Тема 3.
Структуры данных в Python

Структуры данных в Python

- В Python существует четыре типа данных для хранения последовательностей
  - List (список) – изменяемая упорядоченная последовательность. Допускаются одинаковые элементы (Пример, a = [1, 2, 3])
  - Tuple (кортеж) – неизменяемая упорядоченная последовательность. Допускаются одинаковые элементы (Пример, b = (1, 2, 3))

Структуры данных в Python

- **В Python существует четыре типа данных для хранения последовательностей**
  - **Set (множество) – изменяемая неупорядоченная последовательность. Не допускаются одинаковые элементы (Пример, c = {1, 2, 3 })**
  - **Dictionary (словарь) – изменяемый неупорядоченный набор пар «ключ-значение»**

Списки

Список (list) – это упорядоченная изменяемая последовательность элементов.

Списки

- **Особенности:**
  - **может содержать элементы разного типа;**
  - **поддерживают доступ по смещению.**
  - **имеют переменную длину, разнородны и допускают произвольно глубокое вложение.**
  - **поддерживает операторы сравнения: при этом сравнивание производится поэлементно (и рекурсивно, при наличии вложенных элементов).**
  - **представляют собой массивы ссылок на объекты**

Списки

- **Способы создания списка:**

1. **с помощью функции**
   `list([Последовательность])`

Примеры:

```
b = list()
s = list("String")      # s = ['S', 't', 'r', 'i', 'n', 'g']
t = list( (1,2,3,4,5) )   # t = [1, 2, 3, 4, 5]
```

Списки

- **Способы создания списка:**
  2. **перечисление элементов списка внутри [ ]**

```
a = [1,  “str”, 3.456, (1,2,3,4,5)]
```

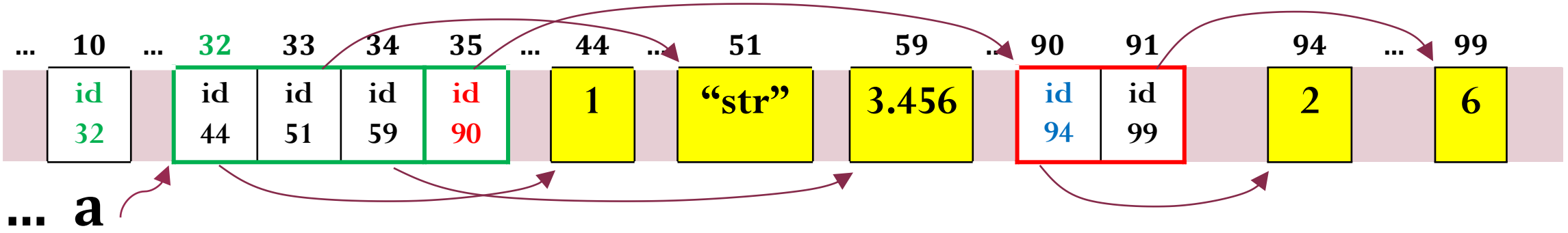
  3. **методом append поэлементно**

```
a = []
a.append(1)
a.append("str")
a.append([6, 96])      # a = [1, 'str', [6, 96]]
```

Списки

- **Создание списка**
  - **При создании списка в переменной сохраняется ссылка на объект, а не сам объект!!!**
  - **Например,**

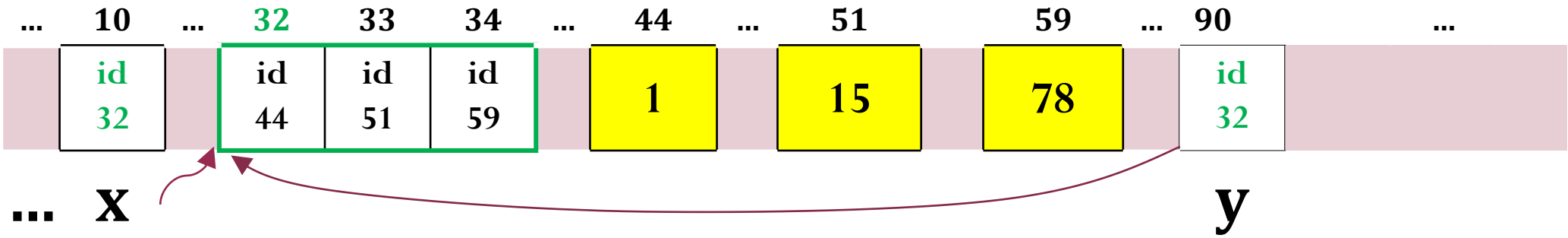
$$a = [1,\ \text{“str”},\ 3.456,\ (2,\ 6)]$$



