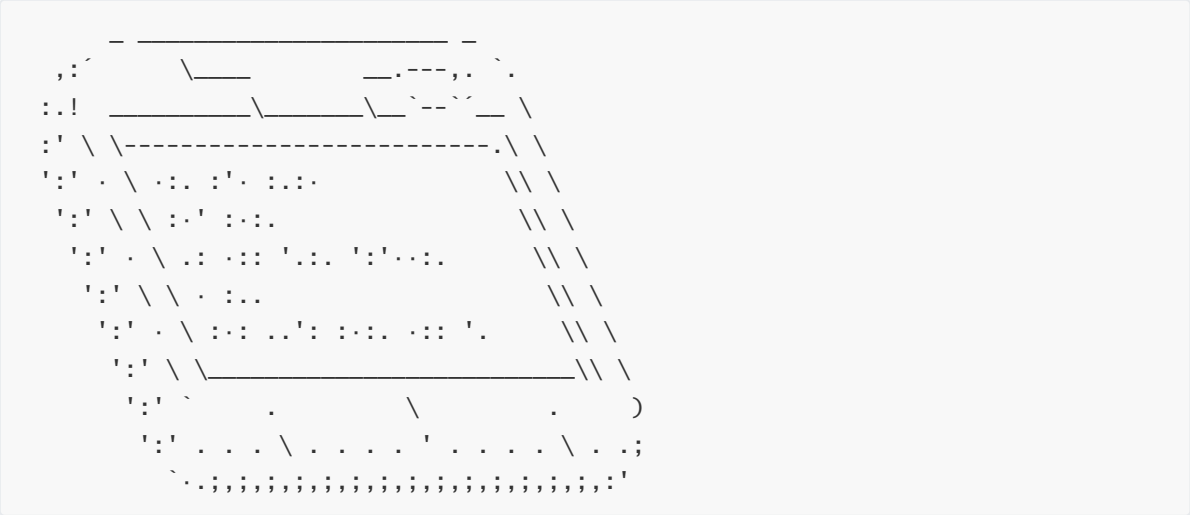
Stone Story RPG 石语言官方文档翻译——石语言

· ·:·: 石语言 Stonescript :·:· ·



石语言Stonescript 用于在意念石Mind Stone 中使用并自动切换装备。Stonescript是石头纪Stone Story RPG中一种极简但功能强大的语言。

如果你想要更多帮助或是合作编写脚本，可以访问我们的Discord，或者中文社区QQ群。

不熟悉石语言？你可以考虑从意念石简介开始。

本手册是所有可用Stonescript功能的综合参考。大多数例子可用来参考了解意念石是如何工作的，可以直接复制粘贴到意念石中使用。

为了方便非常需要本文档的中文玩家使用，译者将尽可能的翻译例子中的英文，如果你想要对比英文例子，可以通过文中标题的超链接访问原网页，查看英文源代码。

· ·:·: 目录 Index :·:· ·

·:·: 1. 例子 Example ·:·:

```
// 在岩石高地中，装备铲子；
// 在恐怖洞窟中，使用装备组合1，
// 但是面对Boss时，
// 换上勾爪和7星D大锤；
// 在亡者之殿中，使用2个魔杖，
// 左手装备毒魔杖，
// 右手装备以太魔杖；
// 但是，如果难度超过了5星，
// 就使用附魔+13的活力长杖；
// 如果生命值低于10，
// 则使用药水。

?loc= rocky
  equip 铲子
?loc= cave
  loadout 1
  ?foe = bolesh
    equip 钩爪
    equip 大锤 *7 D
?loc= halls
  equipL 剧毒魔杖
  equipR 活力魔杖
  ?loc.stars > 5
    equip 活力长杖 +13
?hp < 10
  activate potion
```

·:·: 2. 基础符号 Basics ·:·:

true 真 / false 否

? 进行逻辑判断。如果结果为true，则执行接下来的缩进行的命令（等同于 if）。

```
?loc = caves
  loadout 1
```

: 在" ? " 判断为false的时候，进入逻辑代替分支（等同于 else）。

```
?loc = caves
  loadout 1
:
  loadout 2
```

:?　　具有附加条件的替代逻辑分支（等同于 else-if）。

```
?loc = caves
  loadout 1
:?loc = deadwood
  loadout 2
:
  loadout 3
```

//　　注释。执行脚本时，“//”右边的所有文本都没有逻辑作用。

```
?loc = caves
  loadout 1
//上面写了在恐惧洞窟使用装备组合1（这里随便写什么，不影响代码运行）
```

/* */　　块注解。当脚本执行时，符号之间的所有文本都没有逻辑效果。

```
/*
?loc = caves
  loadout 1
这个脚本什么也不做，因为所有的内容都在块注释中。
*/
```

^　　继续上一行（意念石中每一行能输入的字符数是有限的，因此需要在第二行用^连上，使指令为连贯的，对不能导入外部指令的手机玩家尤为重要）。

```
?loc=caves |
^loc = mine
  equip 连弩

// 以上等同于:

?loc=caves|loc=mine
  equip 连弩
```

·:·:· 3. 游戏状态 Game State ·:·:·

这些判断会告诉你有正在发生什么，以及即将发生什么。

在原文中，这一章节的大部分关键字前面被带上了“? ”，但实际上这些关键词在没有判断的情况下也能直接使用，例如只是将它们打印出来看一看。因此在翻译过程中去掉了这些“? ”判断。

loc地点

loc　　玩家正在访问的地点/关卡

```
?loc = caves
  loadout 1
```

loc.id 当前位置的唯一标识符（英文）

```
var id
id = loc.id
>正在探索 @id@
```

loc.gp 当前循环期间使用过的装备的装备强度总和（只在具有排名的关卡生效）

```
>`0,1,循环中的装备强度为 @loc.gp@
```

loc.name 当前位置的本地化名称

```
>正在探索 @loc.name@
```

loc.stars 当前位置的难度（关卡星级，白5星为5星，青1星为6星，黄1星为11星，以此类推）

```
?loc = caves
  ?loc.stars=4
    loadout 1
  ?loc.stars=5
    loadout 2
```

loc.begin 仅在开始进入任意地区时，time为0的时候，第一帧的结果为true。不包括由衔尾蛇之石自动开始的循环游戏。可以用来初始化变量

```
var i
?loc.begin
  i = 0
  ?loc = caves
    i = -100
```

loc.loop 只有由衔尾蛇之石进行循环之后，第一帧为true

```
var loopCount = 0
?loc.loop
  loopCount++
```

loc.isQuest 如果当前位置是传说或普通任务中的特殊位置，则为true；否则为false

```
?loc.isQuest
  >`0,1,我们正处在一个传奇任务发生的特殊地点中。
```

loc.averageTime 当前地点的平均完成时间。一个位置的平均时间是以加权的方式计算的，其中更新的完成时间更有价值，越旧的时间价值越低

```
>`0,2,平均时间是
^ @loc.averageTime@ 帧
```

loc.bestTime 当前位置的最佳完成时间(您的高分记录)

```
>`0,1,最佳时间是
^ @loc.bestTime@ 帧
```

encounter遭遇

特殊的遭遇目前仅发生在弱点活动中，可能在绿星关卡也将会开放。

encounter.isElite　　返回当前遭遇的是否是精英敌人

```
>`0,1,当前是否遭遇了精英怪： @encounter.isElite@
```

encounter.eliteMod　　显示当前遭遇的敌人是否有特殊的修改器，详细内容见附录H.特殊修改器一览

```
>`0,2,当前的修改器为 @encounter.eliteMod@
```

foe敌人

foe　　当前被玩家锁定的目标敌人

```
?foe = boo
  equip 生命长杖
```

foe.id　　玩家瞄准的敌人的唯一ID(或类型)

foe.name　　目标敌人的本地化名称

foe.damage　　目标敌人每次攻击会对玩家造成的伤害

```
>`0,1,敌人的伤害是 @foe.damage@
```

```
//可以使用这个命令，检测狄斯安琪洛斯完全体的护盾眩晕（其固定造成1点伤害）并躲避
?foe = dysangelos_perfected & foe.damage = 1 & foe.state = 32 & foe.time = 66
  equipR Mind Stone
```

foe.distance　　目标敌人与玩家之间的距离

foe.z　　目标敌人的z位置

foe.count　　40单位（距离）内敌人的数量

foe.GetCount(int)　　*返回int*　　一定距离单位内的敌人数量

foe.hp　　目标敌人的当前生命值

foe.maxhp　　目标敌人的最大生命值

foe.armor　　目标敌人的当前护甲

foe.maxarmor　　目标敌人的最大护甲

foe.buffs.count　　目标敌人身上buff（增益效果）的数量

foe.buffs.string　　关于目标敌人所有buffs的综合信息

```
?foe.buffs.count > 0
  >`0,3,敌人的增益效果是 @foe.buffs.string@
```

foe.buffs.GetCount(str)　　目标敌人的特定buff的层数

foe.buffs.GetTime(str)　　目标敌人的特定buff的持续时间

foe.debuffs.count　　目标敌人身上debuff（负面效果）的数量

foe.debuffs.string　　关于目标敌人所有debuffs的综合信息

```
?foe.debuffs.count > 0
  >`0,4,敌人的负面效果是 @foe.debuffs.string@
```

foe.debuffs.GetCount(str)　　目标敌人的特定debuff的层数

```
>`0,1,冰冻debuff的层数是:
^ @foe.debuffs.GetCount("debuff_chill")@
```

foe.debuffs.GetTime(str)　　目标敌人的特定debuff的持续时间

```
>`0,2,冰冻debuff的持续时间是:
^ @foe.debuffs.GetTime("debuff_chill")@
```

foe.state　　代表目标敌人当前状态的数字

```
?foe.state = 0
  >`0,0,敌人正在休眠
```

foe.time　　目标敌人在当前状态（foe.state）下经过的帧数

```
>`0,0,敌人状态: @foe.name@:@foe.state@,@foe.time@
```

foe.level　　目标敌人的等级

```
>`0,0,敌人:@foe.name@ 等级为 @foe.level@
```

item装备

item.left　　左手上正在装备的物品

```
>`0,1,左手:@item.left@
>`0,2,右手:@item.right@
```

item.right　　右手上正在装备的物品

```
?item.right = 铁头长杖
  activate R
```

item.left.gp / item.right.gp　　装备在左手或右手的物品的装备强度

```
>`0,1,左手装备的装备强度:@item.left.gp@
>`0,2,右手装备的装备强度:@item.right.gp@
```

item.left.id / item.right.id　　装备在左手或右手的物品的ID

```
>`0,1,左手装备的ID:@item.left.id@
>`0,2,右手装备的ID:@item.right.id@
```

item.left.state / item.right.state

装备武器的当前状态，其中1是待机，2是前摇，3是后摇，4是冷却

item.left.time / item.right.time

装备武器在当前状态下的已用帧数

```
>`0,1,@item.left.state@:@item.left.time@
>`0,2,@item.right.state@:@item.right.time@
```

item.potion　　当前酿造好的药剂。如果坩埚上启用了自动酿造，这个关键词就会包括`auto`

```
?item.potion ! empty & item.potion = auto
  activate potion
```

harvest/pickup收取

harvest　　下一个可收获的物体，例如大树和巨石（但是这里不能使用中文）

```
?harvest=Boulder
  equip 石锹
```

harvest.distance　　玩家与最近的可收获物体之间的距离

```
?loc=Rocky & harvest.distance < 7
  equip 石锹
```

harvest.z　　最近可收获物体的z位置

```
?loc=Rocky & harvest.z > 5
  equip 石锹
```

pickup　　当前被玩家选为目标的拾取物品

```
?pickup
  equip 星之石
:
  loadout 1
```

pickup.distance　　玩家与目标拾取物品的距离

pickup.z　　目标拾取物品的z位置

```
?pickup.z > 7
  equipL star stone
```

time时间

time　　在当前地区的帧号（有的地图boss房间和道中是分开的两个地区）

```
?time % 300 = 0
  >每10秒你就会看到此消息
```

totaltime　　在当前大地区的帧号，进入boss房也会累计（进入boss房间时，time会重置为0，totaltime不重置；当重新进入关卡或者循环时，time和totaltime都会重置为0）

```
>`0,0,目前时间： @totaltime@ 帧
```

time.msbn　　Unix时间戳，表示自 1970-01-01t 00:00:00Z后经过的毫秒数（1970年1月1日，世界标准时间上午12:00）；不考虑闰秒

```
>@time.msbn@
```

time.year / time.month / time.day / time.hour / time.minute / time.second

玩家计算机上的本地系统时间

```
>`0,0,@time.year@/@time.month@/@time.day@
^ @time.hour@:@time.minute@:@time.second@
```

utc.year / utc.month / utc.day / utc.hour / utc.minute / utc.second

当前的UTC时间（世界标准时间）

```
var utcZ = utc.year*356*12*24*30 +
^utc.month*12*24 + utc.day*24+utc.hour
var timeZ = time.year*356*12*24*30 +
^time.month*12*24 + time.day*24+time.hour
var timeZone = timeZ - utcZ
?timeZone < 0
  >`0,0,时区: UTC@timeZone@
:
  >`0,0,时区: UTC+@timeZone@
```

