ТЕХНОАТОМ

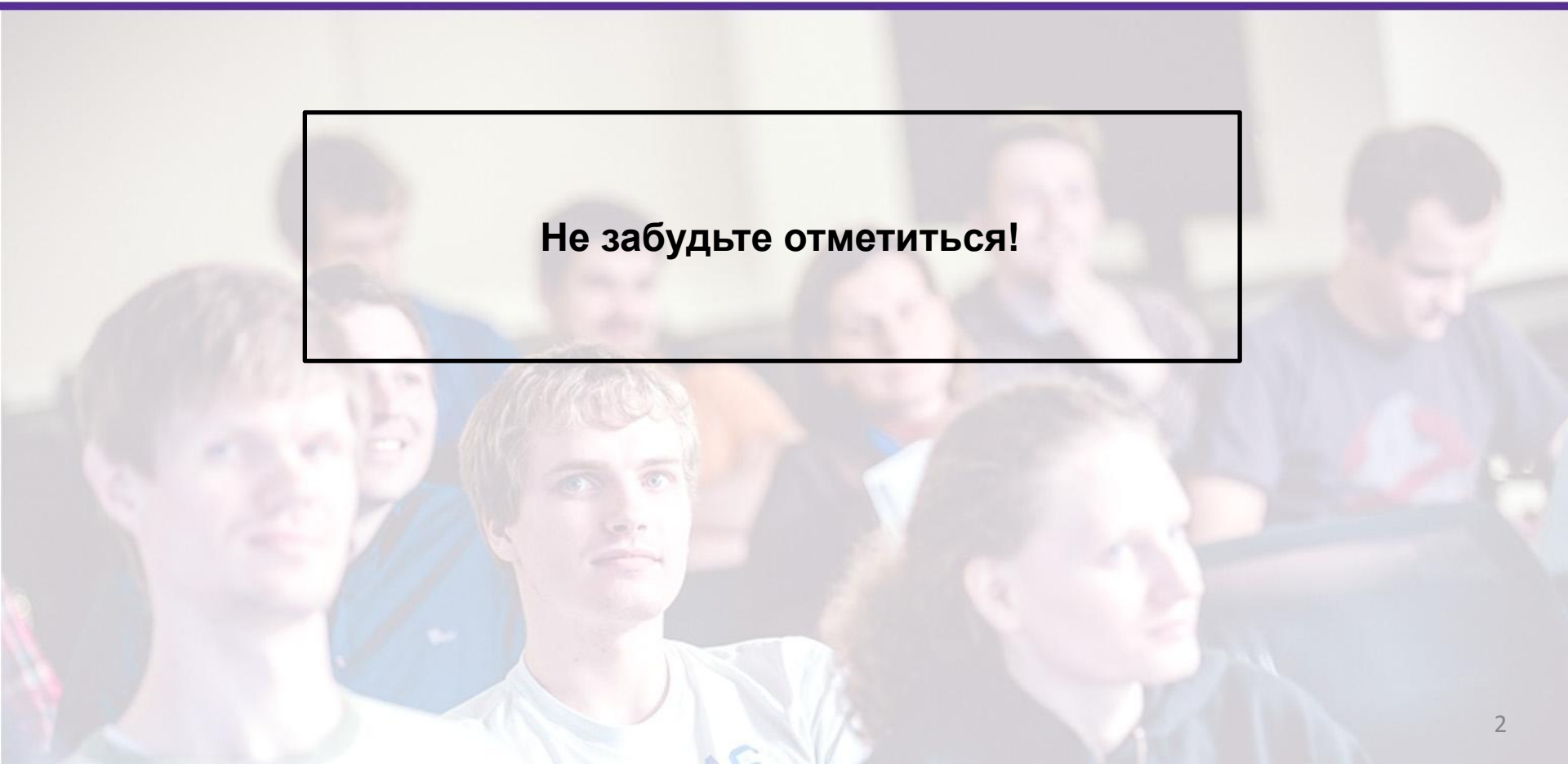
Типы и структуры данных

Урок третий

Антон Кухтичев

Посещаемость

ТЕХНОАТОМ



Не забудьте отметиться!

Программа на сегодня

ТЕХНОАТОМ

- Встроенные типы и структуры данных;
- Обзор стандартной библиотеки. Аннотация типов;

TEXHOATOM

collections

Что такое модуль collections?

ТЕХНОАТОМ

Данный модуль реализует специализированные типы данных контейнера, предоставляя альтернативы для встроенных контейнеров таких как *dict, list, set* и *tuple*.

Что такое модуль collections?