Списки

- **Создание списка**
  - **Групповое присваивание со списками лучше не использовать!**

```
x= y = [1, 15, 78] # на один список [1,15,78] ссылаются x, y
```




```
x, y = [1, 15, 78], [1, 15, 78] # созданы два списка
```

Списки

- **Базовые операции над списками:**
  - **Доступ по индексу (нумерация с 0)**

```
list1 = [1, 15, 78, 56]
print(list1[1])
```

Результат: **15**

  - **Редактирование по индексу**

```
list1 = [1, 15, 78, 56]
list1[2] = 67
print(list1)
```

Результат: **1, 15, 67, 56**

Списки

- **Базовые операции над списками:**
  - **Извлечение среза [X:Y:Z]**

```
list1 = [1, 15, 78, 56,43,534,884,2345,678,63,624]
print(list1[1:8:2])
```

Результат: [15, 56, 534, 2345]

```
print(list1[3:-3])
```

Результат: [56, 43, 534, 884, 2345]

```
print(list1[:4])
```

Результат: [1, 15, 78, 56]

Списки

- **Базовые операции над списками:**
  - **Извлечение среза [X:Y:Z]**

```
list1 = [1, 15, 78, 56, 43, 534, 884, 2345, 678, 63, 624]
print(list1[::-1])
```

Результат:

[624, 63, 678, 2345, 884, 534, 43, 56, 78, 15,1]

Списки

- **Базовые операции над списками:**
  - **Редактирование списков**

```
list1 = [1, 84, 67, 63, 24] # замена элементов
list1[:2] = [2, 3 ]
print(list1)
```

Результат: [2, 3, 67, 63, 24]

```
list1[3:]=[] # удаление элементов
print(list1)
```

Результат: [2, 3, 67]

Списки

- **Базовые операции над списками:**
  - **Редактирование списков**

```
list1 = [1, 84, 67, 63, 24] # вставка элементов
list1[2:2] = [24, 13 ]
print(list1)          Результат: [1, 84, 24, 13, 67, 63, 24]
list1[:0]= list1      # вставка копии списка в его
                      #начало список
print(list1)                          Результат: ?
```

Списки

- **Базовые операции над списками:**
  - **Объединение списков: +**

```
list1 = [1, 84, 67,63, 24]
list2 = list1[2:] + [15, 8, 6, 3, 53, 8]
print(list2)        # Результат: [67, 63, 24,15, 8, 6, 3, 53, 8]
```

Списки

- **Базовые операции над списками:**
  - **Дублирование списков** *

```
list1 = [1, 84, 67,63, 24]
list2 = list1[2:]*3
list3 = [0]*5
print(f"list2={list2} \n list3={list3}")
```

Результат: list2=[67, 63, 24, 67, 63, 24, 67, 63, 24]
list3=[0, 0, 0, 0, 0]

Списки

- **Базовые операции над списками:**
  - **Перебор элементов списка**

```
for i in list:
    тело цикла
```

Пример:

Вычислить сумму нечетных элементов, расположенных после 3-го элемента списка

```
list1=[]
for x in range(int(input("n = "))):
    list1.append(int(input("a = ")))
sum=0
for i in list1[3:]:
    if i%2!=0: sum+=i
print(f"list1 -> {list1}")
print(f"sum={sum}")
```

Списки

- **Базовые операции над списками:**
  - **len(список) – возвращает длину списка**
  - **min(список) – возвращает наименьший элемент списка**
  - **max(список) – возвращает наибольший элемент списка**
  - **sum(список) – возвращает сумму элементов списка**

