



马哥教育

IT 人的高薪职业学院

Python内置数据结构

讲师：Wayne

从业十余载，漫漫求知路

分类

□ 数值型

- int、float、complex、bool

□ 序列对象

- 字符串 str

- 列表 list

- tuple

□ 键值对

- 集合set

- 字典dict



数值型

□ 数值型

- int、float、complex、bool都是class，1、5.0、2+3j都是对象即实例
- int：python3的int就是长整型，且没有大小限制，受限于内存区域的大小
- float：有整数部分和小数部分组成。支持十进制和科学计数法表示。只有双精度型。
- complex：有实数和虚数部分组成，实数和虚数部分都是浮点数，3+4.2j
- bool：int的子类，仅有2个实例True、False对应1和0，可以和整数直接运算

□ 类型转换 (built-in)

- int(x) 返回一个整数
- float(x) 返回一个浮点数
- complex(x)、complex(x,y) 返回一个复数
- bool(x) 返回布尔值，前面讲过False等价的对象



数字的处理函数

- round() , 四舍五入 ?
- math模块、floor()地板、天花板ceil()
- int() 、 //
- 举例 :

```
import math
```

```
print(math.floor(2.5), math.floor(-2.5))
```

```
print(math.ceil(2.5), math.ceil(-2.5))
```

以下打印什么结果 ? 说明什么

```
print(int(-3.6), int(-2.5), int(-1.4))
```

```
print(int(3.6), int(2.5), int(1.4))
```

```
print(7//2, 7//-2, -7//2, -(7//2))
```

```
print(2//3, -2//3, -1//3)
```

```
print(round(2.5), round(2.5001), round(2.6))
```

```
print(round(3.5), round(3.5001), round(3.6), round(3.3))
```

```
print(round(-2.5), round(-2.5001), round(-2.6))
```

```
print(round(-3.5), round(-3.5001), round(-3.6), round(-3.3))
```



马哥教育
IT人的高薪职业学院

数字的处理函数

- `round()` , 四舍六入五取偶
- `floor()`向下取整、`ceil()`向上取整
- `int()` 取整数部分
- `//` 整除且向下取整



数字的处理函数

- min()
- max()
- pow(x,y) 等于 $x^{**}y$
- math.sqrt()

- 进制函数，返回值是字符串
 - bin()
 - oct()
 - hex()

- math.pi π
- math.e 自然常数



类型判断

- `type(obj)` , 返回类型 , 而不是字符串
- `isinstance(obj, class_or_tuple)` , 返回布尔值
- 举例 :

`type(a)`

`type('abc')`

`type(123)`

`isinstance(6, str)`

`isinstance(6, (str, bool, int))`

`type(1 + True)`

`type(1 + True + 2.0)` # 是什么 ? 隐式转换



列表list

- 一个队列，一个排列整齐的队伍
- 列表内的个体称作元素，由若干元素组成列表
- 元素可以是任意对象（数字、字符串、对象、列表等）
- 列表内元素有顺序，可以使用索引
- 线性的数据结构
- 使用 [] 表示

□ 列表是**可变的**

□ 列表list、链表、queue、stack的差异



列表list定义 初始化

- ❑ `list()` -> new empty list
- ❑ `list(iterable)` -> new list initialized from iterable's items
- ❑ 列表不能一开始就定义大小

```
lst = list()
```

```
lst = []
```

```
lst = [2, 6, 9, 'ab']
```

```
lst = list(range(5))
```



列表索引访问

- 索引，也叫下标
- 正索引：从左至右，从0开始，为列表中每一个元素编号
- 负索引：从右至左，从-1开始
- 正负索引不可以超界，否则引发异常IndexError
- 为了理解方便，可以认为列表是从左至右排列的，左边是头部，右边是尾部，左边是下界，右边是上界
- 列表通过索引访问
 - `list[index]`，index就是索引，使用中括号访问

列表查询

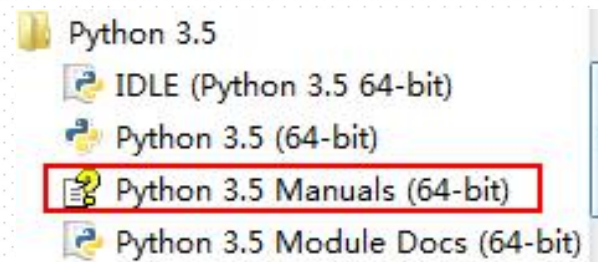
- ❑ `index(value,[start,[stop]])`
 - ❑ 通过值value，从指定区间查找列表内的元素是否匹配
 - ❑ 匹配第一个就立即返回索引
 - ❑ 匹配不到，抛出异常ValueError
- ❑ `count(value)`
 - ❑ 返回列表中匹配value的次数
- ❑ 时间复杂度
 - ❑ `index`和`count`方法都是 $O(n)$
 - ❑ 随着列表数据规模的增大，而效率下降
- ❑ 如何返回列表元素的个数？如何遍历？如何设计高效？
 - ❑ `len()`



