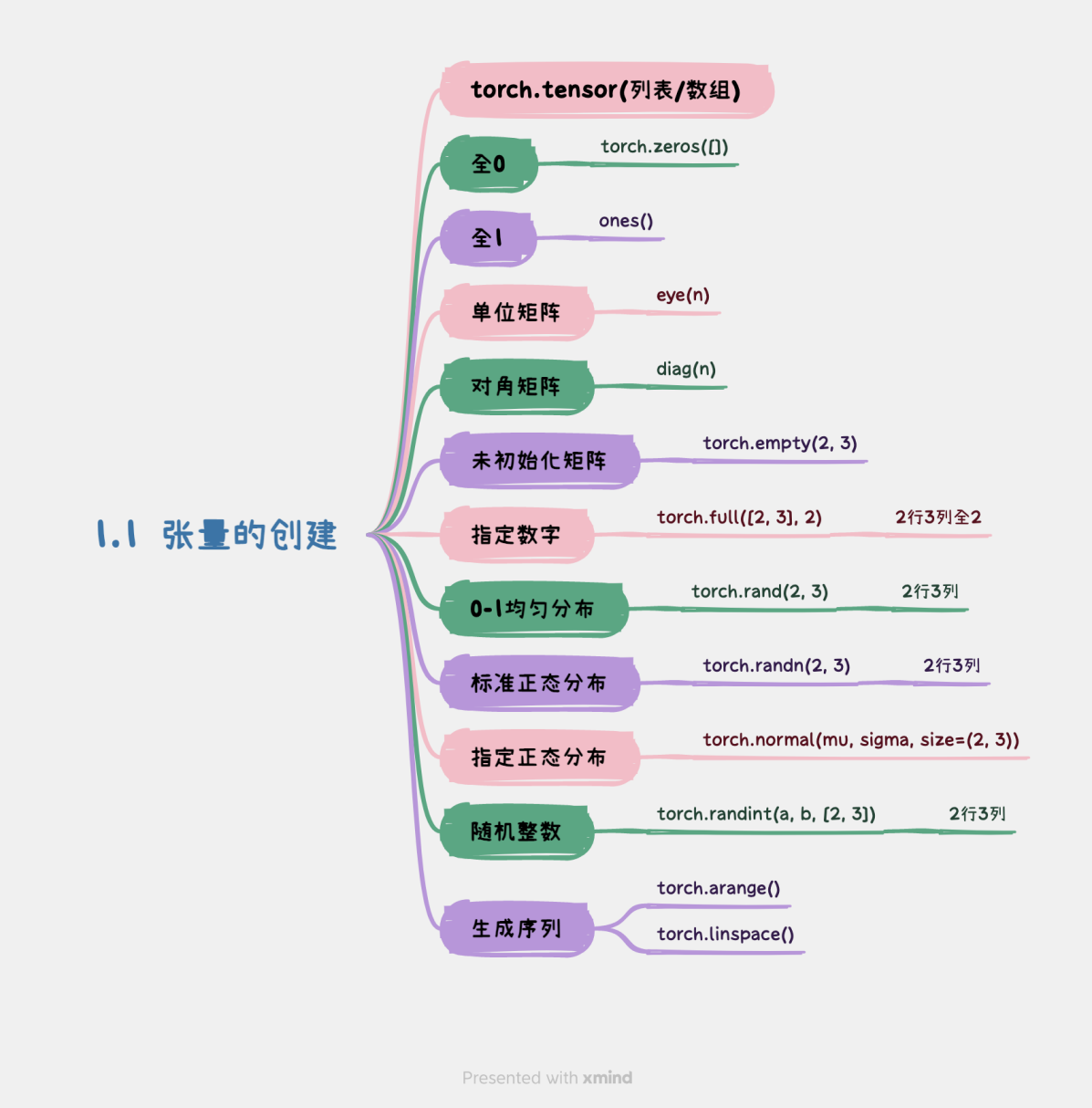
第一章 张量的创建和常用方法

1.1 张量的创建



1.2 张量的类型

调用`t.dtype`查看张量类型。

与数组`np.array`的区别：

1. 整数型：数组默认创建int32（整型）类型；张量则默认创建int64（长整型）类型；
2. 浮点型：数组默认float64（双精度浮点型）；张量默认是float32（单精度浮点型）。

数据类型	dtype
32bit浮点数	torch.float32或torch.float
64bit浮点数	torch.float64或torch.double
16bit浮点数	torch.float16或torch.half
8bit无符号整数	torch.unit8
8bit有符号整数	torch.int8
16bit有符号整数	torch.int16或torch.short
16bit有符号整数	torch.int16或torch.short
32bit有符号整数	torch.int32或torch.int
64bit有符号整数	torch.int64或torch.long
布尔型	torch.bool
复数型	torch.complex64

