

Wifi

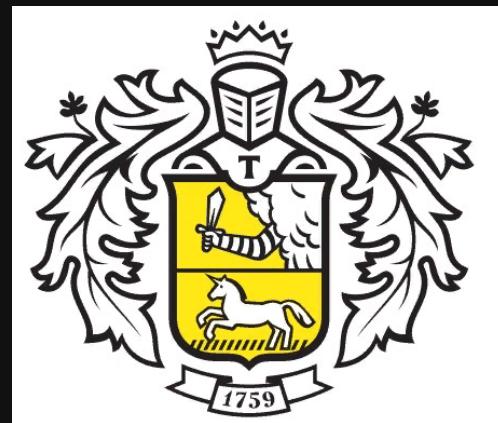
Tinkoff Guest - tinkoff1

Tinkoff Python

Лекция 1

Почему питон?

Типы данных и их особенности



Афонасьев Евгений

План лекции

- Орг вопросы
- Почему питон?
- Типы данных и особенности их реализации

Коротко о нас



какие-то рандомные мужики со змеей

Что мы
разрабатываем?

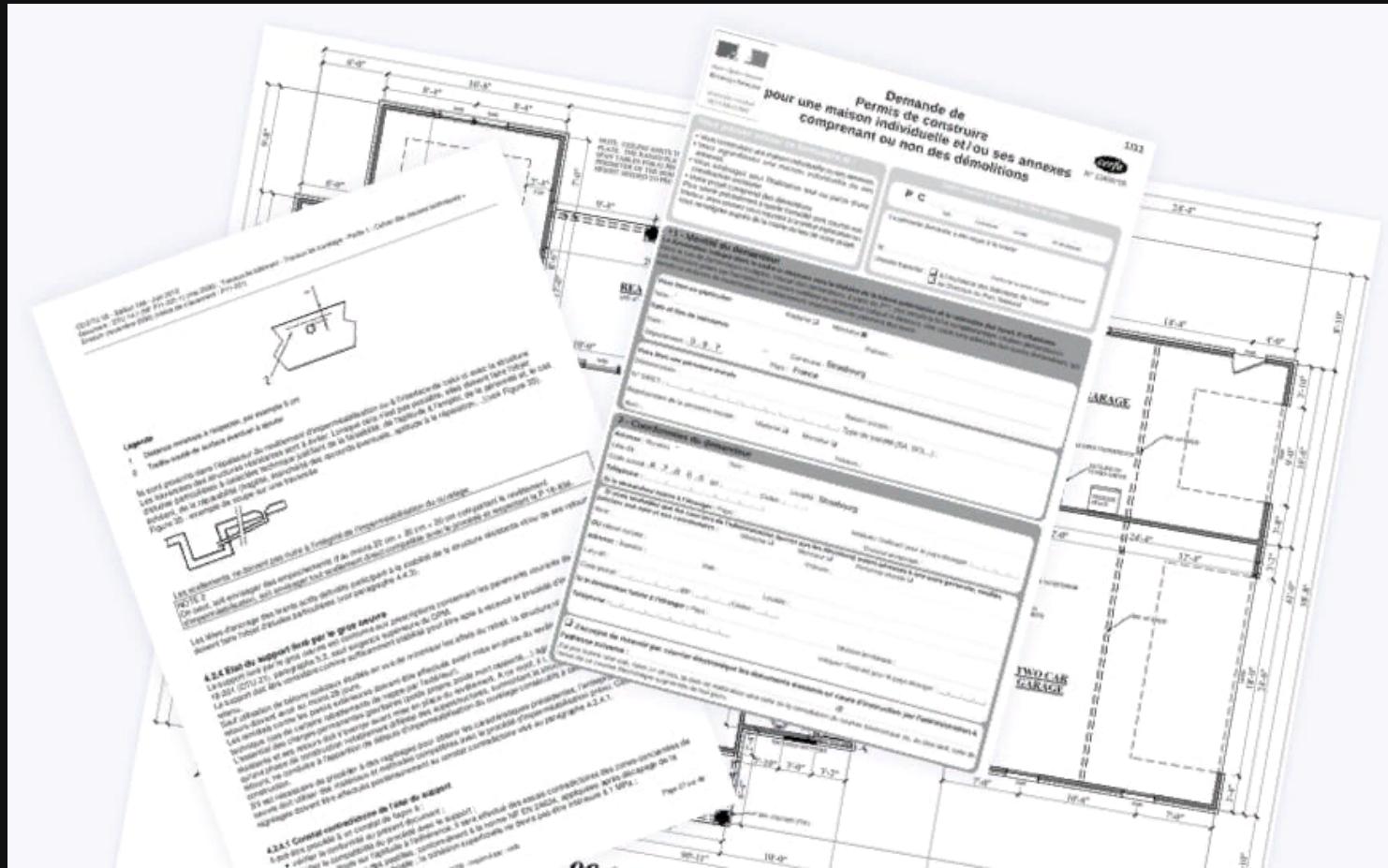
Голосовой помощник Олег



Системы текстовых коммуникаций с клиентами



Системы распознавания документов

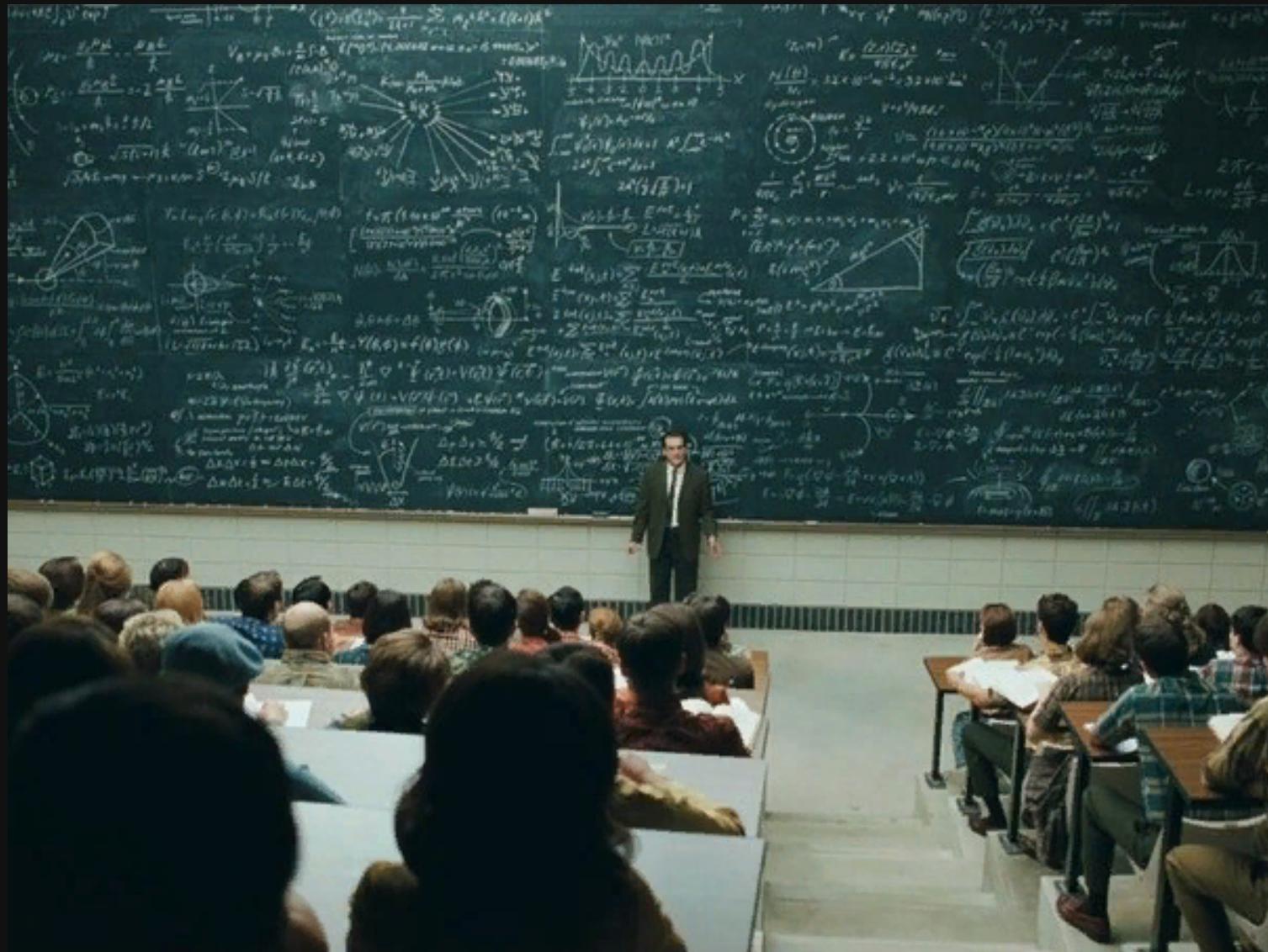


И многое другое!