Списки

- **Базовые операции над списками:**
  - **Функция sorted(список) сортирует элементы списка по возрастанию и возвращает новый список с отсортированными элементами.**

```
list1 = [1, 84, 67, 63, 24]
print(sorted(list1))
print(list1)
```

Результат: [1, 24, 63, 67, 84]
[1, 84, 67, 63, 24]

Списки

- ***Пример*.**
- **Дан список целых чисел. Определить позицию максимального элемента списка, расположенного после второго нечетного элемента.**

Списки

- ***Пример*** **(программа)**

```
list1=[34, 125, 6, 7, 8, 95, 4, 5, 63, 21]
p,k= -1,0
for i in range(len(list1)):
    if list1[i]%2!=0:
        k+=1
        if k == 2: p=i; break

if p>0:
    print(max(list1[p+1:]))
else: print("нет двух нечетных")
```

Перебор индексов списка
Доступ к элементам по индексу

Генерация случайных чисел: Модуль random

- **Методы:**
  - **random.random() – случайное число от 0.0 до 1.0**
  - **random.uniform(start, end) – случайное вещественное число от start до end**
  - **random.randint(start, end) – случайное целое число от start до end**
  - **random.randrange(start, end, step) – случайное целое число от start до end с шагом step**

Генерация случайных чисел: Модуль random

- **Методы:**
  - **random.choice(набор) – случайный элемент из набора (строки, списка, кортежа)**
  - **random.shuffle(набор) – перемешивает набор данных (применяется только к изменяемым наборам)**

Списки

- ***Пример*.**
- **Даны два списка, заполненные случайными двузначными числами. Получить третий список, включив в него все общие элементы первых двух списков. Вычислить сумму элементов полученного списка**

Списки

- **Пример (программа)**

```
import random
lst1,lst2,lst3=[],[],[]

for i in range(int(input("n = "))):
    lst1.append(random.randint(10, 99))
    lst2.append(random.randrange(10, 99, 4))
print(f"lst1={lst1}\nlst2={lst2}")

for x in lst1:
    if x in lst2: lst3.append(x)

print(f"lst3={lst3},    sum = {sum(lst3)}" )
```

Добавляем в список lst1 случайное число из [10,99]

Списки

- **Генераторы списков – это специальные конструкции, позволяющие создавать заполненные определенным образом списки.**

- **Синтаксис:**

список = [выражение for переменная in набор]

Списки

- **Генераторы списков.**

«Классический» способ:

```
a = []
for i in range(1, 15):
    a.append(i)
```

Через генератор

```
a = [ i for i in range(1, 15) ]
```

Списки

- **Генераторы списков**

Примеры:

```
a = [1 for i in range(6)]          # аналогично a=[1]*6
                    # результат [1,1,1,1,1,1]
b = [i*2 for i in range(6)]
                    # результат [0,2,4,6,8,10]
c = [random.randint(1,9) for i in range(6)]
                    # результат [1,8,4,3,5,9]
```

Списки

- **Генераторы списков**

Примеры:

```
d = [i for i range(2,10) if i % 2 == 0]
                        # результат [2,4,6,8]
e = [(lambda x: x*2 if x%2==0 else 0)(i) for i in
range(2,10)]
                    # результат [2,0,8,0,12,0,16,0]
```

