см. рис.~\ref{fig:CoRTdeposition}и).

Отдельно стоит рассмотреть зависимость морфологии поверхности от температуры образца в момент напыления.
В случае низкой температуры рост структур происходил при температуре 150~К, с величиной потока адатомов $F = 0.072$~МС/мин; в результате на поверхности образовывались ветвистые острова случайной формы со средней шириной ветви около 30~\AA\ и высотой $3.8 \pm 0.2$~\AA. Анализ экспериментов по осаждению адатомов кобальта при температурах 170, 225 и 300~К показал, что при увеличении температуры осаждения дендритная форма островов утрачивалась, уступая место треугольной~\cite{Pedersen1997}.

Подробный анализ температурной зависимости морфологии поверхности был проведен в работе~\cite{PhysRevB.95.125423}.
Рост наноструктур Co на вицинальной поверхности Cu(111) был исследован при нескольких температурах между 120~К и 300~К с помощью СТМ при переменной температуре и покрытии менее одного монослоя.
При этом на краях ступеней происходит гетерогенная нуклеация, а на террасах --- гомогенная.
Разница в распределении высот и площадей между этими двумя типами островков с различным механизмом образования объясняется образованием сплава на краю ступени.

На рис.~\ref{fig:physrevb-95-125423} проводится качественное сравнение роста структур при одном и том же значении покрытия.
Подготовленная поверхность представляет из себя набор широких террас, разделенных несколькими узкими террасами (см. рис.~\ref{fig:physrevb-95-125423}а), ширина самой большой террасы составляет около 170~нм.
Плотность ступеней используемого образца составляет приблизительно 1~ступень на 20~нм. Ориентация плотно упакованных рядов определяется по изображениям с атомным разрешением (см. рис.~\ref{fig:physrevb-95-125423}b).
При всех температурах образуются явно разделенные острова на террасах (см. рис.~\ref{fig:physrevb-95-125423}c, e и g), а край ступеней поверхности декорируется (см. рис.~\ref{fig:physrevb-95-125423}d, f и h).
Острова Co на террасах имеют разветвленную форму при 120~К и 150~К (рис.~\ref{fig:physrevb-95-125423}c и e). При 150~К островки растут с тремя толстыми ветвями, разделенными примерно на 120$^\circ$, от которых отходят более тонкие, растущие из толстых (рис.~\ref{fig:physrevb-95-125423}e). Такие углы указывают на влияние на процесс роста направления атомных рядов поверхности подложки~\cite{PhysRevLett.70.3943, PhysRevLett.74.3217,SurfSciRep.31.125,PhysRevB.71.115414}. При 300~К островки растут компактными (см. рис.~\ref{fig:physrevb-95-125423}g).




```
\cite{Pedersen1997}
\cite{PhysRevB.95.125423}
\cite{PhysRevLett.70.3943,PhysRevLett.74.3217,SurfSciRep.31.125,PhysRevB.71.115414}
\cite{PhysRevB.95.125423}
\cite{PhysRevB.47.13043,PhysRevB.62.2126,PhysRevB.56.2340}
\cite{PhysRevB.95.125423}
\cite{PhysRevB.77.125437}
\cite{PhysRevB.73.193405}
\cite{PhysRevB.80.155419}
```