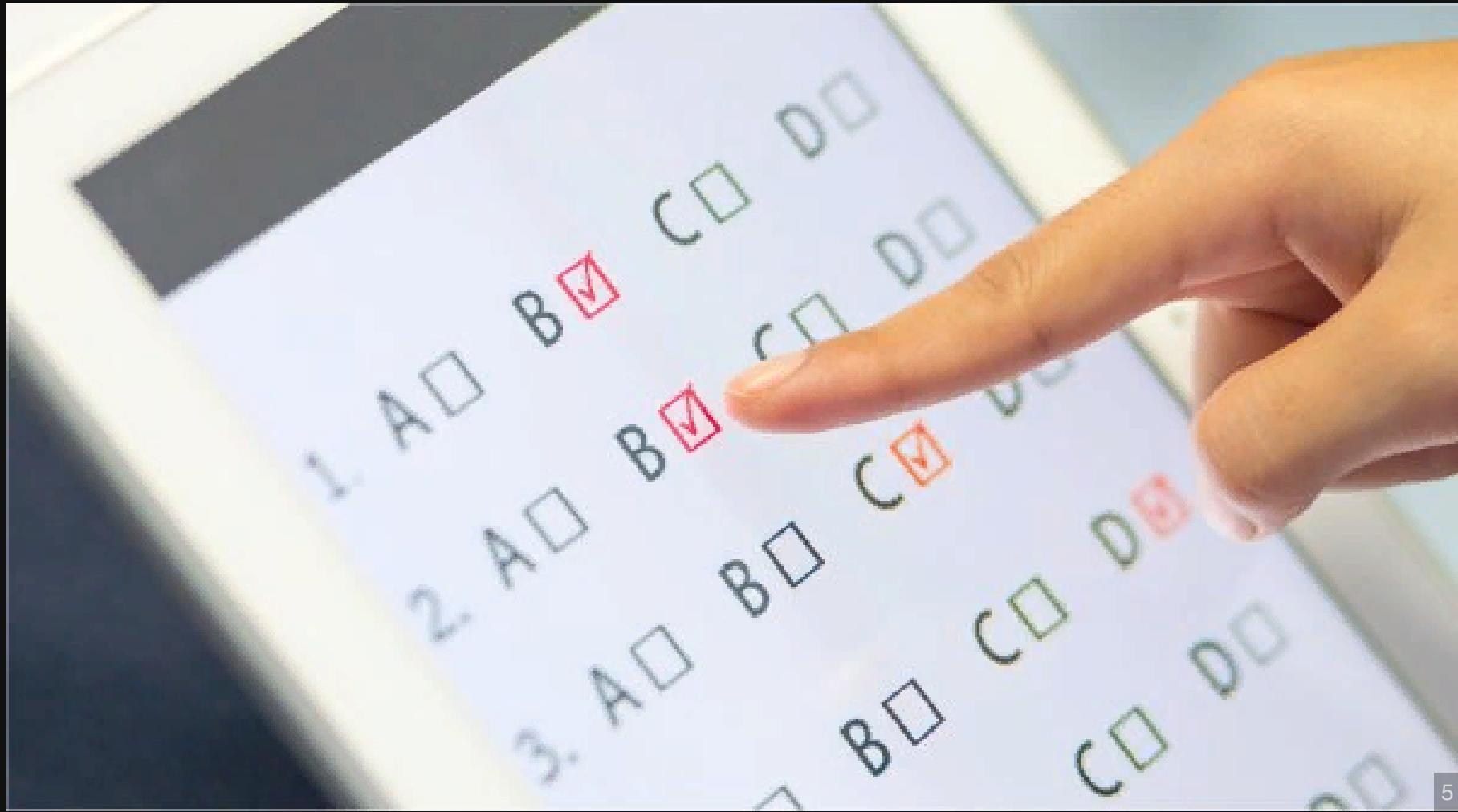
План курса



10 лекций



9 тестов перед началом лекции



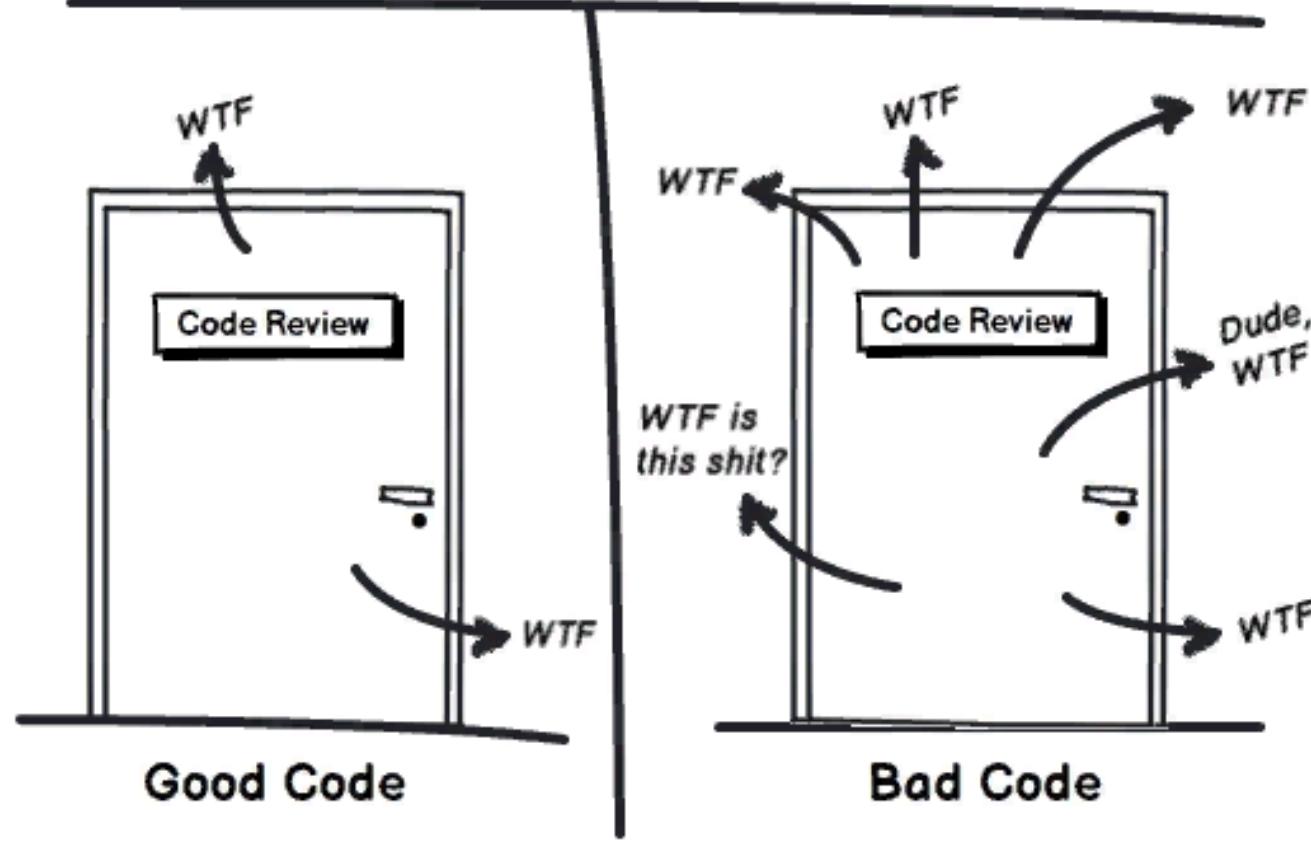
8 домашних заданий

Полноценные программы/
сервисы

Пишем код на *gitlab*
(неделя на задачу, после
штраф по баллам)

Code review

Code Quality Measurement: WTFs/Minute



Курсовая работа

Разработать сервис по собственному ТЗ
или взять ТЗ из списка заготовленных
(примерно на 3 недели)

Условия успешного завершения



1. Все домашние работы
2. Курсовая работа
3. Средний балл больше половины от максимума

Сертификаты выпускникам



Стажировка или найм



Правила общения в Telegram



- Проблемы решаем в общем канале
- Не нужно писать в личку
- Помогаем друг другу
- В нерабочее время мы можем не ответить

Правила поведения в офисе

- Не шататься по офису
- Не шуметь
- Кухня рядом с залом доступна
- Женский туалет в противоположном крыле

