1. ref与out的作用是修饰实参与形参，修饰后，形参改变实参也会改变  
   ref 必须初始化（ref修饰的实参必须有值）out无论实参是否有值，方法里必须赋值
2. 新建物体的方法

GameObject go=GameObject.Instantiate(playerPrefabs) as GameObject;

GameObject go=GameObject.CreatePrimitive(PrimitiveType.Cube);//只能创建Unity内部自带的物体

GameObject go=GameObject.Translate(playerPrefabs);

1. 什么是shader变体

在写shader时，往往会在shader中定义多个宏，并在shader代码中控制开启宏或关闭宏时物体的渲染过程。最终编译的时候也是根据这些不同的宏来编译生成多种组合形式的shader源码。其中每一种组合就是这个shader的一个变体(Variant)。

1. Unity如何收集变体
2. 协程的定义，其与线程的区别

协程是一个分部执行，遇到条件（[yield](https://so.csdn.net/so/search?q=yield&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/weixin_30819085/article/details/_blank) return 语句）会挂起，直到条件满足才会被唤醒继续执行后面的代码。Unity在每一帧（[Frame](https://so.csdn.net/so/search?q=Frame&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/weixin_30819085/article/details/_blank)）都会去处理对象上的协程。Unity主要是在Update后去处理协程（检查协程的条件是否满足），但也有些特例。

协同程序（coroutine）与多线程情况下的线程比较类似：有自己的堆栈，自己的局部变量，有自己的指令指针（IP，instruction pointer），但与其它协同程序共享全局变量等很多信息。

协程(协同程序): 同一时间只能执行某个协程。开辟多个协程开销不大。协程适合对某任务进行分时处理。

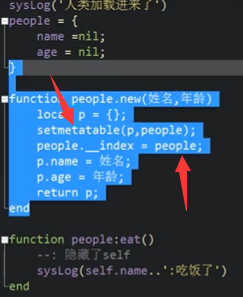
线程: 同一时间可以同时执行多个线程。开辟多条线程开销很大。线程适合多任务同时处理。

1.协程，即协作式程序，其思想是，一系列互相依赖的协程间依次使用CPU，每次只有一个协程工作，而其他协程处于休眠状态。协程实际上是在一个线程中，只不过每个协程对CUP进行分时，协程可以访问和使用unity的所有方法和component

2.线程，多线程是阻塞式的，每个IO都必须开启一个新的线程，但是对于多CPU的系统应该使用thread，尤其是有大量数据运算的时刻，但是IO密集型就不适合；而且thread中不能操作unity的很多方法和component

线程和协同程序的主要不同在于：在多处理器情况下，从概念上来讲多线程程序同时运行多个线程；而协同程序是通过协作来完成，在任一指定时刻只有一个协同程序在运行，并且这个正在运行的协同程序只在必要时才会被挂起。

1. lua继承



7.==和equals函数的区别

对于值类型来说，两者比较的是变量的内容，即值是否相同

对于引用类型来说，前者比较地址，后者比较内容

对于自定义类型来说，前者不同，后者也不同，想要相同则必须在自定义类型里重写equals函数

注：string是个特例，因为内部对两者都进行了重写