ТЕХНОАТОМ

Рассмотрим:

- *defaultdict*
- *OrderedDict*
- *namedtuple()*
- *deque*
- *Counter*

defaultdict


```
collections.defaultdict([default_factory[,                    ...]])
```

Ничем не отличается от обычного словаря за исключением того, что по умолчанию всегда вызывается функция, возвращающая значение.

defaultdict. Примерчик

ТЕХНОАТОМ

```
1.  >>> import collections
2.  >>> defdict = collections.defaultdict(list)
3.  >>> print(defdict)
4.  defaultdict(<class 'list'>, {})
5.  >>> for i in range(5):
6.  ...     defdict[i].append(i)
7.  ...
8.  >>> print(defdict)
9.  defaultdict(<class 'list'>, {0: [0], 1: [1], 2: [2], 3:
    [3], 4: [4]})
```

OrderedDict


```
collections.OrderedDict([items])
```

Похожий на словарь объект, но он помнит порядок, в котором ему были даны ключи.

OrderedDict. Примерчик

ТЕХНОАТОМ

```
1.  import collections
2.
3.  d = collections.OrderedDict(
4.      [('a', 'A'), ('b', 'B'), ('c', 'C')]
5.  )
6.
7.  for k, v in d.items():
8.      print(k, v)
9.
10. d.move_to_end('b')
```

```
collections.namedtuple(typename, field_names, *, rename=False,
defaults=None, module=None)
```

Именованные кортежи являются неизменяемыми подобно обычным кортежам. Вы не можете изменять их после того, как вы что-то поместили в них.

namedtuple. Примерчик

ТЕХНОАТОМ

```
1.  >>> import collections
2.  >>> Point = collections.namedtuple('Point', ['x', 'y'])
3.  >>> p = Point(11, y=22) # instantiate with positional or
    keyword arguments
4.  >>> p[0] + p[1]         # indexable like the plain tuple
    (11, 22)
5.  33
6.  >>> x, y = p           # unpack like a regular tuple
7.  >>> x, y
8.  (11, 22)
9.  >>> p.x + p.y          # fields also accessible by name
10. 33
```

namedtuple. Методы

ТЕХНОАТОМ

- *_asdict()*
- *_make(iterable)*
- *_replace(**kwargs)*
- *_fields*
- *_fields_default*

namedtuple. Выводы

ТЕХНОАТОМ

- collections.namedtuple — краткая форма для создания вручную эффективно работающего с памятью неизменяемого класса;
- Именованные кортежи могут помочь сделать ваш код чище, обеспечивая вас более простыми в понимании структурами данных;
- Именованные кортежи предоставляют несколько полезных вспомогательных методов которые начинаются с символа подчёркивания (_), но являются частью открытого интерфейса. Использовать их — это нормальная практика.

```
collections.deque([iterable[, maxlen]])
```

Очередь из итерируемого объекта с максимальной длиной *maxlen*. Очереди очень похожи на списки, за исключением того, что добавлять и удалять элементы можно либо справа, либо слева.

deque. Примерчик

ТЕХНОАТОМ

```
1.  >>> from collections import deque
2.  >>> d = deque('ghi')   # make a new deque with three items
3.  >>> d.append('j')      # add a new entry to the right side
4.  >>> d.appendleft('f')  # add a new entry to the left side
5.  >>> d                  # show the representation of the
    deque
6.  deque(['f', 'g', 'h', 'i', 'j'])
7.  >>> d.pop()       # return and remove the rightmost item
8.  'j'
9.  >>> d.popleft()  # return and remove the leftmost item
10. 'f'
```

deque. Методы

- *append(x)/appendleft(x)*
- *clear()*
- *copy()*
- *count(x)*
- *extend(iterable)/extendleft(iterable)*
- *index(x[, start[, stop]])*
- *insert(i, x)*
- *pop()/popleft()*
- *remove(value)*
- *reverse()*

Counter


```
collections.Counter([iterable-or-mapping])
```

Это подкласс dict для подсчёта хешируемых объектов.

Counter. Методы

ТЕХНОАТОМ

- `elements()`
- `most_common([n])`
- `subtract([iterable-or-mapping])`
- `update([iterable-or-mapping])`

Counter. Примерчик

ТЕХНОАТОМ

```
1. >>> import re
2. >>> words = re.findall(r'\w+',
   open('hamlet.txt').read().lower())
3. >>> Counter(words).most_common(10)
4. [('the', 1143), ('and', 966), ('to', 762), ('of', 669),
   ('i', 631),
5.  ('you', 554),  ('a', 546), ('my', 514), ('hamlet', 471),
   ('in', 451)]
6.
```

Очередь с приоритетом. Реализована через кучу (heap).

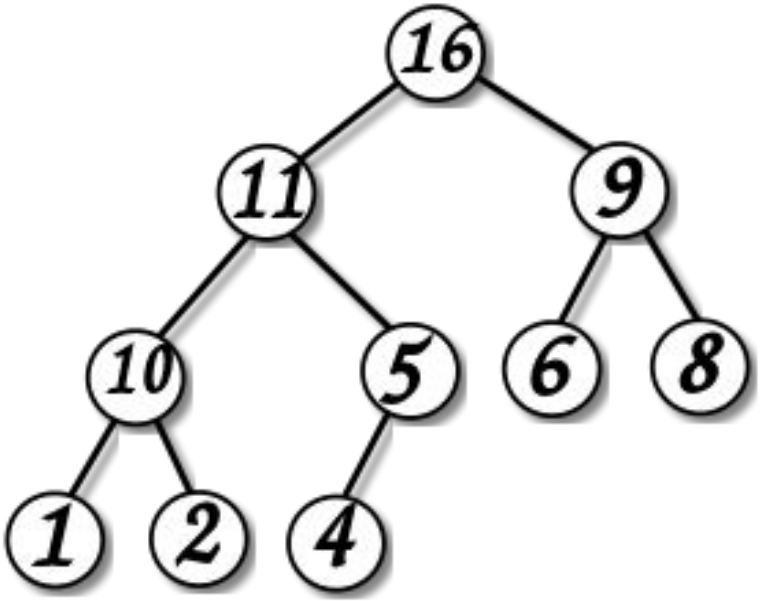
heapq. Двоичная куча (heap)