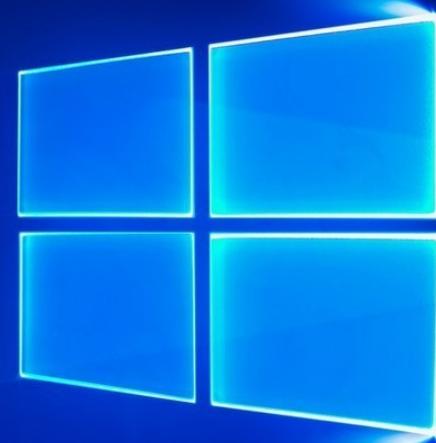
C_Python 3.8

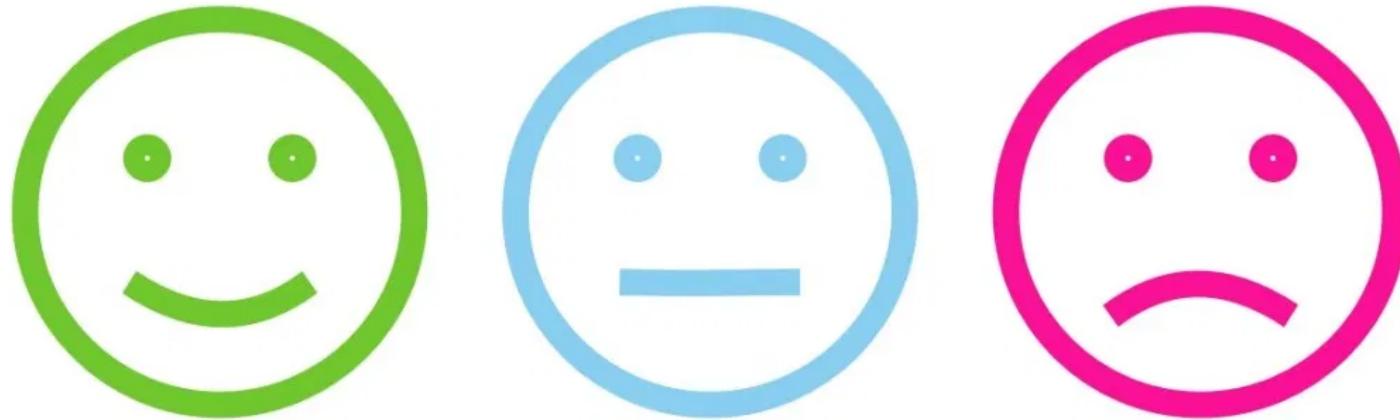


<https://www.python.org/downloads/>

Windows problems

↖(ツ)↗



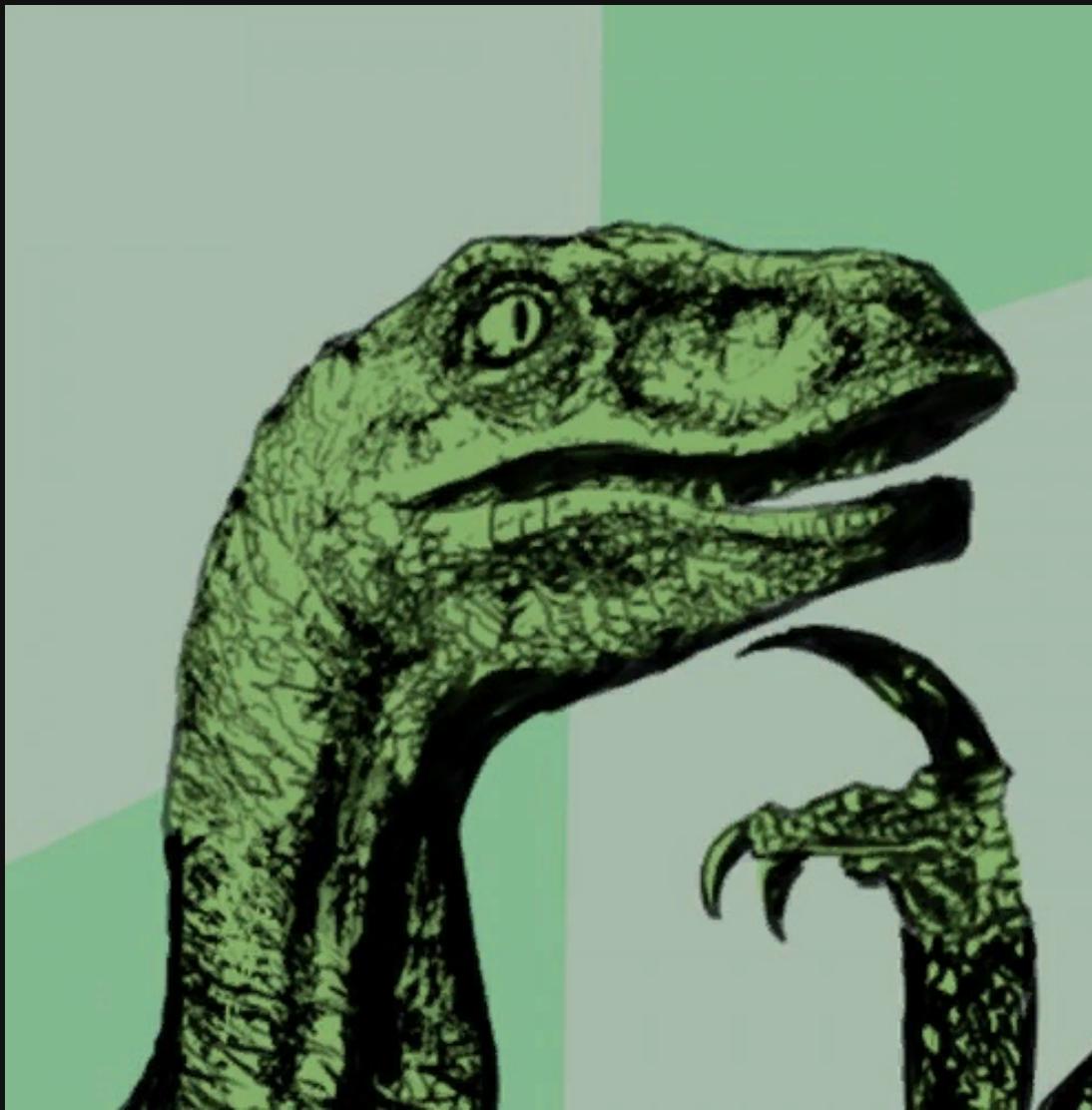


FEEDBACK



ПОЕХАЛИ !

Почему Python?

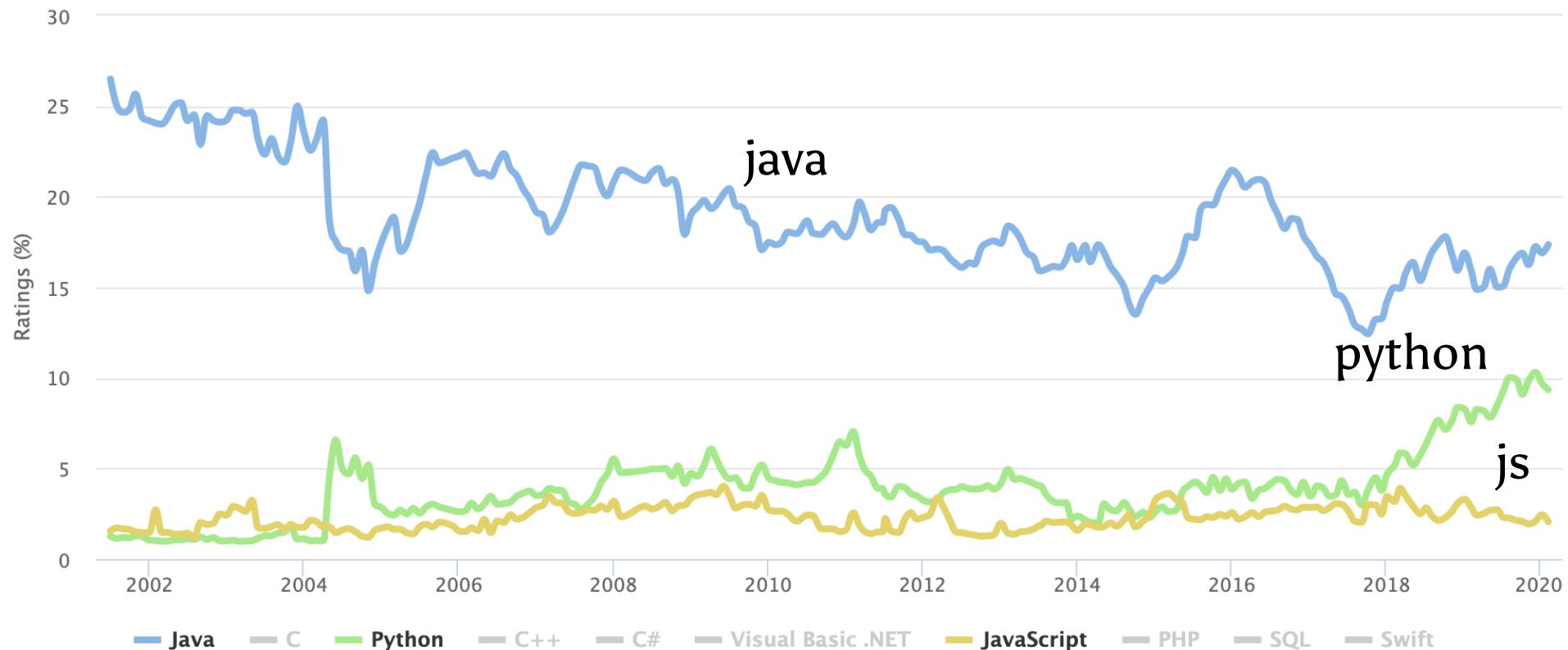


ОДИН ИЗ САМЫХ ПОПУЛЯРНЫХ ЯП

Feb 2020	Feb 2019	Change	Programming Language	Ratings	Change
1	1		Java	17.358%	+1.48%
2	2		C	16.766%	+4.34%
3	3		Python	9.345%	+1.77%
4	4		C++	6.164%	-1.28%
5	7	▲	C#	5.927%	+3.08%
6	5	▼	Visual Basic .NET	5.862%	-1.23%
7	6	▼	JavaScript	2.060%	-0.79%
8	8		PHP	2.018%	-0.25%
9	9		SQL	1.526%	-0.37%
10	20	▲	Swift	1.460%	+0.54%

Tiobe index

И показывает стабильный рост



The best second language

применяется почти во всех областях
современного IT

ну, кроме мобильных
приложений =(



Слабые стороны



Интерпретируемый язык высокого уровня

В течение курса мы будем подробно
рассматривать как же python работает
внутри

Большой расход памяти

(всё объект)

Все значения обернуты в структуры с
рядом дополнительных данных

```
int {  
    счетчик ссылок int  
    ссылка на тип  
    массив чисел для хранения значения  
    etc.  
}
```

