陈久宁(00:00:00): 呃，在我解释REF是什么之前，呃大家关于。这样的一个这样两种行为能够？你理解吗。或者说关于这样两种行为。啊，有没有问题？不能不能用那种全局变量的资料，里面没有全局变量的感觉，呃，有全局变量。在这里。胳膊弯它就是一个。模块内部的全局变量。就是我们。当我们去谈全局变量的时候，我们其实是在一个模块的概念下谈。就是，比如说我们去定义global。Global x. 等于一的时候，我们其实是在谈模块下的X。就是你在你在？啊，下面你是拿不到拿不到X？如果在这个里面定义一个。啊，我们举个例子my my。萝卜Y等于二？啊，在这在这种情况下，你明显拿不到。喂，或者说你这边去定？啊你你无论怎么定义，你都必须要以一个。都必须要以一个啊，完整的方式。去获取到它？就是它的这个全局global这个概念只是相对于这个模块内部。呃，所有函数或者说。所有啊，其他的作用。我们来举个例子。刚才。呃，全关于全局变量的话。我想想要不待会，待会待会我再继续讲吧，我们先先还是等一下，就是刚才那两个预编译的。啊。行为大家有没有问题?

陈久宁(00:02:05): 当时第二次执行这个。等于AN的那个步骤，它是没有在这个。里面直行车，但是他好像。要去加载的时候还是有一个东西的过程。

陈久宁(00:02:35): 上面的那个定义上没有在？啊，这个问题很好，有没有同学能够理解？呃他呃我我来复述一下他这里呃刚才刚才呃这叫什么名字？人人品。啊，人品同学提的一个问题是说我。我我那个模块模块定义模块中定义的内容只会执行一遍，对吧？那么，如何执行一遍的话，我的?

陈久宁(00:03:11): 第一次是一个NAN的状态。我后面第11大模块加载了一次之后我的Y按理来说就应该被第一次模块加载的时候。啊所创建的那个？啊随机数给负值，所以举个例子，我这一次。比如说这一次我我这边创建的是0.26。那么我下一次为什么他不是？0.26这个我我第一次。第一次的这个Y的打印为什么不是0.26是NAN？这是他刚才的问题。我们可以，我们可以验证一下。它这里很明显不是0.26。那么为什么会发生在这个？这是一个很好的问题。

陈久宁(00:04:12): 呃，有没有。有想法的同学。呃，和C无关。哦，比如说我把它移除，我们来重新验证一下。这是第一次是模块的预加载。然后。第二次第二次这个样子。因为他的模块外面他的优先级相交率更高。呃，你这个该怎么理解？要不是他可能在模块外面，他模块内部对他进行了修改，然后是一个暂时性的操作。嗯。说对了一半我我或者说我没理解到你另外一半的意思是什么？你刚才说的是呃隐匿在函数里面，对它的修改是一个临时性的操作。啊，这一点是对的。就是说。

陈久宁(00:05:06): 呃，我们函数与编译的环节。他他这个背后其实就意味着函数与编译环节究竟发生了什么？呃，预编译环节它其实发生的是从接代码。从一个从一段点击量的代码吗？呃，他会去生成一段预编译。结果。这是一个二进制？的数据。这在这个二进制数据里面，我我的。它是一个NAN的一个状态，对吧？他和我什么时候被？被使用。啊，是没有任何关系的，因为我在我。我从我在代码块里面就定义好了。第一次执行就是以AN的状态区域。所以预编译的结果里面一定是NAN。那么。这是这是第一件，第一个事实，第二个事实是我的。所有的。啊，受家的。啊，所有的。呃rapple。大的操作。都不会影响于别人。就都不会影响啊!

陈久宁(00:06:22): 模块的语言对，大概是。这个意思就是说我说的不会影响模块与编译是指。不会修改。旧的内容。就是说我们在旧的内容里面。他是NAN。呃，我第一次修改了他，但是他不会存下来，他只会在在当前。我这个就让一下有效，在我下一次打开最大的时候他又会从。预编译的结构开始。开始做。所以你就会发现，第二次打开，九点的时候他还是从NAN开始。啊，这个能理解吗？就我我这个表述不是特别的准确，但是呃这样表述可能会更容易理解一些。哎，陈老师就昨天教我的那个鬼外子，可以认为他修改了一天一结果。啊对。就是热，加载型就是在。在对预编译结果做修改。所以我这边说所有apple下的操作都不会影响模块的预编译。呃，他他只是部分情况下，或者说我们从直觉上理解他是这个样子。嗯，关于关于这个模块的预编。还有没有其他的新?

陈久宁(00:07:48): 呃，盈利函数它不是预编译。影影音的函数我们说的它是只会在含模块加载的时候被调用。他在预编译的时候是完全不会被调用。嗯，你可以，你可以认为的是。呃，模块的初始化分为两个阶段。第一个阶段的初始化是预编译环节，第二个阶段的初始化是模块加载的环节。然后预编译环节的初始化，它只会发生一次模块加载的环节初始化。每次打开新的酒驾都会发生。对，这这一点是和其他语言。呃，很不一样的。这个预编译是除了？外的所有的地方都会被执行到。啊。对对。就是。呃，你说的就是呃预编译就是一个模块，内部的所有东西都会参与预编译。包括盈利函数也会参与预编译，但是盈利数不会被调用。隐匿的函数的定义会被预编译进行。但是因为它函数的调用不会。就是模块的这个函数的定义被预编译并不意味着这个函数会不会执行。啊，这一点能理解吗？都是这样，没有函数，它没有特点，它唯一的特殊点就在于它会在的时候被调用。这是他唯一的特殊点。其他情况下，它和正常的函数没有任何差异。

陈久宁(00:09:45): 嗯，在开始下面的之前。再看看有没有问题。

陈久宁(00:10:03): 嗯，那好，我是讲我们是讲下面，那刚才刚才是你有一个问题，我临时没有讲对吧，是啥问题还是全局变量？因为刚才不是说。之前你讲的时候讲的这个？然后这个就是小。呃，就比如说在在一个函数里面用不到。用不到函数以外的变量。如果这个不要过来就行。就想喝全球变量？在就那里面，我们有所谓的作用域的概念。这个东西。对于我来说也是一个。不是那么熟悉的一个点。啊，当然我可以尝试的去讲一下。就是呃，刚才。我们来，我们来做一个假设，当我去定义一个。格格后，他声明了一个所谓的全局变量的概念。当我去定义一个global Y等于。啊，一的时候。我在。在一个函数，比如说F内部。或者说。X加Y。嗯，他能说正常出息吗？可以正常执行。得先上一下在这个。呃，我们猜测一下它就是11种一种可能性是它的返回值的结果是二。另外一种可能性是他在这里会报错。大家认为会是哪种可能性？

陈久宁(00:11:46): 呃，认为会报错的几个手，我看一下。呃，它这里是返回值12。在函数内部里面的啊，不需要在函数内部。对啊，这点能够理解吗？我能够理解对吧？啊，那那我那可能就要来一个新的问题了。不够直接写Y等于一？然后再去病也去玩。在这种情况下。他会报错还是会？返回啊。

陈久宁(00:12:42): 报错的几个词？

陈久宁(00:12:47): 大概一个一个同学，两个同学。啊，他这个也是返回啊？那么。那么我我们再来一个新的问题。Y等于一方向？就返回什么已经不重要。如果我。调用一遍之后。如果调用一遍F二之后。请问这里的Y？一还是二。就我一开始创建Y等于。然后在这里。啊YSE对吧。那我怎么样去让它能够修改这个Y？啊，在这里其实就是Y，比如说Y等于。好，我还等于function。乖的也是。啊，在这种情况下。啊不还会被修改？啊，能够理解吗？对，就是当你去声明全局或者说局部的时候，你其实都是去。等于是要是是告诉你的内部啊，你要去拿到一个全局的Y还是一个局部的。你在外面是声明？就是。当你是写Y等于就直接在最外面写Y等于一的时候，而你直接在最外面写global Y等于一啊？对，一模一样的。如果我没有理解错的话是一模一样的。对它有差别的点在于你在函数内部。啊你是。说这是一个全局的还是一个局部的？然后有一个行为。呃，我我们班。我把这一段。呃，有一个行为我自己有时候都搞不太清楚。啊，我可以尝试的演示一下。比如说X等于一。我们写一个for循环。I一大师。

陈久宁(00:15:44): 啊，请问这个时候的X？会被修改吗？就是他是一还是？一加二加。他会不会修改？然后还有一种情况是。X等于一。我爱印尼大使。对，这这这里也是我现在。没有完全搞明白的一个点。啊，不是，它是一个作用力的问题。嗯。这个软重要。

陈久宁(00:16:47): 是什么问题？

王钊(00:16:48): 陈博士是直接运行那个GL文件会出现这个问题，就是你在那个？

陈久宁(00:16:49): 直接运行jar文件会出现问题是吗？

王钊(00:16:55): 对你，你把那个脚本直接运行，它好像就会出现这个问题。

陈久宁(00:16:57): 哦，好像是好像是好像是好像是对。

王钊(00:17:02): 对，就是这个就一直不是很清楚的地方。

陈久宁(00:17:02): 我来看一看啊！X等于一，然后for I in一到十X加等。

陈久宁(00:17:24): 啊，我们在这里。直接运行脚本我看一下。啊对，你可以看见这里。他甚至是直接报错了。对，这是我们，我们从。这个脚本。如果我们直接点运行，它是报错的。这里能够。这是一个非常。在家里面非常特殊的一个现象。他说的是。他说的是外面外面的，这个是一个全局变量。然后里面的是一个？就是他的意思是说呃。不仅仅是函数的定义，循环内部，它也是一个新的作用。所以能够理解这里大概为什么会报错吗？它这个就很像是说你去定义一个函数。啊函数F。然后for爱印一到十？他他汇报说是一个原因。就是呃，循环内部它本身也是一个新的作用，我们把它叫做。叫做软中。然后。函数啊模。我们把它叫做硬。呃，实际上软作用于还有。嗯嗯。还有好几种就是叫。啊循环循环！循环然后if else判断。啊，这些都属于软作用。转作用于一个非常非常特殊的点，刚才大家已经看到了，就是。

陈久宁(00:19:56): 啊，这一段代码如果我是在。下面。通过IP L直接复制粘贴的话，它是能够正常运行的。但是如果我在，我直接点这个。运行按钮它是会报错的。这是专用于一个非常非常特殊的一个点。呃，在这个问题上，我我我甚至。不是很确定我这句话说的对不对，就是究竟哪些是。我们可以看一下官方文档。

陈久宁(00:20:36): 呃，官方文档专门有一页叫做就是。啊，作用我们可以看一下。呃，变量作用就是在官方文档里面有专门有一个叫变相作用语，它其实讲的就是这样一个现象。他说的是我们有一些是全局的，比如说模块。然后。结构体是一个软作用。然后for循环这些也是软作用。然后还有一些什么函数？然后。嗯，包括。一些其他的特殊的语法，它都属于一个硬作用与的概念。软作用力和硬作用的概念，它可能。是现在。啊，是否会被覆盖这个行为啊？覆盖的话它其实就是说。如果。X等于一，然后负I一到十，然后X加。啊，这里的破循环它是一个软。啊，它是软装的话，所以这里的X啊，其实是使用了。外面的。等于是。四岁真的。X被覆盖了？大概是这样理解。

陈久宁(00:22:29): 然后。然后。呃，硬作用于的话，我们可以举一个新的例子。比如说我们在一个函数里面。X等于一。我再再在这个函数里面再定一个新的函数。啊X等于二？然后。那么。就说调一遍。这种情况下我再返回X啊这个。深造啊，但是他的意思是说。这里的X等于二，它用的是？新的自己。而不是。能理解的吗？

陈久宁(00:23:23): 呃，能理解我是怎么造这个东西的吗？然后。如果说G里面的X等于二用的是自己的X，而不是F中的X的话，那么我调用一遍G之后并不会修改。修改X的。那么在这种情况下，他应该还是。如果我没搞错的话。他应该是这个样子。嗯嗯。和我想的还不一样。X等于二。嗯嗯。在这种情况下，他可能是要声明一个人叫local。对啊。是有问题的，那意思是不声明的话可以可以直接调到外面，对对对。反正玉印这样鱼。呃，这个东西我一直。搞得不算非常清楚。然后我一直是。尽量避免这个问题的出现。就比如说通过。通过改名。或者类似的行为。我们可以暂时把这个东西当成一个。当成一个线上。这这种情况下，F是等于一的。我我可以跟大家说说我是怎么理解这个问题的。就我理解这个问题的方式，就是告诉他不要去。不要去尝试跟他搏斗。这这是一种，这是一种结果，然后第二种结果是。啊，我如果不加。用的是。在这种情况下，它就会修改值。什么都不是我传的。啊对。然后。刚才其实是说软作用和硬作用，它存在这样两种概念，然后这两种概念很明显，我刚才演示完之后发现我自己都没有把它完全搞清楚。

陈久宁(00:26:15): 呃，然后我我自己日常的一个经验就是把这个东西封装成一个函数。封装成一个函数之后，比如说我把它叫做main函数。啊，在这种情况下，它是？比如说。嗯。就正常情况下，我可能我可能就通过这种方式去把。啊一个。一个一段代码，它在。他在那个正常？就是。如果不封装函数会报错，或者说变量找不到这种行为。啊给绕开我就会通过这种做做法，比如说一开始一开始我们是叫。是要这样这样的一个样子。我们直接是隐私心这个东西他会报错，遇到报错的话怎么办就是。把它分成一个。外面套一个函数，然后再去执行这个函数。执行这个函数的话，它就等于因为封了函数的话就意味着。我把软装鱼变成了硬装。大概。

陈久宁(00:27:28): 大概是这样一个点，就是说他如果如果说你对编程语言的一些细节非常感兴趣。啊，你可以去慢慢慢慢读这个东西。但是我读了大概两三遍。啊，没有认真的去。啊，还是没有认真的去见得到他的点在哪，我大概知道，但是。啊，没有搞明白它的规则。所以我。日常的编程的时候，我的策略就是绕开他，不要去跟他搏斗。然后遇到了？

陈久宁(00:28:02): 就是我们可以把它总结为一个，就是按照我自己的经验啊，就是不同的人可能会有不同的观点，我自己的经验是不要啊，和软。反应。作用于不懂。然后在在必要的时候。啊插入。Local和。来调整。作用与范围。啊，大概确实。然后。就大概是。占了一个策略。然后如果是按照这样的策略去做的话。很少很少很少会遇到。啊，像刚才那样的。啊，这样说的话大概呢，大家能能理解到吗？

陈久宁(00:29:11): 呃，我不是很。反正我自己也不是在这一块，也不是特别懂，然后我自己也不愿意花太多时间在这上面。啊，所以。呃，我我能给我给的建议就是这样一个建议。如果说有兴趣的话，我们可以。后面啊，一点一点的去钻研，就是他这里的行为非常。还是有蛮多挺诡异的地方。我们看看有没有新的问题。啊说。哦，对REF还没讲呃RE它其实就是一个。

陈久宁(00:29:56): 你就把它当成是一个长度为一的。长度唯一的下降！这个意见就好了。它背后其实是说我的任何一个。如果学过C或者C加加的话。任何一个东西他都可以。含地址或者传引用的方式。别使用对吧？那么REF它其实就是所谓的reference的概念。那么你的任何对象都可以被？都可以被引用。这个其实就是所谓的X，比如说叫它叫。啊，如果你引用了一个对象，你就可以通过。起到这个对象的方式去修改它。所以他就有一点像是如果我们从。啊，从模块的角度来来理解这个问题的话。嗯嗯。比如说一开始我们写了啊Y的一我们从。把它。一直往一直往回退。

陈久宁(00:31:26): 呃，在这种情况下，我已经声明了Y是一个常量对吧？所以如果我尝试去对Y做修改的话。我是没法修改。呃，没法修改的情况下。如果我又想要。不得到所谓的。就是像是，比如说我希望我还是一个随机值，但是我希望它是一个每次加载的时候声明的一个随机值。啊，怎么办呢？我可能。有一种方式是我直接在这里写。这是一种。啊，然后我是说Y等于。啊，我们把它叫做。

陈久宁(00:32:26): 在这种情况下，我们还需要写一个global。呃，它能够达到？

陈久宁(00:32:39): 拿到我们。每次加载的时候重新生成新的随机值的目的对吧？这个和前面是一模一样。但他会有一些问题在于。啊我我可以修改这件事情是对的。嗯。哦，他自己还不让修改。呃，实际上当然你可以通过筛，可以通过一些非常。非常奇怪的方式。去修改它。啊，他甚至这里也不行。对，他不行不行的话，如果说我们确实想要修改这个东西怎么办?

陈久宁(00:33:26): 啊，我们可能会举个例子，我们把它。封装成一个？比如说长度为一的。一个下降。我说这是一个下。呃，那么我们就说。它的第一个元素我就假设我只要它的第一个元素就行。那么我确实也达到目的，就是比如说我每次也是初始化了它，但是如果需要的话，我可以手动的去修改它，比如说我把手动把它修改成。这样的一个。啊，这样的一个行为。然后我每次只要用它是一，只要用它的第一个元素就行。然后长度唯一的这个向量。就是我们，我们。我们的目的就是要一个长度为一的向向量，或者说我们要的是一个。要的是一个对象，然后可被修改。我被外部修改。就是要一个这样的东西，然后我们说诶像降好像可以满，就是长度唯一的下降，可以满足这个需求。啊，然后我们发现诶rap也蛮多的需求。啊，然后就这样。就是rap本身它是一个CC的一个引用的概念，直接加引用的概念。

陈久宁(00:34:55): 然后rap和我们的问题本身。其实并没有一个非常强的关系，只是rap的一些特性。满足了我们对于。啊，可被外部修改，然后并且它是一个对象。这样的一个。需求所以我们才才使用的。所以比如说我在这里写一个。那么他也是。再说my conversation。对他，他其实只是为了为了满足啊，可被外部修改，或者说他是一个。啊，单个对象这样一个概念。能理解吗。你就可以就是从。Graph这个词本身它是指C加加的reference。或者说类比一些加加的。然后。但是在实际编程中你会发现。学大部分人用rap来。来达到。我们去创建一个可被修改的全局。全是这样的一个概念，然后这个东西你甚至还可以要求它是一个常量。就是这里可能会有一个。可能会有一个点在于。虽然声明了是常量，但是它的内容可被修改。这一点不知道大家能不能理解？这一点大家能理解吗？啊，不好理解是吧？哦，我们俩。我们来换一个新的例子，比如说我是声明X等于一的时候。啊，我肯定没法修改它。那如果。啊，这也是的一个。看啊比较。比较特殊的地方。我们来看一下。X等于八。

陈久宁(00:37:21): 呃，在这里它它是允许运行。说到这个我们就可以简单，我们就简单。他就俩他有一个。交互模式和非交互模式的概念。然后。像交互模式的话。我们通过IP L执行，一般就是它就是所谓的交互模式。在交互模式下。啊。可以被修改。然后在交互模式下软作用。啊，有特殊。对，我们可以，刚才就就是遇到了。遇到了一些类似于这样的。然后他非交互模式？非交互模式指的是说？比如说我们有这样一个脚本，这个叫做。我们如果是。Include. 叫叫成猪了？CG L的话。他这里直接报错了。我他他允许允许我们干这件事情。嗯嗯嗯。

陈久宁(00:39:08): 那看来是只能只能在。这种模块里面。还不允许。模块里面啊？它在定义的时候也允许我们干这件事情。嗯，只是他会爆出一个扔出一个警告。然后但是它在模块里面，如果我们。如果我们是YY等于二修改Y的话它。我可能要看一下康熙的文档。

陈久宁(00:40:33): 我看一下就那怎么打飞？它的交互模式和非交互模式指的是什么?

陈久宁(00:41:18): 如果它是一个函数定义的话。也许。也许是对的。这里不能走。FX等于一。他也允许。呃，看的这个行为跟我。之前对i的理解有一点不一样，它背后大概这几个版本发生了一些变化。呃，我可能不知道该怎么解释。就是说。呃，在。某些情况下，这里的他是会直接报错的，而不是扔出一个警告。至于是在什么情况下，我现在一下子找不到了。但大家知道有这么一个现象就可以了，就是说，如果你声明一个东西是cost，你就不要尝试着去修改它。我现在只能说到这一步，因为我确实没有把地址找出来。呃，然后看它有一个点在于我们声明一个东西是看的时候。我们其实只是做了一个变量绑定的生意。啊，这件事情怎么理解，就是说。如果我们声明他是。他是一个。呃，在这种情况下，我们不能说我们不不再能说X等于二了。因为如果说X等于二的时候。诶，他这个时候他又是。他说不允许他就直接报错了。我刚才花了好久没有找到，为什么在什么情况下呢？会报错？没找到，然后现在他就报错了。就总之，我们可以认为。啊，看着他。

陈久宁(00:43:37): 你一旦声明一个东西是靠死你，就不要尝试着去。啊，修改它。对，在这里的话就是说我，我们声明了它是抗的，它确实是不能修改了。但是。呃，我们就在某种意义上可以修改它。在什么意义上呢？我们我们是可以修改这个。这里面的狙？在这个意义上，我们是可以修改它。

陈久宁(00:44:04): 呃，不知道大家能不能理解这个现象?

