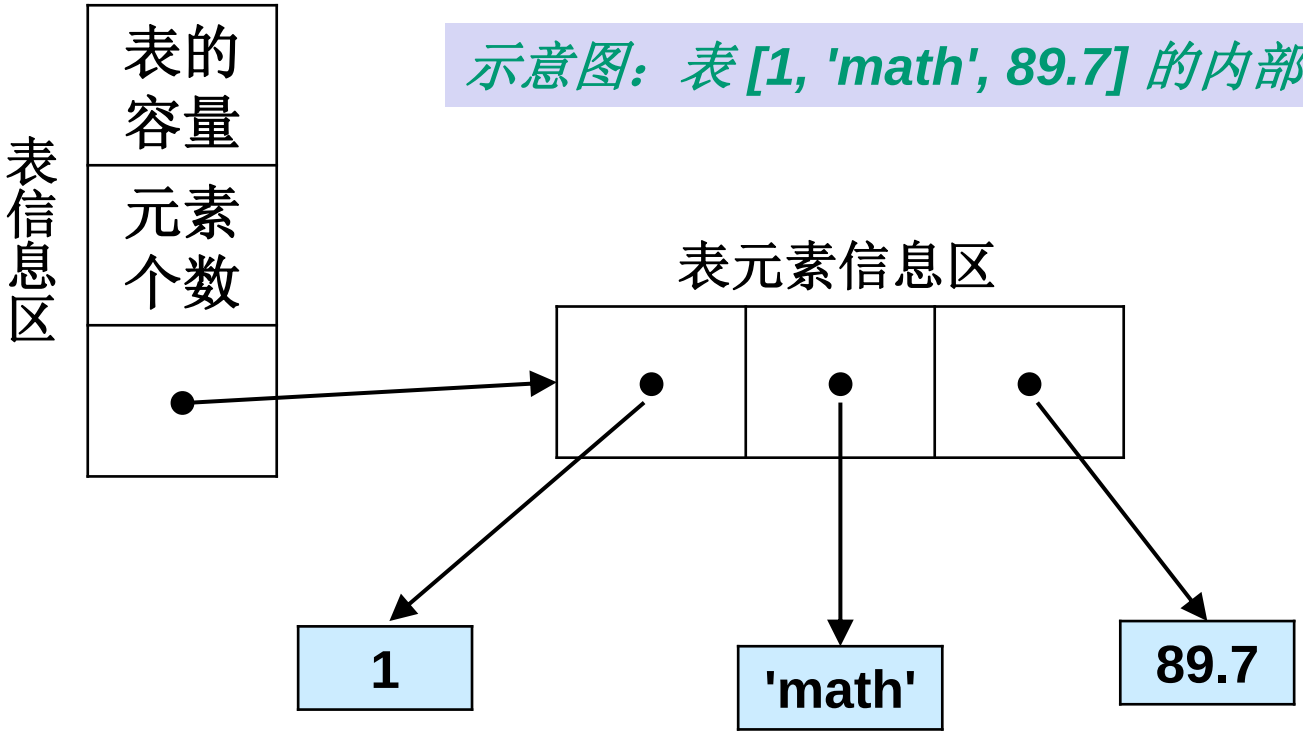
数据的组织和操作

- list (表)
  - 构造、基本操作
- 元组
- 序列、不变对象和可变对象
- 字典
- 集合

表 (list) 的内部实现 (基本情况)

- Python 的 list 属于动态、顺序、引用型数组
  - 在 Python 的官方实现里，采用分离式结构 (非官方的实现可以采用其它技术)



示意图：表 [1, 'math', 89.7] 的内部实现

通过操作构造表对象 (1)

- 调用函数 **list(...)** 构造 (实际是做类型转换)
  - 实参应该是一个序列、迭代器等可迭代对象 **(iterable)**

```
>>> list('abc') # 相当于写 ['a', 'b', 'c']
['a', 'b', 'c']
>>>
>>> list(range(1, 10, 2))
```