Сравнительно
медленный

```

1 static PyObject *
2 long_add(PyLongObject *a, PyLongObject *b)
3 {
4     PyLongObject *z;
5
6     CHECK_BINOP(a, b);
7
8     if (Py_ABS(Py_SIZE(a)) <= 1 && Py_ABS(Py_SIZE(b)) <= 1) {
9         return PyLong_FromLong(MEDIUM_VALUE(a) + MEDIUM_VALUE(b));
10    }
11    if (Py_SIZE(a) < 0) {
12        if (Py_SIZE(b) < 0) {
13            z = x_add(a, b);
14            if (z != NULL) {
15                /* x_add received at least one multiple-digit int,
16                   and thus z must be a multiple-digit int.
17                   That also means z is not an element of
18                   small_ints, so negating it in-place is safe. */
19                assert(Py_REFCNT(z) == 1);
20                Py_SET_SIZE(z, -(Py_SIZE(z)));
21            }
22        }
23        else
24            z = x_sub(b, a);
25    }
26    else {
27        if (Py_SIZE(b) < 0)
28            z = x_sub(a, b);
29        else
30            z = x_add(a, b);
31    }
32    return (PyObject *)z;
33 }

```

Global interpreter lock



Есть не только в Python и не так
страшен, как кажется

Это и многое другое
мы еще обсудим

Сильные стороны



Простой, легкочитаемый

```
if not message.from_client:  
    logger.warning('...')  
    return  
  
if support_cms_client.check_user_banned(message.author.id):  
    logger.info('...')  
    return  
  
if support_cms_client.message_is_stop_phrase(message.text):  
    logger.info('...')  
    return
```