player玩家

armor　　玩家当前的护甲值，向下取整。例如：如果护甲值为"2.4"，armor 取值为2

armor.f　　玩家当前的护甲值的小数点后第一位。例如：如果护甲值为"2.4"，armor.f 取值为4

buffs.count　　玩家身上buff（增益效果）的数量

buffs.string　　关于玩家的所有buffs的综合信息

```
?buffs.count > 0
  >`0,1,玩家的增益效果是 @buffs.string@
```

buffs.GetCount(str)　　玩家身上的特定buff的层数

buffs.GetTime(str)　　玩家身上的特定buff的持续时间

buffs.oldest　　玩家身上最早附加的buff的ID（只有存在的时候才有返回值，否则会报错）

```
>`0,1,最早的buff: @buffs.oldest@
```

debuffs.count　　玩家身上debuff（负面效果）的数量

debuffs.string　　关于玩家的所有debuffs的综合信息

```
?debuffs.count > 0
  >`0,2,玩家的负面效果是  @debuffs.string@
```

debuffs.GetCount(str)　　玩家身上的特定debuff的层数

debuffs.GetTime(str)　　玩家身上的特定debuff的持续时间

debuffs.oldest　　玩家身上最早附加的debuff的ID（只有存在的时候才有返回值，否则会报错）

```
>`0,1,最早的debuff: @debuffs.oldest@
```

hp　　玩家的当前生命值

maxhp　　玩家的最大生命值

maxarmor　　玩家的最大护甲值

pos.x　　玩家当前的X坐标

pos.y　　玩家当前的Y坐标

pos.z　　玩家当前的Z坐标

ai.enabled　　如果意念石AI启用，则为true；如果意念石AI关闭，则为false（例如，在过场动画期间）

ai.paused　　如果意念石AI暂停，例如在等待宝箱掉落时，则为true

ai.idle　　如果玩家空闲，等待攻击之类的事情完成，则为true

ai.walking　　如果玩家正在移动，则为true

bighead　　如果玩家使用了月晷石开启了大头模式，则为true

face　　玩家当前的面部表情

```
?face = "^^"
  >高兴
```

res.stone　　石头 / res.wood　木头 / res.tar焦油 / res.ki 气 / res.bronze 青铜

以上是玩家当前库存中的五种资源的数量

```
?loc = Deadwood
  >木头: @res.wood@ 个
```

```
>`0,1,#magenta, ♦ @res.crystals@
```

player.direction　　指示玩家面对的方向；朝右时返回值为1，朝左时返回值为 -1

```
?player.direction = 1
  >`0,0,朝右走
:
  >`0,0,朝左走
```

player.framesPerMove　　玩家向前移动一个位置所需的帧数

```
>`0,1,玩家移动力: @player.framesPerMove@
```

player.name　　玩家给自己起的名字

```
// 这个例子让玩家能够在自己头上打字:
var name
var x
name = player.name
x = string.Size(name) / -2
>o@x@,-2,@name@
```

player.GetNextLegendName()　　玩家尚未完成的下一个传奇任务的本地化名字

totalgp　　根据装备星级和附魔等级计算得出的总“装备分数”

```
>我的装备分数为 @totalgp@
```

summon召唤物

summon.count　　游戏中当前召唤的盟友数量

```
?summon.count = 0
  equipL talisman
  activate L
```

summon.GetId(index = 0)　　返回给定索引处召唤的ID，index参数是可选的，默认为0(第一个召唤物)；如果该索引处没有召唤物，则返回null。

```
?summon.GetId() ! "cinderwisp"
  equipR fire talisman
  activate R
```

summon.GetName(index = 0) 返回给定索引处召唤的本地化名称，index参数是可选的，默认为0(第一个召唤物)；如果该索引处没有召唤物，则返回null。

```
>`0,1,召唤物:@summon.GetName()@
```

summon.GetVar(varName, index = 0) 返回一个召唤物的自定义变量（召唤物的技能）的层数；不同类型的召唤物有着他们独特的能力，因此有不同的变量。index参数是可选的，默认为0(第一个召唤物)；如果该索引处没有召唤物，则返回null。如果varName与有效变量不对应，则显示错误。

```
?summon.GetId() = cinderwisp & summon.GetVar("ignition") > 2
  activate cinderwisp
```

summon.GetState(index = 0) 返回一个代表当前召唤物状态的数字。index参数是可选的，默认为0(第一个召唤物)；如果该索引处没有召唤物，则返回null。

```
>`0,1,召唤物状态为 @summon.GetState()@
```

summon.GetTime(index = 0) 返回召唤物在当前状态下经过的帧数。index参数是可选的，默认为0(第一个召唤物)；如果该索引处没有召唤物，则返回null。

```
>`0,1,召唤物状态时间为 @summon.GetTime()@
```

screen屏幕/other其他

key 自定义输入的状态（详情请看本文的15.自定义输入）。

```
// 以下显示当前输入：
>@key@
```

rng 返回一个0到9999之间的随机数

```
?rng < 5000
  >头!
:
  >尾巴!
```

```
// 下面这个例子可以生成5到24之间的随机数：
var min = 5
var max = 24
var n = min + rng % (max - min + 1)
```

rngf 返回一个0到1之间的随机浮点数（小数）（应当是返回一位小数）

```
>随机浮点数： @rngf@
```

```
// 下面这个例子可以生成5.0到24.0之间的随机浮点数：
var min = 5.0
var max = 24.0
var n = min + (max - min) * rngf
```

input.x　　输入设备（鼠标 / 触摸）在屏幕ASCII网格上的X方向位置

input.y　　输入设备（鼠标 / 触摸）在屏幕ASCII网格上的Y方向位置

```
> (@input.x@, @input.y@)
```

screen.i　　游戏中的场景屏幕的位置编号，当玩家到达屏幕右侧而屏幕滑动时，这个位置会增加

```
>`0,0,屏幕编号为 @screen.i@
```

screen.x　　游戏中的场景屏幕的位置

```
>`0,0,屏幕位置为 @screen.x@
```

screen.w　　屏幕的ASCII网格的宽度

```
var sw = screen.w
>屏幕宽度 = @sw@
```

screen.h　　屏幕的ASCII网格的高度

```
var sh = screen.h
>屏幕高度 = @sh@
```

·:·:· 4. 命令 Commands ·:·:·

这些代码指令能够告诉游戏系统应该做点什么。

activate (ability)　　激活物品技能。(ability)可以取以下值：potion, P 药水, left, L 左手技能, right, R 右手技能，F召唤物技能，同时也能搞使用技能的ID来使用技能（参考附录A.能力冷却ID）

```
activate R
// 激活右手装备的物品，例如斧头的技能
```

brew (ingredients)　　将药水瓶重新按照指定的成分炼制。仅在开始运行脚本时，时间为0的时候执行，即 `loc.begin = true` 时。成分可以是石头stone、木头wood、焦油tar或青铜bronze，应该用+分开。成分名称可以用英语或设置中选择的本地化语言书写。

```
?loc.begin
  brew bronze + tar
```

equip (str)　　装备一个物品。(str) 为装备英文或本地化名称，其可以增加最多7个条件限制，双手装备必须使用这个形式的指令

```
equip 活力重弩 *8 +5
```

equipL (str)　　在左手装备一个最符合条件的优先级最高的物品

```
equipL 剧毒长剑
```

equipR (str)　　在右手装备一个最符合条件的优先级最高的物品

```
equipR 活力盾牌
```

equip @var@　　根据字符串变量提供的标准装备物品；装备和其他物品搜索命令支持减法标准

```
var weaponName = "poison sword *10 -big"
equipR @weaponName@
```

loadout (n)　　激活指定编号的装备组合。在背包界面装备武器后，按Ctrl+数字即可将组合保存到对应数字

```
?loc = caves
  loadout 1
?loc = deadwood
  loadout 2
```

> (str)　　在屏幕顶部打印一个字符串(str)。

```
> Hello World!你好，世界！
```

> @varName@　　打印一个引用了变量值的内容的字符串。单次打印可以插入多个变量。请在变量名旁加上@以引用。

```
// 以下示例将打印有关当前目标敌人的信息：
var foeInfo
foeInfo = foe
>敌人是 @foeInfo@
```

>(abcd　　在玩家上显示自定义的面部表情，需要大脑袋模式。

```
>( OwO
```

>oX,Y,[#rrggbb,](str)　　相对于玩家位置，在图层最顶层进行打印。X和Y是坐标偏移。#rrggbb是十六进制表示法的文本颜色。颜色也可以使用以下常量设置：＃white, ＃cyan, ＃yellow, ＃green, ＃blue和＃red。如果要以彩虹色打印，请使用#rainFF，其中后2个字符定义亮度。

```
//  下面的示例用红色字色写了“我们上！”。相对于玩家的位置向左偏6个单位距离，向下偏3个单位距离：
>o-6,3,#red,我们上！
```

>hX,Y,[#rrggbb,](str)　　类似于“>o”，不同的是，和大脑袋绘制在相同图层上（比顶层图层更低）；**是画帽子等配件的理想选择**

```
//  以下代码在玩家头上绘制了一顶黄色的帽子（需要大脑袋）：
>h-2,-3,#yellow,ascii
##_
#| |
_|_|_
asciiend
```

```
// 使用两种不同的代码顺序来观察">o"和">h"的区别，可以发现">o"的图层总在上面
>h3,3,这是第一个执行的图层
>o3,3,这是第二个执行的图层
```

```
>o3,3,这是第一个执行的图层
>h3,3,这是第二个执行的图层
```

>`X,Y,[#rrggbb,](str)　　相对于屏幕左上角位置，在图层最顶层进行打印

```
以下例子打印“你好，世界!”，但使用变量作为坐标和颜色：
var posX = 10
var posY = 5
var color = rainE1
>`@posX@,@posY@,#@color@,你好，世界!
```

>cX,Y,[#rrggbb,](str)　　相对于屏幕中心的位置，在图层最顶层进行打印。类似于“>`”。**注意，所有这些都是颜色选填的**

```
// 本示例演示颜色如何为选填，默认为白色：
>c0,0,你好，世界!
```

>fX,Y,[#rrggbb,](str)　　相对于目标敌人头部位置，在图层最顶层进行打印。

```
// 以下代码在目标敌人上绘制了红色十字准线：
>f-2,0,#ff0000,ascii
##!
-#·#-
##¡
asciiend
```

var (variable)　　声明一个可以在数学，逻辑和字符串运算中使用的变量。有关变量的作用范围、生命周期和行为的详细信息，请参见 7.变量部分，这可能与其他编程语言有所区别

```
var message = 你好，世界!
>@message@
```

func (function)　　声明一个可以在之后调用的函数，详见 9.函数

```
func Print(message)
  >@message@

Print(你好，世界！)
```

for v = a..b　　创建一个循环，将变量v从值a迭代到值b。出现在循环范围内的代码将运行多次

```
var a
a = 0
for i = 1..5
  a = a + i
>a = @a@
```

import (script)　　加载并执行外部脚本的单一副本（即对一个外部脚本的所有的import 都是去加载同一份副本），详见11.导入外部脚本

```
// 这个示例导入的钓鱼小游戏位于
//（游戏存档）/Stonescript/Fishing.txt
import Fishing
```

new (script)　　加载并执行类似于“导入” 的外部脚本。但是，加载了“ new”的对象每个都是单独的副本，每个对象只会执行一次。

```
//本示例创建并打印一个矢量对象：
var v = new Components/Vector
v.Init(10, 5)
>Vector = @v@
```

disable abilities　　阻止药剂和武器能力的激活。也使界面HUD按钮变灰

enable abilities　　恢复被先前的“disable abilities”禁用的能力激活

disable banner　　禁止在地点开头和结尾显示位置名称的水平横幅

enable banner　　恢复显示名称的水平横幅的渲染

disable hud(opts)　　隐藏和禁用游戏用户界面元素。接受可选参数以指定要禁用的元素集：p =玩家生命值和减益，f =敌人生命值和减益，a =能力按钮，r =资源，b =旗帜，u =多用途腰带。

```
disable hud // 停用所有HUD元素
```

```
disable hud ru // 仅禁用资源(r)和多用途皮带(u)
```

enable hud(opts)　　恢复由上一个“disable hud”命令隐藏的所有用户界面元素，接受与disable命令相同的可选参数。

disable loadout input　　禁止使用输入键保存或调用武器装载组合

enable loadout input　　恢复通过输入键保存或调用武器装载组合

disable loadout print　　隐藏调出装备组合时出现的消息

enable loadout print　　恢复打印调出装备组合时的消息

disable npcDialog　　隐藏并自动跳过NPC对话框气泡（需要注意使用loc进行限制，以防止在进行任务的时候跳过任务对话）

enable npcDialog　　恢复NPC对话框气泡

disable pause　　隐藏用户界面的暂停按钮，但使用[P]快捷键仍可暂停

enable pause　　恢复被之前的“disable pause”命令隐藏的暂停按钮

disable player　　隐藏玩家角色。对战斗没有影响，这仅仅是视觉效果

enable player　　如果先前的“disable player”命令隐藏了玩家角色，则将其恢复

play (sound) (pitch)　　播放可选音量值的声音效果，具体见附录B.音乐效果；音量高默认值为100，数字越大音高越大，数字越小音高越小。

```
?key = primary
  play buy
?key = up
  play buy 200
```

```
var pitch
?time%30 = 0
  pitch = rng/100 + 50
  >@pitch@
  play buy @pitch@
```