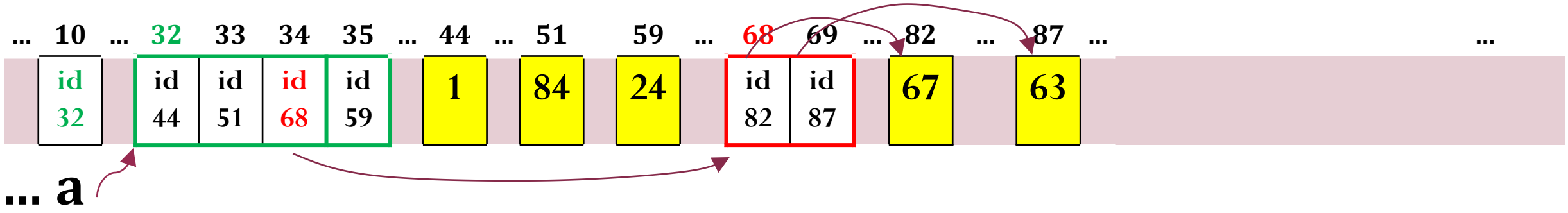
Списки

- **Базовые операции над списками:**
  - **Копирование списков:**
    - ***поверхностное*** **– создает новый объект, заполненный ссылками на элементы оригинала**
    - ***глубокое*** **– создает новый объект и рекурсивно создаются копии всех объектов оригинала**

Списки

- **Базовые операции над списками:**
  - **Поверхностное копирование списков:**

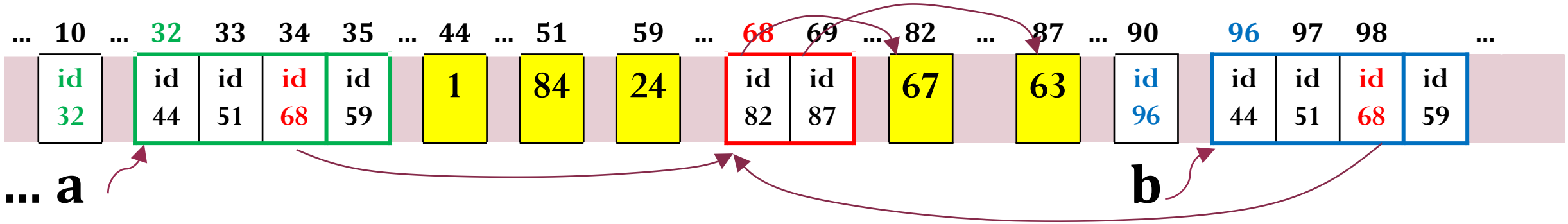
```
a = [1, 84, [67, 63], 24]
b = a[:] # поверхностное копирование
c = a
```



Списки

- **Базовые операции над списками:**
  - **Поверхностное копирование списков:**

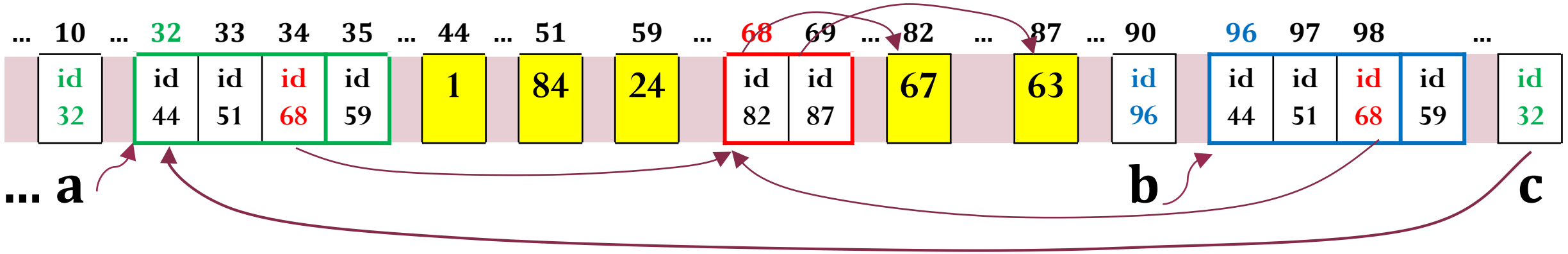
```
a = [1, 84, [67, 63], 24]
b = a[:] # поверхностное копирование
c = a
```