Строгая типизация

```
In [8]: 1 + '1'
```

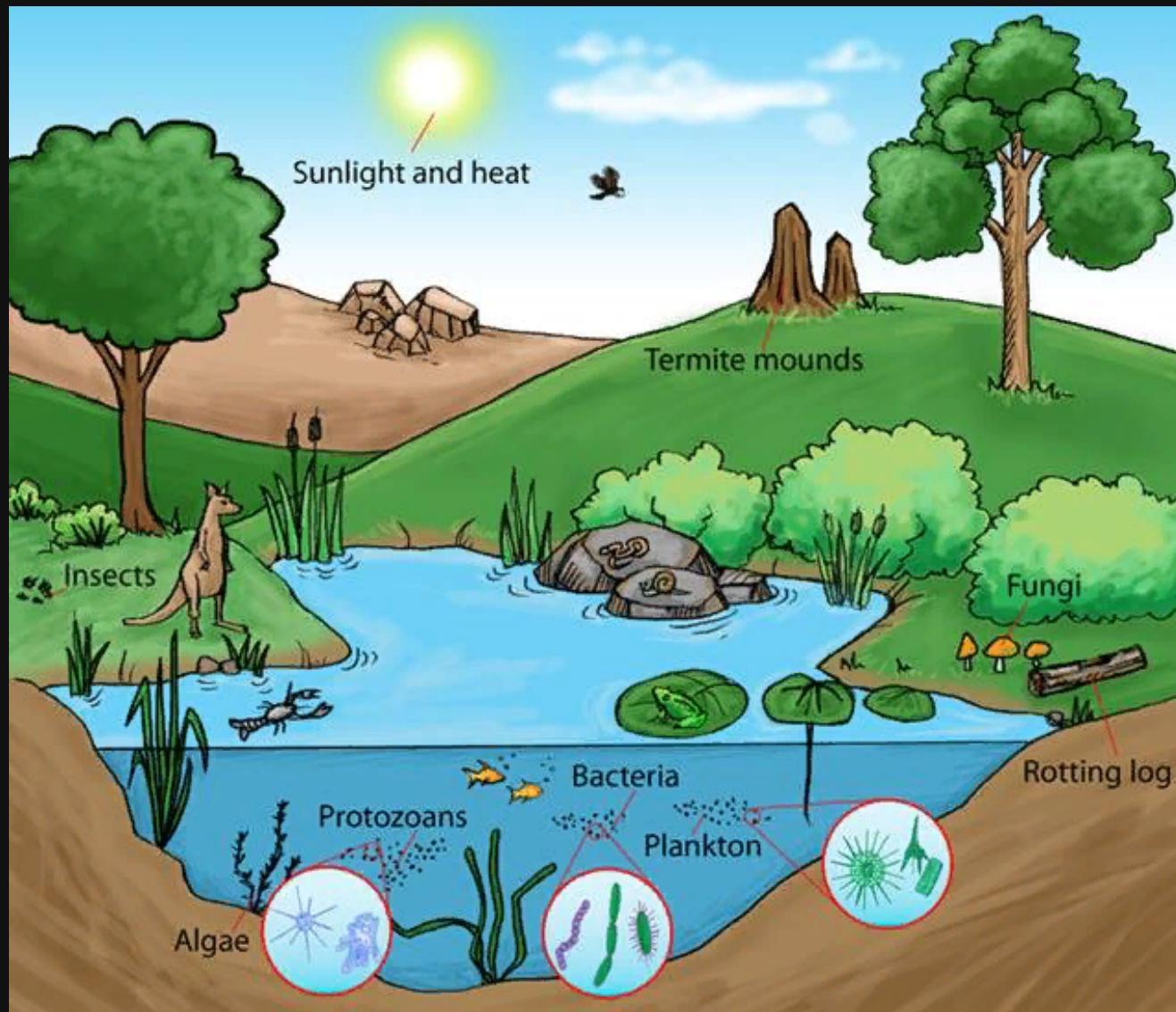
```
-----  
TypeError
```

```
Traceback
```

```
<ipython-input-8-7ff5cb60d31b> in <module>  
----> 1 1 + '1'
```

```
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Богатая экосистема



Достаточно быстрый
для большинства
задач*

С учетом библиотек на других языках

При наличии современных средств
статического анализа можно писать
не только быстро, но и надежно

test.py

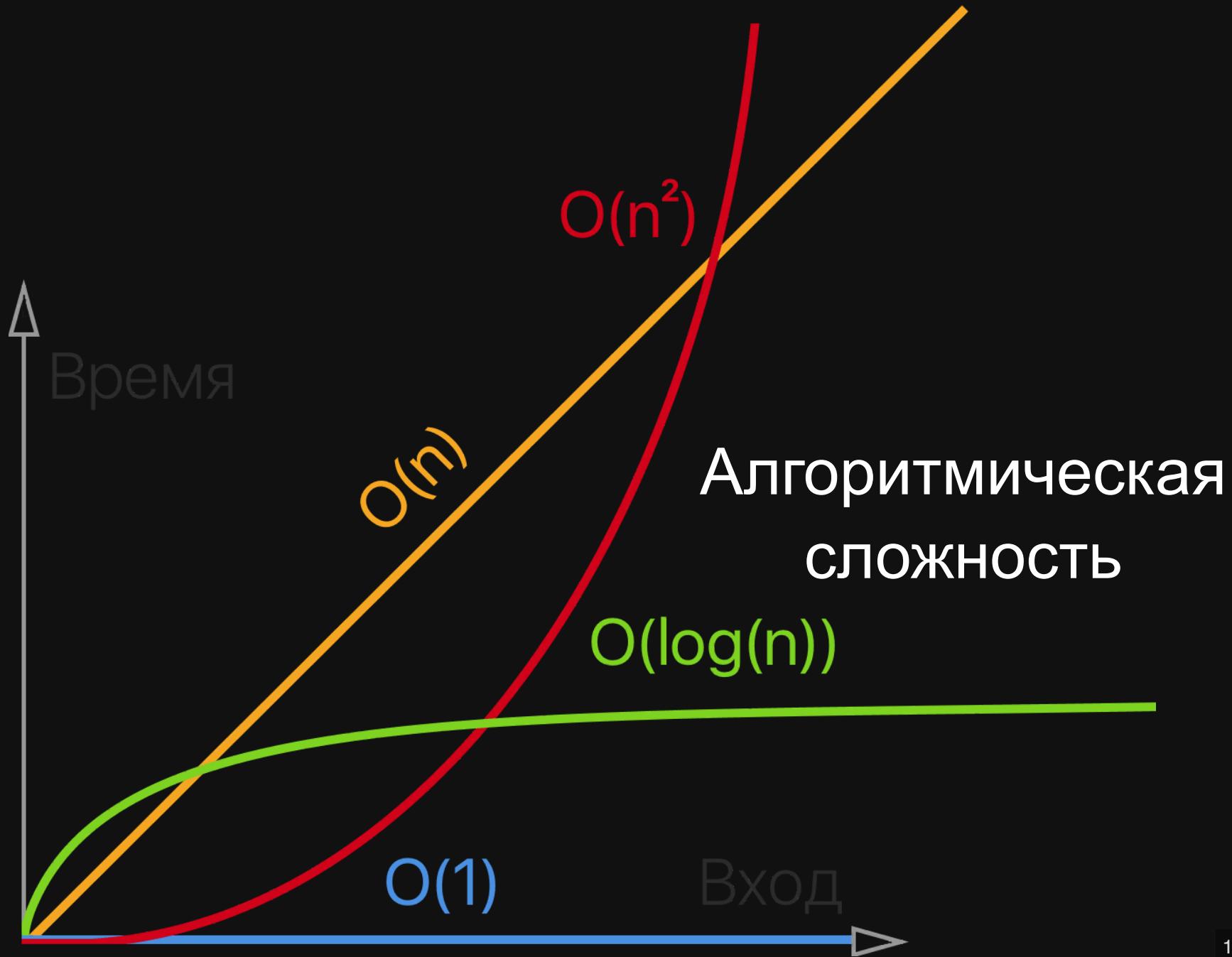
```
1 import datetime
2 name = 'John'
3 age = 49
4 birthday = datetime.date.today()
5 age = 'fifty'
6 if age < birthday:
7     something = name + age
```

Unsupported operand types for + ("str" and "int") ⌂

Severity	Provider	Description
Error	mypy	Incompatible types in assignment (expression has type "str", variable has type
Error	mypy	Unsupported operand types for > ("date" and "int")
Error	mypy	Unsupported operand types for + ("str" and "int")

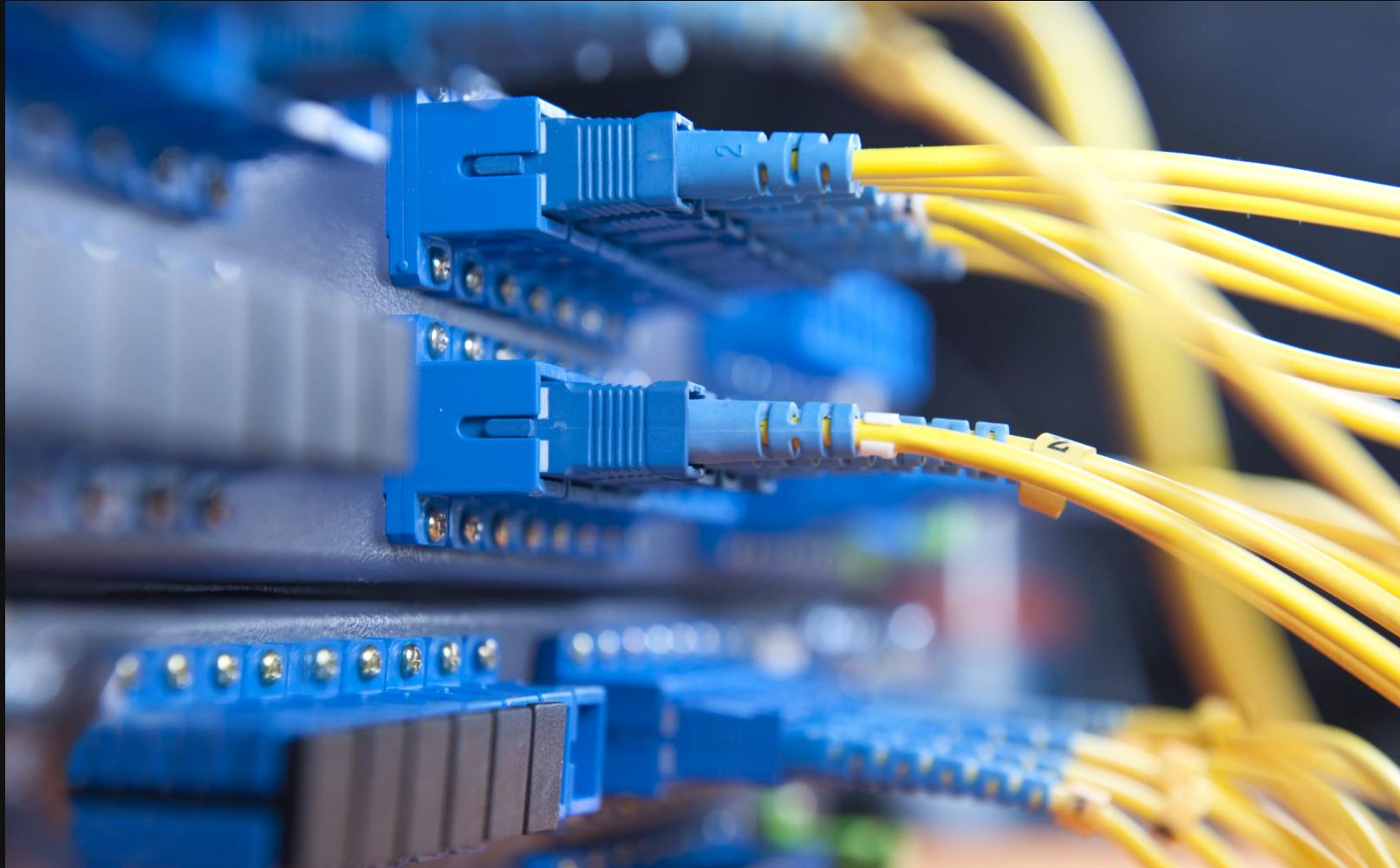
Что такое медленный?





В реальной жизни все не
так однозначно

В веб-разработке мы ограничены в основном IO операциями

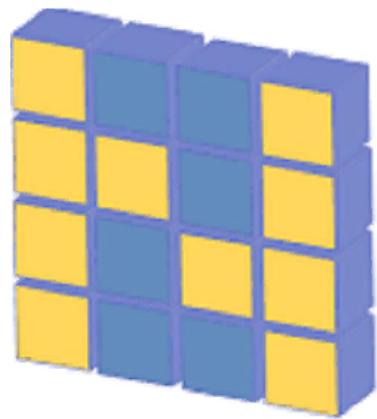


С которыми python
справляется отлично

Неподходящие задачи

- Много CPU
- Не пишем базы данных
(брокеры сообщений)
- Не пишем ОС
- Не пишем драйвера

но если очень хочется, то можно!

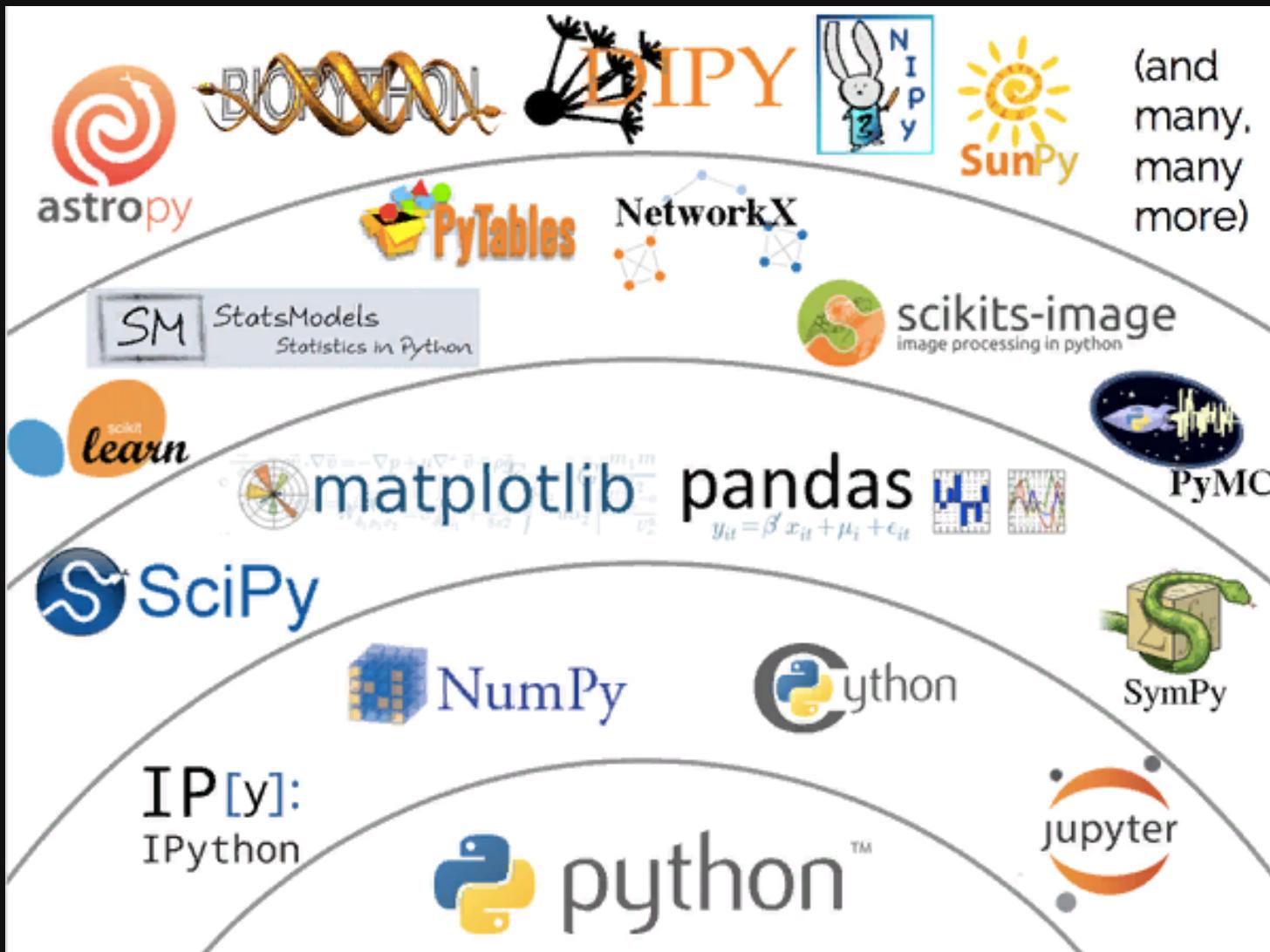


NumPy



pypy

И многие-многие другие



Питон как супер клей помогает
специализированный код на разных
языках и превращать его в реальные
прикладные решения

Работа с параметрами командной строки

```
import sys
if len(sys.argv) < 2:
    print('no args')

~$ python args.py
no args
~$ python args.py print
['args.py', 'print']
```

Argparse

```
import argparse

parser = argparse.ArgumentParser()
parser.add_argument('action')

if __name__ == '__main__':
    args = parser.parse_args()

    if args.action == 'print':
        print('hello')
```

Click

```
import click

@click.command()
@click.option('--count', default=1, help='Number')
@click.option('--name', prompt='Your name',
              help='The person to greet.')
def hello(count, name):
    for x in range(count):
        click.echo('Hello %s!' % name)

if __name__ == '__main__':
    hello()
```

Перерыв?



Какие типы данных в python знаете?



sys.getsizeof

