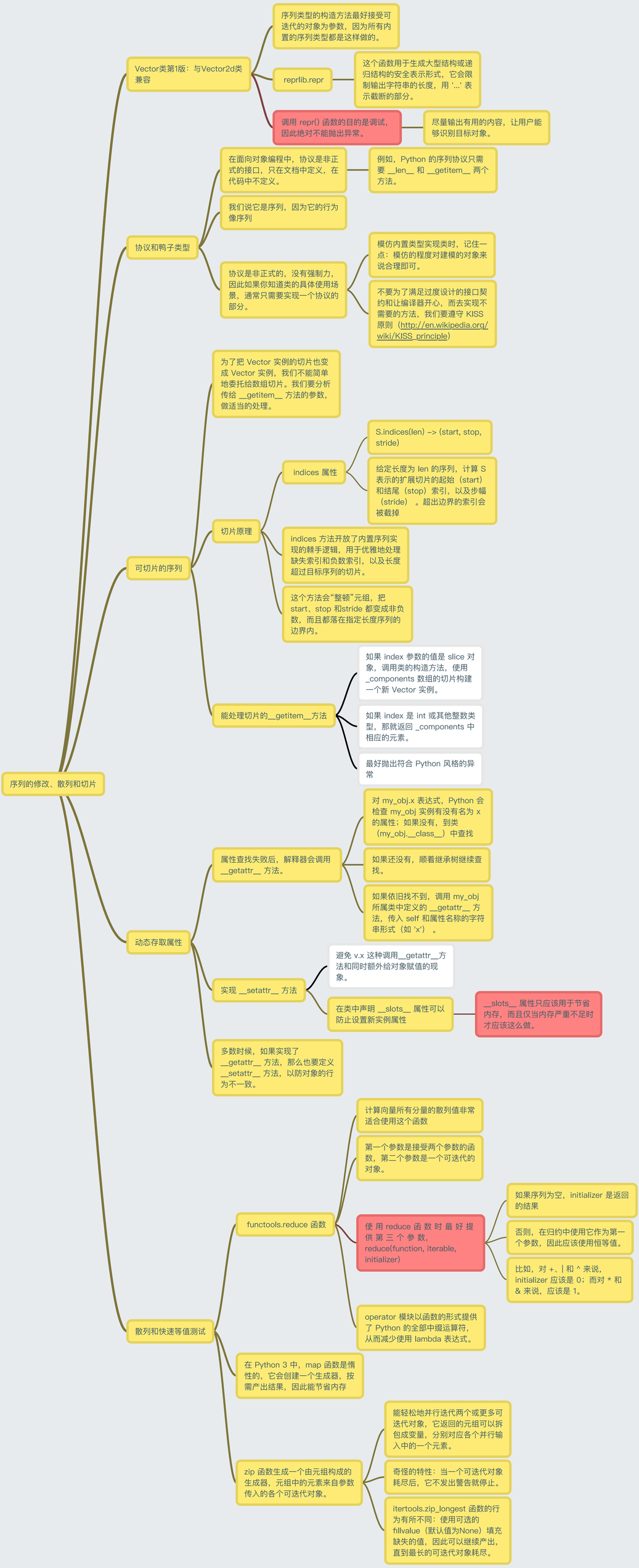
序列的修改、散列和切片
Vector类第1版：与Vector2d类兼容
序列类型的构造方法最好接受可迭代的对象为参数，因为所有内置的序列类型都是这样做的。
reprlib.repr
这个函数用于生成大型结构或递归结构的安全表示形式，它会限制输出字符串的长度，用 '...' 表示截断的部分。
调用 repr() 函数的目的是调试，因此绝对不能抛出异常。
尽量输出有用的内容，让用户能够识别目标对象。
协议和鸭子类型
在面向对象编程中，协议是非正式的接口，只在文档中定义，在代码中不定义。
例如，Python 的序列协议只需要 __len__ 和 __getitem__ 两个方法。
我们说它是序列，因为它的行为像序列
协议是非正式的，没有强制力，因此如果你知道类的具体使用场景，通常只需要实现一个协议的部分。
模仿内置类型实现类时，记住一点：模仿的程度对建模的对象来说合理即可。
不要为了满足过度设计的接口契约和让编译器开心，而去实现不需要的方法，我们要遵守 KISS 原则（http://en.wikipedia.org/wiki/KISS_principle）
可切片的序列
为了把 Vector 实例的切片也变成 Vector 实例，我们不能简单地委托给数组切片。我们要分析传给 __getitem__ 方法的参数，做适当的处理。
切片原理
indices 属性
S.indices(len) -> (start, stop, stride)
给定长度为 len 的序列，计算 S 表示的扩展切片的起始（start）和结尾（stop）索引，以及步幅（stride）。超出边界的索引会被截掉
indices 方法开放了内置序列实现的棘手逻辑，用于优雅地处理缺失索引和负数索引，以及长度超过目标序列的切片。
这个方法会“整顿”元组，把 start、stop 和stride 都变成非负数，而且都落在指定长度序列的边界内。
能处理切片的__getitem__方法
如果 index 参数的值是 slice 对象，调用类的构造方法，使用 _components 数组的切片构建一个新 Vector 实例。
如果 index 是 int 或其他整数类型，那就返回 _components 中相应的元素。
最好抛出符合 Python 风格的异常
动态存取属性
属性查找失败后，解释器会调用 __getattr__ 方法。
对 my_obj.x 表达式，Python 会检查 my_obj 实例有没有名为 x 的属性；如果没有，到类（my_obj.__class__）中查找
如果还没有，顺着继承树继续查找。
如果依旧找不到，调用 my_obj 所属类中定义的 __getattr__ 方法，传入 self 和属性名称的字符串形式（如 'x'）。
实现 __setattr__ 方法
避免 v.x 这种调用__getattr__方法和同时额外给对象赋值的现象。
在类中声明 __slots__ 属性可以防止设置新实例属性
__slots__ 属性只应该用于节省内存，而且仅当内存严重不足时才应该这么做。
多数时候，如果实现了 __getattr__ 方法，那么也要定义 __setattr__ 方法，以防对象的行为不一致。
散列和快速等值测试
functools.reduce 函数
计算向量所有分量的散列值非常适合使用这个函数
第一个参数是接受两个参数的函数，第二个参数是一个可迭代的对象。
使用 reduce 函数时最好提供第三个参数，reduce(function, iterable, initializer)
如果序列为空，initializer 是返回的结果
否则，在归约中使用它作为第一个参数，因此应该使用恒等值。
比如，对 +、| 和 ^ 来说，initializer 应该是 0；而对 * 和 & 来说，应该是 1。
operator 模块以函数的形式提供了 Python 的全部中缀运算符，从而减少使用 lambda 表达式。
在 Python 3 中，map 函数是惰性的，它会创建一个生成器，按需产出结果，因此能节省内存
zip 函数生成一个由元组构成的生成器，元组中的元素来自参数传入的各个可迭代对象。
能轻松地并行迭代两个或更多可迭代对象，它返回的元组可以拆包成变量，分别对应各个并行输入中的一个元素。
奇怪的特性：当一个可迭代对象耗尽后，它不发出警告就停止。
itertools.zip_longest 函数的行为有所不同：使用可选的 fillvalue（默认值为None）填充缺失的值，因此可以继续产出，直到最长的可迭代对象耗尽。