陈久宁(00:44:27): 啊，大家能理解吗？呃，他背后寻说的是。变量的负值。啊，其实是一个。绑定。啊变量名的？然后。只是对绑定。并不会对。这个变量所指向的对象内部的那一段数据起作用。呃，如果我们要画。一个。一张图的话，我们可以这么理解。当我们是声明。

X等于一的时候。我们其实是在定义一个变量X啊！像样一的时候。我们其实定义了一个变量X，然后这个变量X。一段内存。当然我们可以是说X等于12。这段内存就是。长成这个样子。然后当我们说。当我们定义看的时候啊！就看着这个东西，它其实指的是我的这条线。不能改。就是说我不能。再把X指向。你上另外一个？2234的东西。就是。这个是。不能做的。241个叉。就是当我们去声明一个变量是一个常量的情况下，我们其实就是在说这样一件事情。能理解吗。对，但是我们又可以以某种方式说。这一段数据内的每个值都是可以变的，我们可以去写。啊，不就我就我们直接去写X的第一个词等于二。我们就可以把它。转换成。就可以转换成2234？啊，这样解释大家能够理解吧！

陈久宁(00:47:12): 对，所以啊。所以我们。结论其实就是就是刚才说的那两个结论就是啊，变量的副词。啊，其实是。不好的吧。然后看是影响。我好好的把你。

陈久宁(00:47:47): 嗯ok吗。

陈久宁(00:48:10): 就那确实有一个交互模式和非交互模式的概念，但是我又。不然找不到。怎么样给大家演示，所以大家可能可以留个心。然后以后遇到了可以去想想这个方向。看看有没有新的问题。如果有新的问题，我们就继续解释问题，如果没有新的问题我们就。

陈久宁(00:48:57): 那么我就认为大家。前面就今天讲到维持的东西都是能够理解的，对吧？呃，我们今天就讲到目前为止我们就讲了。大概核心的点在两个，一个是。然后。还有一个是。当然，周玉这个东西我的建议是绕开他。嗯嗯。然后我们回到我们的课程大纲里。呃，我们就等于是到目前为止我们就把。把这一部分。基础部分其实昨天就已经覆盖。然后今天的话我们？啊，就那常用的开发工具其实也是我们昨天说的。那两个一个是？Eggs, benchmark tools. 啊，它是提供一个性能的测试的。啊，其实还有一个叫做。它也是作为性能测试，不过它的使用我们可以往后放一放。就在周玉和什么模块预编译这个概念我们刚才介绍了。

陈久宁(00:50:21): 然后再往后的话我们去介绍一下就大的核心的编程风格。呃，当然在开始之前我还是先问一下大家有没有。一个油的问题。说那个国语？另一个孩子，然后在他发出之前的教育。呃，你可以再重复一遍吗？还是就是比如说我定义一个排除这个动作是能实现。传入两个数字，然后实现他们的相加。对，就是，然后然后我在他这个这个代码的这个函数前面。是a的嘛，然后在他前面就用这个a的。呃，我有点没太理解啊，你说的是？

陈久宁(00:51:25): 我还。比如说C等于FER，然后FE。Y等于是加Y吗？对，咱们刚开始是说到这个问题。哦，我以为我刚才已经讲完了呃。就是。这样一件事情会报错吗？你你是想说这个吗？呃，大家的。他写的时候他这个事不错。

陈久宁(00:52:25): 他这里。这个应该是一个。一半这边。像拍四个加四个加的话，他一般就是在最上面的时候，我们同文件的时候。呃，在就那里面我能说的只有。如果你直接去。第一，要用一个还不存在的函数。就是你直接去执行？啊，肯定是会报错的。这一点是我我我能确定一定以及肯定的。你必须要先定义一个函数。啊，比如说X解答完你才能够去调动它。这个是已经？这个我确定一个你看病的一个东西。然后我不确定的是这里。呃，会不会报错这一点？呃，从我的观点来说。呃，他可以报仇也可以不报仇。或者说啊，我我我的，我跟我的认为是他是一个预定义的行为，预定预定义的行为是指啊，我们不应该。依赖于。或者不报错。嗯嗯。我不知道这个有没有解释你的问题。意思是这种情况就要先定义这个是不是？

陈久宁(00:54:07): 就是所有的函数调用你肯定都要先先定义。但是在模块内，它它可能会是一些特殊情况。然后我的意思是在模块内？我们不应该依赖他的这个。当然，如果如果确定要用，那你就直接这样写。就反正能能work就行。呃，像这个他好像选择。这里是不是一个他是不是一个预定义的行为我？只是我个人的态度。但我，但我倾向于认为它是一个呃未定义的行为。嗯嗯。呃，再再看看有没有新的。一个有问题吧？

陈久宁(00:55:22): 啊，没有新的问题。那我就继续了。呃，我这边。呃，后面这个去了核心编程风格这一块的话，呃，我就我计划的是拿一个以前的PPT去。然后。我们内部应该是。有人。是看过的。

陈久宁(00:56:22): 呃，这个。这这个ppt的话是？啊，可以看到是去年年底做的。它不是一个非常非常新的东西，但是它。依然是一个我认为很有用的一个素材，然后也。可以。在很大程度上。啊，解释为什么我们要使用质量？以及我们该如何把就要写好。这样一点，所以这个的话呃，我可以我倾向于认为它是属于就俩核心编程风格的。第一派内容。就是说我们怎么样写出？呃，高性能的酒料代码，或者说怎么样写出好的酒料代码？

陈久宁(00:57:18): 呃，那么我就我就直接就念这个PPT了啊，就是在开始之前的话就是啊，昨天确实是同学问问了一个非常非常好的问题

。是我们为什么要使用？啊，他是一个非常特殊的一个。应用驱动的一个。领域领域驱动的一个语言就是说写高性能的数值计算啊，在九点之前我们做高性能数值计算，大家知道有什么样的方案吗？就是所谓的高性能代码，一般是使用什么样的语言或者什么样的方案去去做的。大家有了解过吗？

陈久宁(00:58:04): 呃，在J二之前，大家一般用C语言C加加或者fortune去做。这三门语这三个？是属于呃，非常非常古老，非常非常经典的解决方案。然后在。呃CC加加。呃之外呢，我们其实更多的时候大家会用matlab和python对吧？那么。为什么。我们一边说CC加加快，一边又在用matlab和。或者为什么？他没有我们想象的那么慢。就就是我们实际使用起来好像matlab和P。做一些数值计算啊，做一些建模仿真啊，也没有慢到不能接受嘛，对吧？之所以他没有那么慢的原因，是因为。那泰勒姆和python他背后是以某种方式调用了。别人用谁写好的代码？所以最慢的地方其实是用谁去跑的，所以。所以在这个意义上，matlab和python慢并不会影响。

陈久宁(00:59:11): 啊，整体的性能？啊，这这也是在九点之前大家最最最典型的一种啊！方式就是。啊，一个解释型的，一个慢的解释型语言。接一个快的。编译型语言。就是所谓的python加C或者matlab加C的方案。对，这是。这是一个点？

陈久宁(00:59:34): 然后matlab和python它有一个非常好的一个。特性在于，它是一个动态交互的一个。啊语言，所以我们就是可以写一行代码看一行结果，然后这种东西是对于建模仿真对于科学计算来说是非常重要的一个。核心核心特特性，它和我们去写互联网里面的业务代码是完全不一样的一个东西。对于我们需要动态交互的特性。

陈久宁(01:00:03): 然后我啊，我刚才也说了，拍场很慢，他本身是很慢。然后经典的方案是python加C加C。然后但是这里也有一个问题，就是呃C和C加加他。呃，放在今天来说，它依然是一个比较难写的东西，从就是它属于一个有经验的程序员会认为C和C加加特别好用。但是对于一个没有经验的CC加加程序员来说。呃，会觉得进入到这个领域会特别的困难。因为它有非常多。

陈久宁(01:00:44): 我那么。容易处理的一些，比如说指针的问题，比如说变量的定义。然后你会发现我明明只要五行，明明在python或者matlab下就是五行代码搞定的事情，我在C和C加压下可能会出现100行。这种这种情况都是很容易出现的。甚至有一些我们。我们觉得可以做的事情，谁和谁？佳佳说你不不，你不能这样做，然后他。他要你需要花费一些啊，非常大的啊！精力去绕开它的限制。这种都是C和CC加比较难写的一个原因。

陈久宁(01:01:25): 然后。经典的方案也是刚才说了加C的方案。呃，python加C的方案它它意味着我们要先先我们先写，先写，先把核心代码用C去写。然后我们要。把写好的C代码封装成一段。啊，可以被拍成调用的。然后再。在在正常的拍摄里面去写它。那么在这种情况下，我们去封装这一段C代码，本身又是一个额外的工作。呃，然后这里也是，就是并非所有做数值计算的人，他都是一个好的程序。在大多数情况下。我们都是做出demo就就够了。所以。

陈久宁(01:02:19): 然后然后再往后就是新的问题就出现了，如果如果我们确实是一个。啊，非常非常大规模的问题，比如说ai。我们现在大家也知道训练一个。举个例子，训练一个什么G PC这样的大模型？他可能光光训练一遍就是三个月，六个月。这样的一个问题，如果我不加呃，三个月六个月是不是已经加速了？很多很多很多的一个成果，如果不加速的话，他可能是三年。这样的一个时间周期，我们肯定是不能接受的。所以我们必须要把。呃，一些东西加速到极致。

陈久宁(01:03:00): 就是这些把把把性能加速到加速到极致，它不是一个。啊靠，时间可以。他不是他不是说啊，我们等新的英伟达的芯片出来，我们就可以免费加速的。一个事情，他必须要有一个软硬件的一个适配。所以我们必须要有一部分程序员去写出非常高性能的C加加代码。呃，这是目前在高性能计算领域。啊，一个非常。呃，非常怎么说呢？非常困难的一个问题，然后但也是现实。呃，在这种在这种意义上，我才会有九这样的一个编程模式出现，就是julia它的那个模式其实就是高性能。呃，然后动态交互，然后做数字计算。就是大部分需要高性能，就是大部分绕不开高性能计算的。领域一般都是跟数字计算有关。所以一个核心的设计。呃，都是围绕数字计算去做的，所以你可以发现就在很多地方用起来。可能会更像用户。啊，更更像matlab一些，但是它又跟matlab会不太一样，因为matlab它并不围绕高性能去设计，就俩是围绕高性能设计的。所以在jar中我们会看到非常多呃，为了高性能。呃，做出的一些。

陈久宁(01:04:29): 特殊的设计。然后这个PPT其实就是说啊，有哪些设计是我们必须要在写代码的时候去思考。那么在开启之前就是说。呃，刚才刚才是一个背景啊，就是说我们为什么会有这这门语言，然后现在就说的是绝了这门语言，它的上限是非常非常高的。呃，我们这边就等于是说呃随手写了。随手写了几段代码，呃就俩的牌子。然后呃的大概我们可以看到是130微秒。在在1.20乘1.20的这样的一个矩阵上。做数组的求和是135秒这样的一个时间。然后matlab如果我们自己去手写这样的一个求和运算的话，它是1.7毫秒。呃，拍场如果手

写求和运算的话是161毫秒。我们可以看到这里面是有一个十倍。1000倍的一个差异。这个能够看到吗？所以我们在这个意义上我们说呃matlab和python是非常非常慢的语言。嗯。

陈久宁(01:05:43): 看一下。

陈久宁(01:05:45): 呃，然后这里面这这这一页它对比的呢是？呃，也是呃，它对比的是matlab和python内置的求和。就刚才刚才一页是说我们手写？Matlab的求和和手写python的求和它是很慢的。呃，这一页说的是呃，即使对于呃内置的那些。求和运算来说，我们只有俩随便写一个就比。的C代码求和要快。就比pass的C代码求和要快。啊，这里可以发现。所以在这个意义上，我们是说的上限是非常高的。然后我们实际上在130微秒的前提下。用质量我们还可以再进一步的加速。就是比如说我们在八线程的情况下，就可以加速到30微秒。Matlab只能加速到75位秒。所以你可以发现。就是在一个。一个非常非常小的问题上。我们就居然就非常容易的去把。工程师花了非常多时间去优化的一个点。呃，再得到一个。两倍数的一个提升。对，所以这是最大上限非常高的一个表现，然后这个表现在各种问题上都是为真的。

陈久宁(01:07:12): 呃，C的话我去年的时候我没写，没没去花时间写写C所以在这里我就没没放C。这东西，但是呃，我们可以基本上认为ja的性能和C的性能是相当的。呃，主打性能和C的性能在大多数情况下。呃，或者说大多数人写出来的代码。都是相当的，但是实际上。啊，如果。呃，比如说让我去写的话，我可以找到非常多的场景就可以比C。快，比如说五倍十倍。呃，甚至我们之前在信号库写了一个代码。呃，巨大的代码比matlab的C代码要快1万倍。对，所以这个是？性能的一个体现。但大多数场景质量和谁的性能，你们可以认为是相当的。就互有优劣的这样的一个。

陈久宁(01:08:05): 那么这里刚才说的是jar的上线的啊，这边要说的是jar的下限。呃，第一到什么情况呢？就是说。呃，我们写好就好的，就是同样是一个数组求和。呃，这里给了两个版本，一个是好的版本，一个是。啊，不那么好的版本。呃，你们可以发现这里面。只有第一行有差异。然后它的性能？慢了十倍。

陈久宁(01:08:32): 这里为什么会出现这样一个情况？就是，这也是很多人不理解的一个点，就是初初学就俩不理解的点。然后我们后面会去解释。然后我们实际上。右边这个还不是最慢？我们可以再慢十倍，再慢再慢十倍。啊，这里又发生了什么呢？对。然后我们可以再慢两倍。就是。

陈久宁(01:09:01): 到39毫秒这样的一个程度。就就呃它它只是一个求和，但是你可以发现一个非常简单的运算不同的写法。啊，它会有不同的性能。然后一个非常残酷的现实就是。大多数居。初学者写的是左下角的这个版本。然后这个PPT其实目的就是说我们怎么样从左下角？呃，按照一定的。呃，编程思路。啊去写到左上角。呃，所以这个就是这些这个报告之之前给出这样一个报告的一个背景。呃，它的一个围绕的一个策略大概是。啊，这样五个点就是类型稳定具体数值，然后不可变的结构体内存顺序下降化编程这样的一些东西。

陈久宁(01:09:58): 呃，这个报告也可以认为是高级的绝大开发者一般不会犯的错误，就是这些内容在在我的脑子里面已经成了一种常识，或者说一种下意识的习惯在。对，比如说红轩或者说呃那些。啊。天天跟他打交道的人里面都是属于一个常识性的一个思维方式。然后我们就需要去把这些东西吸收。然后慢慢慢慢就就是了解怎么样写就在他们会是好的。因为否则的话我们刚才就。就如果我们只能举具体例子的话，你我我可以举出一个julia比C快1万倍的例子，然后你可以举一个julia julia39毫秒这样的一个比还要慢慢。办30倍的一个例子，然后你说就好，很很垃圾，然后我们就开始对骂这个这个其实就没有没有价值了，我们这边其实要说的价值就是呃，怎么样把自己的决赛代码写得更好。

陈久宁(01:11:02): 丢了的话。呃，大家大家呃，昨天写了，然后今天也看了啊！第一个印象就是觉得他是一个。他是他像麦拉比一样，是一个交互式类的一个编程语言，对吧，那么这个交互是？呃，对于ja来说，ja依然是一个编译型的语言，它不是一个解释型的语言。呃，编型语言意味着什么呢？意味着我的代码需要先编译。然后去执行编译后的结果。呃，这句话大家能理解吗？

陈久宁(01:11:38): 那么，作为一门编译型的语言，它一定有两个状态。一个或者说有两个阶段，一个阶段是代码的编译期间。还有一个阶段是代码的运行。就是呃，执行期我们叫他。呃compare time和runtime？那么。为什么要划分这样个阶段呢？因为它有一个非常强的一个。呃，限制在于我们去做代码编译的时候，我们并不知道这一段代码。中变量的词是什么？我们只知道。我们比如说我们去定义一个函数F的时候，我们会说函数F的第一个输入值，它是一个int类型的常量。哦，它是一个int类型的数据输入，但是我不知道这个int类型的值。是一还是二还是三？在这个意义上，我们说我们在代码编译的期间，我们只知道数据的类型。但是我们不知道数据的值，数据的值一定是

在代码运行期间知道的，对，这是。啊，编译型语言的一个。

陈久宁(01:12:39): 呃，典型特征。在市区了，他有一个啊，不那么。像边形语言的一个点在于它可以摇摆。它可以在编译型和解释。呃，现象中做一个摇摆摇摆的意思是说呃。我们可以去定义这样的一个事情，就是说，如果我们如果X小于0.5的话，我们返回一个int类型的零。如果X大于0.5，我们返回一个64类型的一呃这件事情，如果你要用C或者C加加加去，你会发现。几乎不可能，或者说他会非常的麻烦，但是对于我们。呃，做实验的人来说，这是一个非常非常自然的一个现象。我们写写的时候，就比如说你去用python或者用matlab的时候，你完全不会去想这个问题。

陈久宁(01:13:34): 呃，我们举个例子。啊，他就像是。他就像是我们在这里面去说。啊啊。就像昨天写的123。啊，我退出共享了。啊，没有啊，它就像是我们昨天我们昨天举了一个例子，说X等于。等于123。

惠立新(01:14:03): 诶，陈博你好像共享确实退出了。

陈久宁(01:14:03): 哦，共享退出了是吧哦好！

惠立新(01:14:10): 好，现在可以看见了。

陈久宁(01:14:10): 它就像我们昨天说的X等于一对，我们昨天说X等于123，然后我们是说，唉，X二等于一。这件事情在matlab和python下都很不可思议，对不对？但是他就会给一个这样的一个约定，或者说这样的一个约定，对于CC加加来说是一个非常自然的一个约定。我们我们声明好了一个数据的变量啊，数据的类型是什么，我们就不能改它了。但是matlab和python它很容易的去修改。它它可以允许你去做一个动态修改。那么我们继续?

陈久宁(01:14:53): 呃，所以呃，这样的一个右边这个代码段，它不是一个高性能的代码段，但是它是我们。呃，日常做实验中。我们对于一门交互式语言。所期望的一种。啊，行为就是我们希望这样的一个语言能够支撑这种行为。但是啊，刚才解释这么多，其实也是要说啊，这样的一个行为，它对于CC加加来说，它不是一个高性能的。啊，代码写法，因为它背后。你没有办法去，在代码编译的期间就决定。你的返回值是int类型还是fl64类型？你必须要在代码运行的时候你才知道。它的返回值类类型是什么？那么你必须要在代码运行的时候才知道的话，你就没有办法去优化它了。因为在这个阶段，你的所有的代码优化已经结束了。因为就是我的意思是说所有的代码优化都是发生在。

陈久宁(01:15:50): 代码编译器？但是。呃，我们就说在代码编译期间，你没有办法知道函数F的返回值是什么，你就也就没有办法去优化函数F。所以我们大概可以。这样理解就是当数据的。类型不确定的时候。呃，编译器会放弃优化或者实际上它。他不会去。对你，你就可以认为它会放弃优化。就我们直觉的理解就是这样一个现象。呃，这样一个现象，对于python和matlab来说，他们是完全不在意的，因为它本身就不是针对高性能去设计的，它就是一个普通的解释型语言，怎么方便怎么来。然后但是对于九俩来说。呃，就是你你你1000万不要这样做。就是啊，你你我们在做实验。我们在快速的搭圆形的阶段。我们可以这样做，但是一旦我们把我们的原型代码实现好之后，我们要去做性能调优的话，我们就要逐逐步的把这种。不那么好的代码替换成。呃，更稳定的更好的办法。对，所以它是一个呃，你你既可以写烂，也可以写好，但是当你要去做呃更更激进的性能优化的时候，你就要把那些写的烂的东西找出来，然后把它改成写的好的版本。嗯嗯。

陈久宁(01:17:20): 所以在嗯这个这个看一下。呃，刚才刚才的这个。就是函数F的返回值不确定这样一个现象，我们在jar中一般会称作类型不稳定。类型不稳定的话，它就会导致啊，编译器它它会放弃优化，然后。这是一点，然后第二点是它会有一个额外的在代码运行期间做的一个检查，这个检查会检查它的类型。就比如说检查类型是int还是64，如果是的话怎么样怎么样，如果是64的话怎么样怎么样。但无论如何，它是一个额外的一个运算。然后。更重要的是内心不稳定，它也会影响它影响的其实不是自己的性能，而是影响的是用户的性能。就是说。嗯，你你比如说我写了一个函数F。啊，然后函数F被击使用了。函数F写的烂。呃，关于函数F支撑没有问题。他他会把鸡的性能变慢。这个是类型不稳定的一个典型的一个场景。那么我刚才只是说呃类型不稳定，但是呃没有给出一个非常强的一个说明，所以这里再举几个例子说，呃，什么样的东西，它是一个类型不稳定的东西。

陈久宁(01:18:46): 呃，在这里的话，我们举第一个例子就是说。呃，我们关于类型不稳定，可以有一个最简单的一个理解是。呃，函数的返回值的类型是不变的。呃，在这个例子里面，我们去他做的事情是，呃随生成一个随机数，要么是，要么是。所以在。在左边的这个地址里，我的函数F的返回值。要么是零，要么是1.0。然后需要注意的是零，它其实是一个int类型，它是一个整数类型。1.0它是一个浮点数

类型。所以在这个意义上，函数F它不是一个。内心稳定的东西，因为它的返回值的类型是摇摆的。能理解吗。

陈久宁(01:19:32): 对，我们可以通过code one type这个宏去看呃返回值的类型，所以这里我们可以看到红色的代码段。啊，说它是一个body introduce，它其实说的就是函数F的返回值类型在在。浮点数64和整数64之间啊摇摆。然后红色表示它是一个会影响性能的类型，不稳定。呃，所以我们可以看得到的是。呃，我们比如说函数F本身的执行效率它非常快，3.9纳秒。但是我们去做一个求和之后，我们会发现它的时间变到了，变成了微妙的级别。就按理来说，如果我是1024的话。呃，它其实就是应该是三微秒。呃，最多500秒这个样子。但是它其实到了一个21微秒。那么这中间一定是出现了一个性能开销。啊，能理解吗?

陈久宁(01:20:37): 呃，如果要快的话怎么办，我们就很简单把。零改成零点零。他就好了。呃，改成零点零的话就是我，我虽然不知道我的返回值是零还是1.0，但是我知道我的返回值的类型一定是。所以在这个意义上，它是一个内心稳定的。呃，代码。然后它的性能你可以看见它是一个69纳秒的一个。又非常非常快对吧？呃，能理解吗？那么刚才的是刚才是我们关于类型稳定的。第一种理解。啊，或者说是最最简单的一种理解，就是说返回式的类型是不变的。但实际上返回的类型可以变。呃，但是它有前提就是。呃，返回值的类型可以根据函数输入输入的类型来唯一确定。这件事情怎么理解呢？就是。

陈久宁(01:21:45): 同样的，我们左边的这这一个它的返回值的类型是。啊。就是它要么是int类型，要么是fl64类型，但是它也是没有办法确定。然后我们可以去。调整。我们可以调整成这个样子。呃，zero和one这两个函数的目那个效果是返回一个跟X相同的零或者跟X相同的。就是说返回的类型就是说我这里虽然不知道X的类型是什么，但是我知道我返回值的类型一定跟X的类型相同。所以在这个意义上，我们说返函数的返回值类型是根据。呃，是可以由X的函是可以由函数的输入X的类型来唯一决定。所以在这个，所以我们可以看见啊，右边的这个版本它同样是一个内心稳定的一个实现。然后它的性能也是非常很快。我比如说我们这里可以看见的是。最右边啊，给了两个例子，一个是F11个是F1.0我们可以看见，如果我传进去的X是一的话是整数类型，一的话我的返回值类型是64。如果传进去的数数据类型是1.0的话，我的返回值的类型是浮点数的64。就是我的，我的函数F的返回值的类型可能有多种，但是它是可以。唯一的被。还是输输入的类型来决定的。那么在这个意义上我们？依然是说它是一个类型稳定的东西。这个大家不知道能不能理解？呃，能够理解对吧?