# itertools

| | | | |
|---|---|---|---|
| accumulate() | p [,func] | p0, p0+p1, p0+p1+p2, ... | accumulate([1,2,3,4,5]) → 1 3 6 10 15 |
| batched() | p, n | (p0, p1, ..., p_n-1), ... | batched('ABCDEFG', n=3) → ABC DEF G |
| chain() | p, q, ... | p0, p1, ... plast, q0, q1, ... | chain('ABC', 'DEF') → A B C D E F |
| chain.from_iterable() | iterable | p0, p1, ... plast, q0, q1, ... | chain.from_iterable(['ABC', 'DEF']) → A B C D E F |
| compress() | data, selectors | (d[0] if s[0]), (d[1] if s[1]), ... | compress('ABCDEF', [1,0,1,0,1,1]) → A C E F |
| dropwhile() | predicate, seq | seq[n], seq[n+1], starting when predicate fails | dropwhile(lambda x: x<5, [1,4,6,3,8]) → 6 3 8 |
| filterfalse() | predicate, seq | elements of seq where predicate(elem) fails | filterfalse(lambda x: x<5, [1,4,6,3,8]) → 6 8 |
| groupby() | iterable[, key] | sub-iterators grouped by value of key(v) | groupby(['A','B','DEF'], len) → (1, A B) (3, DEF) |
| islice() | seq, [start,] stop [, step] | elements from seq[start:stop:step] | islice('ABCDEFG', 2, None) → C D E F G |
| pairwise() | iterable | (p[0], p[1]), (p[1], p[2]) | pairwise('ABCDEFG') → AB BC CD DE EF FG |
| starmap() | func, seq | func(*seq[0]), func(*seq[1]), ... | starmap(pow, [(2,5), (3,2), (10,3)]) → 32 9 1000 |
| takewhile() | predicate, seq | seq[0], seq[1], until predicate fails | takewhile(lambda x: x<5, [1,4,6,3,8]) → 1 4 |
| tee() | it, n | it1, it2, ... itn splits one iterator into n | |
| zip_longest() | p, q, ... | (p[0], q[0]), (p[1], q[1]), ... | zip_longest('ABCD', 'xy', fillvalue='-') → Ax By C- D- |

## Навороченные итераторы

**Infinite iterators:**

| Iterator | Arguments | Results | Example |
|---|---|---|---|
| count() | [start[, step]] | start, start+step, start+2*step, ... | count(10) → 10 11 12 13 14 ... |
| cycle() | p | p0, p1, ... plast, p0, p1, ... | cycle('ABCD') → A B C D A B C D ... |
| repeat() | elem [,n] | elem, elem, elem, ... endlessly or up to n times | repeat(10, 3) → 10 10 10 |

| Examples | Results |
|---|---|
| product('ABCD', repeat=2) | AA AB AC AD BA BB BC BD CA CB CC CD DA DB DC DD |
| permutations('ABCD', 2) | AB AC AD BA BC BD CA CB CD DA DB DC |
| combinations('ABCD', 2) | AB AC AD BC BD CD |
| combinations_with_replacement('ABCD', 2) | AA AB AC AD BB BC BD CC CD DD |

https://docs.python.org/3/library/itertools.html

more_itertools

```python
import itertools

params1 = list('AB')
params2 = list('cd')
params3 = list('123')

for p1 in params1:
    for p2 in params2:
        for p3 in params3:
            print(p1, p2, p3)

# ИЛИ

for p in itertools.product(params1, params2, params3):
    print(p)
```

# functools

## Функции высшего порядка (заявка на функциональное программирование)

```
functools.partial(func, /, *args, **keywords)
```

```
>>> from functools import partial
>>> basetwo = partial(int, base=2)
>>> basetwo.__doc__ = 'Convert base 2 string to an int.'
>>> basetwo('10010')
18
```

Каррирование

```
@functools.cache(user_function)
```

```
@functools.lru_cache(user_function)
@functools.lru_cache(maxsize=128, typed=False)
```

```
functools.reduce(function, iterable, [initial, ]/)
```

```
reduce(lambda x, y: x+y, [1, 2, 3, 4, 5])
```

(((((1+2)+3)+4)+5).

```
@cache
def factorial(n):
    return n * factorial(n-1) if n else 1

>>> factorial(10)      # no previously cached result, makes 11 recursive calls
3628800
>>> factorial(5)       # just looks up cached value result
120
>>> factorial(12)      # makes two new recursive calls, the other 10 are cached
479001600
```

https://docs.python.org/3/library/functools.html

# pathlib / os.path

Работа с файловой системой

```
from pathlib import Path
p = Path('.').resolve()
print(p)
p2 = p.parent.parent
print(p2)
p3 = p2 / 'Мой диск' / 'Lammps' / 'References'
print(list(p3.glob('*.pdf')))
```

Last executed at 2025-02-09 21:38:44 in 14ms

```
G:\Мой диск\Спецпрактикум


G:\


[
    WindowsPath('G:/Мой диск/Lammps/References/Insight_into_induced_charges_at_metal_surfaces_and.pdf'),
    WindowsPath('G:/Мой диск/Lammps/References/CoAu111.pdf')
]
```

https://docs.python.org/3/library/pathlib.html
https://docs.python.org/3/library/os.path.html