```
In [9]: import sys
```

```
In [10]: sys.getsizeof(1)  
Out[10]: 28
```

```
In [11]: sys.getsizeof(1.)  
Out[11]: 24
```

```
In [12]: sys.getsizeof('1')  
Out[12]: 50
```

```
In [13]: sys.getsizeof([1])  
Out[13]: 80
```

type

```
In [16]: type(None)
```

```
Out[16]: NoneType
```

```
In [17]: type(1)
```

```
Out[17]: int
```

```
In [18]: type(1.)
```

```
Out[18]: float
```

```
In [19]: type('1')
```

```
Out[19]: str
```

```
In [20]: type([1])
```

```
Out[20]: list
```

`id`

```
In [36]: x = 1
```

```
In [37]: y = 2
```

```
In [38]: id(x)
```

```
Out[38]: 4326149920
```

```
In [39]: id(y)
```

```
Out[39]: 4326149952
```

```
In [40]: id(3)
```

```
Out[40]: 4326149984
```

ссылки на участок памяти

None



int

```
In [14]: sys.getsizeof(2)
Out[14]: 28
```

```
In [15]: sys.getsizeof(2*100_000_000)
Out[15]: 28
```

```
In [16]: sys.getsizeof(2*1_000_000_000)
Out[16]: 32
```

Нельзя переполнить!

int

```
In [17]: -5 is -5
Out[17]: True
```

```
In [18]: 256 is 256
Out[18]: True
```

```
In [19]: 267 is 257
Out[19]: False
```

Числа от -5 до 256
кэшируются

bool

```
In [20]: sys.getsizeof(True)
Out[20]: 28
```

```
In [21]: sys.getsizeof(False)
Out[21]: 24
```

```
In [22]: True is True
Out[22]: True
```

Унаследован от int

bool

```
In [23]: True / 10  
Out[23]: 0.1
```

```
In [24]: False ** 100  
Out[24]: 0
```

```
In [25]: True + False  
Out[25]: 1
```

float

```
>>> import sys  
>>> sys.getsizeof(1.0)  
24  
  
>>> sys.float_info  
sys.float_info(max=1.7976931348623157e+308,  
                max_exp=1024,  
                max_10_exp=308,  
                min=2.2250738585072014e-308,  
                min_exp=-1021,  
                min_10_exp=-307,  
                dig=15,  
                mant_dig=53,  
                epsilon=2.220446049250313e-16,  
                radix=2,  
                rounds=1)
```

Может переполниться при возведении в степень

```
In [41]: 10.0**309
```

```
-----  
OverflowError
```

```
Traceback
```

```
<ipython-input-41-35b22fdfd3d0> in <module>  
----> 1 10.0**309
```

```
OverflowError: (34, 'Result too large')
```

-inf, inf, nan

```
In [43]: 1e100  
Out[43]: 1e+100
```

```
In [44]: 1e1000  
Out[44]: inf
```

```
In [45]: -1e1000  
Out[45]: -inf
```

```
In [47]: 1e1000 - 1e1000  
Out[47]: nan
```

Не используйте для точных
расчётов (денег)!

Decimal

Для точных вычислений

Точность можно регулировать!

decimal

```
In [70]: sys.getsizeof(Decimal('1.0'))  
Out[70]: 104
```

bite string

```
In [63]: sys.getsizeof(b'')
```

```
Out[63]: 33
```

```
In [64]: sys.getsizeof(b'a')
```

```
Out[64]: 34
```

```
In [65]: sys.getsizeof(b'ab')
```

```
Out[65]: 35
```

bite string

```
In [6]: b = b'\xd0\x9f\xd1\x80\xd0\xb8  
        \xd0\xb2\xd0\xb5\xd1\x82'
```

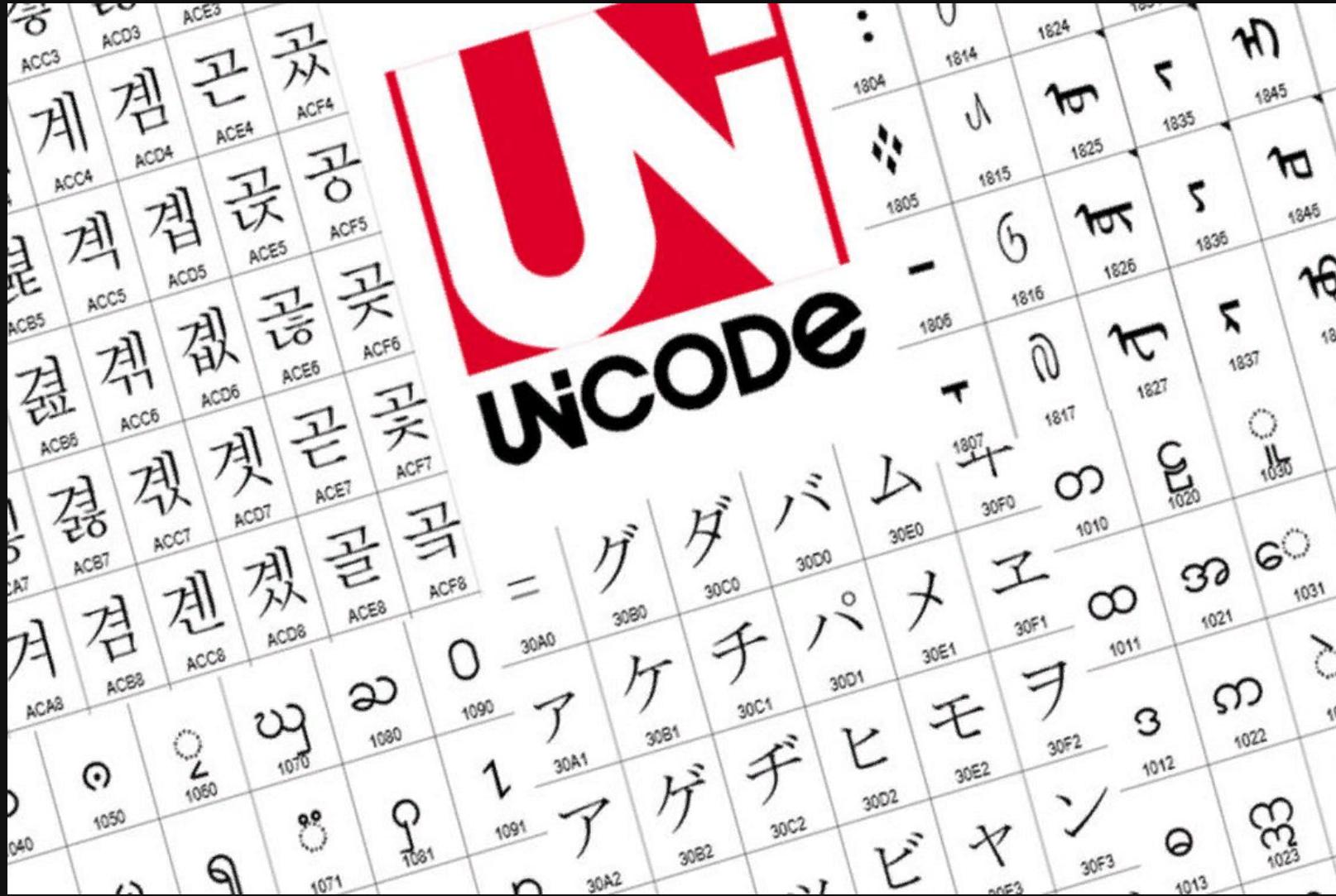
```
In [7]: b.decode()
```

```
Out[7]: 'Привет'
```

str

Представляет юникодные строки

Могут быть интернированы (сохранены во
внутреннем кэше)



wiki

- Первые символы совпадают с ASCII
- Символы кодируются 1/4 байтами (ascii 1 байт, кириллица 2 байта, etc.)

str

```
In [8]: 'Привет'.encode()
Out[8]: b'\xd0\x9f\xd1\x80\xd0\xb8
          \xd0\xb2\xd0\xb5\xd1\x82'
```

Строка незменяемая

```
>>> s[7] = 'M'  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
TypeError: 'str' object does not support item assignment
```

str

```
In [9]: sys.getsizeof(' ')
Out[9]: 49
```

```
In [10]: sys.getsizeof('a')
Out[10]: 50
```

```
In [11]: sys.getsizeof('ab')
Out[11]: 51
```

str

```
In [11]: sys.getsizeof('ab')  
Out[11]: 51
```

```
In [12]: sys.getsizeof('АБ')  
Out[12]: 78
```

```
In [13]: sys.getsizeof('АБВ')  
Out[13]: 80
```

str

Размер символа в памяти зависит от размера самого большого символа в строке по стандарту юникод (1, 2 или 4 байта).

Нужно для легкого
индексирования по
строке

str