Списки

- **Базовые операции над списками:**
  - **Поверхностное копирование списков:**

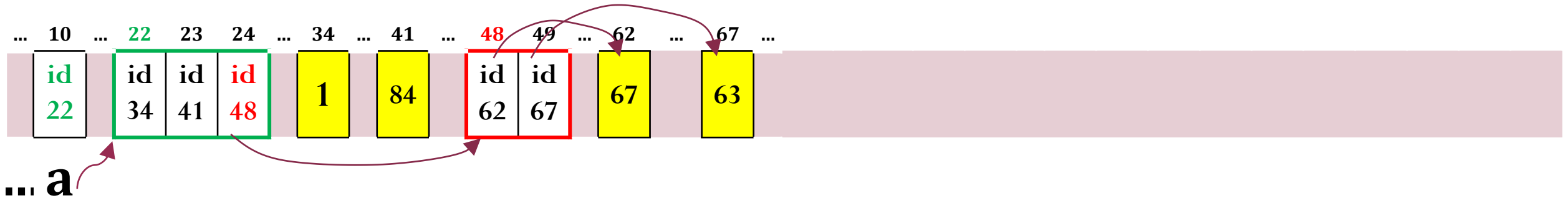
```
a = [1, 84, [67, 63], 24]
b = a[:] # поверхностное копирование
c = a  # c является псевдонимом списка a
```



Списки

- **Базовые операции над списками:**
  - **Глубокое копирование списков:**

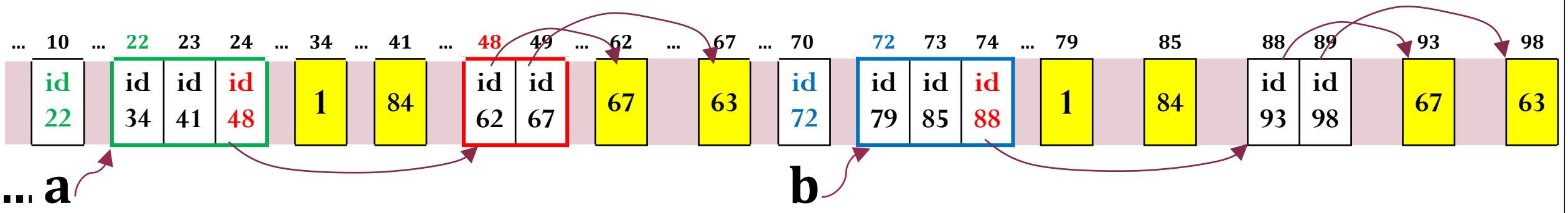
```
import copy
a = [1, 84, [67, 63]]
b = copy.deepcopy(a) # глубокое копирование
```



Списки

- **Базовые операции над списками:**
  - **Глубокое копирование списков:**

```
import copy
a = [1, 84, [67, 63]]
b = copy.deepcopy(a) # глубокое копирование
```



Списки

- **Методы для работы со списками: эти методы меняют список на месте**
  - **lst.append() – добавляет в конец списка lst указанный элемент.**

```
list1 = ['1', '84', '67', '63', '24']
list1.append(45)
print(list1)                    # list1 = ['1', '84', '67', '63', '24', 45]
```

Списки

- **Методы для работы со списками: эти методы меняют список на месте, ничего не возвращая**
  - **lst.extend(list_) – расширяет список lst, добавляя в конец все элементы списка list_.**

```
list1 = ['1', '84', '67', '63', '24']
list1.extend([45, 678, 123])

print(list1)  # list1 = ['1', '84', '67', '63', '24', 45, 678, 123]
```

Списки

- **Методы для работы со списками:**
  - **lst.insert(ind, элемент) – вставляет элемент в список в позицию ind.**

```
list1 = [1, 84, 67,63, 24 , 84]
list1.insert(3, 555))

print(list1)
                    # list1 = [1, 84, 67, 555, 63, 24, 84 ]
```

Списки

- **Методы для работы со списками:**
  - **lst.remove(элемент) – удаляет из списка lst первое вхождение заданного элемента. Если такого элемента в списке нет, то ValueError.**

```
list1 = [1, 84, 67,63, 24 , 84]
list1.remove(84))

print(list1)          # list1 = [1, 67, 63, 24, 84 ]
```