陈久宁(01:23:40): 呃，刚才说了我们可以通过code one type来检查类型不稳定，那么这里的话就回到我们一开始的那个。呃，你就是说为什么同样是数组求和我有很我就把。我就稍微改改，改个一行，或者说改个几行，它的性能就变差了。是因为在这个例子中。当我写result等于零的时候，意味着。I S T的类型，它有可能是整数。也有可能是。数组a的元素类型。

陈久宁(01:24:17): 所以如果我们想象一下，如果数组a的元素类型是浮点数类型的话，那么的类型它就会在两种类型之间摇摆。能理解吗。所以我们去看呃，通过code one type来看就是说。呃，我的my son。然后传一个浮点数64的矩阵进去，我们就会发现。它的返回值的类型它是不稳定的。然后中间中间它还有一个百分耗时说的类型，也是一个红色不稳定的一个状态。那它这个就是属于一个性能的一个。呃，可以优化性能的一个点。或者说。呃，至少我们代码没写好的一个点。它就是类型不稳定导致我的整个。啊，实现并不是一个特别高效的一个实现。当然在。在这个历史中。嗯嗯。它为什么慢？呃，其实是有其他的原因，但是呃表现出来呃，我们写错的地方其实就是因为类型不稳定导致。一系列的性能问题。呃，然后。呃，在同源的代码库里面有非常多像这样的代码段。

陈久宁(01:25:43): 呃，我们可以看一下，就是我传了一个。啊，比如说关键词参数。然后我根据games的词的不同。我去做不同的算法。然后这个不同的算法会返回一个不一样的数据。数据类型。比如说。如果定等于一的话。我就返回一个向量。如果定是等于二的话我就返回。啊，如果是其他情况下的话。我就返回一个标量。就是向量和标量，它是两种数据类型。呃，在这个意义上，它也是一个内心不稳定的点。但是这种代码在。在同样的那一步非常。非常常见这个是？啊，一个没有办？办法的一个。或者说它是一个？呃，妥协吧，因为同。的代码库里面有非常多，是要想。就是有非常多的函数做出来的风格是matlab的风格。然后迈他的补的风格就决定了，我们是需要有这种。啊，关键词参数去控制算法行为，然后返回。返回不同，根据不同的算法，行为行为，返回不同的数据类型的这样的一个现象，然后。啊，这个现象对于matlab的用户来说是很自然的现象，但是对于九的开发者来说，他是一个。性能的一个隐患。就是我。对于不同的问题来说。呃，类型不稳定会带来多大的性能?

陈久宁(01:27:22): 粉丝。啊是可以。啊对比，然后去判断的，但是整体。从一般性的角度来说。啊，类型不稳定，它是会导致性能损失的。啊，然后这边说要说的是。呃，类型不稳定这个东西。它是一个呃，有传播链的一个现象。啊，这个现象该怎么解释呢？就是说我这里有三个函数。啊左边。啊是一个函数。然后它是一个写的非常好的，我用绿色框起来表示它是一个类型稳定的东西。然后第二个函数是生成数据的函数J data它也是一个类型稳定的东西。然后。呃，类型不稳定的函数是？它我们刚才说了，它的返回值的类型可能是整数。也

有可能是浮点数，但无论如何，它是一个类型，不稳定。然后当我们把my sum data还有拼在一起的时候。呃，得到了含税H这个时候H它是一个类型，不稳定。因为。因为F是类型不稳定，所以它的传播电导致整个H。也变得类型不能比较。然后这个时候我们可以发现H的性能是？比较慢的35毫秒。但右边如果我们把。F slow改成类型稳定的版本。我的HE也会跟着类型稳定，我的性能也会跟着变快。啊这个。

陈久宁(01:28:56): 这一页例子可以理解到吗？这一个这这一页历史说的就是呃类型稳定，它很容易从底层传播出来。所以如果我们要写的是一个底层代码。呃，我们需要把内心稳定这件事情。啊，非常非常的在意！如果我们要写的是一个高层算法，是一个APP是一个用户。啊，不太可能复用的一个脚本。能跑就行。就目前目前来说就是。

陈久宁(01:29:29): 大概是这样的一个现状吧？

陈久宁(01:29:33): 因为确实是呃，内心稳定的一个东西，他。呃，从性能的角度来说，我们需要做类型稳定，但是从方便的角度来说，类型稳定有时候会让代码变得不那么方便。就是使用起来不那么方便，所以我们在具体写代码的过程中是需要权衡。

陈久宁(01:29:52): 然后我这边要说的点就是说如果你要写的是底层代码。好，一定把内心放在心上。如果你不是鞋底生态嘛，你可以去讨论，你可以去做不同的。啊，对，这一页说的也是如果。确定要引入一个类型不稳定，你最好在接近用户侧的地方。接近API的地方去，去引入你不要在代码的核心实现。这样。

陈久宁(01:30:28): 呃，刚才说的是呃类型稳定这样一个现象，这个现象也是九大的一个非常非常特殊的一个点。啊，大家有。有有问题吗？上一张PPT。啊哪里。再上一个。啊哎，你是说F fast吗？先data。它就是生成了一个。N乘N的批？登上了一个N乘N的急转。然后居正的每个元素元素是通过F区。是随机出来的。对我，我只是为了把它强行筛成一行，所以用了一个可能看起来不那么熟悉的一个语法。

陈久宁(01:31:34): 呃，有其他问题吗？呃，那么好，我我们就假设大家已经知道类型稳定是一个非常重要的一个。呃，话题了。就后面大家如果去写就要代码的时候，请思考这个问题。然后关于九俩的第二个非常重要的话题在于。啊，具体的数字类型。这件事情指的是什么？我们来看这个例子。啊，这个例子里面我们写了a等于方括号。呃，它其实就是一个。啊，性能的隐患点。因为方口好。啊，意味着它是一个any类型，我们可以。我我我来分。为什么有一退出他之后？就是我们去看，当我们去写。嗯，X等于方括号的时候。你可以看见这里有一个。Any的类型标记，它意味着我们可以往X中。丢任何的，比如说我可以标一个数，丢一个整数我可以标一个。啊数组我可以给我一个字符串。呃，这样一个现象，从一门比如说python和matlab来说，呃挺好的，但是从高性能的角度来说，它是不好的。啊，我们继续。然后如果我们要把。左上角的代码变快，我们只需要给a的。这个向量的类型给一个标记。呃，当我们写int方括号的时候，我们创建的是一个空的数组，但这个空的数组的类型。一定是一个int类型，你不能往里面存。

陈久宁(01:33:31): 其他的数据类型。你不能往里面存？啊字符串你只能存一个int类型的整数。然后在这种情况下我们会发现诶。代码的性能变快了。他其实就是啊！说我们的数据类型要尽可能是一个具体的。就是你的确定性越高。呃，你的代码性能可能会越好。如果你需要特别强的灵活性。

陈久宁(01:33:59): 那么其实你在一定程度上，你就在，你就是在牺牲性能。这边的话啊，给第二个。啊，就是说我们右边是一个结构体。呃，这个结构体定义了一个二维空间的点。和Y有两个？两个这样的。然后我们去。在左边我们定义。定义了这个结构体就是空间中的两个点的距离。反正他就是欧式同欧式距离。啊，就是大概就是相减平方，然后相加再求。更好这样的一个。这个东西。对，然后我们去。生成一堆点。然后去计算每个点和原点的区别对啊，这个就是函数F的事情，就是生成一堆点。F干的事情就是每个点离原点的距离。我们可以看到它的性能大概是在119微秒。呃，然后这件事情。如果我们让结构体。的类型变成具体类型。呃，它这个性能会变成1.6。就这里的差别在于。如当我们去定义一个结构体的时候，我们如果不给结构体类型声明的话。

陈久宁(01:35:36): 这个题的类型。呃是艾米。我们给他一个具体类型声明的话，他的。呃，它的类型就是，比如说这里是64。那么在这种情况下啊，具体类型就意味着编译器知道该如何去优化它。那么我们就可以拿到一个更快的。啊，能够理解吗？

陈久宁(01:36:09): 这里其实有。有一些点啊，我看看。来我来问问这单，这单位问一些问题。我们来，我们来做一些呃。或者说问题学？这个是是不是具体类型？它是具体类型对吧对。啊X等于。比如说一。2.0。他是不是具体在？呃，他实际上是。因为他会，他会在创建的时候。啊，提升到福电。啊X等于比如说。Int12.0。他会在创建的时候。强行转成int？然后。比如说。呃，居然里面有一个叫做的东西。对，这个是不是具体类型？啊，我们可以通过。就是四具体类型。这样一个函数来判断。X的类型啊？

陈久宁(01:37:49): 啊哦，我这里我这里写错了。啊对对。这是这这个呃，这个也是我刚才要说的一个点就是。啊，我的这个题vector。啊，里面如果存了的话。那么我的。向量的里面的元素它其实不是一个具体类型，那么它就不是一个高效的点，但是。啊，这里是因为？它不是一个具体类型。但是对于来说。这个我们可以暂且当做一个事实。呃，我不是非常想。往里面解释，因为它涉及到就是语言学家的一些。恶趣味吧。就当他们就是语言学家非常严谨的一些地方，他会是说is concrete type。这是一个。这是一个具体类型。就是。啊，这是你们，你们可以把这个当做一个事实，或者说一个特殊情况去理解。就可以了，不需要去拯救他。

陈久宁(01:39:02): 但反正但是我我我要说的是，呃，尽管它是一个具体类型，但是我们要说要说的是any它不是具体类型，然后它也会导致整个的性能也会变慢。大体来说只是这个意思。好呃，我们回到ppt。然后。呃，刚才说了两个，就那里面呃典型的一个注意事项，一个是类型稳定，一个是具体类型。这两个点。啊，它都是属于，如果我只是一个普通的matlab或者拍摄用户的话，我可能很难去注意到这些点。但是从ja写代码的角度来说，你想要把ja代码写快，你必须要注意这些点。就是你可以说我的第一个版本。的算法。可以，不管这些点，我只要把把结果算对就好，但是如果你我们要做一个好的工具箱，我们就是需要保证我们的第二个版本或者第三个版本把这些点全部给处理掉。这个是。呃，一个点我有时候我我会跟别人说这叫渐进式的优化就是。逐步逐步优化了对吧？他不需要你在第一天就把函数写到一个完美无缺的程度。但是你可以啊，在第二天或者第三天慢慢慢慢把它迭代出来。

陈久宁(01:40:32): 然后这里要说的是第三个。第三个点在于。啊，可变结构体和不可变结构体。呃，这个在java里面也是一个呃，就是从从C程序员的角度来说，这是一个非常容易理解的话题。但是对于jar程序员或者说程序员来说，这个话题可能。不是那么容易理解，但站起来说，我们可以把它当做是一个。啊，寄给你的事实。记下来就行，就是说呃，如果没有特殊的必要。呃，尽可能的用不可变结构体。你在确实需要的情况下，你可以用可变结构。呃，这里的意思是说，呃不可变结构体。啊编译器更容易优化。

陈久宁(01:41:23): 或者说不可变结构体在？在计算机的内存中它。有更大的可能性是在所谓的占内存。啊，就是我们的内存有两种结构，一种是叫赞，一种是叫堆。然后暂时很快的，暂时一个静态的内存。啊，内存布局，然后堆是一个动态分配的内存布局。啊，所以如果我们用不可变结构体的话。我们的数据有很大的可能性，是在站内层中。也就意味着我们。数据的读取和修改。他都可能会非常快。啊，他们其实没有修改，如果在这样的日程中是不能修改的，但是读取会非常快。所以这是为什么用不可变结构体的一个原因？

陈久宁(01:42:13): 呃，这个例子里边，它我们可以看到，呃可能性能差异不是非常的明显。但是。好看一下这里。啊，这这里的话给了一个，给了一个例子。呃是说。啊，我们通过不可变结构体是可以创建一个所谓的。呃，零开销抽象的一个概念。就是我们围绕比如说四类型，我们可以创建一个新的数据类型。这个新的数据类型上面我们可以定义加法。然后定义数组求和啊，所有的这些操作。然后我们发现就算我们对它做了一层代码封装。呃，它的性能和我们去做一个裸的。计算的性能是一样的，就是一个是143秒，一个是144秒，它的性能是一模一样的。

陈久宁(01:43:11): 就是也，这里要说的点在于啊！这个抽象就是我们，我们提供一个代码抽象，但是这个抽象不会带来额外的开销。这件事情对于matlab和拍摄来说也是。啊，一个不太可能做得到的一个点，但是对于约来说，呃，不可变结构体啊。在这一点上。啊，效果还是挺不错的。它背后其实也是因为qa是一门编译型语言。然后相比而言，如果我们用的是可变结构体的话，你可以看见这是它的性能会从143微秒。呃，急速的下降到6.9毫秒。这里面是一个大概200倍的性能差异。所以如果不是必要的情况下。尽可能使用可啊，不可变结构体。

陈久宁(01:44:07): 当然我这里举的例子是一个。这这个例子属于底层算法的一个例子。所以，如果你是高层写高层API写用户API做做一个脚本，做一个APP。啊无所谓爱怎么写怎么写，只要能用就行。呃，然后这里的话给的是一个。底层的。叫做L LV M的一个代码表示就是说我们。只有两代码在编译成二进制之前的一个结果。然后在这个结果里面。啊裸数据。它是一个非常干净的，就是11行两行就两行代码。就是。加法在。在那个底层L LV M这个级别就是两行代码，叫做fad，就是大概这个意思。呃，然后如果我们封装的封装的那个。不可变结构体它会多了一些东西，但是多的这些东西它是一个可以被优化的一个东西，如果是不可变结构体的话，它会更复杂，然后这些东西它是不可被啊，很难被优化的一个点。

陈久宁(01:45:21): 啊这些。啊，这个可以。我用。了解它为什么会长成这样一些呃，它生成的这些东西具体是什么意思？但是呃知道。大概有这么一个现象就可以。对，刚才说的是啊，可变结构体和不可变结构体的一个变。看看有没有新的问题。哦，可以啊，休息一会。我们休息十分钟。

陈久宁(01:46:05): 呃，关于内存有问题吗？没问题啊。呃，那么我们下面到？第二个话题。最后一个话题就是向下画编程。呃，你们之前有有有在matlab和P上写过香港化的代码？啊，听说过这个概念吗？听说过。呃，向量化编程它其实指的是我的一个。呃，向量化编程这个概念，或者说向量化这个概念它有两种。截然不同的含义。呃，一种含义是不是cpu的？向量化指令。然后第二种含义是指。的这种。函数的输入和输出都是向量。这样的一个模式，这种这样的一个编程模式，我们这边谈的项链和编程是指和开的这种模式。在matlab和python中我们要优化一个段代码，呃，如果你。尝试在matlab中优化。性能的话你就会发现呃，大家给的建议一般就是把循环改成向下画带嘛。所以还是写过向下化代码对吧，对啊，很好啊，这个是和python的一个非常典型的性能优化的手段呃。

陈久宁(01:47:36): 左上角的这个函数F，它其实就是一个向量化编程的一个范式，然后它在修那里面对应的逻辑其实是左下角的这个逻辑。就是如果X大于Y。啊返回。呃X乘Z，否则返回X加Y。就就这样一个，非常简单的一个。啊函数啊，在线下化编程中表示的是这样子。呃，然后。对于朱娜来说，很有趣的一点在于。如果是向量化代码和。所谓的标量逻辑代码的话，你会发现标。就是标量加广播的模式会比向量化更快。啊，能发现吗？啊，就就能理解能理解这一点吗？对，就我们我那个为什么向量化？慢这件事情我们后面会解释。但是在这里的话。呃，说的是。

陈久宁(01:48:44): 看一下编程这个手段。在jar里面并不是一个性能优化的手段。就这是我们关于向下编程的一个认识。就是说啊，大多数时候我们可以通过向下画编程的方式去写出一段。啊。可能是说简短的代码，但是这一段简单简短的代码并不一定定会。加速你的。呃，这里面的解释是说。呃，我们左边的这一段向量化的代码。他如果真的展开来的话。它其实背后是一段一段的for循环。呃，右边是它展开之后。呃，等价等价的一个逻辑。就是说我先比如说我第第一行代码是M等于X。然后对于gay来说，它其实背后发生的是先创建一个像像M。就先创建一个矩阵M，然后对这个矩阵。啊，对，计算X的每一个元素。呃，对矩阵X和Y的每个元素去判断，然后把判断的结果传到M里面。啊，这个就是。呃，像样化代码中第一行。实际背后发生的事情。然后我们类似的我们就可以发现，呃，整个代码段它背后会出现。M等于similar X一次内存创建。然后C等于C面的X两次内存创建。然后它一共会有两四个内存创建以及123。是它一共会有四个循环。能够理解吗?

陈久宁(01:50:35): 这个是象象化编程，就是我们写，当我们写出一段向象化的代码之后，在这里面背后实际上发生的是这样的一个事情，实际上对于matlab来说也非常相似，对于，比如说，比如说我们对于python或者来说。当我们在一个python或者matlab里面写C等于a乘B或者D等于C大于零的时候。呃，我们实际上是把这个循环放在了。C语言里面是？执行罢了。把循环放在C语言里面去执行，会有什么样的好处呢？大家有有没有？能能不能理解到这里?

陈久宁(01:51:28): 比如说我们回到。回到一开始说的。Matlab中向量化编程它可以加速代码性的，呃，为什么能够加速代码性呢？呃，因为matlab和python中的for循环是很慢的。做循环很慢，这件事情我怎么样避免它，或者说我怎么绕开它？绕开它的方式很简单。我把for循环放在C里面去执行就可以了。

陈久宁(01:51:59): 所以向量化编程的本质就是。将一个for循环的。展开。放在一个更高效的语言中去做。能够理解到吗？对这张图说的就是这样一个一件事情。然后将for循环的展开放在更高效的语言中去做。它就能够加速matlab和python代码，因为这个加速不是因为项链画编程好，而是因为python和matlab的。不喜欢太慢。

陈久宁(01:52:36): 呃，能够理解吗？然后。呃，我们知道。性能很高。那么这就出现一个新的问题了。呃的for循环本身就已经足够快了，我们就没有必要。再再做额外的向阳号编程去做展开了。所以为什么我们所以这就回到我们上一张图。这张图里面。为什么。

陈久宁(01:53:02): 呃，通过下面画编程写出来的代码性能会更慢。是因为我们没有必要在这中啊，强行去写向量化的代码。我们只需要去按照一个简单的标样逻辑去写，然后很自然的把它广播就可以了。这也是呃同源目前呃正在。呃，正在进行时的一个优化，就是说早期啊，同源的代码库有很多是。啊借鉴了。然后matlab有很多的代码，它是matlab的代码，它是优化了的。优化的代码，它一般都是向下画编程的模式。然后同源早期的代码啊是借鉴了，所以同源早期有很多向量化的代码写法，但这些代码写法其实从julia的视角来看。完全没有必要，我可以用一个更简单的。呃，负循环或者说更简单的标样逻辑的方式去写出来，然后性能反而还比。呃，这种呃冗长的向量化要更更高效。对，所以并不是所有情况下，我们都应该使用香蕉花变成。呃，在某些情况下。呃，像样化编程可以让代码变得更简短。然后并且性能损失不大。在这种情况下，我们可以用香港化编程的手段。但是如果说我们在写一个底层代码的话，我们可能就需要去思考这个问题了，就是我们究竟需不需要向量化变成。嗯。能理解吗。

陈久宁(01:54:35): 当然我这里只说了一半。另外一半是？啊，下面号编程？这实际上我刚才给的结论。放在一个更更大的场景下，它其实是错的。就我刚才给的结论是说我们在很多时候。呃，向下编程并不是一个必要的东西。这个话这个结论只对CPU来说是对的。为什么只对CPU来说是对的，因为有非常多的，比如说GPU的运算。它它的数据接口就是一个。向量化的接口，在这种情况下。你没有办法绕开？呃，向阳化编程。然后。在这种情况下，你只要把数据接入到向。GPU里面你的性能自然。呃，这个。这个是一个怎么说呢？呃，更更具体的一些情况吧，它是和GPU编程，然后异构计算可能会相关的，但是至少从一般性的july的角度来说啊，相应化编程它背后其实。啊，并不是一个啊，绝对！它，它并不是一个优化性能的手段，它只是一个。呃，简简化代码的一个手段而已。关于向量化编程。啊，我后面应该下午可能会在。再找几个例子去跟大家讲。然后。嗯，我们先这个就是这个PPT的一个全部内容。我们看一下有没有？有没有问题？呃，项链号编程它不是性能优化的手段，它是简化代码的手段。

陈久宁(01:56:49): 就是有时候你可以把一个很复杂的for循环用一行代码去表达。在这种情况下，向量化编程是一个。啊，让代码变得更可读的手段，但是它并不会让你的代码变得更快。在在那个在哪？平时？呃，这个。这个看具体情况吧，并不是说所有的情况下我们都需要一个。

陈久宁(01:57:22): 最短的代码？就是这个代码，只要。能够一眼看懂，或者说在别人看起来不累，那么他就是一个好的代码。

陈久宁(01:57:33): 它跟你用不用向量化编程？用不用循环？没有关系。比如比如说这段代码，它就算再复杂，你写一个。呃，特别好的注释，那其实也也也是挺不错的。呃，比如说我可以给大家看一个。呃，看一个史记的一个例子。在。

陈久宁(01:58:14): 这里大概有一个。419行。到516行有。有一个这么长的一个。然后这一段。呃，代码实现呢，它很难读懂。然后你会发现这里面有？大量的做事在里面去说我们干的是什么事情？然后啊。不，不需要去看懂，但是你你我们可以给出一个答案，就是说这里有100行的代码，这100行的代码，它实际上如果我们用最简单的方式去写，一行就写完了。对它背后的那个数学表达是用正常写，就是一行。然后我们为了把它加速到一个非常快的程度。把它展开成了100行是做专门的。就像刚才说内存的排列顺序可能是一个性能影响的因素之类的。然后这个东西它就是会很复杂，就是一个高度优化的算法，它的实现就是会很复杂。

陈久宁(01:59:15): 那么我怎么让他能够看懂，我就去给注释？给比如说我，我是说这个算法来自于哪篇文章？给出各各种各样详细的注释，然后让他能够变得读不懂，或者说就算我读不懂，这段这段代码没有关系，我知道他大概干了一件什么事情。它也是一个好的。呃，我们看看有没有别的问题。

陈久宁(01:59:42): 第一块讲的是类型的题，我第二块讲的是它的具体数据类型。这个第二分的具体数据类型数就是进阶版的类型的。呃，问题是说具体数没没有，呃，我我先我先复述一下问题啊，就是呃这里问的问题是呃第一块我们说了内心稳定这个概念。然后第二块我们说具体类型这个概念，然后问题问的是具体类型是不是就是一个进阶版的类型稳定啊，他们在一定程度上是的。因为它其实都是在说编译器有没有办法去优化我的代码。呃，如果它不是一个具体类型的话。编译题是不知道？它是什么类型的？那我就当然没有办法去优化了。如果是一个具体类型，编译器就知道啊，我的数据的类型是什么，我就有办法是有的，因为从编译器的视角来说，不存在所谓的类型也不存在类型，它只有int或者64这种。具体的数据类型。所以我们写代码的时候就尽量是这个类型之后，呃，具体定的。对，所以在写代码的时候，除非必要。呃，尽可能写类型文件的代码，尽可能写具体的数字类型的代码，然后尽可能的写不可变的结构体。你你不介绍了，你说扣着我。

陈久宁(02:01:18): 呃code one type这个红它它需要。需要介绍吗？嗯嗯。比如说我们，我们把刚才的那个。这个红的意思其实就是说我，我去检查代码的每一段。它是否是它的返回值的类型大概是什么样子？然后如果说呃预存在类型不稳定的情况？我用红色或者黄色把它标记出来。这就是抠的，我还不干的事情，就俩存在非常多的把代码。

陈久宁(02:02:08): 代码展开的手段，比如说cold。啊，这是一种。啊，就是cold type也是一种。就是。像这种东西都是说julia的代码是怎么样，一步一步被编译到。底层二进制的，然后code one type其实就是说我们把jay的代码展开到一定程度之后，我们去分析它的每一个阶段。啊，它的形状或者说它的特性是什么？对这种东西啊！它不是一个？呃，怎么说呢，它可能很难讲，因为它就是一个靠实践来。来得到的一个东西就是，呃，比如说为什么会在这里会变成什么慢点，然后慢点，不，然后这个东西就是只要能看懂就行，你不需要去理解它为什么会生成这个样子。你只要能够连蒙带猜知道哪一段对应于哪一段大概就够了。然后像这里我们比如说，比如说这里有一个呃问号，就是这个三元运算符。问号的这个对于jana的代码展开来说，它就是go to。

陈久宁(02:03:21): Go to第三段。然后if not什么什么一就是这种东西啊，大概背后对应的就是一个if else的判断。就不需要知道为什么这样展开或者说啊这种。它一段代码它会编译成什么样子，它是一个你不断的去读，然后来猜测来积累经验得到的。嗯，看看有没有新的问题。

田佳辰(02:03:56): 嗯，陈博士我想问一下就是。

田佳辰(02:04:01): 呃，我是做测试代码稳定性的时候嘛，然后。就是有两个变量，它是取了一个就是输入的那个。输入的一个切片嘛，相当于是，然后在那个就是稳定性检查里面，它就那两个变量标红了，但是剩余的那些其他部分。它都是绿的，但是说明就是整体的函数还是稳定的。而且只是取取切片的操作，我不知道为什么它会。

陈久宁(02:04:26): 呃，这里呃，它其实不是内心稳定的，就是你你这个去切片操作肯定是呃，有一个可能是你的数据源本身就是内心不稳定。

陈久宁(02:04:42): 啊，我举个例子，我们可以。

陈久宁(02:04:44): 我们可以看，就是去切片操作这件事情一定是类型稳定。

田佳辰(02:04:49): 呃，是用了一个view的宏，然后不知道是不是因为是这个还是因为就是我取切片的时候，它是呃，我是取它的其中一部分index，然后用的是那个就是。

田佳辰(02:05:01): 比如说一到二到and这样的表达式。

陈久宁(02:05:02): 你你是不是你你是呃你你你做的是不是类似于这样的操作?