如何查帮助

□ 官方帮助文档

□ 搜索关键字



□ IPython中

□ help(keyword)

□ keyword可以是变量、对象、类名、函数名、方法名

```
In [15]: help(list.count)
Help on method_descriptor:

count(...)
    L.count(value) -> integer -- return number of occurrences of value
```



马哥教育
IT人的高薪职业学院

列表元素修改

- 索引访问修改
 - `list[index] = value`
 - 索引不要超界



列表增加、插入元素

□ `append(object)` -> None

- 列表尾部追加元素，返回None
- 返回None就意味着没有新的列表产生，就地修改
- 时间复杂度是 $O(1)$

□ `insert(index, object)` -> None

- 在指定的索引index处插入元素object
- 返回None就意味着没有新的列表产生，就地修改
- 时间复杂度是 $O(n)$
- 索引能超上下界吗？
 - 超越上界，尾部追加
 - 超越下界，头部追加

列表增加、插入元素

- `extend(iteratable) -> None`
 - 将可迭代对象的元素追加进来，返回None
 - 就地修改
- `+ -> list`
 - 连接操作，将两个列表连接起来
 - 产生新的列表，原列表不变
 - 本质上调用的是`__add__()`方法
- `* -> list`
 - 重复操作，将本列表元素重复n次，返回新的列表

列表 * 重复的坑

□ * -> list

□ 重复操作，将本列表元素重复n次，返回新的列表

```
x = [[1,2,3]]*3
```

```
print(x)
```

```
x[0][1] = 20
```

```
print(x)
```

```
y = [1]*5
```

```
y[0] = 6
```

```
y[1] = 7
```

```
print(y)
```

上面代码运行结果是什么？为什么？



列表删除元素

- ❑ `remove(value)` -> `None`
 - ❑ 从左至右查找第一个匹配value的值，移除该元素，返回None
 - ❑ 就地修改
 - ❑ 效率？
- ❑ `pop([index])` -> `item`
 - ❑ 不指定索引index，就从列表尾部弹出一个元素
 - ❑ 指定索引index，就从索引处弹出一个元素，索引超界抛出IndexError错误
 - ❑ 效率？指定索引的时间复杂度？不指定索引呢？
- ❑ `clear()` -> `None`
 - ❑ 清除列表所有元素，剩下一个空列表

列表其它操作

- reverse() -> None
 - 将列表元素**反转**，返回None
 - 就地修改
- sort(key=None, reverse=False) -> None
 - 对列表元素进行**排序**，就地修改，默认升序
 - reverse为True，反转，降序
 - key一个函数，指定key如何排序
 - lst.sort(key=functionname)
- in
 - [3,4] in [1, 2, [3,4]]
 - for x in [1,2,3,4]

列表复制

□ 先看一段代码

```
lst0 = list(range(4))
```

```
lst2 = list(range(4))
```

```
print(lst0==lst2)
```

```
lst1 = lst0
```

```
lst1[2] = 10
```

```
print(lst0)
```

lst0==lst2相等吗？为什么？lst0里面存的是什么？

请问lst0的索引为2的元素的值是什么？

请问lst1 = lst0这个过程中有没有复制过程？



列表复制

□ `copy()` -> List

□ shadow copy 返回一个新的列表

```
lst0 = list(range(4))
```

```
lst5 = lst0.copy()
```

```
print(lst5 == lst0)
```

```
lst5[2] = 10
```

```
print(lst5 == lst0)
```

lst0和lst5一样吗？

□ 对比左右程序的差别

```
lst0 = [1, [2, 3, 4], 5]
```

```
lst5 = lst0.copy()
```

```
lst5 == lst0
```

```
lst5[2] = 10
```

```
lst5 == lst0
```

```
lst5[2] = 5
```

```
lst5[1][1] = 20
```

```
lst5 == lst0
```