1.3 张量的类型转换

1. 转换为浮点型

   `t.float()`转换为默认浮点型（32位）

   `t.double()`转换为双精度浮点型（64位）

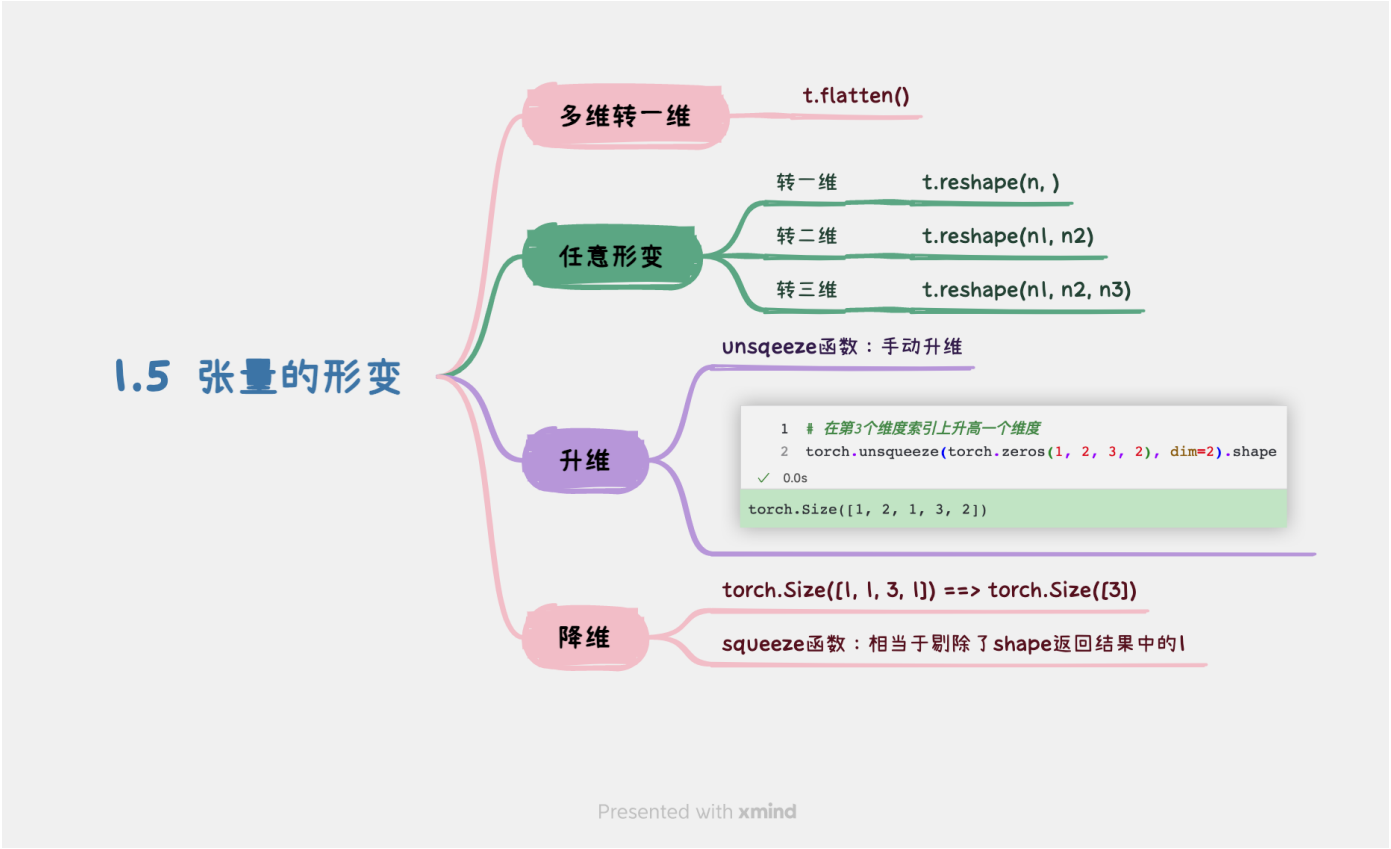
2. 转换为整数型

   `t.short()`转换为16位整数

1.4 查看张量的结构

1. 维度：`t.ndim`
2. 形状：`t.size()`、`t.shape`
3. 元素个数：`t.numel()`

1.5 张量的形变



1.6 张量和其他相关类型之间的转化

1. **转换为数组**：`t.numpy()`、`np.array(t)`
2. **转换为列表**：`t.tolist()`、`list(t)`（列表里是0维张量，不是标量）
3. **转换为标量**：`t.item()`

1.7 张量的拷贝

1. **浅拷贝**：`t2 = t1`
2. **深拷贝**：`t2 = t1.clone()`
3. **视图**：`t2=t1.view(*shape)`

   “视图”的作用就是**节省空间**，和原张量对象共享一块数据存储空间。改变tensor中的元素，也会影响new_tensor。

   可以使用 `view` 方法改变 tensor 的形状，也就是**改变 tensor 的维度，但不改变 tensor 中元素的个数**。

   `view` 方法的使用方式如下：

```
new_tensor = tensor
```

   其中，`tensor` 是需要改变形状的 tensor，`shape` 是一个元组，表示新 tensor 的形状。

   例如，将一个形状为 `(2, 3, 4)` 的 tensor 转换成形状为 `(3, 8)` 的 tensor，可以使用以下代码：

```python
# 创建一个形状为 (2, 3, 4) 的 tensor
x = torch.randn(2, 3, 4)

# 将 x 转换成形状为 (3, 8) 的 tensor
y = x.view(3, 8)

print("x 的形状为：", x.shape)
print("y 的形状为：", y.shape)
```

这段代码的输出结果为：

```
x 的形状为：  torch.Size([2, 3, 4])
y 的形状为：  torch.Size([3, 8])
```

需要注意的是，`view` 方法返回的是一个新的 tensor，原 tensor 的形状并没有改变。另外，`view` 方法中的参数需要满足一定的条件，具体可以参考 PyTorch 官方文档。