田佳辰(02:05:07): 呃，对嗯，类似，但是就是我那个是比比较具体一点，它是比如说是。

陈久宁(02:05:09): 啊，在这里它是一个内心稳定的一个东西。

陈久宁(02:05:19): 呃，我看一下。

陈久宁(02:05:34): 对，这切片操作本身是类型绑定。

田佳辰(02:05:38): 要不要不我把那个具体问题我共享一下?

陈久宁(02:05:43): 如果如果说内心不稳定的话，意味着你的数据源肯定是不稳定的，你可以共享。

田佳辰(02:05:44): 啊，好的好的，我把这个。

田佳辰(02:06:12): 就是它结果的话是运行呃是类型文件，但是它中间部分的话。

田佳辰(02:06:19): 就是这边它是一个。不知道为什么它会变成一个union，但是结果的话是有影响的，这样的话会对性能有一定影响。

田佳辰(02:06:29): 这边的话操作的话是就这样吃了一个切片，然后。

陈久宁(02:06:30): 非常好的老师主持是？然后。

田佳辰(02:06:34): 看的话好像我看一下。

陈久宁(02:06:36): 这样子话都上了。

田佳辰(02:06:38): 电量就剩。

陈久宁(02:06:40): 电脑设置。

田佳辰(02:06:41): 输入的这个X来的。

田佳辰(02:06:44): 我就说直接说的。

陈久宁(02:06:46): 不是，现在也不行。呃，那那在这个例子里面很典型啊，就是我问你X是什么类型?

田佳辰(02:06:58): 嗯，X就是跟输物的类型来嘛，输物的话是float，它就是float。

陈久宁(02:06:59): 签字就是跟书的类型，数字的话是说站在一起啊，你你你给我你给我看一下你具体输入的是什么类型。

田佳辰(02:07:12): 这是一个random?

陈久宁(02:07:15): 对，那那那个比如说155到162行里面，它有一个分支，对不对?

陈久宁(02:07:28): 对，那那这里面的temp X它的类型是稳定的吗?

田佳辰(02:07:34): 嗯，看X也是稳定的，这上面的话是一样的，它就是X的数据类型。

陈久宁(02:07:53): 是稳定了吗?

田佳辰(02:07:54): 这好像它这个lo变量它是写的是问题。好，这边因为呃这是一个。

陈久宁(02:08:07): 哦，我需要看。

田佳辰(02:08:11): 就是，就是它做了个判断，因为我输入的是偶数的，它就直接等于X，或者如果可能输个奇数的话，那可能会。

陈久宁(02:08:12): 就是。是的。

田佳辰(02:08:22): 低速长度的那个?

陈久宁(02:08:32): 你你这里你这里摊牌肯定。肯定是类型不稳定的，就单看155到162。

陈久宁(02:08:38): 摊牌一定是类型。

陈久宁(02:09:03): 那么T类型不稳定的情况下，你的a和B。呃，他肯定也是类型不稳定。

田佳辰(02:09:11): 嗯，那我。

陈久宁(02:09:12): 呃，我目前只能判断到这里。

田佳辰(02:09:14): 好的好的，我是不是这边?

陈久宁(02:09:16): 就是我对我，我就是我能说的是我能说的是view本身去切片操作肯定是类型稳定的，但是类型不稳定是会传播。

陈久宁(02:09:24): 所以你需要去看你究竟?

陈久宁(02:09:27): 第一个热情不稳定是从哪里出来的?

陈久宁(02:09:29): 我能够看到的是TFX它是类型。

田佳辰(02:09:30): 好的好的话。

陈久宁(02:09:47): 嗯，我们看看有没有别的问题。

陈久宁(02:10:02): 嗯，没有别的问题的话。上午的时间是不是到了上午迟到七点？第一个啊试试。啊那那要不我们上午就到这里结束吧，有几个问题我放到下午去讲。

陈久宁(02:10:29): 好，如果说没有问题的话，我们来我这边，上午一开始有两个一个问题，我是没有回答的。第一个一个问题是。嗯。我我们直接来就在里面啊，我就是。Int呃三D点，比如说3.0。对，就是说。3.0。他们是什么差差别？一个是转的是，一个是下面是异性的邀请。呃对。这是。比如说。Int3.0啊3.0这个我们把它叫做构造函数。叫做啊constructor。然后。X比如说X等于三或者。啊H等于三？它其实都是叫做啊have annotation。呃，我不太确定是不是具体的官方的叫法是不是叫他，但是反正大概类似就是大概意思就是类型标题。然后这个类型标记和python会不一样，就是python的类型标记它是不会报错的，但是jar的类型标记是会报错的。嗯，这个是表层语法上的一个差异，就是说一个是。构造函数一个是类型标记，那么在具体使用上的话。

陈久宁(02:12:12): 呃，刚才呃看到的一个点点在3.0它会直接报错对吧？那么在具体使用的情况下，我们还有一些点在于。啊，如果一个算法。很难去推断出他类型稳定或者怎么样的情况，我我可以举个例子。呃，比如说我们写一个类型不稳定的还是iphone。然后如果X大于大于零。啊返回。返回X，否则的话返。Return返回一个。然后一个零？呃，这个例子肯定是类型不稳定的，能够理解吗？比如说我们这里可以看。它一定是一个类型，不稳定。呃，在有些情况下我们。确确实实就是没有办法避免类型不稳定。其实是在核心代码。有些特殊的场景下，我们确实就是没有办法避免。呃，那么在没有办法避免他内心。呃，不稳定的情况下，比如说我就是想要让F能够高效运行的

话，比如说。嗯。我想想看啊啊！你就写写一个这样的吧！

陈久宁(02:14:13): 呃，在这个例子里面，这是一个非常非常呃。

陈久宁(02:14:21): 这是这，这是一个捏造的例子。这个捏造的为，为什么说它是一个捏造的例子呢？因为。呃，我调用。诶，我这里为什么会有一个？啊，这里是没有a的，在这个捏造的例子里面，我。我永远不会进入到这个分子。论这个函数它返回的是零到一。所以我永远进入的都是善良自由分子。呃，能够理解吗？但是因为我的F是一个类型，不稳定。所以会导致我这一步其实运算会比较慢。运算比较慢。我们可以来验证一下。Using benchmark. 呃191纳秒。呃191纳秒在算快还是算慢，我们只需要做一个对比就行了，比如说我们可以去看。嗯，可多太君。他这里就是会告诉你他是一个类型，不稳定。那么我们怎么样让它类型稳定?

陈久宁(02:15:29): 或者说我们在没有办法避免类型不稳定，类型稳定的情况下，我们有没有办法去标记一下，比如说。我可以通过这个东西去标记。它是一个。Root64。举个例子我们就。捏捏一个这样的句子出来。那么。呃，他快了一点，之前是我印象中是一是190。现在是150。然后。如果我们去看科多皇太一的话。嗯嗯。啊对。呃，就是F的返回值它依然是类型不稳定的，但是。他我们后面。其实是一个13，它是一个。呃，稳定的一个四。在这种情况下，我们就是说，在没有办法类型稳定的情况下，我们通过类型标记引入了一些额外的类型信息。

陈久宁(02:16:39): 就是等于就是说我们告诉的类型信息越多，它越有可能优化。或者我们换种换种说法，就是你的代码写的越像C。

陈久宁(02:16:50): 像C是什么意思呢？就是C语言的代码的一个特点，就是你说所有的代码写法都是有一个非常。都比如说所有的变家都都必须要声明类型。嗯，像这种就是你，你就在里面，如果你你给所有的变量都声明类型，你的这个修改代码基本上就是跟C。没有太大差异了，在这种程度上。嗯，你就算是一个类型不稳定的你你你的性能其实也是很高的。所以在这个意义上，我们可以把这个T可以选理解成是C代码的那种类型声明。或者说类型电影呃能够理解吗？对，然后相比而言。呃，我们是完全可以，我这里做了一个非常危险的操作，我们是完全可以进行一个，比如说。等于。呃，在这种意义上的话，比如说1.5。他直接报错了。直接报错了。对，所以我说这是一个非常危险的操作，对，但但是呃我我我本来是想演示，想说的意思是说呃构造函数。想说的意思是?

陈久宁(02:18:18): 构造函数并不需要。返回。啊，对应的类型？就没有，没有任何质量，没有做任何这方面的要求说。啊，意思是说int一个东西的时候？并不一定。真的是。就是它的返回。的返回值并不要求一定适应。就是从。从编程规范，规范的角度来说，或者说，从语法的角度来说，九俩的编译器并不要求一个构造函数一定返回。对应的数据类型。能够理解吗？呃，我们来捏一个新的东西。这个这个地址刚才我做的事情太危险了，我们做一个可能安全医院。比如说。不知道，特别要是。呃，我可以，我完全可以去写一个，比如说。然后P。My point IB等于？我就等于洗就好了，我啥也不干。这也是一个构造函数，但这个构造函数它其实并没有返回int类型。啊，能够理解吗?

陈久宁(02:19:50): 对，就是我可以不遵守这个规范。但是我这这里说的是啊！最大编译器不会禁止这件事情。然后绝大多数程序员会遵守这个规范，但是。我们不能。从从那个julia的这个类型稳定或者之类的方式上去理解的话，你不能要求。他一定会返回一个。啊，这样说能理解吗？它是一个。呃，关于构造函数的一个点。就是相对应的。这个是他们的一个呃细微的一个差异。在大多数时候，你你都可以把它当成是相同的东西去用。嗯。看看有没有问题。就刚才讲的这一点。就是构造函数和类型标记的差异。如果没有的话，我们就进入到一个可能会比较难的一个话题了。

陈久宁(02:21:05): 呃，之前上周五的培训过程中大家有没有听说过？多重派八这个这个词。听说过吗。点头或者摇头。哦，没听说过是吗？那我这边就要。就需要慢慢慢慢跟大家讲。我们先先来说第一个。作为导演吧！JI的函数的定义方式。有两种。一种是F。XY比如说。叫FX等于。就说X平方。这是一种函数定义的方式，对吧？还有第二种函数定义的方式叫FX。比如说十平方。这也是一种它无非是一个单行的定义和一个多行的定义。这次还居然还是定义了一个基本方式。能理解吗。很简单的对吧？应该很简单，然后。啊，我们我们说的这个其实是一个所谓的。呃位置参数，然后关键词参数，比如说我加通过分号。我可以去引入一个，比如说Y等于。X加Y比如说我可以写成这个样子。那么我在使用的时候我必须要写Y等于，比如说Y等于二。啊，我才能够用它。如果我直接去写FER的话。啊，他是会报错的。因为在这个在这个函数定义里。Y是一个关键词参数。啊，能够理解吗？哦，我可能。我我我们。我们先先就是在讲多重派发之前。

陈久宁(02:23:02): 我们先讲第一个话题就是。函数的基本定义方式。刚才说的。两种，第一种是FX等于。比如说我们可以去写FXY等于X

加Y。这个是。这个是。X和Y都是位置参数。在这种情况下，我们是直接去使用。我这边是使用ER我们就可以。就可以用的一个东西。嗯，然后。然后第二种?

陈久宁(02:23:51): 第二种词，所谓的关键词参数，关键词参数是通过。通过分号。风格的。比如说fx分号Y。等于X加Y？呃，在这种意义上的话。我们就定义了一个所有的。嗯，关键是参数。的函数，这种情况下我们直接去下敲FER是不work？不work的原因是因为呃我们第二个参数必须要作为关键词参数，你必须要显示的去把它写出来。能理解吗。那那我们再进一步，比如说。这里必须要写完的你啊！然后如果我们去定义一个，比如说。呃，关键词参数。

陈久宁(02:24:53): 与位置无关。呃，是什么意思呢？就是我们去BF啊，我们去B吧。甚至我我都不需要写X，我就直接先打一个分号，先打一个分号，就是后面所有的东西都是为关键值参数。就是比如说XY。等于X加Y加G啊我们。可以把它拆分一下。呃，在这种情况下，我们可以写X等于一，我们可以写Z等于三，再写Y等于二。就是说啊。关键词参数跟位置无关，我可以按照任一个顺序去去写，比如说我先写啊，先写这一再写再写X这都是可以的。啊，这个能够理解吗?

陈久宁(02:25:49): 对，然后相比较一点的话。位置参数啊，一定是跟顺序有关的。就说我们写一个H。当选的去HYG？在这种情况下，我第一个，第第三个，第二个他一定是。按照固定的顺序复制的，这就是位置参数的一个特性。对，然后关键词参数和位置参数不能复用。关键词参数与。为此阐述不能混用。呃，不能混用的意思是说。呃，对于XH这个函数来说，我们不，我们不能写X等于一。等一下。等于27等于三这种学法。因为我们定义的是位置参数。我不能以关键词参数的使用方式去使用它。然后对于G这个函数来说，我不能以位置参数的方式去使用它。能够理解吗？就是说不能混用。对于python来说是可以混用的。但是九俩不允许呃，我们后面。啊。如果大家还记得这个问题的话，我们后面。就去讲为什么，但是现在先不讲。

陈久宁(02:27:20): 这个是关于函数定义的一个基本的一个铺垫。有了这个基本铺垫之后，我们就开始进入到所谓的多重派发的一个概念。我们我再我再重新开一个新的流量。呃，我们可以去定义，比如说F。等于。然后在这种情况下，我可以去，比如说一和二，我传两个整数进去，它会相加。我传两个浮点输进去他也会想家。啊，他也能够用对吧？呃，他的行为是一样的。但是实际上我们可以去定义一个更多的行为，比如说我可以要求。啊，如果传两个浮点数的话。等于X乘Y？呃，这里发生了两件事情。呃，我们可以先看，比如说二点零三点零。它的结果是六。6.0因为他。他用了乘法。我如果长二和三的话。他的结果。事物因为他用了加法。这里就就有两个现象。

陈久宁(02:28:43): 就是首先第一个现象是啊！第一个现象是啊！函数的定义没有被覆盖。啊，或者说没有被完全覆盖。就函数的定义没有被完全覆盖。增加了一个新的方法。呃，我们可以看到前面在最上面是一个method。就在。在我们整个这个代码的最上面是一个。然后接下来，当我去创建一个新的定义的时候，它是两个M。也就是说它们共同组成了函数。能够理解吗？然后当调用？函数的时候。B要用哪个方法？最佳匹配。的方式去，谢谢！啊，所谓最佳匹配的方式去进行，就是说如果我啊。啊，如果我不做任何的，如果我不做任何的类型声明，比如说。定义FX等于X平方？呃，等价于。就如果我不做任何的类型声明的话，它其实是在说X是一个案例类型。

陈久宁(02:30:17): 都可以，是任意类型都可以。然后我下面的下面的这个有声明的情况下，它就是有具体的数据类型声明的。所以当调用函数的时候，我如果发现，唉，2.0和3.0它能够。它其实最匹配的是这个方法，那它就会调用这个方法。如果他不最匹配的方法，不是，是其他的方法，他就会调用其他的方法。所以在这个意义上。呃，二和三作为整数输入，它最匹配的方法是。上面的，因为它匹配不到下面的。能理解吗。对这个现象。啊，在就那里面叫做多重派发？就是。通过这个这个多重派发指的指的意思就是。通过所有。位置参数的输入类型。来共同决定。要调用的对象？这个就是多重派发的一个定义。

陈久宁(02:31:27): 啊，我可以先暂停一下，然后你们看看有没有问题。我觉得你们一定会遇到问题。啊，对，他一定要是位置参数，我待会可以给一个给个例子。呃，有新的问题吗？嗯，那那这个例子就是呃，这里的问的问题是F。23.0会。调用哪个方法？我们可以猜一下。没有，就是比较那个。呃呃，那我们直接问答案吧，他他他是五。它是5.0的几个手，认为这认为这个结果是5.0的几个手。认为这个结果是6.0的举个手。这这个结果肯钉钉是5.0啊！就是啊，第二个方法。它要求的是X，一定要是福建。我传的22作为整数进去他肯定不匹配。不匹配的话我肯定不会这样，他对不对？对对对。然后关于位置参数就是为什么这里我们强调一下，它一定是通过位置参数来匹配的。我们可以戳一个戳一个DHX。比如说。嗯。Y等于一。等你还进家玩？然后我再定义Y it等于。呃，你会发现的一个点在于。他这里。依然是一个。所以它其实是完全覆盖了铁。他没有创建新的方法。确实。如果。如果。所有的。位置函数。

陈久宁(02:34:01): 都相同。啊，它会会覆盖。就这个意思。所以我们可以想象一下，如果再取一个HX等于X平方的话。啊，我们会发现它

依然是一个方法。依然是一个方法，会意味着什么呢？我之前定义的。Y等于二的使用是不？他为什么会支持？是不是哪里写错了？

陈久宁(02:34:58): 哦，我这里。好像还有点小问题。啊FOOXY。这个行为和我想的好像不太一样。

陈久宁(02:36:01): 这个行为和我想的好像不太一样。我需要。后面去。思考一下。应该是我哪里？做错了。我刚才想说的是，比如说如果我们去定义了一个。定义了一个FX。XY等于X加Y的话。呃，我们正常写23它肯定是，如果我们再次重新定义。他我也相传的话。还会变成六？这个是会被覆盖的，就是说如果我的所有的位置参数是相同的情况下。他会被覆盖？然后刚才我不是很确定为什么？这里没有被覆盖，因为如果被覆盖了的话。按理来说，这个是要报错的。嗯，这个问题我。一下子突然。卡住了，我不知道该如何解答，我可以后面。找到了，然后去。再解释一下。对呃，我们再看看有没有。有没有问题？

陈久宁(02:37:40): 呃，那么就。就是多重派发这个行为大体上是了解的对吧？

陈久宁(02:37:47): 对好呃，为什么我们要把它叫做多重开发呢？呃，原因是因为它是由所有位置参数的输入来共同决定。它相比而言，有一个概念叫做。或者叫单重开发？啊，他其实就是比如说。我们来看，比如说拍摄里面会怎么做？在python里面我们可能会定义一个class就是说叫做。叫做。叫做FO。然后然后这个class里面定义了一个。叫F的函数吧，这叫F好了。然后。然后他有一个。然后我这里比如说返回。X的平方。然后我会定义，比如说。再定一个新的类型。看那是在三次方。呃，比如说我定义了一个。或者叫F。哦哦，等于。

陈久宁(02:39:25): 诶，我这里哪里又写错了。为什么会给赢啊？我哪里，我觉得我哪里又卡？卡住了。他是这个。

陈久宁(02:40:18): 呃，这个是python的一个面向对象的一个行为，对吧？

陈久宁(02:40:21): 能够理解吗？那这里的话其实啊，有一个问题在于。或者说有一个实际的一个现实。呃，情况是FFOOR它其实背后是等价于。啊啊。他们背后是完全一模一样的一个一个东西。所以我甚至实际上我可以写。这个倒是可以的。他返回的是他，第二个是FOO就是。他其实就是调用这个。对，但但是我们，我们从从正常的逻辑来，从正常的逻辑来说，FO点F它第二个呢是。For这个类型？定义了F函数。B调用的是B这个类型定义的函数。如果我们换到绝大视角。他其实就是。啊，很简单的就是。我定一个结构题。然后我去定义一个一个函数FO。然后X等于？X的平方。我再定一个结。X的三次方对于对来说就是这样表示。能够理解吗？所以它这里。为什么我们说？多重派发的原因其实就是。呃，开始了这种。呃，面向对象或者说对象方法的这种概念逻辑。放在最大的视角下，它其实就是关于第一个位置参数。所以在这个意义上，我们把它叫做单。能理解吗。嗯，需要思考一下。

陈久宁(02:42:40): 所以。所以我呃，如果如果想不清的话，我这边就直接给结论，结论就是说。修炼啊，不需要面向。的设计模式。

陈久宁(02:42:57): 因为多重派发。对，这是为什么居然没有概念的原因？因为因为我们做面向对象的这种设计模式。我们其实希望的就是通过面向对象来提供一个所谓的。派发的一个逻辑，但是这个派发的逻辑从多重派发的视角来看。它是没有太多价值。我们完完全全可以通过多重派发来实现一样的派发逻辑。当然有一个点是。呃，多重判罚或者说巨大的模式。啊，不太能够处理到好的点。就是状态机的概念。就是我们可以举个例子。比如说我们可以。这个该怎么解释呢？哦，我知道了class point啊，这叫classpoint二D吧！呃，在拍摄里面我们会。诶，这个是叫limit吗？还是叫杠杠一名？我们再定义一个方法，比如说叫move。

陈久宁(02:44:43): 然后。Move out然后我们就就直接。就直接写X等于X二？再减Y等于Y加一吧！啊，这就是一个非常简单的一个。状态题的模式。呃，为什么这么说呢？比如说我定一个定一个点。比如说。23。然后当我去写的时候。嗯，不好意思。

陈久宁(02:45:36): 呃，我我每次调用，每次调用，每次调用它，其实内部的状态都一直在改。比如说这里的。他其实已经到了七。我再再改，他又到了八。啊，这个能够理解吗？就是一个非常简单的一个就是说啊，面向对象的这种class的这种设计理念。他可以。有一个有一个数据类型是存储。