·:·:· 5. 搜索过滤器/关键词 Search Filters ·:·:·

这些是在搜索敌人、地点和物品时使用的：

```
?foe = insect | foe = poison
  loadout 3
```

对于中文来说，如果你直接利用中文搜索并不一定能符合（大概率在代码里面跑不通），所以建议对照着去搜索英文。

英文	中文
poison	剧毒
vigor	活力
aether	以太
fire	火焰
air	气
ice	寒冰
arachnid	蜘蛛类

英文	中文
serpent	蛇形
insect	昆虫类
machine	机器类
humanoid	人形
elemental	元素种
boss	头目
phase1	第1阶段
phase2	第2阶段
phase3	第3阶段
spawner	产卵者，巢穴类
flying	飞行的
slow	缓慢的（例如超大蜗牛，炸弹推车）
ranged	远程的
explode	爆炸的（例如毛毛，炸弹推车）
swarm	蜂群，一大群的
unpushable	不可击退的（例如巢穴类）
undamageable	不可损坏的
magic_resist	魔法抗性
magic_vulnerability	弱魔法的
immune_to_stun	免疫击晕
immune_to_ranged	免疫远程攻击（如枯木峡谷的石龟子）
immune_to_debuff_damage	免疫中毒debuff（中毒debuff是减攻击的debuff）
immune_to_physical	免疫物理伤害（如亡者之殿的鬼魂）
*[number] star level (location or item)	*[数字] 地点或物品的星级
+[number] enchantment bonus (item only)	+[数字] 物品的附魔等级

更多相关内容请参照附录E.中英文对照表。

·:·: 6. 比较符号 Comparisons ·:·:

与游戏状态结合使用来制定决策

= 比较数值是否相等，或字符串是否包含。如果是，则返回true

```
?hp = maxhp
  loadout 2
```

! 比较数值是否相等，或字符串是否包含。如果否，则返回true

```
?foe ! poison
  equipL 长剑
```

& 逻辑运算符“与”。

```
?loc=caves & foe=boss
```

| 逻辑运算符“或”。如果“&”和“|” 混合在单个复杂的表达式中，所有的“&”优先

```
?foe=slow | foe.count>3
  activate potion
```

> 比较数值是否大于。可以用来比较关卡的难度，敌人的数量，健康状况等参数

```
?foe.count > 10
  equip 巴迪什 shiny
```

< 比较数值是否小于

```
?hp < 6
  activate potion
```

>= 比较数值是否大于或者等于；“>”和“=”的组合成一个比较

```
// 以下两个示例是等效的：
?loc.stars >= 6
  equipR 活力盾牌
```

```
?loc.stars > 6 | loc.stars = 6
  equipR 活力盾牌
```

<= 比较数值是否小于或者等于；“<”和“=”的组合成一个比较

```
?hp <= 6
  activate potion
```

·:·: 7. 变量 Variables ·:·:

变量是一种存储数值以备后用的方式。用关键字“var”声明一个新变量：

```
var myVar = 10
// myVar是变量的名称，为其赋初值为10
myVar = myVar + 5
// 现在myVar 等于15
var secondVar = 3
myVar = myVar - secondVar
// 许多变量可以用于数学运算中，myVar现在等于12
```

一个变量被声明后，它只会被初始化一次，即第一次执行“ var”的时候

```
var i = 0
i++
>i = @i@
```

在这个示例中，声明了变量“i”并为其初始赋值为0；在运行的每一帧期间，“i”的值增加1，然后打印到屏幕上。

只有当你离开关卡并手动开始新的运行时，变量才会重置为其初始值。当利用衔尾蛇之石循环时，这些变量都不会重置。

变量可以选择用引号声明为**字符串**类型的值。这时变量中可以包含特殊符号和空格。

例如：

```
var a = 10
var b = 5
var myVar = a + " x " + b + " = " + (a * b)
>@myVar@
// 此实例的输出结果是 "10 x 5 = 50"
// 此示例还演示了如何使用运算符'+'连接字符串
```

导入脚本中的变量包含在那些脚本中，并且变量名称不会与其他脚本中的变量或函数冲突，详情请见 11. 导入外部脚本。

在Stonescript中你可能遇见不同的变量类型，这些变量类型都可以在被“var”声明的时候通过初始化来实现。当使用变量的时候，也可以通过重新幅值不同类型的数据转换类型。以下是在使用过程中可能使用到的变量类型：

变量	描述
int	int32类型，只能是整数，范围为-2147483648~2147483647，当进行数学运算结果为小数的时候会自动舍去小数点及以后的部分。当变量声明为如“1”、“6”、“-10”这样的整型时使用。
float	单精度浮点数float32类型，可以是小数，最大范围能够取到-3.4e38~3.4e38，能表示的最小值为1.4e-45，具表示。精度较低，仅有7位有效数字，超出部分都是用科学计数法进行表示。当变量声明为如“1.0”、“6.3”、“-10.5335”这样的浮点数时使用。

变量	描述
bool	布尔类型，只有true和false两种取值，当使用其他类型转换为布尔型时，0为false，非0为true。
string	字符串类型，用双引号引起来的几个字符，如"Abc","一天"，在stonescript中双引号可能在显示上被简化。
array	数组类型，有序的元素序列，储存多个相同类型的变量，例如int[3]就是储存了3个int类型的变量。数组的第一个序号从0开始，例如长度为10的数组中的数据储存在a[0]~a[9]。详细见14.数组。
BigNumber	大数类型，一种特殊的用于处理超过32位的大整数的方法。以字符串的格式创建超过int32范围的整型数字，使用专门的函数与其他数字的计算，没有重载普通的数学符号。详细见BigNumber大数型

·:·:· 8. 数学运算 Math Operations ·:·:·

运算符能够对数字进行修改。其可以与变量结合使用，也可以直接在游戏状态的表达式中使用。

+　　将两个数字或变量相加

```
var a = 2 + 3
// a等于5
```

-　　将两个数字或变量相减

```
?hp < maxhp - 5
  equip 活力长剑 dL
```

*　　将两个数字或变量相乘

```
var a = 2
var b = 5
a = a * b
// a等于10
```

```
var a = 2 * 0.4
// a等于0.8
```

/　　将一个数字或变量除以另一个，结果向下取整。

```
var a = 8
a = a / 4
// a等于2
```

```
var a = 5.0
a = a / 2
// a等于2.5
```

++　　递增一个变量

```
var a = 3
a++
// a等于4
```

--　　递减一个变量

```
        例如:
var a = 3
a--
// a等于2
```

%　　取模。即给出将一个数字除以另一个数字后的余数

```
var a = 5 % 4
// a等于1
```

```
?time % 8 < 4
  > \O)
?time % 8 > 3
  > (O/
// 这将绘制一个动画表情符号
```

()　　括号可用于更改运算的优先级

```
var a = 2 * (3 - 1)
// a等于4
```

!　　否定。在是否判断（布尔bool表达式）前面使用时反转该值

```
?!ai.enabled
  >未启用AI.
```

·:·:· 9.函数 Functions ·:·:·

当脚本变得越来越复杂时，自定义函数起到了重要的组织作用。函数可以减少脚本重复，增加脚本可读性。声明函数时，其包含的内容不会被立即执行。只有在之后调用函数时，函数内部的脚本才会执行。**要注意，由于函数内部的代码都属于这个函数，所以应当注意函数声明和函数体的空格。**

本示例创建了一个每帧增加1的计数器。当按下任何自定义输入键时，调用函数ResetCounter()，通过这个函数使计数器值归零：

```
var count = 0
count++
>计数为 @count@

func ResetCounter()
  count = 0

?key=begin
  ResetCounter()
```

另一方面，函数可以返回一个值。在此示例中，我们声明一个简单的函数来计算我们在道中区域（非boss的区域）使用了多长时间：

```
func NonBossDuration()
  return totalTime - time

var duration
duration = NonBossDuration()
>时间是 @duration@
```

通过 `return` 来返回参数，同时也标志着函数的结束；如果在函数脚本中有多个“return”，则只会执行到第一个，这个“return”后面的代码将都不再执行。在此示例中，位于不同位置的信息将分别打印，展示“return”对函数体的中断作用：

```
func ReturnValueCheck()
 ?true
  >`0,0,第一个返回前的内容
  return 执行了第一个返回
  >`0,2,第一个返回后的内容
 :
  return 执行了第二个返回

>`0,1,@ReturnValueCheck()@
```

函数还可以接受任意数量的参数/参量，从而使其功能更强大，并且可以对任意的输入值进行操作。在这里，我们声明一个效用函数，该函数会生成一个输入其参数的范围内的随机数，然后我们再用它生成5到10之间的随机数：

```
func RandomRange(min, max)
  // 当最小值大于最大值时，直接返回最小值
  ?min >= max
    return min
  //
  return min + rng % (max - min + 1)

var randomValue
randomValue = RandomRange(5, 10)
>RNG: @randomValue@
```

当引用属于外部脚本的变量时，可以在函数内部使用前缀 `this` 。“this”的使用是可选的，脚本变量可以由不带前缀的函数访问。但是，在以下示例中，变量“a”同时出现在函数的内部和外部，因此要使用“this.a”进行区分：

```
var a = 1
func TestScope(a)
  >脚本中的var的值为 @this.a@,函数中的var的值为@a@

TestScope(3)
```

当函数调用其他函数或它们自身时，这将创建一个执行堆栈，该堆栈的规模会不断扩大，直到应用程序崩溃为止。为了保护计算机资源，Stonescript的堆栈限制为215，如果超出该限制，则会报错。

·:·:· 10.本地函数 Native Functions ·:·:·

在我们可以通过脚本定义自己的函数的同时，Stonescript也附带了一组预定义/本机函数，其行为类似于命令，但不同之处在于它们更明确地按主题分组，可以接受参数，有些还有返回值；这些函数可以接受的参数类型和返回值类型的说明都包含在7.变量当中。

ambient背景音乐

ambient　　*返回string*　　返回所有活动环境音频id的逗号分隔列表

```
>`0,0,环境音效为 @ambient@
```

ambient.Add(str)　　*无返回值*　　使用给定的声音ID向环境音频层中增加音效，且最多4层；如果添加了第5层，则最老的层将被删除

```
?loc.begin
  ambient.Add(ambient_crypt)
```

ambient.Stop()　　*无返回值*　　清除所有音频环境层

```
?time = 3
  ambient.Stop()
  ambient.Add(ambient_mines)
```