ТЕХНОАТОМ

1. Значение в любой вершине не меньше, чем значения её потомков;
2. Глубина всех листьев (расстояние до корня) отличается не более чем на 1 слой;
3. Последний слой заполняется слева направо без «дырок».

heapq. Двоичная куча (heap)

ТЕХНОАТОМ



16	11	9	10	5	6	8	1	2	4

heapq. Методы

ТЕХНОАТОМ

- `heapq.heappush(heap, item)`
- `heapq.heappop(heap)`
- `heapq.heappushpop(heap, item)`
- `heapq.heapify(x)`
- `heapq.heapreplace(heap, item)`
- `heapq.merge(*iterables)`
- `heapq.nlargest(n, iterable[, key])`
- `heapq.nsmallest(n, iterable[, key])`

heapq. Примерчик

ТЕХНОАТОМ

```
1.  >>> def heapsort(iterable):
2.  ...     h = []
3.  ...     for value in iterable:
4.  ...         heappush(h, value)
5.  ...     return [heappop(h) for i in range(len(h))]
6.  ...
7.  >>> heapsort([1, 3, 5, 7, 9, 2, 4, 6, 8, 0])
8.  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

ТЕХНОАТОМ

Бинарные деревья

Бинарные деревья поиска. Binary search tree (BST)

БДП — это бинарное, или двоичное дерево, для которого выполняются следующие дополнительные условия (свойства дерева поиска):

- Оба поддерева — левое и правое — являются двоичными деревьями поиска.
- У всех узлов левого поддерева произвольного узла X значения ключей данных меньше, нежели значение ключа данных самого узла X.
- У всех узлов правого поддерева произвольного узла X значения ключей данных больше либо равны, нежели значение ключа данных самого узла X.

Аннотация типов

Аннотации типов просто считываются интерпретатором Python и никак более не обрабатываются.

Аннотация типов. Особенности

ТЕХНОАТОМ

- Пишутся непосредственно в коде;
- Повышается информативность исходного кода;
- Аннотации переменных:

```
price: int = 5
title: str
titles: List[str] = ["hello", "world"]
```

- Аннотации функций:

```
def isBST(root: TreeNode) -> bool:
    # Проверка, что БД является БДП.
```

Домашнее задание

ТЕХНОАТОМ

1. Реализовать очередь с максимальным приоритетом (max heap) без использования heapq;
2. Реализовать LRU cache (least recently used); Для кого это просто, тогда LFU cache;
3. Написать класс для нахождения медианы в потоке данных;

Полезные ссылки

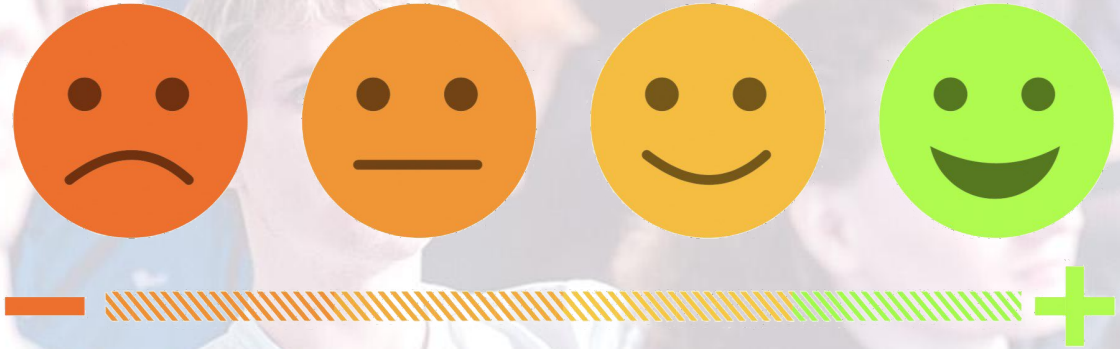
Модуль collections
Модуль heapq
Алгоритмы. Построение и анализ | Кормен Томас Х., Лейзерсон Чарльз И.

Для саморазвития (опционально)
Чтобы не набирать двумя пальчиками

Оценивание занятие

ТЕХНОАТОМ

Не забудьте оценить занятие!



ТЕХНОАТОМ

Спасибо за внимание!

Антон Кухтичев

a.kukhtichev@corp.mail.ru