陈久宁(02:46:03): 啊，我的一个。一个系统的一个。状态。这个系统的状态可能在这个例子里面很简单，就是一个点它的XY的坐标，但是对于一个复杂系统来说，它有可能是一个，比如说一个工件，或者说一个电梯的一个，比如说它的。各种各样的状态。控制工具箱可能就会涉及到很多这样的东西，然后包括信号处理工具箱，可能也有很多这样的东西。它它不可避免的是需要存在的。那么同样的这个，比如说move up这个函数我们放在这下，我们会怎么做？呃，放在这一下，我们首先我们先先同样的先先给出一个X。啊，就比如说。啊，放在就量下，我们会，我们会这样做，就是说我们直接写一个mob。啊，然后写P？一点玩家里。

陈久宁(02:47:11): 呃，这里的意思是我每次都创建一个新的结构。这是九俩的点和？开审这种。啊，面向对象的设计模式非常大的一个差

别。所以我每次比如说。23那么我可以写MRP等于P。12等于move up？然后P三等于？他就是一直创建新的对象，一直创建新的对象啊，这个的话就是所谓的。它就涉及到一个呃，额外的一个编程范式。啊，函数是编程？与面向对象变成。你你们可以。简单描述一下这两种。模式之间的差异嘛，我们已经我已经把这个代码放在这里。对，这就是我们刚才已经演示了拍摄的这种模式和巨大的这种模式在实现，同样一个呃。系统就是系统从一个状态向另外一个状态变换的一个非常简单的一个。啊模式，一个是拍摄的模式，一个是最大的模式。你们看完之后有什么样的感觉？呃，哪个很麻烦？呃，你是只巨大的这种模式很麻烦对吧？对，这是一个，这这是一个评价就是。有俩的啊move up写起来。是不是用起来麻烦？觉得用起来，因为它要反复创建一个新的对象，逐渐循环下去，而且他还不能去覆盖它，给它删掉它。不能不能覆盖？也无法申请。

陈久宁(02:49:35): 对，这是一个很好的评价点，我们其他同学有没有其他的观评价点？就是啊，随便随便评价。就是看完之后的感觉如何？

陈久宁(02:50:11): 但是。太腼腆了。啊，没有新的评价。没有新的评价。那我就继继续自己做自己的啦！像jar的这种模式，我们一般把它叫做函数是编程。但是函数的编程并不是。并不是指随便就在里面随便写一个东西就弹出去，便是我们指的是。函数F或者说Y等于FX。只要X不变。如果。我想想。如果。Y一等于。那么。X一等于X二？有个X一等于。本人杀的话。那么。FX一它是等于？就是。基本的一个理念是这样一个理念，就是我们可能。呃，在一些非常。奇怪的。情况下，我们会说这个是还是我们可能会去争论一个东西是不是函数是变成，但是从基本。的理念来看，或者说从整体的情况来看，我们都会认为。呃，所谓的函数是编程，就是说当你的输入不变的情况下，你无论调用多少次。他都他他的结果都是不变的。所以换句话说。我第一是要用。对啊。然后。完了之后再调F。啊，效果是一样的。啊，这个怎么理解呢？

陈久宁(02:52:21): 理解的就是说。呃，我当我是写move up西医的时候。哦，我拿到是二点零五点零。当我再调用move up PE的时候，依然是2.00电影。它就像一个普通的数学函数一样。啊，能够理解吗？对相比而言。我这里。是get move up的时候。和第二次掉move up的时候，第三次掉move up的时候。内部其实有一个东西变了。但是我们无法观察到。我只我只有。去看的时候我才知道哦，他其实每次都要用他其实。结果是不一样的。能够理解这里的差异点。

陈久宁(02:53:16): 那么。哪种方式好，哪种方式不好？

陈久宁(02:53:25): 没有一个。绝对统一的一个。我们目前来说只能说是在就那下。你找函数时，编程的方式会更容易。然后如果你强行。做一个。类似于拍摄的那种编程模式，面向对象的这种编程模式，你会发现。呃，很难去。很难去14。我举个例子，如果我们要做。要要想做成类似于函数，是编程的方式，我们会怎么做我们会写成。会创建一个。就说。我会创建可能。呃，实际上。还不对。就是很多时候我们会直接。在同源的代码库里面有大量的这种写法，因为它不能要求它。如果是要一个可变的状态，这样的时候它的下一步要求就是它的数据类型也要可变。因为很有可能，比如说我的X一开始是。零点零或者说它是一个NAN，或者说它是一个。它是一个不存在的一个一个词。然后等我掉了某个函数之后，它才存在。所以。大多数时候，如果我们需要在julia里面模拟一个python的面向对象的编程模式的时候，我们就会去。

陈久宁(02:55:07): 啊，诉诸于这个这个结构。然后而且我们会要求这个是一个可变的啊，是一个。抽象类型的结构。然后这个时候我们去实现moba，然后比如说P。欢迎他二弟！啊，我们去说啊P点Y等于。然后就看屏对。然后你就会发现啊问题。就是你就会发现诶！确实可以用。对吧move up。然后你每次都都可以用，就是你确确实实可以在julia里面通过，然后。以这种方式去构造一个像开一样的体验。但是问题在哪？问题在于，这样的一种写法已经违背了我们上午说的高性能的。两个基本原则。一个是。用不可变结构体，第二个是用。具体的数。所以，所以我刚才的结论是说。在jar里面你尝试去以拍摄的思维方式去做这种状态机的话你会发现。你不得不。

陈久宁(02:56:26): 去构造一个？呃，很慢的代码实现。这个很慢的代码实现你你很难去绕开它。能理解到这里的点吗？觉得可能大家是需要思考一下。或者说有没有？

田佳辰(02:56:53): 呃，陈博士啊就是。

田佳辰(02:56:56): 呃，就是如果说就是这个结构体里面，它每一次新建的话，它计算量都特别大，就是我开发到的一个函数，它是这样，它就只能用可变的一个结构体去实现，但是它的就每次它生成一个新的结构体的话，它速度会很慢。

田佳辰(02:57:10): 就不是一个简单的这样一个它的某一个属性加一的这样的运算，它可能还有其他很多的运算，那这样的话是不是只能用

可变的结构体去做？但这样的话确实就是。

陈久宁(02:57:20): 呃，那那比如说我要问你为什么你会认为创建一个新的结构体的开销很大呢？

田佳辰(02:57:31): 呃，就是可能。呃，它的就是其他的一些属性，就是它每一次新建的时候就是会要，呃，就是一个滤波器组的一个结构嘛，它要计算很多一个滤波器的一些参数嘛，每次新建都要去重新算，但是它实际上的话，它只是其中的一个属性变了，它不需要去。

田佳辰(02:57:49): 去再去计算那些，就是可能它那个属性的话是不影响它的。对，我去组织结构的。

陈久宁(02:57:56): 呃，我明白你的点在哪？啊，我来做一个例子吧！它它这里说的点在于，如果我们去构造一个，比如说point三D。啊，我们可能会有一个XY，我们可能会有一个。比如说叫。啊origin。可能会有一个这样的属性，就是说离原点的距离。我们可能会希望在这个结构体里面，我们去起到这个属性。所以当我们去去定义point的三，比如做三D。这样的一个。我们去定义一个构造函数，说这个构造函数其实就是。XY然后第三个属性我们是需要构造出来的。是算出来的，它其实就是一个。X的平方加Y的平方。还更好对吧？然后田嘉诚刚才提到问题是一个非常典型的一个问题，就是我如果每次去。比如说构造一个结构体的时候，它都需要去算这个属性的话。我。从性能的角度来考虑。我不应该。我不应该让他每次去构造一个新的结构体，因为他就每次都会去算很多东西，然后算到他这些东西，其实我一直都没有变过。啊，你要说的地点是这个点对吧？

田佳辰(02:59:28): 呃，对，就是它呃就可能和这个还不太一样，就是它那个要计算的属性和其中的某几个参数是没有关系的。

田佳辰(02:59:37): 就说我，我要变的参数的话可能是。不需要去额外计算。嗯，是这个意思。

陈久宁(02:59:44): 啊对，但但我的，但但是你你要说的点其实就是这个点，就是比如说这里啊，当然这里啊，我们也可以。啊，我这里可能。写写的太狠。太随意了，确实。那那那我就是比如说我们就。

田佳辰(03:00:09): 呃，就比如说有可能再加一个the的参数，我要算的是。

陈久宁(03:00:10): 对啊对啊就。对，我比如说我们就我们就就写一个这样的，然后我们写一个。啊，这写一个函数，然后。三D然后等于？呃，这个叫啥叫叫叫叫呃？C加等于。这大概这个影响。对啊，这就是信号库目前一个最典型的一个代码写法。然后关于这个的话。呃，说实话呃。它有两个，有两个因素是需要考虑在里面的。一个因素是？我们是否真的需要按照？Matlab的api去设计？去设计用户使用体验。啊，这是一个因素要考虑的第二个因素是。啊我们。哪些。真的需要放在？这是第二个，第二个因素要考虑。

陈久宁(03:01:42): 呃，我说的是。啊，第一个因素的话可能和整体的一个。就是比如说这个工具箱的设计者。下构思是决定。他排版的说我们要做的一模一样。那我们就按照这种方式去做？没有任何。可以讨论的余地就是按照这个去做，没有任何可以讨论的余地，但如果说我们要做一个。原生的高性能的东西的话。我们肯定不是走。那么在这种情况下，我们怎么样去解决这个问题，解决这个问题的思路就在于我们这个这个属性它真的要放进来吗？它完全不需要放进来。我完全可以去写一个。这样的一个函数，然后先算。去去算去算这个属性的词就可以了。对，我直接我直接去写这个函数就行，我为什么一定要在构造函数构造这个结构体的时候去去把它算出来呢？因为你因为比如说。你你相信你确定用户一定要这个属性吗？如果用户不需要这个属性的时候，你为什么需要去算它呢？我们肯定是希望在用户需要这个属性的时候才去算它，对不对？

陈久宁(03:03:16): 呃，不知道田嘉诚你有没有理解我这边？

田佳辰(03:03:22): 呃，我理解嘛，就是这个确实。呃，怎么说呢？这可能就是一个方便用户的一个设计吧，因为这是一个类比较基础的一个就是。嗯，结构吧，很多的话就是可能其他函数调用它的话。

田佳辰(03:03:38): 就不需要用户再去，就是比如说。那他要算一个实物的东西，我给他一步就弄出来，是这个样子。谢谢陈博士！

陈久宁(03:03:46): 没有我，我的点在于你们，你们可能会认为这个是方便用户。呃，但是从我的视角来看，它并没有真的方便用户，我完完全全可以通过一段函数去做到同样的功能，你们认为的方便用户是因为你们要要求。呃，按照matlab的API去设计。如果我抛开matlab的API，我不认为这个东西是方便用户的。

田佳辰(03:04:19): 嗯，了解了解。

陈久宁(03:04:20): 对啊。所以啊，我们这边也就说到了就是。在状态期这个概念下，我们有两种典型的实现方式。一种是按照。函数是编

程的方式去实现，函数是编程方式的话去实现，就是我的函数的每一次输入和输出都是一个独立的。啊，不可变结构体，这个不可变结构体是我的状态。从旧的状态变成新的状态。它是我函数的输入和输出，然后还有一种模式是我们python的这种类的设计模式。他就是说。啊，所有的状态存在同一个数据结构里面，然后我通过。修改这个数据结构的。去修改它的状态。呃，从一个小系统的角度来说。嗯，这个怎么评价呢？

陈久宁(03:05:26): 我目前没有办法去判断。哪种方式在超大规模系统上？能够做到更好！因为我知道的是。这种模式在九两下。啊，在超大规模系统上它一定。快不起来。然后。但我也知道的是的这种。可变结构体的这种模式。在。

陈久宁(03:06:00): UI绘图这边其实用的是非常非常多的。就比如说我去做UI绘图，我去修改它一个状态量的时候我去触发一个。新的。啊绘图指定？就是说比如说我修改了。比如说我有一个。比如说我定义一个。我说是我们的长宽。那么我我非常希望的是，比如说我有一个。啊600乘八百？这样的一个屏幕。帮我去说我修改。修改这个啊，屏屏幕的长。为靠高为，比如说1.24。我希望的是，当我去敲下回车的时候，我的整个屏显示屏自动的拉伸。会有一个这样的一个响应式的一个编程模式的。这个其实是我们。啊开审或者说。啊，面向对象的这种。状态期模式下。最需要的呃，更需要的一个点，它其实是一个set和get的一个功能。这种响应式，响应式的一个编程模式。在举了下能不能做到呢？这就是。

陈久宁(03:07:16): 这其实也是我，我一直不认为。啊信号库这边的代码设计做对了的原因。就是当我们直接去说，P点就是X等于多少多少的时候，我们其实并没有去。真的把这个状态机的响应。和更新给做进去，我们我们只是写了PDS等于六。等于我们并没有去说。当PDX等于五的情况下，我后续需要做什么样的事情？这件事情当然也可以做，我可以去定义一个，比如说叫做。啊，然后PD。然后如果不等于等于。X那么我我就。我就说PDMPDX等于X。然后我去，我去钓。我去执行一个，比如说。呃PDX exchange。比如说我执行一些其他的额外的一些动作。然后。如果。啊，如果是Y的修改的话，PDY等于Y。然后这里比如说。这里我们就直接报错好了。那在这种情况下，我确实可以写P减X等于三。我我这里确实可以写平诶啊哦，不好意思，我我给你错了。还有一个是方。

陈久宁(03:09:49): 对，我确实可以通过这种方式去做到既修改又触发新的啊，新的事件，但是你就会发现啊，所有做做的这些事情，它都是在jar里面去模拟一个其他语言的特性。

陈久宁(03:10:11): 这件事情它其实并不是高效的东西。然后我。我的观点是？他能做，但是做不大也做不好。然后我非常。呃，我其实个人是非常希望我有时间去把这个问题给处理掉的。因为因为如果我只是在这里一直讲的话。哦，我知道很多人是不相信，因为确实目前的信号库。就是围绕的这种设计模式去做的。这一点我也是知道。然后呃，关于这个我可能就评价到这里。可以先看一下其他人有没有更多的问题，如果有更多的问题我们可以继续讨论。特别想说我感觉我看到你那写的是。呃P一等于二P，然后P二等于二P。可以直接写P等于P。啊，这个是啊，那你说你说的这个点其实是。并不影响。嗯，我我我来。我来复述一下。呃，这里说的点是甚至。呃，你刚才说的是P啊，比如说P等于move。P然后P等于RP你可以，我们可以一直这样做对吧，但是它并不影响。他他并不改变，背后有一个新的结构体被创建这个事实。都叫P，只是变量名变了。就是呃，回到我们上午我们上午说的。它只是一个变量，从一个对象指向了另外一个对象，但是并不改变，有一个新的结构里被创建。啊，看看有没有新的问题？

陈久宁(03:12:41): 呃，我可以补充评论一个就是在这里面。函数是编程是主流。

陈久宁(03:12:52): 这点是一个就是啊，百分就是整个九大社区写的代码里面可能70%以上的代码。都是属于函数，是编程的风格。呃，函数的编程风格它有一个好处在于。因为呃，这个可能很难。很难给出一个具体的例子去说，但是。啊，我我们可以，我们可以直接上这样去得到这样一个结论，就是隐藏的状态量越多。越南。真正的问题存在。啊，这这是一个函数，函数是编程的一个优势，或者说啊，这是。啊，状态及拍摄状态期模式的一个劣势。因为如果你你设想一个设想一下，你在一个你在处理一个特别大的一个复杂系统。你并不知道。每一个组件里面有没有状态？在这种情况下，你如果是对蛋白做做第八个，你可能很难知道。里面某一个状态下发生了变化，你可能很很难把这个系统给浮现出来。那么从函数是编程的角度来说，我可以非常容易的去把这个系统给浮现出来。然后非常容易的去找到究竟是哪一个？导致了。我的行为跟预期的不一样。所以这个debug的难度也是呃，函数是编程的一个。优点。关于这个可能评价就暂且做到这里就是说。

陈久宁(03:14:51): 啊，围绕的多重我们是？或者说吉利亚的这种主主要的编程模式就是多重开发。和函数是编程？然后就那里面没有拍摄的？面向对象的这种class的设计模式。然后就在里面，你可以通过mutable struct去模拟一个状态机。这个东西。存在一定是有使用场景的

，但是我目前。我个人不认为。啊，我们信号库？应该按照这种场景去使用。大概是这样一个结论。

陈久宁(03:15:40): 呃，有新的问题吗?

陈久宁(03:15:55): 呃，如果没有新的问题，那么我来抛一个新的问题。就是。红我我们来猜测一下。从性能的角度来说，F。X加Y。哦，我把它重新写一下吧！对，就是我们有两个函数，一个是FX，一个是GAGA。GXY它们唯一的差别在于一个有类型标题，一个没有类型标题。那么我要问的是，从性能的角度来说。和G12。他们。在性能上有差别吗？我们可以猜测一下。呃，认为他们有性能差别的。举个手。就基本上所有人都认为。虚会更快对吧？因为车有车已经非常明确的说了它是X五啊，它是int类型。但是如果我们实实际上去看这个生成的代码的话。你会发你会发现？它是一模一样。毫无差别。对，为什么。呃，背后。背后其实是因为就这个也是julia的一个非常强大的一个特性，叫做。半型编程。

陈久宁(03:17:59): 啊，之所以我提这个问题，也是因为。呃，我们前面说了具体类型，然后类型稳定，说了一大堆，然后包括我说呃，只要我们按照C的编程风格去写。你肯定在慢慢起来，大概会说了很多东西，然后我我也在之前做九大培训的时候，我也发现有很多人。就会得到一个结论，说我就把所有的。所有的东西全部。全部写写完就是int然后int到底。就比如说，比如说C等于。就就我我我知道有的人他就会想想想把这个的东西全部给出类型描写。然后。然后这就会带来一个问题就是。你如果所有的东西都给类型标题的话。我有的。都给类型标？呃，它会带来两个后果，第一个后果丝。呃，他不再是。一个。如果用python的说法叫做。鸭子类型。就是我们，我们可以想象一下，如果想象一下你写的一个算法。

陈久宁(03:19:15): 你你说这个东西一定要是。一定要是X加Y的话，那么用户他一定会说。诶，我能不能写一个二？就是用用户说诶，我我敲一个23是ok的，我能不能敲一个浮点数的20和三？然后发现不能不能用。然后用户就会说哎，这个写的不好。对，这就是一个，它不再是一个大大产品或者说它。知识的数据。输入表现。有限他他就不在是啊，不在。用户友好。这是所有类型，所有变量都给类型标记的一个坏的点。好的点在于。啊，我们把它叫做。好的点在于。好，我们非常清楚。自己在处理怎么样的？然后。可以更容易的。的把代码写对。这个能够理解吗？对，所以。在这样的一个前提下，我才问大家，刚才我们说F。Int等于X加Y和BHY。我我才问到大家F和狙会不会存在性的差异？呃，答案一定是不存在性能差异。

陈久宁(03:21:01): 然后然后写下来我们去解释为什么不存在新的差异，在这里面我们刚才在讲到多重派发的时候，其实有有一个概念。我没有去介绍。它实际上是说在这里面我们的函数。或者说。我们。分分为三个。三个城区。分为呃存在三个概念。一个是叫function？一个是叫。呃，第三个呃待会再说我们，我们就是比如说我去定义。FOOX等于X平方，然后FOX。等于X的三次方。在这，在这个意义上，我们就说我们定义了一个函数FOO。然后这个函数存在两个方法。对吧。所以函数就是说。所有方法的集合！嗯，所有。同名方法的集合大概是这个意思。然后方法他方法是？这个该怎么定义？方法就是方法，方法就是源代码。大概就是我们写。他其实。大大概是这一种，我我真不知道该怎么样给方法再给一个定义了。但但是就是意思就是说，当我们去写这个的时候，我们其实创建的是一个新的方法。然后当我们写下面这一行定义的时候，我们创建了第二个新的方法。然后九俩实际上还存在第三个概念叫做method。Instance. 啊方法学历？

陈久宁(03:22:56): 这就是一个啊！呃，大家为什么会认为？之前F和G有性能差异的一个原因。就是因为它其实是有所谓的方法，实例的概念方法，实例的概念概念，我们可以做一个简单的对比。呃源代码或者说方法呃，我我我真的不知道方法该怎么定义了这个方法它其实是一个C加加模板。模板实现。啊，如果用过C加加的话。啊可以。按照这种模式去理解。呃，说它是模板实现是什么意思呢？是说？呃，我们来举一个例子啊！

陈久宁(03:24:16): 呃，这个包是可以帮助我们去快速的去检索。或者是说去分析方法实际的创建，呃，我们可以随便创建一个函数就叫。比如说my son。就叫啊，比如说。想想什么我写。FXYFXY等于X平方加Y平方。嗯。当我们敲下这一行代码的时候，我们创建了一个方法。呃，这个方法就是一个。一段元代嘛？我们可以简单这么理解。然后叙利亚有一个特性，昨天其实大家也提到了就是。九的函数第一次执行的时候很慢。对吧。呃，这个大家知道吗？知道这个知道这个现象吗？对，就是。对了，它的函数第一次执行很慢，比如说32我们可以做一个碳。我找的例子太。太过于简单了！没有办法复。但但是这个这个现象可能大家是知道的，所以我就不不多演示，呃，他之所以会出现。函数第一次执行很慢的原因。

陈久宁(03:26:01): 在于。第一次执行的时候。这个生成的过程，其实就是编译的过程。所以通过我们可以通过呃通过这个。来诊断分析这

个现象。当我去定义FXY等于这个的时候，我们可以说method instances。呃，它不存在任何的方法实例。然后当我们是第一次调用完调用这个函数之后。我们去看a，他多了一个。你可以看见它，它其实创建了一个叫做iphone intel的东西。但我们可以类似的去定义int Y等于X平方。加Y平方。那么G12。你会发现它生成的也是同样的方法识别。所以在这个意义上，从方法实例的角度，或者说从编译后的代码的结果的角度来说。F和Y IT iphone和G其实是没有任何差异的。因为我们最终执行的东西是编译后的这一段。所以他们不会有任何性能的差异，差异能够理解吗？首次执行慢。背后其实原因就是因为。啊，我们有。我们涉及到了方法实例的编译。或者说方法实例的创建。然后这里像比如说我们如果如果。如果传一个新的数据类型2.0和3.0进去。他又会编一个新的发。方法识别。然后传一个呃三进去，它又是一个新的。就是在这个意义上，有俩能够把泛型和性能结合的很好。它就是通过模板。模板的这种方式去做的，它其实背后背后很简单。

陈久宁(03:28:21): 呃，背后的背后的思路就非常相似的，就是它其实背后就是当你去创建一个方法实例的时候。呃，你其实就是在把。你就是在创建？一大堆的诶？等于X加Y，然后iphone。啊，然后弗洛特要是。等于X加Y然后？大概我们就是创建了一大堆这样的。干了实现。

陈久宁(03:28:51): 如果是就是如果是C语言的话，我们其实就是干了这样的一件事情，但但是我们写这样的很写一大堆这样的东西其实是很蠢的嘛，对吧？所以我们。从所以谁加加他有一个所谓的。哦，我忘记具体到具体写法了，好像就是叫弹出来。