BigNumber大数型

大数型是一种特殊类型的对象，它提供了一种处理超过32位的大整数的方法，可以以字符串的格式创建超过int32范围的整型数字（并上不封顶）。虽然大数型只能保存正整数和负整数，但它们的算术和比较操作可以处理浮点数、整数和其他大数。大数型与其他数字的计算需要使用专门的函数，没有重载普通的数学符号。

b.Add(num) / b.Add(BigNumber)　　*返回大数型*　　把一个数字加到大数型上（+）

```
var bn = math.BigNumber(12)
bn.Add(5)
>@bn@
// 17
```

b.Sub(num) / b.Sub(BigNumber)　　*返回大数型*　　从大数型减去一个数字（-）

```
var bn = math.BigNumber(12)
bn.Sub(5)
>@bn@
// 7
```

b.Mul(num) / b.Mul(BigNumber)　　*返回大数型*　　用一个数字和大数型相乘（*）

```
var myBigNum1 = math.BigNumber(12)
var myBigNum2 = math.BigNumber(12)
myBigNum1.Mul(5)
myBigNum2.Mul(1.5)
>@myBigNum1@ @myBigNum2@
// 60 18
```

b.Div(num) / b.Div(BigNumber)　　*返回大数型*　　用大数型除以一个数字（/）

```
var myBigNum1 = math.BigNumber(12)
var myBigNum2 = math.BigNumber(12)
myBigNum1.Div(5)
myBigNum2.Div(1.5)
>@myBigNum1@ @myBigNum2@
// 2 8
```

b.Eq(num) / b.Eq(BigNumber)　　*返回大数型*　　判断大数型是否等于一个数字（=）

```
var bn = math.BigNumber(5)
>@bn.Eq(5)@ @bn.Eq(3)@
// true false
```

b.Gt(num) / b.Gt(BigNumber)　　*返回大数型*　　判断大数型是否大于一个数字（>）

```
var bn = math.BigNumber(5)
>@bn.Gt(3)@ @bn.Gt(10)@
// true false
```

b.Ge(num) / b.Ge(BigNumber)　　*返回大数型*　　判断大数型是否大于等于一个数字（>=）

```
var bn = math.BigNumber(5)
>@bn.Ge(3)@ @bn.Ge(5)@
// true true
```

b.Lt(num) / b.Lt(BigNumber)　　*返回大数型*　　判断大数型是否小于一个数字（<）

```
var bn = math.BigNumber(5)
>@bn.Lt(3)@ @bn.Lt(10)@
// false true
```

b.Le(num) / b.Le(BigNumber)　　*返回大数型*　　判断大数型是否小于等于一个数字（<=）

```
var bn = math.BigNumber(5)
>@bn.Le(10)@ @bn.Le(5)@
// true true
```

b.ToFloat()　　*返回float*　　将一个大数型转换为float，如果数字太大或者太小就会提供一个报错

```
var bn = math.BigNumber(5)
var fNumber = bn.ToFloat()
>float = @fNumber@
```

b.ToInt()　　*返回int*　　将一个大数型转换为int，如果数字太大或者太小就会提供一个报错

```
var bn = math.BigNumber(5)
var iNumber = bn.ToInt()
>integer = @iNumber@
```

b.ToString()　　*返回string*　　返回B大数型的字符串形式，可用于将其序列化并以供存储和读取

```
var myBigNum = math.BigNumber("123456789123456789")
storage.Set("myBN", myBigNum.ToString())
---
var bnStr = storage.Get("myBN")
var myBigNum = math.BigNumber(bnStr)
```

b.ToUI()　　*返回string*　　返回用于用户界面的大数型的缩短字符串表示形式

```
var myBigNum = math.BigNumber("123456789123456789")
>@myBigNum.ToUI()@
// 123.5Qa
```

color颜色

Stonescript中的颜色用RGB转十六进制颜色表示法中的字符串表示，如#ff0000，或者用简化的预置表示，如#red

color.FromRGB(r,g,b)　　*返回string*　　将颜色从三个整数(0到255)转换成字符串#的形式

```
var c = color.FromRGB(255, 0, 128)
>`0,0,@c@, @c@
```

color.ToRGB(string)　　*返回int[3]*　　将颜色从字符串转换为三个整数(0到255)

```
var c = color.Random()
var rgb = color.ToRGB(c)
var r = rgb[0]
var g = rgb[1]
var b = rgb[2]
>`0,0,@c@, @c@ \n @r@ \n @g@ \n @b@
```

color.Lerp(c1,c2,t)　　*返回string*　　计算从颜色c1到c2的在比例t (百分比)处的线性插值

```
var c1 = "#ff4400"
var c2 = "#8888ff"
var t = 0.5
var c
t = math.sin(time*0.1) / 2 + 0.5
c = color.Lerp(c1, c2, t)
>`0,1,@c@,@c@\n ██████
```

color.Random()　　*返回string*　　返回随机颜色

```
var c
c = color.Random()
>`0,0,@c@,@c@\n ██████
```

draw绘制

draw.Bg(x, y, color)　　*无返回值*　　设置屏幕上指定坐标的背景色

```
draw.Bg(5, 4, #red)
```

draw.Bg(x, y, color, w, h)　　*无返回值*　　设置屏幕上指定矩形区域的背景色

```
draw.Bg(5, 4, #cyan, 10, 6)
```

draw.Box(x, y, w, h, color, style)　　*无返回值*　　在指定的位置绘制指定大小的矩形。矩形的边框取决于颜色color和样式style号。负样式号会使矩形的填充透明。警告：此时，顶层打印类型会始终画在这一矩形盒子的上面。

```
// 用此示例通过按键盘的[左]/[右]来查看不同的style号：
var style = 1
?key = leftBegin
  style--
?key = rightBegin
  style++
draw.Box(10, 5, 30, 15, #333333, style)
>`12,6,#ffffff,当前Style号：@style@
>`12,8,#888888,按左/右\n改变style号以查看
```

draw.Clear()　　*无返回值*　　清空整个屏幕

draw.GetSymbol(x, y)　　*返回string*　　返回屏幕位置的（x，y）的符号

```
//本示例选择一个屏幕坐标并在左上角绘制在那里找到的符号。选择[]可以由玩家被移动：
var s
var x = 20
var y = 10
var drawX
?key=leftBegin
  x--
```

```
?key=rightBegin
  x++
?key=upBegin
  y--
?key=downBegin
  y++
s = draw.GetSymbol(x, y)
>`0,1,Symbol = @s@
drawX = x - 1
>`@drawX@,@y@,[#]
```

draw.Player() / draw.Player(x,y) *无返回值*　　在脚本的特定位置绘制带有装备和插件（以`>( >o >h`打印的内容）的玩家角色。可选参数为相对于原来玩家绘制位置的偏移量x和y。如果要绘制到绝对屏幕位置，请参阅screen屏幕，并导出从玩家的本地位置转换为屏幕位置的偏移量。

int整型

int.Parse(str) *返回int*　　将数字字符串转换为整数值。如果输入的字符串不是数字字符串，则会报错。

```
var s = "999"
var i = int.Parse(s)
i++
```

event活动

event.GetObjectiveId(int)　　*返回string*

event.GetObjectiveProgress(int)　　*返回int*

event.GetObjectiveGoal(int)　　*返回int*

通过所需目标的索引返回有关社区或季节性事件中活动目标的信息——任务ID、任务进度或者任务目标。事件通常限制为最多3个活动目标，因此函数的所需参数应该是0、1或2。

```
var id
var p
var g
id = event.GetObjectiveId(0)
p = event.GetObjectiveProgress(0)
g = event.GetObjectiveGoal(0)
>`0,1,任务ID @id@ 的任务进度/任务目标为 @p@/@g@
```

item物品

item.CanActivate()　　*返回bool*　　在某些时刻如果物品技能可以使用，则返回true，否则返回false。在一些游戏场景中，所有的物品技能都会被禁用，即使它们冷却已经好了，比如在boss大战前或者电影中。

```
?item.CanActivate()
  equip Bardiche
  activate R
```

item.CanActivate(str) *返回bool* 如果可以激活特定的物品，则返回true。**只有在装备了该物品的情况下才能够使用，使用的能力字符串参数，请参见附录A“能力冷却ID”严格填写，而不能随便填写。**有些物品的机制要求它们除非满足特定的条件，否则不能被放技能。这个函数是item.GetCooldown()的一个子集，因为一个物品的冷却时间可能是零但某些时候不让用，但是如果冷却没有好，那一定不能放技能。

相关可以填写的参数中，"potion"是这一函数的特例，由于药水不用被装备，因此当可以喝药时候会返回true，动画期间或药水空瓶都会返回false.

```
equip bardiche
?item.GetCooldown("skeleton_arm") <= 0
  equip skeleton arm
  ?item.CanActivate("skeleton_arm")
    activate R
```

item.GetCooldown(str) *返回int* 返回给定物品技能的剩余冷却时间（以帧为单位），所有可用的技能ID见附录A.能力冷却ID。

```
?foe = boss & (item.GetCooldown("bardiche") <= 0 | item.GetCooldown("bardiche") >
890)
  equip bardiche
  activate R
```

有关 item.CanActivate(str) 和 item.GetCooldown(str) 使用的能力字符串参数，请参见附录A“能力冷却ID”严格填写，而不能随便填写。注意:无效的能力字符串将返回-1，如果没有对应的武器，技能结果也将返回-1；同时，附录A表格中列出了各个物品的冷却时间和伤害动画时间，使用时需要参考以上例子，判断是否武器已经完成动画并打出伤害；部分物品为瞬发，不需要考虑动画时间。

item.GetCount(str) *返回int* 返回库存中某个物品的数量。如果找不到任何相关物品，就返回0

```
var searchCriteria = "sword *0 -big -socket"
var swordCount = item.GetCount(searchCriteria)
>我有 @swordCount@ 把基础的长剑
```

item.GetLoadoutL(int) / item.GetLoadoutR(int) *返回string* 返回快捷武器栏腰带中预存的武器。整数参数是要查询的预存位置。如果该加载项在该槽中没有项目，则返回空字符串。

```
>`0,1,左手快捷武器栏1为：@item.GetLoadoutL(1)@
>`0,2,右手快捷武器栏1为：@item.GetLoadoutR(1)@
```

item.GetTreasureCount() *返回int* 返回当前库存中的宝箱数量

item.GetTreasureLimit() *返回int* 返回库存中宝箱的总数限制，或者说，最大容量

```
var trs
var max
trs = item.GetTreasureCount()
max = item.GetTreasureLimit()
>`0,2,宝箱数量/宝箱上限为 @trs@/@max@
```

key按键

key命名空间允许定制标准的游戏输入和快捷键。该系统基于动作action(缩写为“act”)和key按键，其中每个动作对应于一种类型的输入/快捷方式，每个按键对应于物理按键。可以在Unity文档找到可以分配给动作的所有可能的按键的列表。对动作绑定的更改在运行的时候保持不变(它们当前不保存到存储)。出于优化目的，建议不要每帧都更改绑定。

动作	默认按键	默认按键2
暂停	P	空格
终止	L	
背包	I	
意念石界面	M	
药水	Q	
左手物品技能	E	
右手物品技能	R	
上	W	Up方向键
下	S	Down方向键
左	A	Left方向键
右	D	Right方向键
主界面	回车键	小键盘回车键
返回	X	
能力1	左Shift	右Shift
能力2	左CTRL	右CTRL
撞击1	Z	
撞击2	C	
动态1	F	

动作	默认按键	默认按键2
动态2	T	
动态3	G	
动态4	V	
动态5	B	

key.Bind(act, key1) / key.Bind(act, key1, key2) *无返回值* 为特定动作分配一组新的按键；如果另一个动作已经占有了这些键中的一个，那么已经在使用的键将与原始动作取消绑定；一个动作最多可以分配两个键。

```
?loc.begin
  key.Bind("Potion", "P")
  // 在这个例子中，原本用于暂停的“P”键不再暂停游戏，而是激活药剂。此外，药剂原来的“Q”键不再有效，“Q”现在不和任何行动绑定。
```

key.GetKeyAct(key) *返回string* 返回绑定到给定键的动作；如果给定的键没有绑定到任何操作，则返回“None”。

key.GetActKey(act) *返回string* 返回绑定到给定动作的第一个键；如果给定操作没有绑定任何键，则返回“None”。

key.GetActKey2(act) *返回string* 返回绑定到给定动作的第二个键；如果给定操作没有绑定第二个按键，则返回“None”。

key.GetActLabel(act) *返回string* 返回一个面向用户的标签，表示绑定到给定动作的第一个按键。当前的实现返回了绑定键的第一个字母，但这在“LeftShift”这样的情况下可能会引起混淆。

key.ResetBinds() *无返回值* 将所有操作重置为默认的键绑定。

loc地点

loc.Leave() *无返回值* 离开关卡，就像玩家手动离开一样

loc.Pause() *无返回值* 游戏暂停，就像玩家手动暂停一样

math数学运算

math API可以处理整数和浮点数。当一个数字用小数点表示时，意味着它是一个浮点数(如 var a = 0.5)。如果没有小数，那么它将被视为整数(如 var a = 2)。在这一小节中，函数返回值为数字代表其返回的可能是int或是float

math.Abs(num) *返回数字* 返回给定数字的绝对值

```
var number = -2
number = math.Abs(number)
// 现在number等于 2
```

math.Sign(num) *返回数字* 如果给定的数字是负数，则返回-1。否则，返回1

```
var sign = math.Sign(-21)
var n = 10 * sign
// n等于 -10
```

math.Min(num1, num2)　　*返回数字*　　返回两个数字中较小的一个

```
var number = math.Min(3, 10)
// number等于 3
```

math.Max(num1, num2)　　*返回数字*　　返回两个数字中较大的一个

```
var number = math.Max(3, 10)
// number等于 10
```

math.Clamp(num, min, max)　　*返回数字*　　将数字num限制在最小值min和最大值max范围内。如果该数字已经在该范围内，则它将返回而不会更改

```
var number = 50
number = math.Clamp(number, 0, 10)
// number被限制范围降低，将等于 10
```

math.Round(num)　　*返回数字*　　将数字四舍五入到最接近的整数

```
var number = math.Round(2.7)
// number等于 3.0
```

math.RoundToInt(num)　　*返回数字*　　将数字四舍五入为最接近的整型数字int

```
var number = math.RoundToInt(2.7)
// number等于 3
```

math.Floor(num)　　*返回数字*　　将数字向下舍入到第一个比它小的整数

```
var number = math.Floor(2.7)
// number等于 2.0
```

math.FloorToInt(num)　　*返回数字*　　将数字向下舍入到第一个比它小的整型数字int

```
var number = math.Floor(2.7)
// number等于 2
```