值有规律的整数表

```
>>> list(1234) # 实参不是 iterable 则报错
Traceback (most recent call last):
  File "<pyshell#36>", line 1, in <module>
    list(1234) # 实参不是 iterable 则报错
TypeError: 'int' object is not iterable
>>>
>>> list(str(1234))
['1', '2', '3', '4']
```

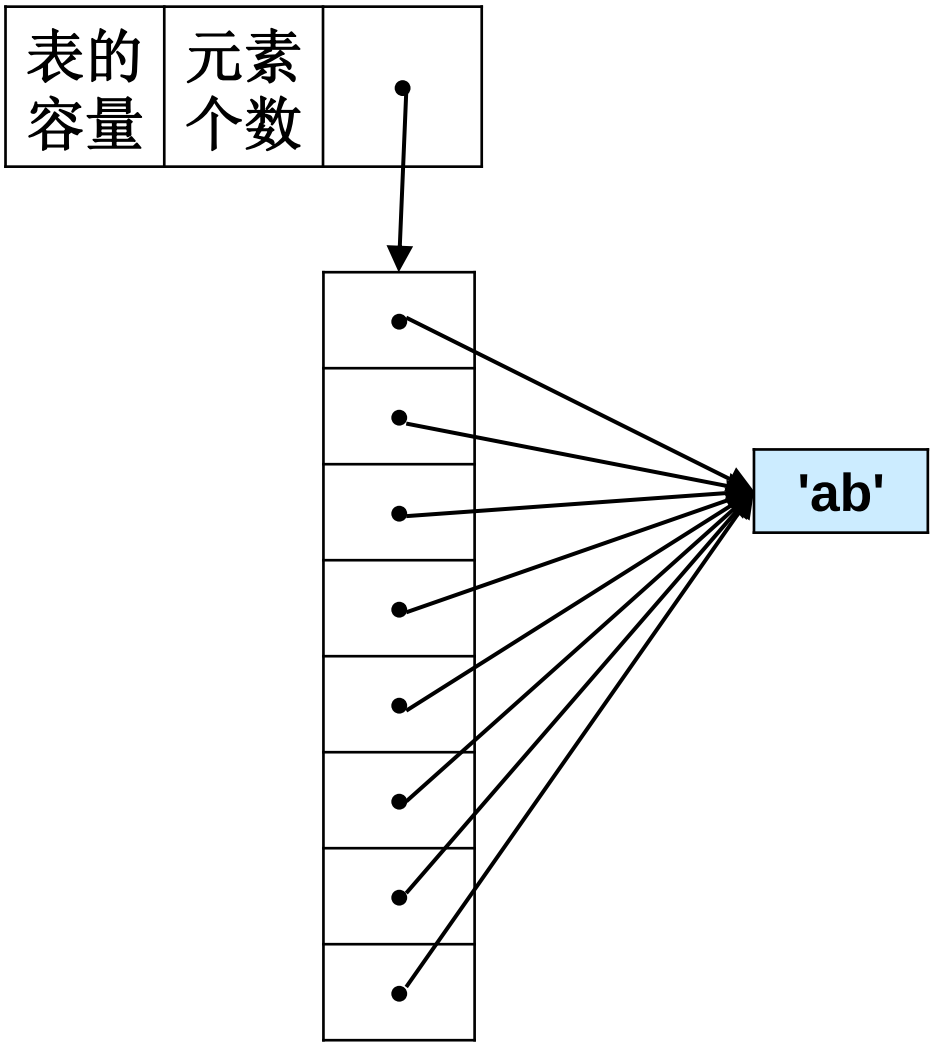
通过操作构造表对象 (2)

```
>>> lst1, lst2 = [1, 2], list('ab')
>>> lst1 + lst2  # 表的拼接，得到一个新表
[1, 2, 'a', 'b']
>>>
>>> lst1 * 3 # 表的拷贝拼接，得到一个新表
[1, 2, 1, 2, 1, 2]
>>>
>>> 2 * lst2 # 拼接和拷贝拼接操作适用于所有序列类型
['a', 'b', 'a', 'b']
>>>
>>> lst1 * 0 # 此时得到一个空表
[]
>>> [None] * 3 # 构造给定长度，元素无特定值的表
[None, None, None]
>>>
>>> lst1.append(3) # 在表尾添加一个元素 (lst1 被修改，无返回值)
>>>
>>> lst1
[1, 2, 3]
```