陈久宁(03:29:12): 安排的T然后什么？Function. 是叫方形吗？我都忘记是不是叫方形啊？嗯！这是什么。EF然后XT。也是。啊，大概是一个。如果是C加加的话，大概是一个这样的东西。提提然后。啊T等于。啊，好像是我，我忘记具体。怎么个写法了，但是啊C加加其实就是这样的一个模板，就是通过模板的话，我们可以批量的生成一大堆。代码实现。然后这一大堆代码实现每一个本身都是非常高效的。所以这一对这一大堆代码实现和我们开始问，大家说FXY。和GXY。之间是否存在差异，呃，结论是不存在差异，因为他们最终都是。呃，非常具体的一个版本。呃，说到这里大家能跟上吗？就我这边说描述的非常的。模糊或者说简洁，因为它涉及到。修改的一些非常底层的机制，然后这些机制。不是。三言两语就能说，说完，或者说说准确的。所以我这边说的非常模糊。但是大概背后的意识不知道大家能不能理解到。啊，问这个地方不需要？对啊，对，这是一个很好的点，就这个结论我我时不时的我会提就是说。

陈久宁(03:31:13): 呃，具体数值类型它只在。两个地方。呃，一个类，一个地方是结构体的定义。第二个地方是啊！大概是容器的地。东西的使用，比如说。大的比方。结构体的定义就是说第二个，然后。然后里面是必须要的这种写法。然后。在方法上，在方法定义上。数据类型。

陈久宁(03:32:08): 数据类型。多重派发。意思是说。如果说我们存在两种算法，我举个例子，如果我们存在两种算法。我换一个地址吧！对，我先把结论写一下，就是结论就是说啊，数据类型就是。补充数据类型。呃，只用于多重派发的目的，在绝大部分情况下，这一点都是真的。嗯。他其实就引入了一个新的。呃，鉴定是性能优化。代码实践。呃，我们来举一个例子就是。叫比如说，比如说我们有一个。我们有一个矩阵，就是对角矩阵。Using linear. 说123。啊，它确实存在，那比如说我，我创建一个长度为。100的对角矩阵。对吧。那么。我如果要对这个长度为100，对小时计算进行求和。啊，大概就是。这个样子。

陈久宁(03:33:41): 那要问的一个问题是。我。有没有更快的方法？就我怎么样去让这个对角矩阵的求和？上的更快。对，是算对角线。呃，那么我怎么样去判断一个矩阵是对角矩阵？就是就是我们肯定是想要写出一个通用的。通用的求和求和。函数对不对？我们会希望这个通用的求和函数。如果输入是一个对角矩阵，我就只算对角线。他绝对非常快，如果他不是对角矩阵，我就我就全部一个一个老老实实算。对吧。那么这里就有一个点了，就是。啊，我们需要写一个。我们需要写一个函数，假设说。A然后正常情况下就是比如说等肌肉。然后SM D。这个是啊，今天上午？给的那个最快的版本，正常情况下，我们就是写一个这样的东西，那么。我们怎么样引入？对甲级战这个感觉。呃，我可以给一个提示。

陈久宁(03:36:04): 在这里面我们是通过一个非常非常典型的一个。优化问题的思路是我们引入一个新的结构。啊，居然已经有了所谓的对角矩阵的结构体。叫做diagonal最低到100。这就是一个对角矩阵。但是它的类型不是叫array，它是叫diagonal。就是所谓的对角矩阵的一个数据类型。我们要做的事情就是。可能变一个。就说a等于a的。A的对角线的元素其实就是。所以买上。这个事情做起来也非常容易。

陈久宁(03:36:56): 等于a加A？好，完了结束。能理解吗。我们可以来验证一下。带你去马克？又是谁零零？然后我们先我们先给出第一个，第一个函数就是。啊，通用的含通用的实现。然后我们刚才说。对角线这个东西。啊，比如说如果a等于。好了，BGM一到200的话。

陈久宁(03:37:49): 啊，它的项目是很慢的啊，我们可以。它是一个575纳秒的一个时间。然后九两内置的求和是737纳秒的一个时间。然后

哦，刚才说了。我们我们很清楚，知道他是一个对角的。但是。呃，这一点我们需要，首先我们需要把它表达为一个对角计算的数据类型，它不能是一个。不能是一个类型，我们必须要表表达成一个最小矩阵的数据类型，所以它就要达到更多。

陈久宁(03:38:34): 它是一个对角矩阵的数据类型，那么在这种情况下。啊，它还是慢的。它还是慢的原因是因为它依然是调用了之前那个通用的方式。我还没有我我我现在还没有把下面这个加进去。我把下面这个。

陈久宁(03:38:57): 实现只要加进去之后。它就会变快。能搞定一些吗？呃，就是下面这一段代码可以看懂吗？就是一个函数调用了函数自己。他其实背后发生的事情是我先。就是当我第一次去调的时候，我先去把它先先走了这条路。然后当我把a点D加AG给取出来的时候，诶他一看。它好像匹配不到下面的方法。匹配不到下面的方法的时候，它就会匹配上面的方法。然后去调用上面的实现。所以我们就会发现。我们去优化一个对角线的对角线矩阵的求和，我们只需要多写一行代码，你就够了。然后。哦，可以去想象一下这件事情在。或者麦太不下。会会做成什么样子？简单点说就是几乎做不了。所以拍摄和mat他们没有办法做到提示的性能。

陈久宁(03:40:18): 那么就可以做到体制的性能就有一个前提，前提是我们要。啊，围绕结构体。和多重派方？去设计蛋。然后进行。的性能优化。这个是Q里面一个非常非常典型的故事。所谓的渐进式的性能优化就是说。呃，我的麦上其实从第一天开始我就能用，只是他不够快而已。然后第二天我加了一个新的方法实现，让它变得更快。这就是所谓的间。就我我。就从第一，就从设计之初就已经考虑到了这一点，他不要求我们在在代码实现的第一天就把所有事情做到完美。我们可以根据自己的需求，然后。补充一些特定的优化。然后这些特定优化的补充它其实啊，不需要你对整个系统进行大规模的重构。我觉得可以到这里先稍微暂停一下，比如说我们休息一下，然后如果有问题的话。

陈久宁(03:59:14): 呃，我们继续吧！呃，关于刚才讲通过多重派发的这种设计模式，大家有没有问题？可以，或者说有没有听完之后有一些观点或者说看法可以抛出来讨论讨论。

陈久宁(03:59:43): 没有一个没有任何一个人有想法啊！打二个呢，对，就是要非常熟，你要知道它有哪些？不一定。就是说，如果你知道的话，你可以节约一定的代码工程量。但是如果你不知道的话，你自己写一个其实。

陈久宁(04:00:14): 也不难的，我可以跟你说打这个函数的实现。大概就100行不到。对一就是julia的所有的julia的内置代码。库都是非常简单的。他可能需要一定的。呃，这次怎么说呢？我觉得我觉得我讲完就比如说。昨天包括今天讲完了这些东西。想完之后，其实你就有能力去阅读巨大的代码库。当然。呃，可能会遇到一些呃，比较旮旯的点，他用到了一些特殊的语法，但是大多数。

陈久宁(04:00:53): Ja代码库就是我，我说的是ja标准库的实现，它就是用纯句来写的。所以你阅读起来就像看。更和其他人写的就爱代码是一模一样的，就像你看我的写的就爱代码是一样的。我们。我可以先留一个点就是。32个呢。实现的解析。我们待会有时间的话我，我给大家演示一下，打到更多这个数据类型会怎么设计？

陈久宁(04:01:24): 呃，有啊，有嗯，你说的是这个东西吗？还是说？就是，比如说你去看官方文档，然后。你去看。不过。这里也有个点，在于它是一个英文文档。所以如果比如说如果你关心线线性代数。他就你就去看linear algebra的文档，然后它里面会有，会有一大堆，然后慢慢看。大体大体上就知道他有什么了。在这里的话，大家可能。对，你可以看见。还有。它有非常多，不只是对角元素，它还有双对角。

陈久宁(04:02:21): 然后三对角？啊，各种各样的东西。就是啊，我们可以，你不需要，不需要，因为你不知道这些特殊的对象是否存在，然后觉得你没法去用多重派发或者怎么样，因为刚才刚才我们去讲这个例子。背后的核心思路就是。如果你知道一个。一段数据的特特征是什么？像这个例子，刚才的例子里面是我们知道这段数据的特征是只有对角线有有元素。那么我们就可以去想着围绕这个特征去构造一个结构体。然后我们就可以把它通过都从派发的方式接进去。对，就是它的因果关系其实是跟你们想象的，是反过来的，不是因为你知道了diagonal的存在才去。

陈久宁(04:03:25): 才去用多重派发加速。代码，而是因为。我们知道这段数据它有一个非常强的特征，所以我们围绕这个特征。就够去设计一个结构，然后再接入多层排法。然后只是恰好。呃，我举的这个例子里面。打二个呢是存在的。所以。好，我们就做做出来了。像那个信号库这边做的一个。呃，信号库这边做了一个代码实践，它背后就大量的使用了多重派发。然后大概就是300行的就料袋嘛！比C代码快。快1000倍实际上现在做到现在已经。如果加上GPU的话，可以快到10万倍。这个东西背后就是通过多重开发去做的。如果有兴趣可以在媒体上可以去搜。然后不过这个代码。信号库的代码目前在我们内部。啊，是私有的，所以不是一个。不是一个公开大码库？

陈久宁(04:04:47): 但你们可以去。可以大概的看，比如说我们是怎么样去优化的，然后最后我们可以看到一个。

陈久宁(04:04:55): 呃，可以看到一个性能差异吧，就是说同一个代码实现他们在。就是优化前和优化后，它会存在一个131微秒和两微秒的一个性能差异，就同样是用U来去写。它，它会存在20倍的性能差异。然后在20倍的性能差异。跟卖菜的不比还要更快。

陈久宁(04:05:16): 但但这些背后其实都是围绕啊，比如说围绕一个结构体。然后设计成一个更好的结构体。之类之类的。之前旧的版本就是用了可变结构进行做。然后我去优化之后呢，我先把。可变结合体变成了一个不可变的结合体。然后再把一些不需要的。不需要，就是比如说这种都是属于啊状态量之类的东西，他们其实呃就像刚才呃填。呃，田家诚问呃信号库这里面有很多很多属性，那我我我第一个问题就是。这样的属性它真的需要吗？我所以我我我们就可以把这个属性。不必要的属性都是可以移除的。然后当我们把结构体压缩到一个只有一个属性的时候，然后所有的问题都变得非常简单，然后它背后就是通过多重派发的方式去定义。呃，加减乘除四则运算，然后把整个呃所谓的卡上的数值计算给建立起来了。那么它做完之后它就会比啊，买下来的C代码还要。还要快。这背后呃我我我举这个例子，其实并不希望大家能够。完全get到这个设计点，因为它里面。呃，是一个非常综合性的设计，实际上。实际上这个这个例子做出来到现在同源可能。能理解他的。他为什么好？也只有除了我以外，可能也只有另外一个人。

陈久宁(04:06:55): 其他所有人都没有真正get到自己的点。但反正就是呃这个结论或者说这个东西已经做出来，在这里，那么大家就知道。呃，修改代码可以做的比C快，而且还要快，非常非常多。然后这背后的一个非常核心的原则就是我们是围绕一个。呃，九俩风格去设计代码。就是通过多重派发，通过呃不可变的结构体这样的一些基本原则，然后把它组合起来。呃，我们看看有没有新的别的问题。使用不锁定结构就是自动进行优化。让我们先暂时先了解一下。从同源目前的代码库来说。绝大部分人都没有真正的去利用这一。利用这俩的这个优势去做新的优化。目前。宏源大部分代码库做的新的优化都是。把这件事情做好，就是说把通用事先做好。他并没有去给出特定的。优化实现。但是呃。Julia的一个更大的优势就是相比于C语言和C加加语言来说，就是我们把第一步就是把这个。呃，475到481写好我们能够达到什么程度，能够达到跟C和C加加性能相当的程度？

陈久宁(04:08:57): 但是子和C和C加压性能相当，它并不是。的全部优势，最大的全部优势其实是体现在。484号。他能够。在一个极致的性能的情况下。通过传递更多的。类型信息来做出更激进的优化。

陈久宁(04:09:18): 这个是就俩真正的呃优势。啊同源目前我看到的。啊。可以说是所有代码库。都没有把这一点利用上。所以如果你们有一个人在日常的代码实践中把这个东西用上去，我会觉得诶非常开心。对，因为他真正的发挥了质量的优势。但是呃，如果你们只是把第一步。做好其实。已经算是做的很不错了。对，因为单单把第一步做好。呃，就是就已经。从我的评价来看，就已经是一个中级程序员的水平。对，如果你第一步没有做好啊，你，你可能就是一个初级的军团开发者。对你，你第一步做好了，就第一个函数实现的足够高效的就是今天上午的那张图里面，如果你从左下角的那个三个乌龟变成了左上角的那个火箭，那么你就是从一个初级的巨大开发者变成了一个中级的巨大开发者。

陈久宁(04:10:25): 然后从一个中中级的开发者变成一个高级的开发者，其实就体现在。

陈久宁(04:10:31): 比如说你能不能围绕多重派发？去进一步的把性能给挤压出来。大概大概是这个样子，当然我不能说的这么绝对的对。

陈久宁(04:10:46): 不能把这一行代码再讲。在这块呢，运用具体数据。降的比较快，具体数据是怎么样转化的？嗯，这里呃很很简单，就是比如说我们知道。呃，AA是一个我们创建了a是一个类型对吧？呃a里面我我通过tab键我可以补全，我知道通过补全我就知道诶，我可以把这个。这个东西给取出来。取这个东西出来本身是很快的，然后取出来对它求和其实也是很快的，对吧？或者说对他通过my sum求和。呃，这个东西必。这件事情是非常非常快的。啊57纳。然后那那我就，我就可以通过多重派发的方式，我是比如说我可以，我当然可以可以写一个就是叫。呃my son diagonal。然后AA等于？My son a点DIA我当然可以写一个这样的东西啊！呃，它也它也是它的性能其实是一样的嘛，对吧，能理解吗？

陈久宁(04:12:09): 呃，这个这个跟那个可能没有关系。呃，但反正就是说我我其实呃如果说最简单的写法，我就是创建一个新的函数，然后这个新的函数去调用函数，那么实际上我呃，我们并不需要去创建这个新的函数，我们实际上就是。就是用上，然后但是去一个通过多重派发的形式去去说。等于买算。呃，这边的话就会，我们可以把它进一步的拆开。就是说如果不知道该怎么？这样的话就说我们可以在这里写写一条日志。嗯，比如说就开报A。然后。Return my S打a点压？然后我们前面的？我们这里这里套一个吧！呃，秀type of A。呃，我我我就是给两给两个方法都插了一个袖，然后等于是我们来辅助我们的。其实啊，你就会发现它第一步先调用了diagonal这个东西。然

后其实也就是哦，我们我们可以，我们可以再把这个东西再再再。再补的更多！A print line?

陈久宁(04:14:16): 对，你就会发现，我们首先先调用了一个。打个更多特有的方法。

陈久宁(04:14:21): 然后他他传进来的a是一个达到的类型。然后我把这个特有的方法的a点DIAG给取出来。取出来的时候。呃，我们我们要注意到的是，呃多通派发它不会匹配到这个特有方法了。因为它不再是它其实是一个。感觉类型他反正不是。我们也可以不关心unit range是什么，他反正就是一个项项，然后这个项肯定不是达到更多类型。所以它匹配不到这个方法就是。在在我们调用MSADIAG的时候，它绝对不会匹配到的方法，能理解吗？所以他只会去匹配这个通用的方法。所以你就会发现，诶它有一个类似于递归调用的一个模式。呃，这样解释能够理解了吗？就通过这种低调的方法，就可以让这个变得更快。它它变得更快，不是通过自己归调用的方式去做的，变得更快，是因为。因为我们只算了对角线的元素。就是变得更快，就比如说。啊，比如说我，我我没有地位要用，我就正常写一个my diagonal。他也他也很快。对，它是因为我们只对对角线的元素进行了求和。

陈久宁(04:15:54): 这就是一开始我们抛出那个问题的时候就是。一个对角矩阵我们怎么样？对它进行更快速的求和。这个问题的一个初衷。就递归调用，只是为了让。所有的用法能够统一到一个函数名上。就我们可以。举个例子。就我们刚才呃围绕这个东西做了一大堆的设计。实际上。或者说做了一大堆的解释，实际上这个概念。九两内置的其实已经做完了，我举个例子啊，a等于。比如说大，然后这种呃一到100。啊，这是一个对角线矩阵对吧，打个更多类型，我们可以通过collect把它写成B。

陈久宁(04:16:51): 呃，注意到B是一个类型，它是一个草民举证。我们可以通过benchmark two我们来看。对，你会发现它就是有性能差异，其实的内置内是沙米已经做了这样的多重排。我刚才啊，只是。一个。非常小的例子告诉大家，唉，我们可以通过多重派发构造出一个统一的API接口。

陈久宁(04:17:25): 然后这个API接口根据不同的数据类型的输入，我们可以调用不同的优化实现。我们类似的还有比如说除了对角对角矩阵是一种矩阵类型以外，我们还有比如说上三角还有，比如说啊。新数据证，比如说using spars array，然后S PR。0.66。我我不太确定是是是这样写的，那是。比如说0.2。对这个样子，然后我们可以这是一个系数矩阵，然后我们也可以把这个系数矩阵啊。

陈久宁(04:18:06): 啊，吸收就是说。啊，这个地方要么没有值，要么为零啊，要么有值，要么为零吗？然后我们也可以去看到，就是比如说。这里的这个情况下，a是一个吸收矩阵，它是247，然后上B。可以看见它就是有性能差异，然后我们这个例子里面。可能系数性还不够，如果比如说0.01。我们哎呀，这个问题可能就立刻就清楚了。这个现象差距可能立刻就出来了。

陈久宁(04:18:43): 啊，他好像啊，依然没有出来。但但是呃就是没有到那种非常夸张的性能差异，但是这个性能差异确实存在。就是。这也是julie的一个非常好的点，就是我们可以围绕多重派发去设计一个统一的。这个统一的函数名，它可以按照你的。啊，去工作就是它，它能够适用于各种不同的情况。就它是一个？啊啊。它是一个。换行编程的模式就是说你不同的数据输入。反正只要他长得像急症，他就他都能用，不管他具体是什么样的急症。然后，但它好的点在于，如果它是一个特殊的矩阵，如果我恰好知道这样的特殊矩阵如何优化。我就可以把它贴进来。这样这样说能够理解。

陈久宁(04:20:02): 就大家大家是？思考的样子是指没懂还是？可以理解。啊，这是有一种是？他自从开发带来的这个性能的优化是他那个优化的就是他那个。比如说对于这个一点，大概它这个优化的方式带来的优化，而不是不能再发这个事情。对对对，没错啊，这里出现性能优化是因为我们知道只算对角线。它一定更快！因为我要加的东西少了嘛，对吧，所以肯定更快。原来原来我要加一个，比如说100乘100的矩阵我要加1万个元素，现在我只要加100个元素，我肯定。然后多重派发在这里面，完完全全是为了构造一个统一的编程接口。

陈久宁(04:21:10): 呃，有没有其他的理解？就不，不用担心，说错就是。其实就是抛出来，然后我们可以进一步解释讨论。

陈久宁(04:21:35): 如果。呃，没有更多问题的话，我们可以下面演示一个。介绍一下打的类型是怎么样设计？

陈久宁(04:21:47): 不过在在介绍他之前，我们先介绍另外一个。封信叫做。就是参数化类型。汉书画类型。他是一个或者叫。参数化结构体吧？他说的事情是，我们前面是说了我们一个结构体最好定义成。

陈久宁(04:22:15): 啊，比如说X不是？对，我们前面说了就是上午上午讲的是我们的结构体，最好定义成是一个具体类型，对不对？这样的话性能才快。但是。呃，那就有一个问题了，我们其实。比如说X等于我们，我们可能比如说我们有一个。啊，我们，我们希望它是一个int类型的，我们希望它是就希望它是一个质类型的。但它都是首尾的rector。对吧。这件事情是如何做到？

陈久宁(04:22:50): 就是我我我一个结构体它都叫vector。但是他。他一会儿是类型的，一会儿是64类型的vector。然后这种情况下。他和我们去定一个vector F64。比如说我们定一个站的站的，比如说这一场。它会不会存在一个性能差异？这个就是啊，参数化结构题要要处理的事情。我们这边叫。然后实际上我们如果要把它。我们希望就是我们希望这个啊现状。现状是我们有一个的类型。我们希望让他能够。能够接受。这种。这种类型。啊，能够理解就是我们要做的事情。就是我希望我希望我，我不要局限在弗洛特。64，应该局限局限在弗洛斯64。啊，也许。对于当前的例子来说是ok的。但是我们肯定能够找到很多例子。我们希望它是一个非常通用的一个。所以。这件事情是通过所谓的。啊，我想二。来做的。啊，这样就可以。阿迪比如说。不知道他要做什么？然后23。呃，这样的话我们就创建了一个叫做。呃，结合体的元素类型依然是四类型，但是它又是一个。

陈久宁(04:24:50): 啊，通用的一个结构体。如果我们需要一个整数类型的话，我们可以去定义。然后在这种情况下，如果我们写3.5。他肯定是不行的。啊，这里能够理解吗？这是一个特殊的语法，我们就把它当做特殊语法去理解就可以了。就是在这里花括号里面，你就把它当成是一个占位符。这个单位服是需要？用户是产生似的。就是你作为写代码的人，你可能也不知道这个画括号里面是啥？它就是一个模板。但是我我们我们能够知道的是，X的类型和Y的类型一定是。一定是P类型。像这里的话，其实我们。它就非常相似，比如说我们可以去写一个。向量vector。

陈久宁(04:26:03): 比如说vector in。然后。就和12。这里面我们其实就是。干的是同样一件事情，我们可以写。就是为什么我们的矩阵啊？可以，我们的向量或者说我们的矩阵可以支持啊，不同的数据类型。

陈久宁(04:26:27): 其实就是因为向量和矩阵它被设计为了一个参数化的结构体。能理解吗。啊对蒜类型稳定。就是它这里在于。类型不稳定的是。呃，或者说那你你你问你问了一个措施的问题，但是我get到你的意思就你应该问的问题是这个东西算不算抽象类型？是是这意思？对啊，它是一个具体类型。这个。或者说它的性能，它的性能是非常高的。在在这里面，比如说我可以传，我可以传就呃，T是什么这件事情啊，我写代码的时候是不知道的，我是可以通过传传传这样的东西进去。像这里，比如说P等于啊！我也可以传一个，比如说二，三点二，三点五。这都是可以做的，然后它性能快或者不快，完完全全取决于我传进去的这个东西是不是一个具体类型？如果是，那么它就快，如果不是它就慢。这个和之前讲的是一模一样的，你就你就把它理解成。

陈久宁(04:27:54): 地震就好了。就是就像就像是我们上午演示的。啊，这样的一个东西是？然后大的一个东西它？它是一个慢。然后后面接方口号，它其实就是一个，它就是vector。的一个简写而已。呃，能够理解吗？对，所以呃，如果我们要做结构体的话，我们。如果我们要做一个高性能的结构体的话，我们一般是两种写法，一种是这种写法。还有一种是参数化的写法。这里的参数化有不同的。我我们是可以有任意多个参数。在里面的，比如说我们可以写一个。我说T一，T二，T三。啊，这个都是可以的。然后我去说。当地然后。不知道64。啊1.23。这这个都是可以。就是我可以有任意多个参数化。