math.Ceil(num)　　*返回数字*　　将数字向上舍入到第一个比它大的整数

```
var number = math.CeilToInt(4.2)
// number等于 5.0
```

math.CeilToInt(num)　　*返回数字*　　将数字向上舍入到第一个比它大的整型数字int

```
var number = math.CeilToInt(4.2)
// number等于 5
```

math.Lerp(a, b, t)　　*返回数字*　　计算从值a到b的在比例t (百分比)处的线性插值

```
var number = math.Lerp(0.0, 20.0, 0.75)
// number等于 15.0
```

```
var n = 0.0
?key = Begin
  n = 0.0
n = math.Lerp(n, 100, 0.02)
>n = @n@
// 变量n以每帧2%的速度向100移动。当一个按键被按下时，n就会复位
```

math.Log(num, base)　　*返回数字*　　返回数字的对数

```
var number = math.Log(5, 2)
// number等于 2.321928
```

math.Pow(num, p)　　*返回数字*　　返回数字的幂

```
var number = math.Pow(3, 2)
// number等于 9
```

math.Sqrt(num)　　*返回数字*　　返回数字的平方根

```
var number = math.Sqrt(9)
// number等于 3
```

math.e　　*float浮点数*　　常数e，也称为欧拉数或自然常数，约为2.71828

```
>E 等于 @math.e@
// 将e的值打印到屏幕上
```

math.Exp(num)　　*返回数字*　　计算e的n次幂

```
var number = math.Exp(3)
// number的结果等于 20.08554
```

math.pi　　*float浮点数*　　常数π，约为3.1415926

```
>PI 等于 @math.pi@
// 将π的值打印到屏幕上
```

math.ToDeg(num)　　*返回数字*　　将弧度转换为度数

```
var number = math.ToDeg(2 * math.pi)
// number等于360
```

math.ToRad(num)　　*返回数字*　　将度数转换为弧度

```
var number = math.ToRad(360)
// number等于 2π
```

math.Acos(num)　　*返回数字*　　返回以弧度表示的数字的反余弦值。输入范围是-1到1。如果输入值超出界限，则返回“NaN”

```
var number = math.Acos(-1)
// number等于 π
```

math.Asin(num)　　*返回数字*　　返回一个以弧度表示的数字的反正弦值。输入范围是-1到1。如果输入值超出界限，则返回“NaN”

```
var number = math.Asin(1)
// number等于 π/2
```

math.Atan(num)　　*返回数字*　　以弧度返回一个数的反正切值

```
var number = math.Atan(2)
// number等于 1.107149
```

math.Atan2(y, x)　　*返回数字*　　返回x轴与从原点到点（x，y）的直线之间的角度（以弧度为单位）

```
var number = math.Atan2(3, 2)
// number等于 0.9827937
```

math.Cos(num)　　*返回数字*　　返回给定弧度角的余弦值

```
var number = math.Cos(0)
// number等于 1
```

math.Sin(num)　　*返回数字*　　返回给定弧度角的正弦值

```
var number = math.Sin(math.pi / 2)
// number等于 1
```

math.Tan(num)　　*返回数字*　　返回给定弧度角的正切值

```
var number = math.Tan(2)
// number等于 -2.18504
```

math.BigNumber() / math.BigNumber(number) / math.BigNumber(str)　　*返回大数型*
由一个int或者float型数字、或者一个字符串创建一个对应的大数型

```
var myBigNum = math.BigNumber("500")
myBigNum.Add(500).Mul(1000).Mul(1000).Mul(1000)
>@myBigNum@
// 1000000000000
```

music音乐

music　　*返回string*　　返回当前播放音乐的ID

```
>`0,0,当前音乐为 @music@
```

music.Play(str)　　*无返回值*　　用给定的声音ID播放音乐。一次只能播放一首音乐

```
?loc.begin | loc.loop
  music.Play(temple_0)
```

music.Stop()　　*无返回值*　　停止所有音乐

```
?!string.Equals(music, "")
  music.Stop()
```

player玩家

player.ShowScaredFace(num)　　*无返回值*　　如果玩家启用了大头，他们的面部表情会在一段时间内变成害怕，时间为num帧

```
?key = primaryBegin
  player.ShowScaredFace(1)
```

screen屏幕

screen.FromWorldX(int)　　*返回int*　　将X轴上的值从地区空间转换到屏幕空间

screen.FromWorldZ(int)　　*返回int*　　将地区空间Z轴的值转换为屏幕空间Y轴的值

```
var x
var y
x = screen.FromWorldX(pos.x)
y = screen.FromWorldZ(pos.z - pos.y)
>`0,1,玩家在屏幕上的坐标为 @x@,@y@
```

screen.ToWorldX(int)　　*返回int*　　将X轴上的值从屏幕空间转换到地区空间

screen.ToWorldZ(int)　　*返回int*　　将值从屏幕空间的Y轴转换为地区空间的Z轴

```
var x
var y
var z
x = input.x
y = input.y
>`0,1,光标的屏幕位置为 @x@,@y@

x = screen.ToWorldX(input.x)
z = screen.ToWorldZ(input.y)
>`0,2,光标的地区位置为 @x@,@z@
```

screen.Next()　　*无返回值*　　对于占据多个屏幕的地区位置，将摄像机相对于玩家向前移动一个屏幕；上限是前移一个屏幕，只有当玩家出现在此屏幕内才能进行下一次移动

```
?key = rightBegin
  screen.Next()
```

screen.Previous()　　*无返回值*　　对于占据多个屏幕的地区位置，将摄像机相对于玩家向后移动一个屏幕；当玩家走到下一个屏幕的时候会取消效果重新跟随

```
?key = leftBegin
  screen.Previous()
```

screen.ResetOffset()　　*无返回值*　　重置摄像机以跟随玩家，撤消screen.Next()和screen.Previous()对屏幕所做的更改

```
var lastScreenI = -1
?lastScreenI ! screen.i
  screen.ResetOffset()
lastScreenI = screen.i
```

storage储存

保存到永久存储器中的值在您离开某个位置时以及游戏关闭时仍然存在。它们不是primary_save的一部分，而是存在于Stonescript文件夹中的一系列独立文件中。导入的脚本相互隔离地访问存储，允许不同的模块使用相同的密钥，而无需修改彼此的数据。

storage.Delete(string)　　*无返回值*　　删除指定关键词上可能存在的任何值

```
storage.Delete("highscore")
```

storage.Get(string)　　*返回储存值*　　检索存储在指定关键词上的永久值

```
var value = storage.Get("highscore")
?value
  >最高分为 @value@
:
  >没有最高分。
```

storage.Get(string, value)　　*返回储存值*　　检索存储在指定关键词上的永久值。如果没有找到，则返回第二个参数作为默认值

```
var value = storage.Get("highscore", 0)
>最高分为 @value@
```

storage.Has(string)　　*返回bool*　　如果检索的关键词存在于永久存储中，则返回true，否则返回false

```
?storage.Has("highscore")
  var value = storage.Get("highscore")
  >最高分为 @value@
:
  >没有最高分。
```

storage.Incr(string) *返回int* 将存储在指定关键词中的值增加1，然后返回新值

```
?gameOver
  storage.Incr("stat_TimesPlayed")
```

storage.Keys() *返回array* 检索包含当前上下文中所有可用存储键的字符串数组

```
var a
?time % 30 = 0
  a = storage.Keys()
  for i = 0 .. a.Count()-1
    >Key @i@ = @a[i]@
```

storage.Incr(string, integer) *返回int* 将存储在指定关键词中的值增加给一个定量，然后返回新值

```
var amount
?foundCoins
  amount = rng%5 + 5
  storage.Incr("coins", amount)
```

storage.Set(string, value) *无返回值* 将value值保存到指定关键词的永久存储区

```
var score = 1000
storage.Set("highscore", score)
```

string字符串

string.Break(string, integer) *返回array* 给定最大宽度，将一个字符串拆分为多个字符串

```
var s = "The brown fox jumps over the lazy dog"
var a = string.Break(s, 14)
for i = 0 .. a.Count()-1
  >`0,@i@,@a[i]@
```

string.Capitalize(str) *返回string* 将字符串的第一个字母改为大写

```
var a = "foo"
a = string.Capitalize(a)
>@a@
// 会打印出 "Foo"
```

string.Equals(str1, str2)　　*返回bool*　　接受两个字符串参数，如果它们完全相同，则返回true，否则返回false；区分大小写

```
var a = "foo"
?string.Equals(a, "foo")
  >这两个字符串相同
:
  >这两个字符串不同
// 在这个例子中，字符串是相等的,string.Equals()计算结果为true
```

string.Format(str1, ...)　　*返回string*　　通过用其他参数的值替换格式模板来修改字符串，然后返回最终组成的字符串。这是一个强大的功能，支持许多格式选项

```
var str = "我的名字是 {0} ，我总共有 {1} 的力量值!"
var result = string.Format(
^  str,
^  player.name,
^  totalgp
^)
>@result@
```

string.IndexOf(str, criteria)　　*返回int*　　将字符串变量、字符串条件作为参数，并在字符串中查找条件的位置。如果未找到，则返回-1

```
var a = Hello World!
var index = string.IndexOf(a, llo)
// index 等于 2
```

string.IndexOf(str, criteria, startAt)　　*返回int*　　接受字符串变量、字符串条件和起始索引作为参数。从“startAt”索引处开始搜索字符串，查找条件的位置；如果未找到，则返回-1

```
var a = Hello World!
var index = string.IndexOf(a, llo, 4)
// index 等于 -1 ,因为搜索从第4个字母开始搜索
// 因此找不到 'llo'
```

string.Join(s, [])　/　string.Join(s, [], int1)　/　string.Join(s, [], int1,int2)　　*返回string*
获取一个字符串数组 [ ] 并用分隔符 s 将其组合成一个字符串，可以传递可选的整数参数来指定起始索引 int1 和要组合的元素数量 int2；如果没有提供索引参数，则组合整个数组

```
var a = ["Hello", "World", "!"]
var b = string.Join(";", a)
>`0,0,@b@
// 打印 "Hello;World;!"
```

```
var a = ["Hello", "World", "!"]
var b = string.Join(";", a, 1)
>`0,0,@b@
// 打印 "World;!"
```

```
var a = ["Hello", "World", "!"]
var b = string.Join(";", a, 0, 2)
>`0,0,@b@
// 打印 "Hello;World"
```

string.Size(str) *返回int* 接受字符串变量作为参数，并计算它有多长，以整数表示

```
var a = Hello World!
var size = string.Size(a)
>size 等于 @size@
```

string.Split(str) / string.Split(str, s...) / string.Split(str, s..., bool) / string.Split(str, bool) *返回array* 在给定一组字符串分隔符 s... 的情况下，获取一个字符串并将其分解为一个字符串数组。如果没有提供分隔符，那么只要有空格，字符串就会断开。可选的布尔参数指定是否应该丢弃空条目。

```
var a = string.Split("Hello World !")
for i = 0 .. a.Count()-1
  >`0,@i@,[@i@] = @a[i]@
// 将字符串分解为 "Hello", "World", "!"
```

```
var a = string.Split("Hello World !", " ", "l")
for i = 0 .. a.Count()-1
  >`0,@i@,[@i@] = @a[i]@
// 将字符串分解为 "He", " ", "o", "Wor", "d", "!"
```

```
var a
a = string.Split("Hello World !","l","r",true)
for i = 0 .. a.Count()-1
  >`0,@i@,[@i@] = @a[i]@
// 将字符串分解为 "He", "o Wo", "d !"
```

string.Sub(str, startAt) *返回string* 接受字符串变量、起始索引作为参数，并从该起始索引处向前拆分字符串

```
var a = Hello World!
var subString = string.Sub(a, 6)
>substring = @subString@
// subString 是拆分出来的 "World!"
```

string.Sub(str, startAt, length) *返回string* 接受字符串变量、起始索引作为参数，并从该起始索引处向前拆分字符串，在给定长度处停止

```
var a = Hello Worldvar sudokuStore = import Games/SudokuEnchant/SudokuStore
  >`8,1,@sudokuStore.Count()@,@level_@!
var subString = string.Sub(a, 6, 3)
>substring = @subString@
// subString 是拆分出来的 "Wor"
```

string.ToLower(str) *返回string* 将字符串中的所有字母改为小写

```
var a = "Foo"
a = string.ToLower(a)
>@a@
// 在屏幕上打印 "foo"
```

string.ToUpper(str)　　*返回string*　　将字符串中的所有字母改为大写

```
var a = "Foo"
a = string.ToUpper(a)
>@a@
// 在屏幕上打印 "FOO"
```

sys系统读取

（sys系统读取目前可能没有按预期运行）

sys.cacheRemoteFiles　　*bool*　　指示是否应在运行之间缓存远程导入的文件，默认值为true。如果设置为false，那么当您从位置屏幕开始游戏时，远程脚本将会重新下载。在远程部署的新脚本的开发和迭代过程中更改这一点会很有用。这个全局属性在两次运行之间保持不变。

```
// 我们使用“上”键作为切换的输入
// 快速迭代期间远程缓存的开/关
?key = upBegin
  sys.cacheRemoteFiles = !sys.cacheRemoteFiles
  ?sys.cacheRemoteFiles
    >远程缓存已启用
  :
    >远程缓存已禁用
```

sys.fileUrl　　*string (只读)*　　导入脚本时使用的当前文件路径的获取地址。默认值取决于您的设备。在PC上，默认值是`local`。在移动设备上，默认为`https://StonestoryRPG.com/stonescript/`

```
\>`0,1,文件Url地址为： @sys.fileUrl@
```

sys.SetFileUrl(str)　　*无返回值*　　使用import或new命令时，更改导入脚本的来源。只能在心灵石上调用，如果写在导入的脚本上会出错。接受任何URL作为远程位置，但也接受值“local”和“remote”作为快捷方式，分别指向您的本地驱动器或官方Stone Story RPG存储库。如果null作为参数传递，那么文件URL将重置为设备的默认值。这个全局属性在两次运行之间保持不变。

```
sys.SetFileUrl(
^"https://MyCoolDomain.com/scripts/")
import MyCombatScript
```

sys.MindConnect()　　*无返回值*　　仅当sys.MindConnect() 是MindStone中唯一的一行文本时才有效。启用一个等待来自其他程序（如python脚本）的连接的接口。连接后，外部程序可以运行Stonescript命令并评估游戏状态变量。在此处了解更多信息：https://github.com/artificial-potato/SSRPGInterface

sys.isMobile　　*返回bool*　　返回玩家是否是在Android 或者 iOS上玩游戏的

sys.isPC　　*返回bool*　　返回玩家是否是在Win/Mac/Linux这样的PC系统上玩游戏的

sys.os *返回string* 玩家的操作系统类型，例如"Android", "iOS", "Linux", "OSX" 或者 "Windows".