Списки

- **Методы для работы со списками:**
  - **lst.pop([индекс]) – удаляет последний элемент (или элемент с заданным индексом) из списка lst и возвращает удаленный элемент.**

```
list1 = [1, 84, 67,63, 24]      #   63
print(list1.pop(3))
                                # list1 = [1, 84, 67, 24 ]
print(list1)
```

Списки

- **Методы для работы со списками:**
  - **lst.index(x, [start [, end]]) – возвращает индекс первого элемента со значением x (поиск по всему списку или от start до end). Если такого элемента в списке нет, то ValueError.**

```
list1 = [1, 84, 67,63, 24 , 84]
list1.index(84))

print(list1)          # list1 = [1, 67, 63, 24, 84 ]
```

Списки

- **Методы для работы со списками:**
  - **lst.count(x) – возвращает число вхождений в список lst элемента x.**

```
list1 = [1, 84, 67,63, 24 , 84]

print(list1.count(84))          # 2
```

Списки

- **Методы для работы со списками:**
  - **lst.sort([key=функция]) – сортирует список lst на основе функции (по умолчанию, сортировка по возрастанию, т.е. reverse=0). Функция – это функция одной переменной, применяемая к каждому элементу списка перед сортировкой.**
  - **Сортировка по убыванию**

    lst.sort(reverse=1)

***Важно*: изменяется сам список lst, в отличие от функции sorted(), которая возвращает отсортированный список.**

Списки

- **Методы для работы со списками:**
  - **lst.sort([key=функция])**

  - **Примеры**

```
list1=[34, -125, 6, -7, 8, -95, -4, 5, 63, 21]
list1.sort();                 # [-125, -95, -7, -4, 5, 6, 8, 21, 34, 63]
list1.sort(reverse=1);        # [63, 34, 21, 8, 6, 5, -4, -7, -95, -125]
list1.sort(key=abs);          # [-4, 5, 6, -7, 8, 21, 34, 63, -95, -125]
```

Списки

- Методы для работы со списками:
  - lst.sort([key=функция])
  - Примеры

```
list2 = list("Маша Саша и Даша")
list2.sort()
print("".join(list2)) #  ДМСаааааааишшшш
list2.sort(key=str.lower)
print("".join(list2)) #  аааааааДиМСшшшш
```

Списки

- **Методы для работы со списками:**
  - **lst.reverse() – разворачивает список**
  - **lst.copy() – поверхностная копия списка**
  - **lst.clear() – очищает список**

Списки

- **Многомерные списки**
- **Работа с многомерными списками ничем не отличается от обычных списков, только при обращении к ним нужно указывать несколько индексов.**
  - **Пример**

```
s= [1,2,3,'Python',5, [3.4,'a']]
print(s[1],' ', s[5][1])          # 2   a
```

Кортежи

Кортеж (tuple) – это упорядоченная неизменяемая последовательность элементов.

Кортеж

- **Кортеж – это список, элементы которого нельзя менять.**
- **Обычно используется в случаях, когда данные постоянны на всем протяжении выполнения программы.**
- **Кортеж заключается в круглые скобки (…)**
- **Кортеж может быть многомерным**

Кортежи

- **Способы создания кортежа:**

1. **с помощью функции**
   `tuple([Последовательность])`

Примеры:

```
b = tuple()                  # b = ()
s = tuple("String")          # s = ('S', 't', 'r', 'i', 'n', 'g')
t = tuple( [1,2,3,4,5] )     # t = (1, 2, 3, 4, 5)
```

Кортежи

- **Способы создания кортежа:**
  2. **Перечислением элементов в круглых скобках**

Примеры:

```
a = ()                  # a = ()
b = (4)                 # b = 4 – Число, не кортеж!!!
s = (5, )               # s = (5, ) – кортеж
t = 12, "23", 4.5       # t = (12, '23', 4.5) – кортеж
v = 54,                 # v = (54, ) – кортеж
```

Кортеж

- Операции над кортежами и методы кортежей аналогичны средствам работы со строками при условии, что они не изменяют содержимое.
- К ним можно отнести:
  - Объединение кортежей
  - Повторение кортежей
  - Извлечение среза
  - Доступ по индексу
  - …