```
In [19]: '\n Привет \n'  
Out[19]: '\n Привет \n'
```

```
# сырье строки без учета спецсимволов  
# (нужно для регулярок)  
In [20]: r'\n Привет \n'  
Out[20]: '\\n Привет \\n'
```

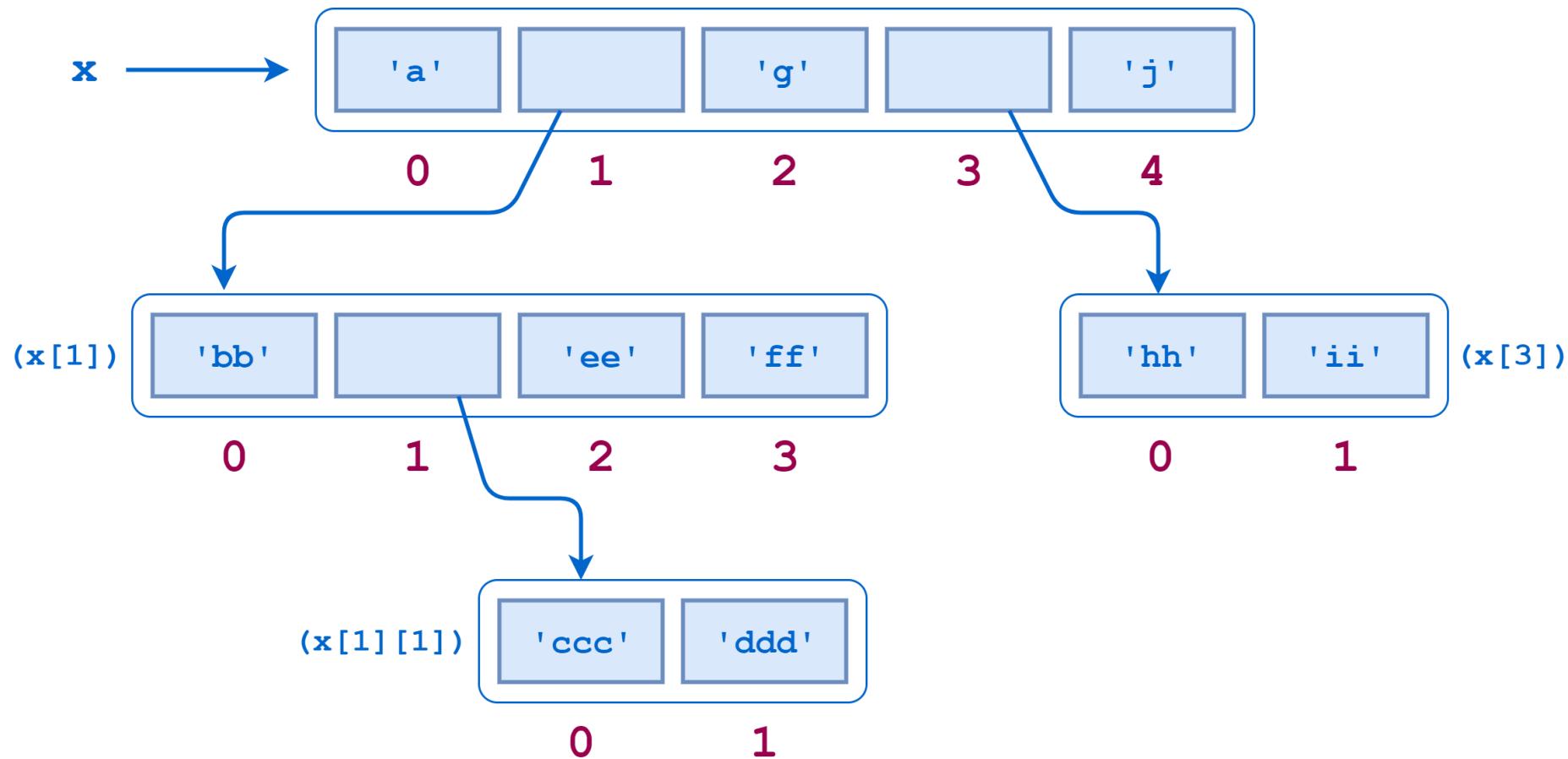
tuple

```
In [26]: sys.getsizeof(tuple())  
Out[26]: 56
```

```
In [27]: sys.getsizeof(tuple([1]))  
Out[27]: 64
```

```
In [28]: sys.getsizeof(tuple([1, 2]))  
Out[28]: 72
```

Хранит ссылки



Неизменяемая*
коллекция фиксированной
длины

Сложность операций

- `t = (1, 2, 3, 1)`
- `t[0]`
- `2 in t` или `t.index(3)`
- `t.count(1)` - кол-во элементов

list

```
In [32]: sys.getsizeof([])
```

```
Out[32]: 72
```

```
In [33]: sys.getsizeof([1])
```

```
Out[33]: 80
```

```
In [34]: sys.getsizeof([1, 2])
```

```
Out[34]: 88
```

При создании выделяется
память на точное кол-во
элементов

list

```
In [35]: l = []
```

```
In [36]: sys.getsizeof(l)
```

```
Out[36]: 72
```

```
In [37]: l.append(1); sys.getsizeof(l) # 4 раза
```

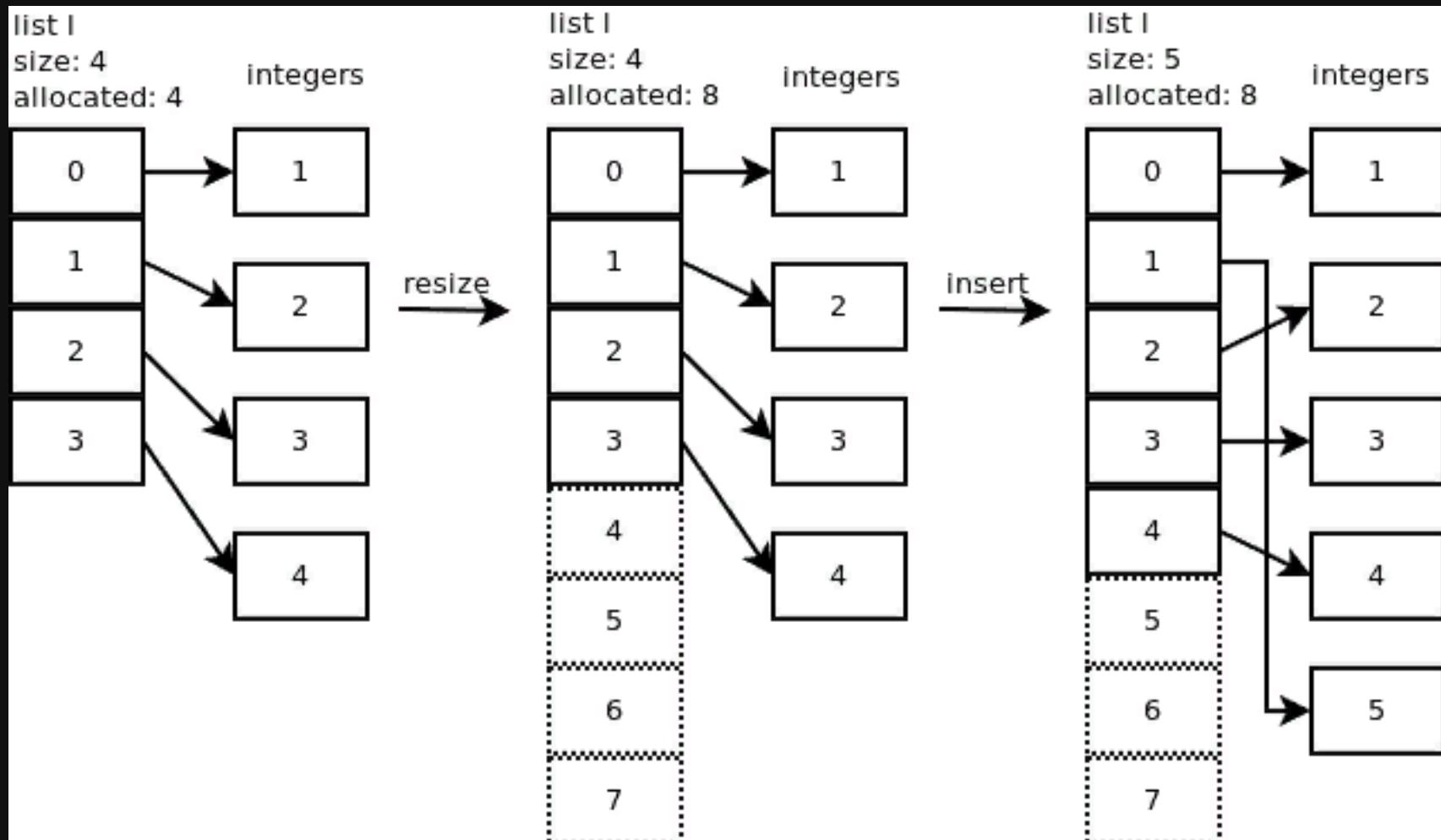
```
Out[37]: 104 # выделили места на 4 ссылки
```

```
In [41]: l.append(1); sys.getsizeof(l)
```

```
Out[41]: 136 # выделили места еще на 4 ссылки
```

При изменении память
выделяется с запасом

Длина динамически меняется



Сложность операций

- `l = []`
- `l.append(1)`
- `l.insert(0, 'value')`

list vs tuple

Если набор значений не будет меняться, то использование tuple на большом кол-ве объектов может сэкономить память

dict

```
In [44]: sys.getsizeof( {} )  
Out[44]: 248
```

```
In [45]: sys.getsizeof( {1: 1} )  
Out[45]: 248
```

```
In [46]: sys.getsizeof( {1: 1, 2: 2} )  
Out[46]: 248
```

```
In [62]: sys.getsizeof( {  
    1: 1, 2: 2, 3: 3, 4: 4, 5: 5, 6: 6  
} )  
Out[62]: 376
```

Всегда берет память с
запасом

dk_indices

Index	Value
0	2
1	-1 (DKIX_EMPTY)
2	1
3	-1 (DKIX_EMPTY)
4	-1 (DKIX_EMPTY)
5	-1 (DKIX_EMPTY)
6	-1 (DKIX_EMPTY)
7	0

dk_entries

Index	Hash	Key	Value
0	2286678025444376703	'a key'	1111
1	2159330848570984162	'dict'	2222
2	-2924782077118122456	'python'	3333
3			
4			

Ключом может быть любой
хэшируемый (неизменяемый)
объект

- int, float, string
- tuple*
- объект с __hash__ методом

Хэш коллекции зависит от хешей элементов