Text Localization文本本地化

在撰写本文时，Stonescript支持12种语言；一些操作允许自定义脚本适应玩家选择的语言

te.language *string* 玩家在“设置”中选择的语言代码。可能值:EN（英语）、PT-BR（葡萄牙语）、ZH-CN（简体中文）、ZH-TW（繁体中文）、FR（法语）、DE（德语）、RU（俄语）、ES-LA（西班牙语）、ES-EU（西班牙语）、JP（日语）、KR（韩语）和TK（土耳其语）。

```
var lang = te.language
>Language = @lang@
```

te.xt(str) *返回string* 将给定的英文文本翻译成玩家选择的语言。如果没有找到翻译版本，则返回输入文本。或者，可以使用文本标识符(TID)作为输入——不过TID的详细列表超出了本手册的范围

```
var button = ui.AddButton()
button.text = te.xt(Play)
// 在“设置”中更改您的语言，以查看此示例的效果
```

te.GetTID(str) *返回string* 返回给定文本的文本标识符(TID)；输入文本应该是玩家选择的语言

```
var tid = te.GetTID("Play")
>`0,1,@tid@
// 将tid_button_play打印到屏幕上
```

te.ToEnglish(str) *返回string* 将给定文本从玩家选择的语言翻译成英文原文；如果没有找到翻译版本，则返回输入文本

```
>`0,1,@te.ToEnglish("Jogar")@
// 如果在设置中选择了葡萄牙语，它将打印“Play”
// 否则它将打印“Jogar”
```

time时间

time.FormatCasual(int) / time.FormatCasual(int,bool) *返回string* 将一定数量的帧转换为人类可读的时间字符串表示形式，如“1m 23s”。第二个参数(bool)是可选的；如果为“true”，则结果中的精度最大化。

```
>`0,0,当前时间为
^ @time.FormatCasual(totaltime, true)@
```

time.FormatDigital(int) / time.FormatDigital(int,bool) *返回string* 将一定数量的帧转换为人类可读的时间字符串表示形式，如“1:23”。第二个参数(bool)是可选的；如果为“true”，则结果中的精度最大化。

```
>`0,0,当前时间为
^ @time.FormatDigital(totaltime, true)@
```

UI用户界面

有关Stonescript的高级UI系统的详细信息，请参见16.用户界面部分。ui名称空间的其他函数:

ui.OpenInv()　　*无返回值*　　打开背包/库存页面

ui.OpenMind()　　*无返回值*　　打开意念石编辑器页面

ui.ShowBanner(str)　/　ui.ShowBanner(str,str)　　*无返回值*　　显示最多包含两条消息的动画横幅。每次调用ui.ShowBanner()时，横幅动画都会重新启动

```
?time = 120
  ui.ShowBanner("Hello World!")
```

Other其他

Type(var)　　*返回string*　　评估变量的类型并返回字符串表示形式，可能的类型包括“string”、“int”、“float”、“bool”、“function”、“object”和“null”

```
var a = 0
?Type(a) = int
  >变量 'a' 是一个整型
```

·:·:· 11.导入外部脚本 Importing External Scripts ·:·:·

您的脚本不必全都放在意念石中。Stonescript支持通过`import`和`new`这两种关键字加载外部文件。为了使外部脚本起作用，它们必须位于你的（游戏存档）/Stonescript文件夹内，并以“.txt”结尾。

在最基本的示例中，外部脚本被导入并像直接在你的意念石头中一样运行，这使得开发挂机脚本变得更加方便:

```
import Rocky
import Deadwood
import Caves
import Forest
```

如果你找不见自己的Stonescript文件夹，你可以在意念石中输入任何以上示例。现在在意念石的左边沿会出现一个小按钮，点击就可以访问你的Stonescript文件夹。当然，文件夹中不一定会有以该名字命名的.txt文档，是否存在脚本文档只取决于你之前是否自己编写过.txt代码文件。

脚本被导入后，意念石会将其复制到一个独立的内存空间中并作为参考。外部脚本中声明的变量是隔离的，并且不会与其他脚本中的变量交互。在此示例中，打印实用程序脚本提供了简化的服务:

```
// 写在PrintUtil.txt中的程序
func LowerLeft(x, y, color, message)
```

```
  y = y + screen.h  //将y相对于屏幕下方的坐标转换为相对于屏幕上方的坐标
  >`@x@,@y@,@color@,@message@

func LowerRight(x, y, color, message)
  x = x + screen.w  //将x相对于屏幕右边的坐标转换为相对于屏幕左边的坐标
  y = y + screen.h  //将y相对于屏幕下方的坐标转换为相对于屏幕上方的坐标
  >`@x@,@y@,@color@,@message@

// 意念石中的主程序
var print = import PrintUtil
disable hud     //禁用HUD
print.LowerLeft(0,-1,#ffffff,"血量: " + hp)   //调用函数打印
```

高级解决方案可以使用"new"命令多次导入相同的脚本，例如，为了实现模块化或面向对象的范例：

```
// 写在Vector.txt中的程序
var x = 0
var y = 0

func init(_x, _y)   //给x,y幅值
  x = _x
  y = _y

func subtract(otherVect)    //用自己x,y减去别的实例的x,y
  x = x - otherVect.x
  y = y - otherVect.y

// 意念石中的主程序
var vectFrom = new Components/Vector
var vectTo = new Components/Vector

vectFrom.init(5, 4)
vectTo.init(8, 2)
vectTo.subtract(vectFrom)

>x = @vectTo.x@, y = @vectTo.y@
```

外部脚本可以实现 `ToString()` 功能，从而可以在高级打印命令中直接使用它们：

```
// 写在Vector.txt中的程序
var x = 0
var y = 0

func init(_x, _y)
  x = _x
  y = _y

func ToString()
  return "(" + x + ", " + y + ")"

// 意念石中的主程序
var v = import Components/Vector
v.init(3,5)
>Vector = @v@
```

可以从Stonescript的子文件夹导入外部脚本：

```
import Games/Blackjack
import Cosmetics/PetFrog
import Cosmetics/Hats
```

虽然相似，但“导入import”和“新建new”之间有两个重要区别。使用“import”对同一外部脚本进行访问时，每次都访问相同的对象；即如果从多个位置导入相同的脚本，都将访问相同的对象。使用“new”导入脚本时都会生成一份新的副本，但是这个脚本的本体只会运行一次，而不是每帧重复访问运行一次。

·:·:· 12.字符画艺术 ASCII-art ·:·:·

在Stonescript中，可以将自定义ASCII-art嵌入脚本中，并使用顶层打印命令在屏幕上进行绘制。有几种方法可以执行此操作，并且某些字符具有特殊的功能：

代表空格，透明，不画任何东西。

\n 为换行符，使绘制继续在下一行。警告：\n是一个昂贵的操作，不应用于在大型绘图中断行，而是改用ascii/asciiend块。

1. 顶层打印

```
// 本示例在屏幕的左上角绘制一个绿色圆圈
>`1,0,#green,ascii
#.-.
(   )
#`-´
asciiend
```

2. 变量打印

```
// 此示例将怪鱼图案保存到变量中，然后将其绘制为红色，显示在屏幕的左上方
var fishSprite
fishSprite = ascii
###(°_##
#_/_ o\#
#´  `'"#
asciiend

>`0,3,#red,@fishSprite@
```

·:·:· 13.循环 Loops ·:·:·

循环功能允许一段代码多次运行。要创建循环，请使用以下格式的'for'关键字：

```
for v = a..b
```

变量“v”以值“a”开始循环并增加值直到达到“b”，然后循环结束：

```
for i = 1..5
  >`O,@i@,i = @i@
```

迭代变量“v”不应在“for”之前声明，而应包含在循环范围内。但是，可以在循环之前声明开始和结束值“a”和“ b”：

```
var min = 1
var max = 4
var sum
sum = 0
for j = min..max
  sum = sum + j
>sum = @sum@
```

循环也可以反方向进行，也可以使用负数：

```
var g
g = ""
for k = 5..-2
  g = g + k
>g = @g@
```

循环可以相互嵌套，也可以与数学表达式内联以形成复杂的算法：

```
for x = 1..9
  for y = x/2 .. x/2 + 6
    >`@x@,@y@,*
```

要尽早退出循环，请修改迭代变量，使其超出范围：

```
var n
n = ""
for i = 1..5
  ?i = 3
    i = -1
  n = n + i
>n = @n@
```

或者也可以使用 `break` 命令退出循环：

```
for i = 1..5
  ?i = 3
    break
```

可以使用以下形式循环遍历数组的元素：

```
var a = [1, 2, 3]
var n
n = ""
for value : a
  n = n + value
>n = @n@
```

·:·:· 14.数组 Arrays ·:·:·

数组是一种特殊类型的变量。它们提供了一种将值和对象顺序组织到分配给单个变量的集合中的方法：

a = []　　初始化一个新数组。更多信息见下文

a[int]　　读取给定位置的值

```
var myArray = [10, 3]
?myArray[1] = 3
  >是的，位于 [1] 的值等于 3
```

a.Add(value)　　将新值/对象添加到数组末尾

```
var myArray = []
myArray.Add(10)
```

a.Clear()　　从数组中移除所有元素，使其为空；这比用 [ ] 重新声明数组更有效。

```
var myArray = [10, 3]
myArray.Clear()
```

a.Contains(value)　　确定给定值是否在数组内；如果能找到则返回true，否则为false

```
var myArray = [10, 3]
?myArray.Contains(3)
  >是的
```

a.Count()　　返回数组中元素的个数

```
var myArray = ["apple","banana"]
var size = myArray.Count()
>数组的大小为 @size@
```

a.Emplace(int, value)　　用新值替换给定位置的值

```
var myArray = [10, 3]
myArray.Emplace(0, 4)
var value = myArray[0]
>位于 [0] 的值现在是 @value@
```

a.IndexOf(value)　　在数组中搜索给定值；返回一个整数，指示该值第一次出现的位置；如果找不到该值，则返回-1

```
var myArray = [10, 3]
var index = myArray.IndexOf(3)
>在位置 @index@ 找到了这个值
```

a.Insert(int, value)　　将新值/对象添加到数组的特定位置；右边的元素被移动到下一个位置

```
var myArray = [10, 3]
myArray.Insert(1, "apple")
// 现在的数组是 [10, "apple", 3]
```

a.RemoveAt(int)　　从指定位置移除数组中的元素；返回移除的值。 零基础提示：myArray.RemoveAt(0)会移除第一个元素，而 右边的元素移动到上一个位置

```
var myArray = [1, 2, 3]
myArray.RemoveAt(1)
// 现在的数组是 [1, 3]
```

a.Sort()　　将数组的元素按升序组织；如果数组包含不同类型的对象，它仍将被排序，但无法达到定义预期的结果，也不保证能够按元素类型分组

```
var myArray = ["Cherry", 2, "Apple", 1, true, false, "Banana", 3]
var value

myArray.Sort()

for i = 0 .. myArray.Count() - 1
  value = myArray[i]
  >`0,@i@,@value@
```