陈久宁(04:29:21): 然后这里就涉及到一个构造，就是你会发现，唉，这样写起来好像很烦，我能不能就这样写？他其实也可以。它会是自动推断。当然，如果说如果说它自动推算不成功。啊，我们是可以。补补稍微补一补的之类的，就比如说我们可以给出一个额外的构造函数，我们说啊。

陈久宁(04:29:51): 我们可以说那个。哦，我还是不说了吧！对，反正反正就是参数化结构性大概就是这个意思，能能能理解了吗？这里如果是自动推荐的，这种性能就不如哎，一模一样一样一模一样。就是你你你你可以这么认为。就是你你只要去看太。它如果没有给你红色，就是从直觉的角度来说，它如果没有给你红色的显示，它就是一个高性能的版本。对这句话可能并不一定完全正确，但是在很多时候。

陈久宁(04:30:45): 它是一个非常有效的一个参考参考点。这个这个也算是容器吧，这个结构。啊对对对，但是刚才我们不是说呃在容器和那个。那种，但是可以用这种。通过具体变量类型来对对提高对参数化类型，它也是一种具体类型。只不过它的这个具体类型是你后面给它指定。但是刚才呃通过这个展示来看的话，呃，如果说是就算是没有给他这个更新指定。他的性能跟指定那个都是一样的。呃，这个点其实在于在于这个我们可以试试，我可以跟你解释为什么？呃function。比如说我们定义一个方向叫point。三D我这里用的是小写啊！三D我们正常情况下我不知道XYZ的类型对吧？呃，但但其实我是可以知道的，我可以通过。就说。第二。第三。我这里好像又引入了一个新的概念。

陈久宁(04:32:05): 我们之前见过这个写法吗？我我可以通过这种方式。然后他也是类新闻给你。对，就是说我，我其实我其实不知道那个。就是在这个函数里面。我知道的是xyz一定有一个类型。然后我通过我这个关键词，或者说通过这个特殊的语法，我可以把XYZ的类型

取出来。然后我们谈论内心稳定和不稳定，我们上午说的是。只要我的我可以，我的返回值的类型可以唯一的被输入的类型来决定。啊，那么我就是一个类型稳定。在这个例子里面，我们知道输入的类型它一定是确定对吧？然后我们我们就知道返回值的类型肯定是确定。嗯，说到这里顺便就是提提一下will这个东西干的是什么？比如说。我们正常情况下。随便写一个函数。二X然后返回，比如说X加一。啊，嗯，我们在这里面并不需要知道X的类型是什么，所以我们就直接这样写就可以了。如果我们在某些情况下，我们需要知道X的类型。我们就可以通过。比如说XP。来来把一个来把T给取出来，比如说啊。但这个例子里面我们就是。

陈久宁(04:33:58): 他拿到的就是印特六的事。这里拿的这个是？就取到的这个东西其实就是一个类型，或者说值。能理解吗。就是啊，这个是一个特殊的语法，然后只要。只要get到。他的目的是什么？我觉得就够了。不需要去做太深的一个理解。呃，这里这个东西这个语法可以拿来做非常非常多的东西，比如说我们有积。

陈久宁(04:34:35): 是占a等于R的数。我们可以通过type of。A来获取到这个矩阵的类型。我们可以通过。来获取到这个的元素。我们也可以通过。A来获取到这个矩阵的维度，它是一个二维的矩阵。

陈久宁(04:34:57): 呃，这三个函数其实背后非常非常简单，我举个例子。啊，比如说my type of A？它是一个。我要****。就完了。他就一句话。能理解吗。就是你你们以为的一个非常非常非常难的一个巨大内置函数，其实它本质上就是一行代嘛。然后a type也是类似的。他就拿出来了。然后。比如说。呃，维度买电。他就拿到了纬度。然后为什么为什么我自己我自己这样行为，因为他背后就是一个。

陈久宁(04:36:45): 思考一下。

陈久宁(04:37:18): 呃，这个吗。啊，我应该怎么解释，或者说你哪里没有？就是我，我们这就等于是说我希望把，因为就是他，他其实背后是因为呃绝了所有的矩阵。符合一个这样的一个参数化的一个设计的模式，就是说第一个第一个参数是它的元素类型，第二个参数是它的维度。

陈久宁(04:37:57): 所以如果我们想要知道一个计算的维度的话，我们只要把第二个参数就取出来就可以了。那么取第二个参数的话。啊，我们就需要。就是按照这个模式去取第二个参数。就是第一个参数是什么？我我不关心，因为我的我的目的是取第二个参数，但是我必须要在第一个。比方放一个放一个单位。呃，是做一个？匹配因为是这样的，就是如果我们没有wow的话。啊，他会说K没有定义。因为他确实没有第一，我们正常情况下，如果我们要去。写一个，比如说。我们正常情况下，我们是写这样的东西。正常情况下，我们要要在这里面写的都是一些具体的。具体的类。已经定义了的东西嘛？但是没有定义这个东西，我我们其实就是通过。我只是做了一个模板。它和前面那个参化形式是非常类似的。就我们前面在做参数化结构体的时候，其实就是在做这样的事情。你就可以把它当做一个特殊就俩特殊的语法理解就好了。就是说没有定义的东西，我们可以作为一个模板。

陈久宁(04:39:22): 嗯，看看有没有新的问题。他他说的是我，我要有一个函数。然后。就是。对，就是它前面是前面是函数的名字名称。然后最后等于N其实就是。啊，对，这是我们单行函数定义的那个模式。呃，如果没有问题的话。我们就进入到。另外一个师弟就是。在organo的实现了，就刚才其实是做了一个准备的介绍。在里面这个东西。啊，很简单，它其实就是一个所谓的。为了区分我叫麦。然后。它是一个区？所以它的这个矩阵的类型是T我不管我不知道T是什么。然后这个集镇的？这个军阵的维度一定是二维的，我知道它一定是二维的。然后我知道这个机这个对角机。只需要对角线的值就可以了。所以我这里存的数据。是一个vector。大概是一个back T的东西。然后。我们能够就通过这个结构体去表达一种。啊，对角矩阵的数据类型这个能够理解吗？就是说我这个结构品其实已经存储了。对角矩阵的所有信息。因为对角矩阵它最重要的信息就是对角线的元素的内容。

陈久宁(04:42:00): 所以我。只需要去存一个项链我就可以，我只需要去存一个项链，我就存储了对焦。这个对角矩阵就能够把这个对角矩阵表达出来。啊，这个能够理解吗？对，至于怎么表达，我们待会待会去介绍。对，然后。然后就关于？计算接口。它有，它有一些特殊的规定。

陈久宁(04:42:27): 比如说。嗯，我们当我们在是写AA一。的时候，当我们去取一个矩阵的词的时候，我们在背后其实是调用了get index。啊，他现在是是a一大概是通过这种方式。我举个例子。A等于二等23？我们只学A。A E D的时候其实背后是get index。他是他背后干的是同样一件事情。对，然后狙还有还有第二个。这种接口是a二对不，比如说a一等于等于二。它背后是index？嗯嗯，a12。就是其实提供了两个函数，一个函数用于。矩阵的值的获取一个函数，用于矩阵的值的设置。Half a day are getting in death, setting in death. 然后九。关于虚算的一个要求在于。啊，我作为一个矩阵，我必须要提供三个。东西一个是叫size的长度。但是a必须要有实现。第二个是get index？呃，

几乎是必须要实现的，然后第三个是set index。

陈久宁(04:43:53): 啊，也必须要实现。啊，当然这里并不是说真的必须要实现，就是说最好是实现。呃，所以这样的话，呃，当然它还有一个要求在于。这必须要是一个抽象的。鸡蛋类型。这个又又引入了一个新的语法。他背后的意思是说呃。这是。小于号冒号它其实是一个类型与类型的继承关系。这样的一个东西，比如说我们知道int类型是一个。是一个实数类型。我知道实数是一个。单个类型。大概是这样的一个意识。然后这里其实就是说我。我声明我们要创建的这个对角矩阵，它是一个抽象的。

陈久宁(04:44:44): 二人类型。然后这个a类型的元素类型就是这里的T。这个a类型的维度是二。能够理解吗？啊，在这种情况下，我们在。

陈久宁(04:45:08): 呃，这里矩阵接口要实现哪些东西？我们可以去九两的官方文档其实是有。有定义的我们可以我这边嗯。我可以找到，大概它有一个叫做。嗯java官方文档在这一节里面是有。是有定义力的，他在。它在这里啊，一个抽象的，其实类型必须要实现的方法有。

陈久宁(04:45:57): Size a这个方法。Get index. 这个方法还有另外一个get in death方法，然后其他的。呃，可选的。呃方法。有一堆这一堆你可以？实现也可以不实现。看具体的情况和D。所以这里我们用一个最简单的模式来做，我们就选，我们就实现size A。

陈久宁(04:46:23): 呃，这里写一个B点size的原因是因为这个size这个函数是。来源于被就是我们提供一个完整的名称。就是当我们写SA的时候。呃，我们实际上就是在调用B。因为因为在base里面大概就是base里面有一个size。所以我们可以直接敲菜就可以了，但是如果我们要去定义一个新方法的话，我们必须要把全全称给写出来。

陈久宁(04:46:53): 这里面的话我们就可以通过。啊，它的尺寸其实很简单，就是。Lunch a点data。啊，为了。让他表现了一次，我把这个名字叫叫大吧！然后。对，这个就是我们这个矩阵的尺寸。维度就是其实就是一个N乘N的。维度然后这个嗯就是我的对角。对角向下的长度。然后。Get index这个东西。我们就可以去写a的。呃IT比如说我们定义这样一个实。他其实就是。这个这个实线也很简单，如果I等于J的话。

陈久宁(04:47:59): 我就知道他是对角线矩阵了。我叫返回。A点D I a G的第二个元素，否则的话。返回。这里能够理解吗？就是，否则的话返回零？但是我们上午说到，我们要做一个类型稳定的一个实现，所以我们返回一个Z。能get到点吗？好，我们把这三个东西就这三行三段代码。接进去我们就做了一个对角线。那么我们可以正常的，比如说这个东西叫a等于。他都他都是有的认识A。就是NDMA。这些都有。然后比如说我可以去算这个集镇的。嗯哦，他他这个定义就是IS line。对，我都可以算，就正常，大多数东西都可以算，之所以可以算呢，也是因为多重派发或者说这种泛型编程带来的一个好处。所以。呃，所以刚才有同学就很很很担心，说我不知道大家跟着我是否存在。我该怎么样去做斗争派啊，实际上就是。啊，这事情你真要做起来非常非常简单。但是当然这里也有个前提，就是你要知道，或者说你要相信自己有能力。把这些东西给拼出来。真正去做了，你会发现在接待下这些这些事情是非常容易的，但是你必须要。

陈久宁(04:49:51): 要尝试着去做才行。然后我们可以。比如说我们之前有一个my sum函数。比如说上a上a他，我们现在是没有对上。做优化的。呃，比如说我们现在没有对上门做优化，它可能会比较慢22纳秒。啊，他其实是把所有的词全部全部算了，就包括那些理全部加起来了。如果我们要对它做一个优化的话，我们就直接写。等于。然后。我不知道。嗯。我们就会发现哎，太快了。能够理解吗？对，如果我们，我们不是很确定的话，我们可以找一个更大的啊！

陈久宁(04:50:56): 比如说一到100。它是26那边食材是非常快的。他和他和那个？呃，和绝大内置的实现。一样的。所以所以的内置实现。它仅仅只是方便，然后它确实也很常用，所以写了一个在那里。但是如果你遇到了一个你特殊的需求，然后就俩的标准符没有怎么办呢？你自己写一个其实不难。对它的代码量也不多，然后它的性能足够好。这个就是diagonal的一个呃，最简单，最简单的一个东西，当然啊内的内置实现它，它把除了上以外，可能把各种其他的函数都给出了优化的实现。但是那些东西我们今天就没有必要在这里讲，就是你遇到了。你去做专门性的，通过多重派发的方式给出一个优化时间就可以了。然后你也会你就会发现，唉，这个东西我先写三行，唉，它能用，然后我发现慢了，我再加一个新的函数，它就变快了。

陈久宁(04:52:10): 然后一点一点一点就是所谓的渐进式的优化。啊ok吗。这样我知道你写一些函数，它后面会有个感叹号那个，但一般来说不叫像飞扬。啊啊，这是一个很好的问题啊，就是就在里面我们会有一些东西，比如说叫我举个例子，X等于。

陈久宁(04:52:38): 等于231。然后我们可以对X进行一个排序对吧？

陈久宁(04:52:44): 呃，它还有一个。呃，排序完的结果是？Dot X我们把它存成Y。呃X是没有变的。然后我们有一个感叹号版本的。他去

完之后的X是变了的。对啊，这个东西啊，我们其实就是说一个含一个算法，它可能会有一个叫做。In place的版本。和一个。呃，普通版本我们就叫或者叫呃我不太确定它那边叫什么版本，就是反正就是copy版本。大概是这一只？然后in的是原地修改。就是我们比如说short感叹号，它就是一个in的版本。然后正常的说它就是一个copy的版本。然后就那有一个命名上的约定，这是一个命名。命名约定。它不是一个语法。他只是因为就那样，你可以。举个例子，我可以写a我的一个呃，我可以定一个函数名叫做a感叹号BCB。就我完全可以定义一个这样的函数，然后就有一个约定说，如果我们要实现的这个算法是一个算法，我们就在后面加一个感叹号作为区分。

陈久宁(04:54:13): 就就这样没别的了，它是一个程序员之间的君子协定，你可以不遵守。但大家大部分人是这么做的。那也就是你当时在那个？对，它是两个函数，它不是一个函数，而这个事情也很容易做，我们来，我们来随便做一个吧，就是说我们做一个。交换，交换。我们就简单写一个，呃，就我，我比如说我写一个X等于12，我要做一个函数，这个函数是把X的第一个元素和第二个元素交换位置。呃，他做起来很简单，比如说我先写一个书读。然后。啊X的temp等于X？然后X二等于一，然后X。一等于碳。啊，这就做完了对吧？Wap然后。我是不是哪里写错了啊？我看一下X等于X等于XC。要11个。看不等于X？

陈久宁(04:55:39): 对啊，他就做过了一个版本的swap。然后呃，正常的手法不会怎么写呢？啊，等于。说话吧。就是很多时候我们不一定要要说。立刻去写一个正常的版本，我们先把版本写完。然后套一个copy就完了？能理解吗。

陈久宁(04:56:34): 嗯，我们看看有没有新的问题。

陈久宁(04:57:19): 有希望听吗？如果没有新问题的话，我就引入到下一个话题，然后讲完下一个话题我们来看看C。我先找一下我的ppt在哪？

陈久宁(04:58:41): 好，我们来到这个PPT。可以分享到。呃，这个PPT是4月份的时候在北京华为海事做的一个报告，然后起的是他们。的代码，然后在这里面我们。有一些就是对上午做的汇报的一个报告的一个更补充版本吧。呃，这些都。多漂亮，这个可以看一眼，就是这个是我的一个。主观性的评价你可以认，也可以不认，但是反正就是一个评价而已。

陈久宁(04:59:29): 大概意思是说。如果我们给出足够的。时间的话。啊C和C加加一定是一个。性能上最好的一个版本。就就绝大部分情况下C和C加加的性能。都是最好的那那个？然后就俩的优势在于。我们可以在一个很短的有限的时间内。就先达到这个性能上限。然后matlab和。呃，缺点在于。它的上限太低了。

陈久宁(05:00:04): 就这张图，反正就是一个。啊，非常非常主观的评价。然后这里面也稍微区分了一下就是。啊，初级程序员和有经验的绝大程序员。写代码的时间和性能的一个。就是版本就是比如说初级的。初级的绝大程序员，他可能一开始写出来的代码性能跟my tab会非常相似，或者甚至比慢。但是通过啊！呃，比较长的时间，它可以优化到一个比较好的一个技能。然后有经验的巨大程序员，他可能。呃，花的时间可能稍微比麦会稍微多一点，也许会比麦还要稍微短。

陈久宁(05:00:53): 但是它的一个性能的一个起点本本身就已经足够高了。在这个足够高的体现下。啊，它也许就不需要再进一步优化了，或者说如果要优化的话，我们啊稍微花一点点时间也就够了，就绝对不需要像C和C加加一样，花费大量的时间在品格优化上。它大概是表达的这个意思。

陈久宁(05:01:15): 就你可以相信也可以不相信。就反正是一个非常非常主观的一个评价，我也没有任何的经验。啊，或者数据去跟你正面看。然后。那个啊。反正。啊，这里这里也是一个，这这是一个点，是需要注意的，就是啊，之前确实是漏掉的啊。在里面这个我是啊，我其实一直想讲，但是一直都忘记，就是如果你们一旦开始去写出来，你们就会发现一个啊匪夷所思的现象。一等于等于一？

陈久宁(05:02:01): 这个现象非常的匪夷所思。然后包括呃。向向量一和。我们。比如说呃，我想想。这是一个向量啊，这是一个航向量。然后这是一个？长向量不等于月向。就是这种，这种都是一个从matlab的视角来说，你你你怎么做成这个样子，你是有bug吧，你写错了吧，或者。对，就是这是一个非常非常匪夷所思的一个点。就是。我们有同学能够？哪个尝试去解释一下？就为什么我们为什么除了要设计成这个样子？就他我我我可以先给结论。他就那肯定没做错。就是julie是故意做成这个样子的，不是因为我们写代码写写写出了bug，而是julie他要故意做成这个样子。那么有没有同学能够猜测一下？

陈久宁(05:03:17): 或者说尝试。解释一下为什么觉得要做成这个样子？啊什么。不对。不对，跟内存没有关系。啊，它和多重派发有关系？它就是在强调它的不同的类别，包括第一个的话就是很典型的是一个数组的一个不同的数。第二个是mattress vector给我推测mattress vector他们是某一个鸡类的，他们的开身。啊对对。对，就是它背后的核心原因。确实是因为他们的类型是不同的。因为一的类型是什么？

它的类型是一个整数。然后向量它这个是一个？然后像航向这样？和列项讲，我们可以看到这里是一个matrix。是是什么？它是一个array？Intel是二，它是一个二维矩阵。然后vector int是什么？它是一个？就是你的数据类型不同。你去做比较，或者说做这种意义是不大的。啊，这个是。最初的就是，这是他们的一个表层的一个点。然后再往深。我们可以。有没有人尝试去解释解释？再进一步解释。他们都是不都是？他给拍摄成两个对。就是通过我们前面说的那个参数化类型的方式是这个。

陈久宁(05:05:43): 有没有想法？嗯，说就是在书城开发。事情过去。有更快的，不同的类型对应的优化。因为。进行优化。我说混在一起的话。其实并不一定。对，通过把它区分开，然后用不同开发的进行。区分的话现在？对，呃，这是一个很好的点，就是这个点说的是呃。皮肤所有的性能是来自于多重派发。我们把。我有一台。对于多重派发有没有好处？就是他不会提升性能，他也不会怎么样，他可能会给用户带来一个。呃，表层的一个应用，应用性就是，唉，看起来一等于等于一等于，像这样是一个很自然的现象。但他在。更大规模的代码组合的时候你会发现。你你没有办法去？去把所有的这种可能现象全部。组合出一个用户满意的状态。就是。这这件事情就是一。就是包括还有就是酒里面。一加二就比如说这件事情为什么为什么不能做呢？就为什么就那样，一定要求你说。就是这样的一个事情，就是觉得。在很早的时候已经做过非常非常多的实验，就包括呃，比如说这个一。啊，一个向量加一个是整数。啊，之前居然是支持的，后来把它删掉了。

陈久宁(05:07:48): 所有的这些东西背后是因为虽然已经做过了非常多的尝试。然后这个尝试就是非常多的尝试去让去让呃用户得到一个看起来符合直觉的结果。就是，比如说我们这里说的符合直觉就是。呃，标量和数组可以相等，可以相加这样的一些符合直觉的结构，然后之前做了非常多这样的呃努力和工作。

陈久宁(05:08:14): 后来发现我们永远没有办法满足每一个用户的每一个需求。就是总有新的需求。说诶，这里不符合直觉你能不能补一补？完全补不了。对，所以到最后啊，我们认为就是整个绝大社区，他基本上都是这么认为，就是说呃，这个事情是没有办法，他是他是一个永远做不完的工作。所以与其与其我们去。做一个这样的永远做不完，而且用户也不会满意的工作，不如在第一天就告诉用户。一和整数一和向量不是同一个东西。大概可以这么理解？这是一个解释。

陈久宁(05:08:57): 第二个解释是啊，歧义的问题。

陈久宁(05:09:02): 题的问题这个怎么理解呢？比如说。如果。一等于等于一的情况下。啊，这个矩阵的维度。咦，他要不要香的？张男孩子不对？不知道。我们在这个问题里面他。它是一个标价，是一个没有维度的。他是名名为向。然后如果我们希望12。啊，等于等于啊，比如说12。一逗号啊。12如果就是一道的话，二是一个列向量。然后12是一个航向。如果我们希望它相等的话，我们，我们可能会希望诶它的维度和要不要相等。比如说S。他也不像的就是。我们可能在某一个地方达达成了一个一致说诶这个地方符合直觉，然后我们就会发现，诶另另外一个地方又蹦出了一个不符合直觉的东西。然后我们永远没有办法在所有地方达成一致。然后就会产生更多的。对，这是第二个解释。

陈久宁(05:10:31): 啊，能够理解吗？对，所以。所以质量的话啊，就是说啊！哦，那我前面前面共享是不是也没看？对，前面共享也没开啊！你记得你记得那个一段？那一段开了吗？那段就没开始？那那就错过了，错过了一些呃，非常不好意思。所以回到我们这个。这个PPT就是说。呃，标样是标量像样是像样？呃，尽可能就是我们同学内部有很多代码库是把它混用的，但是从julia的呃设计设计思维来看。呃，需要尽可能的。把它呈现开来，因为。啊，其实很用。哦，我们好像有同学笑得很开心。就是因为呃，即使混用你也不会产生一个。啊，你理想的那个样子？所以不如在第一天就把它成。啊，然后比如说使用高效函数这些东西，这个东西可以不用。

陈久宁(05:12:07): 啊，这是一个？就是我我我主要是想讲这这一张PPT。这张PPT是一个非常有趣的一个例。我们可以啊！尝试的去。呃，解读一下。V一的版本是我们第一天在海西的代码库中看到的版本，它定义了一个算法。这个算法是？啊，一个方值的计算。然后它的实现是长这个样子，我们对比一下MT的版本就同样的代码。咱们不就有点像？一个是600微秒，一个是200微秒。呃，我忘记这个数据的大小是多少了，可能是1024吧，或者没事，但反正是一个650秒和200微秒的一个。差异。然后我们是想诶，怎么样去优化它？呃，优化它的话，我们会发现这里面。啊有两个。有两个橙色去。一个是ABSA的四次方。还有一个是ABS的平方。那我就想诶，这个的东西我先把ABS的平方。给算出来。