列表复制

❑ shadow copy

❑ 影子拷贝，也叫浅拷贝，遇到引用类型，只是复制了一个引用而已

❑ 深拷贝

❑ copy模块提供了deepcopy

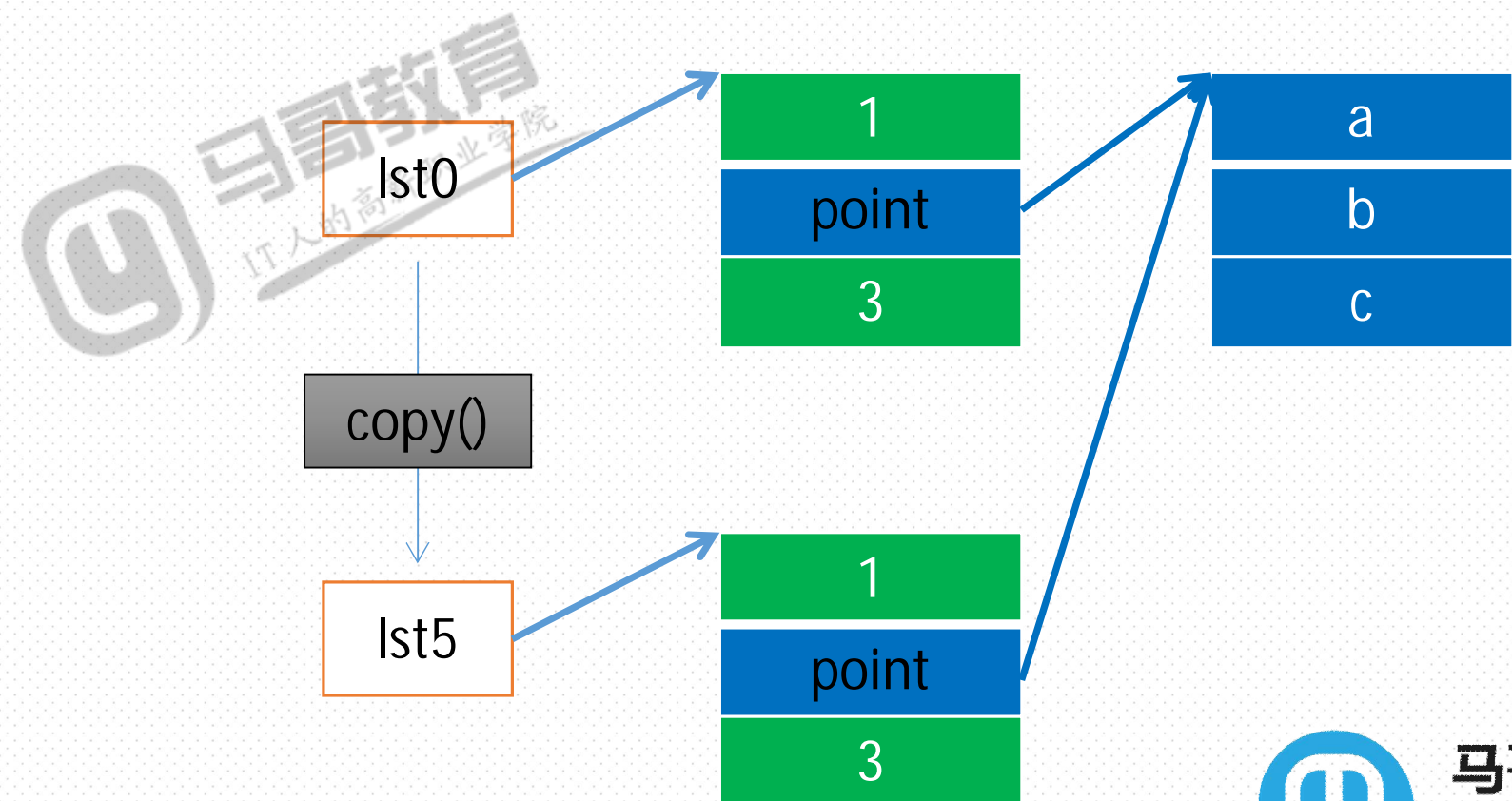
```
import copy
```

```
lst0 = [1, [2, 3, 4], 5]
```

```
lst5 = copy.deepcopy(lst0)
```

```
lst5[1][1] = 20
```