以下是初始化和使用数组的一些不同方法：

```
var emptyCollection = []
```

```
var magicNumbers = [10, 3, 0, 15, -7]
```

```
var someStrings = ["apple", "banana", "cherry"]
```

```
var sameButMultiLine = [
  "apple",
  "banana",
  "cherry",
]
```

```
var redeclaredEachFrame
redeclaredEachFrame = [] // 这种方法对电脑不太友好
```

```
var clearedEachFrame = []
clearedEachFrame.Clear() // 这种方法对CPU和内存更友善一些
```

```
var clearedEachLoop = []
?loc.begin | loc.loop
  clearedEachLoop.Clear()
```

```
var multiDimensional = [[], [], []]
```

```
var objectCollection = [
  new Components/Float,
  new Components/Float,
  new Components/Vector,
]
```

```
var animationFrames = [ascii
——┼
  O/
 /|
 / \
asciiend
^,ascii
   ---.
  O    \
 /|\┼——
 / \
asciiend
^]
```

遍历数组：

```
var myArray = ["Apple", "Banana", "Cherry"]
var count
var value

count = myArray.Count()
?count > 0
  for i = 0 .. count - 1
    value = myArray[i]
    >`0,@i@,@value@

// 本示例将在屏幕左侧打印水果名称
```

多阶数组访问：

```
var a = [[1,2], [3,4]]
var value

value = a[1][0]
>在 (1, 0)处找到值： @value@
```

·:·:· 15.自定义输入 Custom Input ·:·:·

Stonescript可以在游戏状态中使用 `?key` 读取玩家输入。这可以用来驱动高级行为，例如切换AI的不同模式，同时自定义输入允许在Stone Story之上创建全新的体验。

```
// 在此示例中，@符号可以像经典Rogue-like游戏中的主要角色一样在屏幕上移动
// 按键代码（leftBegin等）是指按钮的初始按下状态

var x = 0
var y = 0

?key = leftBegin
  x--
  ?x < 0
    x = 0
?key = rightBegin
  x++

?key = upBegin
  y--
  ?y < 0
    y = 0
?key = downBegin
  y++

>`@x@,@y@,#ffffff,@
```

按键代码表，如果想要调取某一个按键的对应状态，请按照代码表指示使用

Held按住	Pressed按下	Released释放	Default PC 默认现实按键
left	leftBegin	leftEnd	A or ←
right	rightBegin	rightEnd	D or →
up	upBegin	upEnd	W or ↑
down	downBegin	downEnd	S or ↓
primary	primaryBegin	primaryEnd	LMB, Return 左鼠标、回车
back	backBegin	backEnd	X
ability1	ability1Begin	ability1End	Shift
ability2	ability2Begin	ability2End	Control
bumpL	bumpLBegin	bumpLEnd	Z
bumpR	bumpRBegin	bumpREnd	C

·:·:· 16.用户界面 User Interface ·:·:·

按钮、文本、动画……Stonescript提供了一个构建复杂布局和高性能用户界面的系统。默认情况下，在系统的底部有一个不可见的"根"面板。各种其他的UI元素可以被添加到根面板，包括附加的子面板，形成一个树结构。所有元素按照添加的顺序在一个步骤中绘制在一起。

```
UI的结构
root根面板
|
├─ Panel面板
|   ├─ Text文字
|   ├─ ASCII-art字符画
|   └─ Button按钮
|
├─ Panel
|   ├─ Panel
|   |   └─ Text
...
```

UI用户界面

调用ui命名空间中的函数来构建接口：

ui.root　　*面板对象*　　在其上构建整个树的基本UI对象

```
disable hud
ui.root.visible = true
```

ui.AddAnim(string)　　*返回动画对象*　　将动画对象添加到根面板，接受动画图案的列表作为参数。

```
?loc.begin
  ui.AddAnim(ascii
\o)
%%
(o/
asciiend)
```

ui.AddButton()　　*返回按钮对象*　　向根面板添加一个Button按钮对象

```
func OnPressed()
  > 你好，世界!

?loc.begin
  var button = ui.AddButton()
  button.y = 1
  button.text = 按下我
  button.SetPressed(OnPressed)
```

ui.AddPanel()　　*返回面板对象*　　将Panel面板对象添加到根面板，面板是一种重要的对象类型，充当其他元素的容器。

```
?loc.begin
  var p = ui.AddPanel()
  p.color = #red
```

ui.AddStyle()　　*返回int*　　添加可用于绘制矩形组件(如面板和按钮)的新样式，并返回新样式的ID号。会防止多次添加相同的样式，此时样式不会发生任何变化，并且返回相同的ID。因为不同的脚本可能都调用ui.AddStyle()，建议将ID保存为变量，而不是将样式编号硬编码到脚本中。

```
var customStyle = ui.AddStyle("
^123
^456
^789")
?loc.begin
  var p = ui.AddPanel()
  p.style = customStyle
```

ui.AddText() / ui.AddText(string)　　*返回文本对象*　　将文本对象添加到根面板。

```
?loc.begin
  var t = ui.AddText()
  t.text = "Hello World!"
```

ui.Clear()　　*无返回值*　　从主容器中移除所有UI元素。

```
?key = backBegin
  ui.Clear()
```

Component组件

组件是所有其他UI类型的基类型。这意味着其他元素(面板、文本、按钮和动画)都具有以下属性:

component.x　　*int*　　组件相对于其停靠位置（锚点+停靠点）的X位置。

component.y　　*int*　　组件相对于其停靠位置（锚点+停靠点）的Y位置。

component.w　　*int*　　组件的宽度，其默认值因对象类型而异。

```
button.w = string.Size(button.text) + 4
//调整按钮大小以适合其当前文本，每边+2
```

component.h　　*int*　　组件的高度，默认值为5。

```
panel.h = panel.parent.h
// 设置面板的高度以匹配其父面板的高度
```

component.absoluteX / component.absoluteY　　*int,只读*　　组件相对于屏幕的位置

```
var t
?loc.begin
  var p = ui.AddPanel()
  p.anchor = bottom_right
  p.dock = bottom_right
  t = ui.AddText("Foo")
  p.Add(t)
>`0,1,组件相对位置为 @t.x@,@t.y@
>`0,2,组件绝对位置为 @t.absoluteX@,@t.absoluteY@
```

component.anchor *string* anchor锚点，表示组件内部轴心的自动布局属性。这将指导UI系统如何定位组件相对于自身的位置。默认值为`center_center`"中心-中心"。可能的值：`top_left, top_center, top_right, center_left, center_center, center_right, bottom_left, bottom_center, bottom_right`

component.dock *string* dock停靠点，类似于锚点的自动布局属性。但是停靠点表示外部轴心，或父组件内部放置组件的位置。如果有疑问，请对锚点和停靠点使用相同的值，这是最常见的情况。

```
?loc.begin
  var p = ui.AddPanel()
  p.anchor = top_right
  p.dock = top_right
  p.w = 20
  p.h = 9
  var t = ui.AddText("你好世界!")
  t.anchor = left_bottom
  t.dock = bottom_left
  t.x = 2
  t.h = t.lines.Count() + 1
  p.Add(t)
```

component.ax *string* 锚点的X部分，可能的值:`left, center, right`

component.ay *string* 锚点的Y部分，可能的值:`top, center, bottom`

component.dx *string* 停靠点的X部分，可能的值:`left, center, right`

component.dy *string* 停靠点的Y部分，可能的值:`top, center, bottom`

```
var p
?loc.begin
  p = ui.AddPanel()
  p.ax = right
  p.ay = top
  p.dx = right
  p.dy = top
```

component.parent *面板对象,只读* 对组件父面板的引用；如果组件已创建，但从未添加到另一个面板，则可能指根面板`root`。`panel.Add(component)`被称为组件的父对象更改。

component.visible *复合类型,bool/string* 组件的可见性。默认值为“inherit继承”。可能值：`true, false, inherit`。如果设置为`true`，组件将始终可见，忽略其父组件的状态。如果设置为`false`，组件将不可见，不管其父组件是什么。但如果设置为`inherit`，组件将遵循其父组件的可见性。

component.Recycle() *无返回值* 从父面板中移除组件。它将在以后的`ui.Add_()`的调用中重新使用并赋予新的用途。任何对回收元素的变量引用都应该被清空或重新分配以避免错误。

Panel > Component面板组件

面板是用作其他组件的容器的矩形组件，互相添加的面板链形成了树形结构。

panel.children *Component[]* 组件数组，包含所有`panel.Add()`使用添加到面板的子组件。

panel.clip *bool* 指示面板的边界是否应该用于限制子组件的绘制。如果为true，将不会绘制超出面板边界的子组件部分。

```
var p
?loc.begin
  p = ui.AddPanel()
  p.w = 4
  p.h = 3
  var t = ui.AddText("迅捷的棕色狐狸跳过懒狗。")
  p.Add(t)

?time%30 < 15
  p.clip = true
:
  p.clip = false
```

panel.color *string* 面板的颜色，详情见RGB转十六进制颜色表示法。

panel.style *int* 面板当前样式的ID号，默认值为1；可能的值从-8到8。可以通过`ui.AddStyle`()添加其他样式。

panel.Add(Component) / panel.Add(Component, int) *无返回值* 将组件添加到面板。组件成为面板的子组件，面板成为组件的父组件。元素添加到面板的顺序会影响绘制顺序。通过使用可选的整数参数，可以将组件插入到特定的排序位置。没有整数参数意味着组件作为面板的最后一个子组件添加。这个函数也可以用来改变已经是面板子组件的组件的绘制顺序。

panel.Clear() *无返回值* 从面板中移除所有UI元素。以这种方式移除的组件被回收到ui系统中，并将在以后的`ui.Add_()`类型的函数调用中重新使用。对这些元素的任何变量引用都应该被清空或重新分配以避免错误。

panel.Remove(Component) / panel.Remove(int) *无返回值* 从面板中移除特定组件或移除指定索引号处的组件。以这种方式移除的组件被回收到ui系统中，并将在以后的`ui.Add_()`类型的函数调用中重新使用。对这些元素的任何变量引用都应该被清空或重新分配以避免错误。

Text > Component文本组件

多行文本框，支持颜色元数据（数据属性）。

text.align *string* 框内文本的对齐方式。默认值为"左"。可能的值:`left, center, right`

text.color *string* 文本的颜色，详情见RGB转十六进制颜色表示法。

text.lines *string[]* 字符串数组，内容为文本框格式化其内容后分解的文本行。不包括颜色元数据。

text.text *string* 文本框的全部内容。文本的子部分可以用元数据`[color=#rrggbb] [/color]`来着色。

```
?loc.begin
  var t = ui.AddText()
  t.text = "你好， [color=#red]世界[/color]!"
```

Button > Component按钮组件

button.text *string* 写在按钮内部的文本

```
button.text = player.name
// 把你的名字写在按钮上!
```

button.tcolor *string* 按钮内文本的颜色，颜色详情见RGB转十六进制颜色表示法。

```
button.tcolor = #ff0000
// 将按钮文本设置为红色
```

button.bcolor *string* 按钮边框的颜色，颜色详情见RGB转十六进制颜色表示法。

```
>@button.bcolor@
button.bcolor = #880000
//打印当前的按钮边框颜色，然后将其设置为深红色
```

button.hcolor *string* 按钮被按下时高亮显示的颜色，颜色详情见RGB转十六进制颜色表示法。

```
?loc.begin
  var b = ui.AddButton()
  b.hcolor = #yellow
```

button.sound *string* 按下按钮时播放的声音效果。默认为"确认"。

```
button.sound = buy
// 更改按钮，使其在按下时播放购买声音
```

button.style *int* 按钮当前样式的ID号，默认值为1；可能的值从-8到8。可以通过`ui.AddStyle`()添加其他样式。

button.SetPressed(f)　　*function callaback函数回调*　　指定按钮被按下时要调用的函数。该函数可以有任意数量的参数(甚至没有参数)。当调用该函数时，第一个参数将是对按钮本身的引用。

```
var button1
var button2
func OnPressed(btn)
  ?btn = button1
    >Button1 被按下了
  :
    >Button2 被按下了

?loc.begin
  button1 = ui.AddButton()
  button1.y = 1
  button1.SetPressed(OnPressed)

  button2 = ui.AddButton()
  button2.y = 6
  button2.SetPressed(OnPressed)
```

button.SetDown(f)　　*function callaback函数回调*　　类似于`.SetPressed()`，`.SetDown()`指定一个函数，当按钮开始被按下的开始时(用户最开始接触按钮的时间)调用这个函数。

button.SetUp(f)　　*function callaback函数回调*　　类似于`.SetPressed()`，`.SetUp()`指定一个函数，当按钮开始被按下的状态结束，反弹至最顶端时(用户最后接触按钮的时间)调用这个函数。

```
func OnDown()
  > 按下!

func OnUp()
  > 弹起!

?loc.begin
  var button = ui.AddButton()
  button.y = 1
  button.text = 按这里
  button.SetDown(OnDown)
  button.SetUp(OnUp)
```