更多表操作之后介绍

拷贝拼接所构造新表的内部实现

```
>>> lst = ['ab']
>>> lst1 = lst * 8
```



示意图：lst1 的内部实现

表的构造 (3)：表推导式/描述式 (list comprehension)

- 推导式 (comprehension, 类似于逻辑和集合论中的内涵表示)
  - 用于描述一个序列 (不限于表)，其元素的生成具有一定规律
  - 至少包含一个表达式、一个表示重复构造方式的 for 段
- 推导式的基本形式：表达式 for 变量 in 表达式1

```
x**2 for x in range(10)
```

  - for 段：其中的变量控制生成中的迭代，称为迭代变量；表达式1 的值应该是可迭代对象
  - 表达式：描述元素的生成方式 (常含变量)，称为生成表达式
  - 一个推导式：表示一个值序列，变量依次取 表达式1 所给出的值，对每个值计算生成表达式，从而得到序列中各个元素

生成器表达式

- ***Note***：推导式本身并不是完整的 Python 表达式；但 **(推导式)** 是合法表达式，称为**生成器表达式 (generator expression)**

```
>>> (x**2 for x in range(10))
<generator object <genexpr> at 0x000001F09D461DD0>
```

- 生成器表达式的求值并没有实际创建组合对象，其值是一个**生成器对象 (generator object)**
  - 是一种**迭代器 (iterator)**，可产生一系列可用的值；例如，可以作为 **for** 中循环变量的取值源等

```
for x in (i**2 for i in range(100)):
    if x % 11 == 0:
        print(x, end=" ")
```

表推导式

- 推导式可以用来构造各种序列对象，包括表、元组、字典等
  - 表推导式：生成具有规律元素值的表对象

```
>>> squares = [n**2 for n in range(10)]
>>> squares
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

```
>>> lst = list(x*3 for x in 'abc')
>>> lst
['aaa', 'bbb', 'ccc']
```

- 推导式在基本形式之后可以有任意多个 for 段或 if 段

推导式的一般形式

1. 带条件的推导式：表达式 for 变量 in 表达式1 if 逻辑表达式

[x for x in range(200) if x % 7 == 3]

- if 段：其中的 逻辑表达式 应涉及迭代变量，作为筛选条件来筛除 for 段产生的部分不满足条件的值；(也就是，只对满足逻辑表达式要求的变量取值，去计算序列元素)
- 实现枚举 (for 段) - 筛选 (if 段) – 生成 (生成表达式) 的过程

2. 含多个 for 段的推导式，引入多个迭代变量

```
>>> list(s + t for s in "abc" for t in "12")
['a1', 'a2', 'b1', 'b2', 'c1', 'c2']
```

```
>>> [i1 + i2 for i1 in range(4) for i2 in range(i1 + 2)]
[0, 1, 1, 2, 3, 2, 3, 4, 5, 3, 4, 5, 6, 7]
```

推导式和作用域

- 一个推导式引进一个新作用域，嵌套在当前作用域之中
  - 在推导式 for 段中引进的迭代变量是局部变量，其作用域包含：本推导式的生成表达式，以及位于该 for 段之后的部分

```
>>> [i1 + i2 for i1 in range(i2) for i2 in range(4)]
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    [i1 + i2 for i1 in range(i2) for i2 in range(4)]
NameError: name 'i2' is not defined
```

```
def func(n):
    lst1 = [x * n for x in range(n)] # 此处的 n 是形参 n
    lst2 = [n**2 for n in range(10)] # 此处的 n 非形参 n
    print(lst1)
    print(lst2)
    print(n)                         # 此处的 n 是形参 n
```