```
In [50]: hash(tuple())
```

```
Out[50]: 3527539
```

```
In [51]: hash((), 1, 2)
```

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

```
Traceback
```

```
<ipython-input-51-8fb42b6c367f> in <module>
```

```
----> 1 hash((), 1, 2))
```

```
TypeError: unhashable type: 'list'
```

Питон поддерживает
его заполненным
не более чем на 2/3
(чтобы избежать коллизий)

В текущей реализации
упорядочен по порядку
вставки

Сложность операций

- $d = \{\}$
- $d['key'] = 'value'$ - вставка
- $d['key']$ - получение

set

```
In [55]: sys.getsizeof(set())  
Out[55]: 232
```

```
In [56]: sys.getsizeof(set([1]))  
Out[56]: 232
```

```
In [58]: sys.getsizeof(set([1, 2, 3, 4]))  
Out[58]: 232
```

```
In [59]: sys.getsizeof(set([1, 2, 3, 4, 5]))  
Out[59]: 744
```

dict без значений с
дополнительными
операциями для множеств

frozenset

- Неизменяемый и хэшируемый
- Никому не нужен
- Память не экономит

frozenset

```
In [99]: In [59]: sys.getsizeof(frozenset())
Out[99]: 232
```

```
In [100]: In [59]: sys.getsizeof(frozenset([1]))
Out[100]: 232
```

frozendict?

```
In [60]: from types import MappingProxyType
```

```
In [62]: m = MappingProxyType({1: 1})
```

```
In [63]: m[2] = 2
```

```
-----  
TypeError Traceback  
<ipython-input-63-050a8142f5a4> in <module>  
----> 1 m[2] = 2
```

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

Collections

<code>namedtuple()</code>	factory function for creating tuple subclasses with named fields
<code>deque</code>	list-like container with fast appends and pops on either end
<code>ChainMap</code>	dict-like class for creating a single view of multiple mappings
<code>Counter</code>	dict subclass for counting hashable objects
<code>OrderedDict</code>	dict subclass that remembers the order entries were added
<code>defaultdict</code>	dict subclass that calls a factory function to supply missing values
<code>UserDict</code>	wrapper around dictionary objects for easier dict subclassing
<code>UserList</code>	wrapper around list objects for easier list subclassing
<code>UserString</code>	wrapper around string objects for easier string subclassing

ChainMap

```
>>> baseline = {'music': 'bach', 'art': 'rembrandt'}
>>> adjustments = {'art': 'van gogh', 'opera': 'carmen'}
>>> list(ChainMap(adjustments, baseline))
['music', 'art', 'opera']
```

Deque

```
>>> from collections import deque
# make a new deque with three items
>>> d = deque('ghi')
# iterate over the deque's elements
>>> for elem in d:
...     print(elem.upper())
G
H
I

# add a new entry to the right side
>>> d.append('j')
# add a new entry to the left side
>>> d.appendleft('f')
# show the representation of the deque
>>> d
deque(['f', 'g', 'h', 'i', 'j'])
```

defaultdict

```
s = [  
    ('yellow', 1),  
    ('blue', 2),  
    ('yellow', 3),  
    ('blue', 4),  
    ('red', 1),  
]  
  
d = defaultdict(list)  
  
for k, v in s:  
    d[k].append(v)
```

namedtuple

```
>>> Point = namedtuple('Point', ['x', 'y'])
# instantiate with positional or keyword arguments
>>> p = Point(11, y=22)
# indexable like the plain tuple (11, 22)
>>> p[0] + p[1]
33
```

namedtuple

Значительно экономит память при
работе с очень большим кол-вом
объектов

typing.NamedTuple

```
class Employee(NamedTuple):
    name: str
    id: int = 3

employee = Employee('Guido')
assert employee.id == 3
```

sorted collections

```
>>> from sortedcontainers import SortedList
>>> sl = SortedList(['e', 'a', 'c', 'd', 'b'])
>>> sl
SortedList(['a', 'b', 'c', 'd', 'e'])
```

<https://pypi.org/project/sortedcontainers/>

dataclass

```
@dataclass
class InventoryItem:
    '''Class for keeping track of an item in inventory.'''
    name: str
    unit_price: float
    quantity_on_hand: int = 0

    def total_cost(self) -> float:
        return self.unit_price * self.quantity_on_hand
```

Надстройка над обычными
классами, которая генерирует
типовыe методы

Не так эффективны по
памяти как namedtuple

Никакие проверки
типов не делаются!

используйте `mypy =)`

Изменяющие дефолты в аргументах

```
In [21]: def func(l=[]):  
    ...:  
        l.append(1)  
    ...:  
        print(l)  
    ...:
```

```
In [22]: func()  
[1]
```

```
In [23]: func()  
[1, 1]
```

Дефолтные значения для
функций вычисляются один
(!) раз и переиспользуются

Ipython

```
~/Projects/support-operator-gateway on ↵ develop ⏲ 18:10:54
$ ipython
Python 3.7.4 (default, Oct 12 2019, 18:55:28)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.5.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: import os

In [2]: os.ch
        chdir()    chmod()    chroot()
        chflags()   chown()
```

Как измерить скорость?

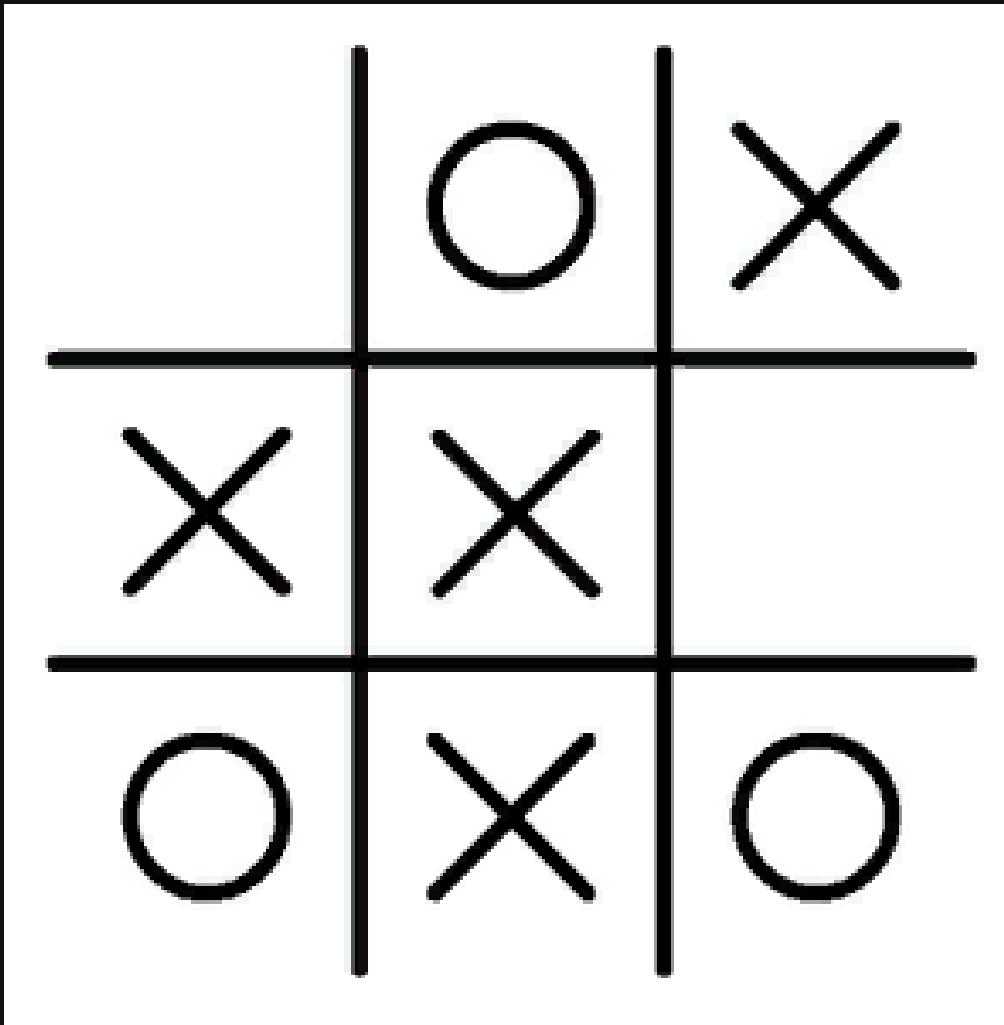
```
$ time python -c 1 + 1
python -c 1 + 1  0.02s user 0.02s system 77% cpu 0.049 total
```

timeit

```
In [1]: %timeit 1 + 1
8.01 ns ± 0.26 ns per loop
(mean ± std. dev. of 7 runs, 100000000 loops each)
```

Попробуйте измерить реальное
время выполнения различных
операций, сравните с
ожидаемым по сложности
алгоритма и удивитесь =)

Домашняя работа



Консольное приложение

Размер поля задается
при запуске

Игра играет против вас
(хотя бы случайными
ходами)

В любой момент можно сохранить игру, чтобы продолжить после закрытия программы

Вариант для храбрых:

- Морской бой
- кол-во кораблей должно зависеть от размера поля

Инструкция по домашним работам
на сайте финтеха. Нужно скинуть в
канал телеграмма свой login в gitlab

Вопросы?