# shutil

Высокоуровневые функции для работы с файлами

Многое можно сделать при помощи `pathlib`!

```
shutil.copy(src, dst, *, follow_symlinks=True)

shutil.copy2(src, dst, *, follow_symlinks=True)

shutil.copytree(src, dst, symlinks=False, ignore=None, copy_function=copy2,
ignore_dangling_symlinks=False, dirs_exist_ok=False)

shutil.move(src, dst, copy_function=copy2)

shutil.rmtree(path, ignore_errors=False, onerror=None, *, onexc=None, dir_fd=None)
```

https://docs.python.org/3/library/shutil.html

# Работа с архивами

- `zlib`
- `gzip`
- `bz2`
- `lzma`
- `zipfile`
- `tarfile`

Разные алгоритмы подходят для хранения разных данных по-разному!

https://docs.python.org/3/library/archiving.html

# Сериализация/хранение

- `json`
- `pickle`
- `sqlite3`
- `shelve`

`yaml`

`peewee`

`SQL Alchemy`

`pyarrow`

`diskcache`

https://docs.python.org/3/library/persistence.html

# Хранение данных (в человекочитаемом виде)

Хранение данных в **одновременно** машино- и человекочитаемом виде

- `json` – структурированные данные
- `csv` – табличные данные

```
total_time,total_steps,current_phase,phase_steps,phase_time,current_temp,atoms,defects,rng_seed,lens,gaps,noc
0.0228034,100,0,0,0.0228034,130,"[68, 145]",[],11638946771510751024,[1 1],[ 76 122],2
0.0324315,146,0,0,0.0324315,130,"[68, 149]",[],11638946771510751024,[1 1],[ 80 118],2
0.0456919,215,0,0,0.0456919,130,"[71, 147]",[],11638946771510751024,[1 1],[ 75 123],2
0.0667131,316,0,0,0.0667131,130,"[72, 147]",[],11638946771510751024,[1 1],[ 74 124],2
0.0968543,464,0,0,0.0968543,130,"[69, 144]",[],11638946771510751024,[1 1],[ 74 124],2
0.140543,681,0,0,0.140543,130,"[64, 146]",[],11638946771510751024,[1 1],[ 81 117],2
0.204465,999,0,0,0.204465,130,"[73, 147]",[],11638946771510751024,[1 1],[ 73 125],2
0.296198,1467,0,0,0.296198,130,"[71, 147]",[],11638946771510751024,[1 1],[ 75 123],2
0.436191,2154,0,0,0.436191,130,"[49, 160]",[],11638946771510751024,[1 1],[110  88],2
0.631982,3162,0,0,0.631982,130,"[70, 187]",[],11638946771510751024,[1 1],[116  82],2
0.926893,4641,0,0,0.926893,130,"[60, 182]",[],11638946771510751024,[1 1],[121  77],2
1.34828,6812,0,0,1.34828,130,"[22, 79, 97]",[],11638946771510751024,[1 1 1],[ 56  17 124],3
2.64889,9999,0,0,2.64889,130,"[84, 87, 88]",[],11638946771510751024,[1 2],[  2 195],2
5.50574,14677,0,0,5.50574,130,"[50, 51, 52, 142]",[],11638946771510751024,[3 1],[ 89 107],2
7.55576,21544,0,0,7.55576,130,"[10, 57, 64, 65]",[],11638946771510751024,[1 1 2],[ 46   6 144],3
14.2927,31622,0,0,14.2927,130,"[59, 60, 61, 65, 156]",[],11638946771510751024,[3 1 1],[  3  90 102],3
```

```yaml
name: example_run # will be used for output toos
terrace_length: 50 # length of terrace
terrace_width: 9 # (997)
initial_atoms_count: 10 # atoms
defects_count: 0 # atoms

barriers: barriers_adaptive4
barriers_heights: # eV ---- Ag/Pt
    E1: 0.235
    E2: 0.295
    E3: 0.241
    E4: 0.370
```

yaml

pandas

numpy

https://docs.python.org/3/library/csv.html
https://docs.python.org/3/library/json.html

# Остальное

- copy
- time, datetime
- enum
- os, sys
- io
- logging
- ctypes
- argparse
- urllib
- tkinter

dateutil

# Веселое задание

1. Подойти к научному руководителю
2. Получить задание
3. Выполнить
4. Защитить результат в конце практики
5. Успех + оценка!