陈久宁(05:13:21): 然后这样的话我就少了一个，少了一次计算对吧？那么我的我我把这个尝试做一下诶，发现性能确实快了快了三倍。Matlab变成了172微秒。九俩变成了啊11微秒。这个样子。然后绝大部分人做到这一步就结束了。或者说做到这一步，对于matlab和python语言

来说就已经结束了。因为他没有更多的办法去处理。呃，对于猪来说，我们还可以做进一步的优化，这一步，这个进一步的优化。在质量下可以做，在C语言下可以做。但是啊。如果在C语言下做的话，意味着我需要有一个。就是P加C的方案或者matlab加C的方案，它可能不是那么容易做出来，但是jl做出来还是比较容易的。

陈久宁(05:14:13): 就是说。呃，我们可以。把它调一个顺序。啊，这里说的第二个顺序性指的是什么呢？就是我们可以啊。呃，我他这里说的是第三个版本，第三个版本我用了一个特殊的，特殊的函数实现方式就是。呃，我在比如说命命这个实现。我可以传一个函数进去。传一个函数进去。的话。我去想想这个该。就是在第二个版本里面有一个有一个点在于。

陈久宁(05:15:04): 我如果去算第一项这个除法的第一项的话。呃，我会先去把。点平方二算出来。太平方二算出来，它其实是一个批证对吧？呃，我们上午说了内存是一个需要考虑的因素。集镇创建这个你是创建了一个新的执政？他其实在整个性能开销里。

陈久宁(05:15:31): 也是一个需要考虑的因素。就是我们月嫂的内存创建。性能可能会越高。所以在这里面我就会去想诶，我能不能把？碳的平方。呃，创建的这个局。给删掉。就是说我我这样的话，我就因为因为我我也是算呃命。

陈久宁(05:15:53): 啊time平方我其实没有必要去。创建一个矩阵，我只要去对整个东西做一个循环，然后呃开一个四次方向相加。啊，这样的事情就可以了嘛，所以最后我们给出一个。呃，就是提供了一个流式的版本，这个流式的版本就是说对于a的每一个元素。

陈久宁(05:16:12): 我去算他的。呃，四次方，然后把四次方。是算平方。就是这呃上VR版本和V三版本，我画线的地方是等价的。但是他们的性能是不一样的，性能不一样去是因为。一个有内存创建，一个是没有内存创建。能够理解吗？啊对。那个广播机制为什么可以买？嗯，就TP之后那个点的？呃VR版本你说这个吗？是天气点哦！呃，这里会有创建，就是这个地方是会创建一个内存的。对我我们要做的就是对于VR来说，我们要优化的就是这一段内存的创建。

陈久宁(05:17:21): 因因为我去，我去求求这个矩阵的平均值。或者说社区的每个元素的四次方的平均值，这件事情从理论上来说，我是不需要去。

陈久宁(05:17:34): 创建一个中间矩阵的，我只需要。我只需要去写一个for循环去做就可以了。所以，所以我们这件事情就是对于就来说就是。呃，用这个流式版本的函数，它就可以优化掉。这两件这两个东西是完全等价的，但是它们性能有差距，这个性能差距就是体现在啊。

陈久宁(05:17:58): 有没有一个额外的内存创建？所以我们可以发现，从VR到V三。从它的性能从11微秒上升到了3.2微秒。然后V三它其实还没有完全做完。V三的话，它其实又还依然是存在一个。额外的就是集成，创建的就是。比如说呃V三里面。我是对a的每个元素。就是我，我们把V三分成两部分。我去算分子这一项的时候我对a做了一次。呃呃循环对吧？我去算分子对象背后一定是以某种方式对a做的一个循环。

陈久宁(05:18:49): 呃，你是在笑什么吗？那那或者我刚才说的VR，比如说我们为什么从VR到V三，你可以说一下。美满足一下所有的那个。啊啊对对。就你你那边是在跟别人聊天还是怎怎么样，我看着你好像很开心的样子。嗯，行行。啊好，我们回到就是回到V三，就是我们为什么要从V三到V四，我们就去分析V三的一个特性。V三的话我们去算分子的时候会对a做一次循环。对吧。然后我们是算V三的分母这一项的时候。也会对a做一次循环。我们。算分子的时候我对a做循环，然后去算呃a的每个元素的四次方。算分子的时候，呃，算分母或者呃算另外一项的时候又对a的每个元素做了一个循环。

陈久宁(05:20:01): 然后就算平方这一项。那么这里就出现了？我能不能只用一次循环？就把所有事情算完。所以这就引出引出了微信的版本。微信的版本他做的事情很简单，就是说。我从头到尾我只对a做一次循环。这一次循环。哦，我把ABA删我把。啊，这个元素的二次方和这个元素的四次方。一起算掉。一起算掉之后，然后分别加到分子分母上。然后最后去做一个？啊。调平均值的操作。它就是微微信的版本。能理解吗。对，然后我们就会发现。啊，把vc版本做出来，它就是一个1.6微秒的一个性能。就是他背后。这里面究竟做了哪些性能优化操作吗？啊，其实也没有太多，就是所有的，或者说这里面背后所有的性能优化的操作都是围绕我们今天上午去讲的那些点。就比如说内存创建就在这个例子里面，它是围绕内存。这个话题去做的。或者说围绕我们上午讲的广播这个话题去做的。

陈久宁(05:21:14): 广播它也许是一个非常简洁的代码写法，但是我们可以看到它的性能是存在。开销的。就是V和VR的版本，它都是属于。呃，matlab和python典型的向量化编程的写法。他能够。呃，很简单的把事情。写完。但是它如果我们需要达到一个更高的性能的话，我们需要去分析这种向量化代码的。背后发生了什么什么样的事情？然后去分析的话，我们只需要分析我们能不能在内存在。这种事情上

去进一步的优化它，比如说我能不能移除重复的计算，我能不能移除不必要的内存开销。

陈久宁(05:21:57): 这样的一些点。然后把这些事情全部揉在一起，就是VC的版本。啊，这样解释能够？能够清楚了吗？对，这个是呃，最后补充的一个陈老师问题就是你习惯在for循环里面加两个。呃，这是这个是？这个是因为啊！就那里面当我们去做。类型就是当我们是做a一或者说。这样操作的时候。呃，就俩会有一个。越界检查。这个越界检查，他大概会判断他大概背后做的是什么，做的是如果一在。啊一群index。有正常工作？有，这天正常工作。否则的话报错，这就是越界检察干的事情。那么这里就有一个EL判断了。这个EFS判断对于底层代码来说。啊，它还是蛮贵的。蛮贵的，大概就是一个。呃，十个指定周期就是二到十个指定周期，这样的一个开销，如果我们做的是正常的加减法。加减乘除这种都是一个指定周期就能搞完的事情，我们引入了一份判断的话。

陈久宁(05:23:32): 事情就变慢了，而且慢的还会比较严重。对，所以对于一些底层运计算来说啊，我们会引引入所谓的inbound inb干的事情就是说，唉，我这个东西已经在数组的。正常。正常下标范围内了，你不要帮我再去做越界检查了。他就干这样一件事情。对，就是inbound这个东西啊，需不需要完全取决于你的算法？如果你的算法决定了，这里是一个安全的。东西就是你这里一定不会出现。做做越界现象的话。你可以加上。然后。当然，当然就是你也要看你是不是一个底层算法，如果你在一个非常高层的地方，你加上一，就是比如说你这个算法需要。需要十秒钟才能跑完。然后你有一个地方有一个是下标，你给它加上银棒子，可能快了点那秒。

陈久宁(05:24:31): 啊，那又如何呢？对吧？所以取决于要不要加引棒子，有两个因素。

陈久宁(05:24:37): 一个因素是我们是不是在讨论一个底层的算法，实现一个基础的算法实现？因为在基础算法时间向上优化if else的时间开销。啊是有价值的。然后第二个因素是我的这个算法，我能不能从算法程度上去确保它一定不会出现数组越界。然后第二个S IM D是九点，呃是利用到了CPU的。呃，向量运算指令。这个事情。我觉得可能没没法做一个更。更更好的一个。呃，解释你，你就大概可以理解成是。CPU我们几乎现在所有的CPU。啊，你都可都有一个内置的并行。并行手段这个并行手段就是S IM B或者说A。对这个并行手段如果。他他可以。啊，极大的提升我们的性能，我们举个例子。嗯，我们a等于六十，六十。呃，大概是一个三微秒。然后如果我们加大一棒子的话，这里其实不会有任何影响，因为我们其实没有，没有，没有，没有做。数组的下标操作在这个这个代码里面。对它其实性能是一样，依然是一个。

陈久宁(05:26:54): 呃，三维秒。对，如果我们举举个例子，我们把它稍微改一改。我们叫for I in。Each dxa. 然后。AI对这里的话会有性能差异，因为正常情况下九。在这个地址里面就那可能优化掉。就就俩知道。这样一个这样一段代码。是绝对不会越界的。呃，我们可以去，比如说我这里可以做一个非常。非常简单粗暴的事情，比如说我们知道他是60。杭州是64吧，我记得好像。这样也许。嗯，他似乎还是对他似乎还是做了一个不会越界的就是。就是他就是要不要加赢，当然也要就是这个演演示出来就是。取决于。这俩能不能优化，然后？取决于我们。性能有没有差异之类的？就是。呃，现在就俩这边其实越来越聪明了，所以他在很多情况下，你就算不加硬棒，他会自动帮你补一个。呃，然后的话。我们可以。尝试一下。你可以发现诶，变快了。呃，就是从三点三点六微秒到236纳秒。他们的结果。稍微有一点变化。一个是教。如果没有SM D的话。嗯。你你会发现最后几位稍微有一点不同的变化。就是S啊，它是因为SM D它是一个并行指定并行指定的话，它能够更大效率的提升。然后但是有一个代价。它的。For循环的顺序是可能打乱的。

陈久宁(05:29:11): 就是，他可能先算。就是它在这里面的I的顺序可能是先算第三个就是I等于三，然后接下来是I等于一，再接下来I等于四。然后再接下来二等于二，它也有可能是以一种乱序的方式去实现的。然后为什么它会有差异，因为福地点数是不满足。浮点数的加法是不满**换率的，就是对于浮点数来说，a加B等于等于B加A。这件事情并不为真。所以你会发现啊，这样两个版本之间存在一点点的。

陈久宁(05:29:45): 精度差异。呃，这也是为什么这个例子其实俩并没有默认给你加上S IM D的原因，因为他并不知道他没有办法去保证。你想要的。就是这个东西。因为它们并不是一个完全意义上的等价，因为浮点数。啊，没有办法。呃，满**换率。所以这在这个例子里面，我们必须要主动加上SM D。啊，它才能够生效。对，然后S IM D它也有一种就是说法就是。啊，他也有一个特点就是。他的要求非常非常严格。严格到什么程度呢啊？我们做一个。我我可能时间好像有点紧张了就。就不掩饰了？我们直接给结论吧，就是SM B这个东西。跟非常类似，它一般只出现在。在地全带吗？就是啊，这个底层代码为什么它会指出现在底层代码是因为SM D指令。对于数据结构。和运算的要求。算了呃和运算。的要求非常严格。呃，严格到什么程度呢？它必须要是比如说。Int类型。啊，他比如说他必须要是家。乘除这种。加减乘除啊这种。啊，普通的基本运算。他甚至你。他甚至不能要求吃。呃，没有。就是你的这个for循环里面，你不能要有一次判断，如果有，有了一次判断，你的S IM D也没有办法启用。然后他会有？就是有各种各样的要求。

陈久宁(05:31:58): 所以这些要求导致。S IM D它一般只会出现在底层代码里。啊，所以。啊，作为一个。刚才这个说的是。嗯，就是原因

吧，然后作为一个建议就是编程建议这个。这一点我在数学库也是看到了。啊，包括在信号科也看到一些就是。啊，有些人诶，我我这边不是演示了吗？说诶，加上SM D我的算法。我的函数实现快了八倍，我是不是所有地方都先不管三七，二十一先套一个？

陈久宁(05:32:33): 呃，我的建议是MD只在。旨在确认有效的情况。因为如果说你套上一个S IM D，但是。它没有起任何作用的话。你对于下一个程序员来说就是噪音。就是，这就是一个代码可读性和可维护性的一个点。加上一个毫无作用的东西，下一个。我维护你代码的人就会想诶，这里为什么要加上S IM B？然后他可能花了半个小时，一个小时，两个小时的时间去研究完了之后。

陈久宁(05:33:13): 他还没有研究明白，最后去问你，然后你说诶，我随手加的也，我也不知道有没有用，然后他可能气死了就。呃，所以我我的建议是只在确认有效的时候。这个是关于S IM D我们看看有没有别的问题。啊，会有一定的影响，就刚才这个演示。

陈久宁(05:33:43): 呃，取决于你在做多少精度的问题。你的问题的精度要求是多多少？就大部分任务可能是没有这种进步要求。呃，有其他问题吗？呃，如果没有问题的话，那今天这个就到这里。就是我我这边讲的所有东西就到这里。啊，我们可以简单回顾一下吧，就今天下午其实就是在讲。啊多重派发这个东西？然后围绕多重派发，我们再谈就来的一个核心的编程模式，就是我们怎么样围绕多重派发去做。代码设计。啊，就是包括那个？J的多重派和python的单派发，然后julia的这种。函数是编程和python的状态，状态机的这种编程之间的。差异和讨论。然后还有就是我们围绕juda的多重派他去做一个渐进式的性能优化，然后这样的一些东西，然后以及就是引出了就是我们结构体的一个定义，就是说啊，比如说这个。

陈久宁(05:35:03): 啊，其实类型我们怎么样去定义它？占了一些点。但核心核心逻辑就今天下午的核心逻辑是。编程风格是一定是围绕多重开发去做的。你可以，你可以把J就当做一个普通的C去用，或者说你就把J当成matlab去用，没关系，你可以，你可以去做出一个呃非常简单的一个函数实现。但是如果你想要把J用的更好。你背后一定是要去思考多重派发的一个。

陈久宁(05:35:33): 设计应该如何规划？然后。呃，再往后的话就是呃，这下午其实就今天的这个内容就是像我们讲啊。就是普通的一个函数，我们怎么样去对它做性能优化？我们写出一个高性能就来代码，需要考虑哪些点？下午讲的是我们的一个整个的系统系统架构，软件架构里面啊，它是围绕什么样的原则去设计的，这个原则就是多重派发。对于巨大来说，就是多重派发。然后那么。这两个东西。就是啊，质量啊，高级编程的。最最重要的两个主题。然后然后中间可能穿插的讲了一些各种各样奇奇怪怪的知识，比如说预编译。呃，就是这两个模块的预编译，然后啊方法实例这样的概念，然后参数化结构体这样的概念，这些东西都是啊，日后面日后工作的时候啊，时不时会遇到，但是并不一定每天都会遇到的一个问题。我回到我们的大纲。就大纲的话，其实这样一看诶，我们的大纲好像全部都讲完了。对，就是啊，比如说广播和for循环这个我们上午讲。呃，下午最后其实也说了，就是说呃，向量化编程这种模式，广播编程的这种模式，它在python和matlab下是一种性能优化的手段，但是我们不要期望它在J下会优化你的代码性能。

陈久宁(05:37:08): 对，我们只要了解这一点就已经。然后就俩的性能优化基础这件事情。啊，上午讲的其实是一半，我其实后面有点想讲更多东西，但是因为也是没有时间准备，所以我就拿着去年的PPT在讲。呃，后面的话有机会我会补充更多的性能，优化的一些呃思路，今天讲的是最核心的那些思路。然后在具体的编程实践中，我们可能会有额外的性能优化，是不是就是性能优化这件事情？呃，可能聊个35天都聊不完的。就是。呃，一般。呃，像高性能计算这个领域。你从入门到到出师，你可能没个两三年。出不来。所以性能优化这个领域是一个非常难，然后非常需要。专家经验的一个领域。这件事情。它就是这样一个现实。所以如果说我今天上午讲的这个性能优化有多少，其实也没有多少，但是它确实覆盖了出来的那些核心的思路。

陈久宁(05:38:22): 那反正就是。围绕昨天和今天要讲的东西，这个课程课课程大纲大概是也算是覆盖完了，就比我想象的比较比较快，或者说今天。

陈久宁(05:38:34): 今天把进度给赶上去了。然后那就到？我我这边的话就是这个课程的总结就到这里。然后吴亦凡你这边是要不你这边来？分享一下屏幕，我看一下你们CSD到底怎么做的？什么都没有做，呃，那就是说呃这个C间CD是需要你一边选现五，我不需要，不需要我的选项运行了以后，我这边就是显示一个惯气状。

陈久宁(05:39:32): 对，你这里下学历没有没有接进去，我看一下你的first package里的代码。这这是这是默认的实验。这不是你，你写的，这不是你配的显示？我看一下，我看一下你的这个仓库的代码，你给我看对。你有你的这个点get up CI是放在哪了？你的内容在哪？哦，不知道。你有没有把那个东西贴过来？对啊，你你现在操作一下吧！

陈久宁(05:40:14): 把这个内容贴过去。然后你去看你最新代码的流水线。在commission那里对那一个月可以找到。对，然后我我看一下你点

开那个CICI那边。嗯，对。呃，你这个怎么又贴错了？你你东西到你的，你的血压到底在哪里？你我我再看，再回到你的代码库。就是，然后点同源呃。点同源那个文件夹下呢？有东西吗。对，你这里有你这里东西到底放在哪里？呃，是是要是要碰到两个文件，就是要放在。你提上去的东西我看一下，我看一下你的仓库内容好不好，呃网网址你给我看你的网址。对对，我去看你的代码，代码内容。你打开我看一眼。对你你你你有没有发现一个点？

陈久宁(05:42:08): 就是你这里有一个点加点YY ML。然后你有一个点诶不对你你你把那个？点到那个就是呃0383的那个就上面看到有有没有一个0383什么什么的对，你把它改改到。对，这里有一个点CI点Y ML，还有一个点同源文件夹，然后你的点同源文件夹下是不是也有内容？对你你你为什么会有两个？这个文件呢，我看一下你点同学文件夹下的这个东西去的。对，那那这个东西你是从哪来的？我可没有。在其他地方教你。只要你们。去复制他。这个这个文件夹也不是我创建的，这肯定是你创建的。你以某种方式创建的啊，如果如果没关系没关系，你把现在的点CI点ya。Y ML的东西。覆盖过去。然后把点外面的这个删掉就行。然后外面这个删掉。对，然后你再提交。

陈久宁(05:43:52): 对，然后点开我们点开这个圈圈。

陈久宁(05:43:56): 啊，你就可以，你就可以看一下，这个东西已经开始跑。然后也贴进去了，然后这个MR格式检查你可以把它做。对，比如说你这里你也演示一下吧！呃，版本检查过了，但是格式检查没过。然后你必须要等你可以点开，上面有个job dependencies。对对，它会呃有一个呃修dependency是把那个勾一下。对，你会发现呃！呃呃，这里看不出来，这里看不出来，但是反正意思是说你必须要等格式检查，通过你的单元测试才会开始跑。所以简单体现我们大家都可以先把格式简单注释掉，对，你可以这边再演示一下。就是可是检查的那个你你在这里面去找这个。反对啊，不是格式检查。我最上面最上面。这个对。你把这一张注释掉就行了。

陈久宁(05:45:00): 对对。

陈久宁(05:45:25): 对，你就会发现这里流水线少了一下。对，又回到490？对，然后一旦格式检查跑完他就会跑单啊，现在单元测试开始跑了。就就很简单，你就把那个那一段代码复制过来，它就完了，你不要给我搞别的新的东西就行。嗯，会自动存储。我不知道会不会自动生成，这个生成肯定是你点了某个奇怪的操作生成出来的。他一个也不是我教你的东西。它一般就是嗯，就只能存在一个这样的ml文件。呃对对，然后它这个Y ML究竟哪个生效，你可以点开你仓库的左下角？你这个仓库左下角有一个？这个对，然后里面C？对，然后在最上面的general拍l对这里定义了究竟哪一个文件是生效的，就是我书上这个游它字都会生成一个同学，我不知道，我刚才说了这个特别出现，我不知道。好，剩下所有人的问题找他。不难吧。那那这个就算解决了。呃，有没有其他要角的，我这边可以再聊个五分钟，然后我要去开会了。就可以，甚至可以聊跟课程有关的东西。也可以。就是今天讲了一大堆关于性能优化方面的东西，但是我们就感觉是看着你操作就是。

陈久宁(05:47:14): 能知道你是在做什么，就是知道你在做什么，甚至有时候可能都不太能跟得上。我只是听到一个知道您做什么，但是我们去实际上考虑的时候感觉会有一种无缝下什么。这个也是没有办法的事情，因为我一开始我刚才也说了，呃高性能计算这个领域，你们不花个两年，你可能出出不了事。

陈久宁(05:47:37): 但是重要的点在于。呃，你们需要知道，比如说我这，我这种人每天脑子里面在想什么，你们在日常的代码实践中朝这个方向靠齐。

陈久宁(05:47:49): 啊就好了，慢慢慢慢朝这个方向靠起就好了你。你不要希望自己一个礼拜出师或者怎么样？一个礼拜如果如果如果所有的高性能计算的东西，你靠一个礼拜就学完。就会用，然后就能够应用到你的工程中，但是在骗你。然后包括我们怎么样设计出一个好的代码，这种都是属于一个专家经验，或者说高级程序员才能够掌握的东西，那么大家从刚毕业怎么样一步一步发展到一个更高的一个专家编程能力。这个是需要日常不断的思考来。来提升的对，没有一个我我我个人是觉得没有一个。写信在里面。就是要不断的打磨，不断的试图，如果你每天想这种问题。那你可也许过一年？就是一个很不错的程序。如果你每天。想的就是怎么样复制粘贴？也许一年后的和现在一也许一个月你就到底。

陈久宁(05:48:53): 就我的意思，一个月到底就是一年后和一个月，或者你没有差异。他就是关于就这这个发展什么，我在网上之前也查过相关的资料，就发现它主要还是在科学计算这一块是比较。是有优势，呃，叙利亚的发展。这个也是因为它的起源是科学计算。所以那些做科学计算的人，他们是围绕就俩去做了非常多的生态的。然后。这并不完全意味着就那只能做科学计算，只能说目前的生态是科学计算做的非常好。Jel实际上我们可以用java去写写网写网页。我们可以用指甲去做各种各样的事情。对，做实验师弟你想做什么都可以。

陈久宁(05:49:45): 呃，我们有没有其他需要聊的？你跟真的就刚才那个版本检查后，然后单一测试以后那个刷布质有肯定这个这个示。啊，那我们没有别的要聊的，我们今天就到这里吧，然后周周四周五大作业，我晚上给一个更详细的一个。说明。八到十米。然后。周四周五在哪里？这件事情到时候我去跟徐平确认一下。

陈久宁(05:50:24): 训练一下之后再在群里跟大家说对。那那就你自己问问题，我，我在那玩玩玩，我就创建了一个新的一个项目，这个问题我们是。关元单元测试以后我看你之前能打出一个。HTML文件很多啊，你在artist里可以找到。

陈久宁(05:50:49): 那那我们这个课程就是周一，周二的这个最大培训的课程就到这里结束了，后面就大家比如说明天是一个，呃，研发部的。流流程的一个培训，然后再往后两天就能做大作业对。