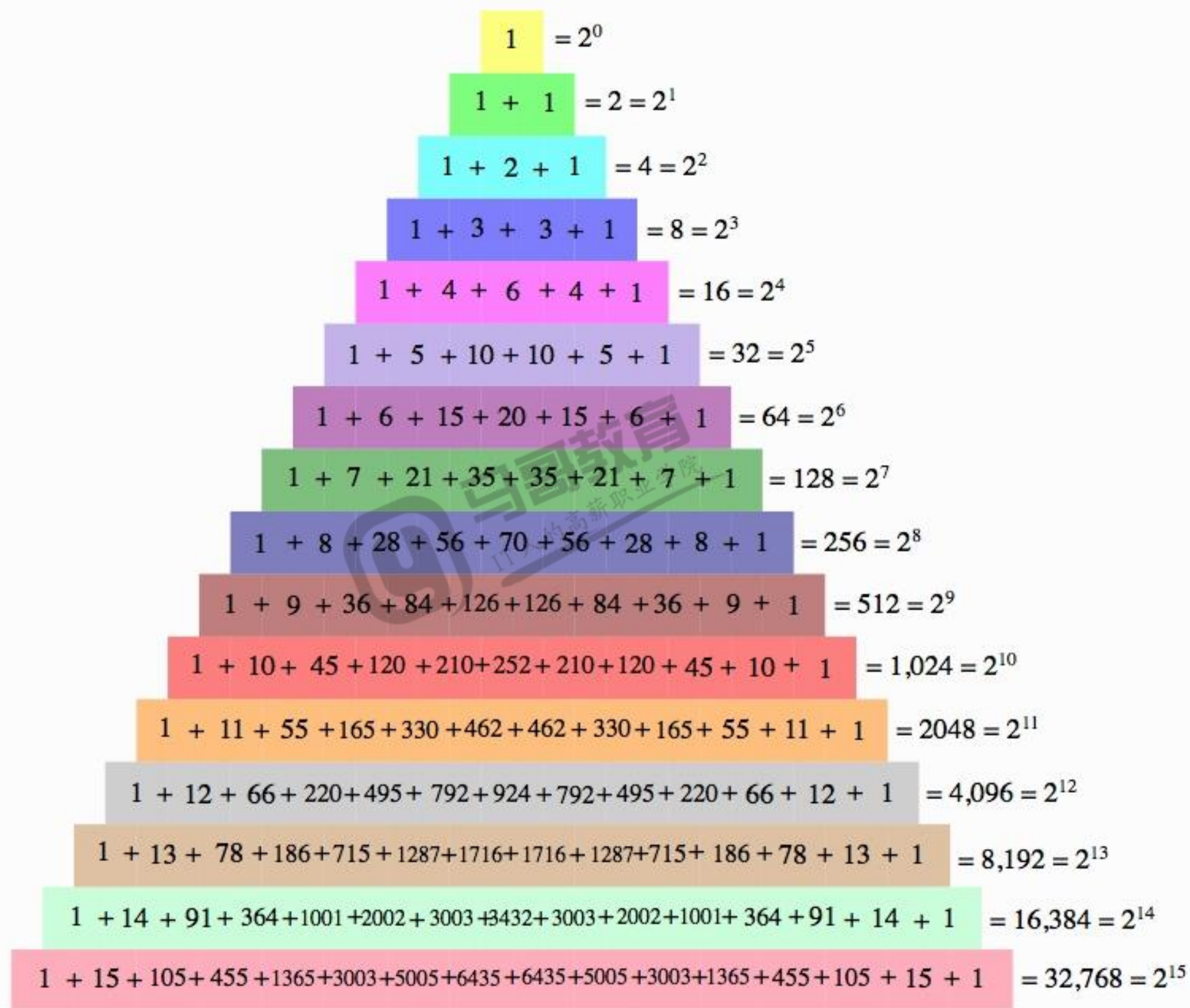
```
lst5 == lst0
```



随机数

- ❑ random模块
- ❑ randint(a, b) 返回[a, b]之间的整数
- ❑ choice(seq) 从非空序列的元素中随机挑选一个元素，比如random.choice(range(10))，从0到9中随机挑选一个整数。random.choice([1,3,5,7])
- ❑ randrange ([start,] stop [,step]) 从指定范围内，按指定基数递增的集合中获取一个随机数，基数缺省值为1。random.randrange(1,7,2)
- ❑ random.shuffle(list) ->None 就地打乱列表元素
- ❑ sample(population, k) 从样本空间或总体（序列或者集合类型）中随机取出k个不同的元素，返回一个新的列表
 - ❑ random.sample(['a', 'b', 'c', 'd'], 2)
 - ❑ random.sample(['a', 'a'], 2) 会返回什么结果

杨辉三角



马哥教育

IT人的高薪职业学院

谢谢

咨询热线 400-080-6560



马哥教育
IT人的高薪职业学院