Anim > Component动画组件

可以添加到UI的ASCII 图像序列动画。

anim.color　　*string*　　动画的颜色，颜色详情见RGB转十六进制颜色表示法。

anim.duration　　*int*　　动画的时间长度，以帧为单位。

anim.flipX　　*bool*　　如果为true，则以其轴心水平翻转图片。

anim.flipY　　*bool*　　如果为true，则以其轴心竖直翻转图片。

anim.frame　　*int*　　正在绘制的当前动画帧位置，可以通过更改以将动画设置到特定帧。

anim.gamePause　　*bool*　　如果为true，当玩家暂停游戏时动画会自动暂停播放，而当玩家退出暂停状态时，动画会继续播放。

anim.loop　　*bool*　　如果为true，动画将在持续时间结束时从头开始播放。

anim.playing　　*bool,只读*　　如果动画当前正在播放，则为true。

anim.paused　　*bool,只读*　　如果动画当前正在播放，则为true，但由于调用`anim.Pause()`而暂停

anim.pivotX / anim.pivotY　　*int*　　可用于微调ASCII艺术相对于其位置绘制位置的中心附加偏移量。

anim.playOnStart　　*bool*　　如果为真，动画将尽快开始播放。

anim.AddLayer(string)　　*返回动画值*　　在此基础上添加一个新的ASCII动画。当动画播放时(或其帧随`anim.frame`改变)，所有动画层保持同步。每一层都有自己的颜色、中心位置等属性集。对复杂的ASCII艺术使用动画层的优点包括改进的性能和更好的代码质量。如果动画被回收，所有层同时清理。

```
var a = ui.AddAnim(asciiArtA)
var layer2 = a.AddLayer(asciiArtB)
layer2.color = #bbbbbb
```

anim.Load(string)　　*无返回值*　　分配一个新的ASCII 动画工作序列。

anim.Pause()　　*无返回值*　　暂停播放当前帧的动画。随后调用`anim.Play()`继续播放。

anim.Play()　　*无返回值*　　开始播放动画，或者在动画暂停时继续播放。

```
var dance
?loc.begin
  dance = ui.AddAnim(ascii
(O/
%%
\O)
asciiend)
  dance.duration = 20
  dance.loop = true
  dance.Play()
```

anim.Stop()　　*无返回值*　　暂停播放并将动画设置回其第一帧。

Canvas > Component画布组件

为绘制任意字形和颜色而优化的容器组件。

canvas.blend　　*string*　　画布的混合模式，当画布后面有元素时。可能的值:`opaque 不透明`、`Multiply 乘`、`Divide 除`、`Add 加`、`Subtract 减`。默认值为`opaque 不透明`。

```
var filter1 = ui.AddCanvas()
var filter2 = ui.AddCanvas()
filter1.w = screen.w
filter1.h = screen.h
filter2.w = screen.w
filter2.h = screen.h

filter1.blend = multiply
```

```
filter1.SetFG(#aa5555)
filter1.SetBG(#dddddd)

filter2.blend = add

filter2.SetFG(#aa6600)
filter2.SetBG(#662200)
```

canvas.Get(int,int)　　*返回string*　　返回画布上特定位置x，y处的符号。

canvas.Set(str)　　*无返回值*　　用给定的符号填充整个画布。

```
?loc.begin
  var canvas = ui.AddCanvas()
  canvas.Set("X")
```

canvas.Set(int,int,str)　　*无返回值*　　将画布上的特定位置x，y更改为指定符号。

```
?loc.begin
  var canvas = ui.AddCanvas()
  canvas.Set(0, 0, "A")
```

canvas.Set(int,int, fg,str)　/　canvas.Set(int,int, fg,bg,str)　　*无返回值*　　用于在特定位置更改画布，同时重设前景色和背景色。

```
?loc.begin
  var canvas = ui.AddCanvas()
  for x = 0..canvas.w
    for y = 0..canvas.h
      var fg = color.Random()
      var bg = color.Random()
      canvas.Set(x, y, fg, bg, ▄)
```

canvas.SetFG(color)　　*无返回值*　　为整个画布设置前景色。

```
?loc.begin
  var canvas = ui.AddCanvas()
  canvas.Set("R")
  canvas.SetFG(#red)
```

canvas.SetFG(int,int, color)　　*无返回值*　　更改特定位置x，y的前景色。

```
?loc.begin
  var canvas = ui.AddCanvas()
  canvas.Set("X")
  canvas.SetFG(2, 1, #ff00ff)
```

canvas.SetBG(color)　　*无返回值*　　为整个画布设置背景色。

```
?loc.begin
  var canvas = ui.AddCanvas()
  canvas.Set("g")
  canvas.SetBG(#00aa00)
```

canvas.SetBG(int,int, color)　　*无返回值*　　更改特定位置x，y的背景色。

```
?loc.begin
  var canvas = ui.AddCanvas()
  canvas.Set("X")
  canvas.SetBG(2, 1, #yellow)
```

·:·:· 17.小技巧 Tips ·:·:·

- 定义 “?” 的作用范围时，空格（缩进）很重要；
- 可以在关卡运行时通过按键盘上的“M”来更改脚本；
- 意念石头右上方的电源按钮可打开/关闭脚本；
- 如果调用了多个装备命令，则以最后一个为准；
- 该脚本每秒执行30次（每帧一次）；
- 要尝试不同的脚本，建议将它们复制到外部文本编辑器（例如记事本）中；
- 常用快捷键（例如Ctrl + A全选，Ctrl + C复制，Ctrl + V粘贴）很有用；
- 在游戏中按住Tab键可为你提供大量有关游戏状态的信息，并显示近期的Stonescript错误列表；
- 通过在文本中使用“ \ n”，可以将打印命令分成多行。

·:·:· 18.默认脚本 Default script ·:·:·

```
import Fishing

?hp < 7
  activate potion
?loc = caves
  equipL sword
  equipR shield
  ?foe = boss
    equip crossbow
```

·:·:· 附录A.能力冷却ID Ability Cooldown IDs ·:·:·

有关 item.CanActivate(str) 和 item.GetCooldown(str) 等多处使用的能力ID的字符串参数，请参见下表"能力冷却ID"严格填写，而不能随便填写。注意:无效的能力字符串将返回-1，如果没有对应的武器，技能结果也将返回-1。

同时，表格中列出了各个物品的冷却时间和伤害动画时间，使用武器技能时需要通过冷却时间判断是否武器已经完成动画并打出伤害；部分物品为瞬发，不需要考虑动画时间，只需要保证在发动的当前帧完整使用而不被替换即可。

部分物品的冷却时间会随着给**技能的附魔等级**上升而减少，亦在下表中标出，附魔等级用 enLv 表示。

物品	能力冷却 ID	冷却时间/帧	动画时间/帧
药水	"potion"	无	无
巴迪什	"bardiche"	900	22
冲撞盾	"bash"	270	无
堕神之剑	"blade"	2400	无
那伽面具	"mask"	2400	无
冲锋盾	"dash"	45	无
斧头	"hatchet"	120	20(多次判定)
重锤	"heavy_hammer"	660	14
意念石	"mind"	360	无
铁头长杖	"quarterstaff"	195 - 3 × enLv	无
骷髅手	"skeleton_arm"	900	无
火焰护符	"fire_talisman"	150	20
煤灰精灵	"cinderwisp"	2700	无
以太护符	"aether_talisman"	150	20
虚空编织者	"voidweaver"	2700	无
灾厄魔杖	"wand_aether"	900	无
爆炸魔杖	"wand_fire"	900	无
冰霜魔杖	"wand_ice"	900	无
疫病魔杖	"wand_poison"	900	无
荡涤魔杖	"wand_vigor"	900 - 15 × (enLv-1)	无
引力魔杖	"wand_stone"	900 - 30 × (enLv-1)	无
延展长杖	"staff_aether"		
炼狱长杖	"staff_fire"		

物品	能力冷却 ID	冷却时间/帧	动画时间/帧
永恒长杖	"staff_ice"		
狂战长杖	"staff_poison"		
防阻长杖	"staff_vigor"		
灵敏长杖	"staff_stone"		

（部分装备未完全实装，在实装后补充相关数据）

·:·:· 附录B.音乐效果 Sound Effects ·:·:·

Stonescript可以根据自定义逻辑播放游戏中的音效。

```
// 在这个例子中，每当玩家失去生命值时，就会播放“不平等”的声音：

var lasthp = hp
?hp < lasthp
  play unequip
lasthp = hp
```

大多数声音都有随机或连续播放的变化。如果同一个声音在一帧中播放多次，那这些播放将被忽略。如果每秒播放超过5种声音，它们将被抑制。

以下是石头纪中可用声音的完整列表:

```
buy      购买
click    单击
confirm 确认
soul_stone   灵魂石
sword_cast   铸造长剑
sword_hit    长剑攻击
wand_cast    铸造魔杖
wand_hit     魔杖攻击
player_kick      玩家踢
player_punch     玩家拳击
stone_throw_cast     扔石头
stone_throw_hit      石头击中
grappling_cast       抓钩铸造
grappling_hit        抓钩击中
grappling_idle       抓钩闲置
hatchet_cast
hatchet_hit
shovel_cast
torch_cast
torch_hit
torch_idle
pickup_stone
pickup_wood
pickup_tar
pickup_success
soul_stone_drop
```

```
wand_drop
key_drop
cross_deadwood_bump
cross_deadwood_row
cross_deadwood_splash
ui_starfirst
ui_starnew
ui_starold1
ui_starold2
ui_starold3
ui_starold4
bronze_gate_open
prompt_choice
waterfall_hook_hit
waterfall_splash
haunted_gate_key_bounce_1
haunted_gate_key_bounce_2
haunted_gate_key_bounce_3
haunted_gate_opening
haunted_gate_point_lost
haunted_gate_key_into_gate
haunted_gate_shuffle
haunted_gate_shuffle_fast
haunted_gate_torch_off
haunted_gate_torch_on
haunted_gate_try_to_open
paint_splat
waterfall_land
waterfall_rope_grab
waterfall_rope_swing
skeleton_boss_death
skeleton_boss_legs_die
spider_boss_death
tree_boss_death
mushroom_boss_death
tree_boss_attack
tree_boss_awake
tree_boss_idle
tree_boss_spike
spider_boss_attack
player_hit
mushroom_boss_awake
mushroom_boss_punch
mushroom_boss_shoot
skeleton_boss_arm1
skeleton_boss_arm2
skeleton_boss_attack
skeleton_boss_idle
skeleton_boss_bone_bounce
skeleton_boss_arm_woosh
equip
unequip
bat_attack_small
bat_death_small
bat_wing
bat_wing_small
```

```
spider_attack
spider_death
spider_death_small
spider_eggs_spawn
spider_walk
scarab_awake
scarab_bite
scarab_death
scarab_horn
scarab_wings
mosquito_attack
mosquito_death
mosquito_loop
bronze_gate_locked
bat_attack
bat_death
progress_1
progress_2
progress_3
progress_4
progress_5
progress_6
progress_7
progress_8
progress_9
life_gain
player_death
logo_full
logo_short
smithy_hammer
sightstone_cast
error
ranting_tree_halt
treasure_close
treasure_item_pop
treasure_item_show
treasure_open
skeleton_boss_awake
skeleton_boss_hand_slam
level_up
insta_kill
spider_boss_awake
metal_drop
treasure_drop
smithy_hammer_fail
xp_tick
crossbow_cast
crossbow_hit
wand_aether_cast
wand_aether_hit
wand_air_cast
wand_air_hit
wand_fire_cast
wand_fire_hit
wand_ice_cast
wand_ice_hit
```

```
wand_poison_cast
wand_poison_hit
wand_vigor_cast
wand_vigor_hit
skeleton_boss_arm_reconnect
skeleton_boss_summon_minions
mushroom_boss_fat_slam
pickup_bronze
temple_npc_chant
temple_npc_clear_throat
temple_npc_talk
first_controller
slave_npc
slave_outro_chatter
slave_outro_voice
haunted_gate_npc_voice
slave_outro_transition
bronze_guardian_attack1
bronze_guardian_attack2
bronze_guardian_attack3
bronze_guardian_death
bronze_guardian_helmet
bronze_guardian_power_up
bronze_guardian_steps
ant_attack
ant_death
ant_walk
snail_attack
snail_attack_small
snail_death
snail_death_small
snail_walk
ghost_death
ghost_death_small
skeletimmy_death
skeletony_death
skeletimmy_attack
skeletony_attack
skeletony_awake_a
skeletony_awake_b
skeletony_walk
ghost_loop
ghost_loop_small
ghost_attack
ghost_attack_small
controller_death
controller_whip_attack
controller_whip_hit
dominotaur_death
dominotaur_whip_attack
dominotaur_whip_hit
mine_walker_death
mine_walker_attack_a
mine_walker_attack_b
mine_walker_attack_hit
mine_walker_awake
```

```
mine_walker_step
fire_elemental_attack
fire_elemental_attack_hit
fire_elemental_awake
fire_elemental_death
mine_walker_helmet_break
ice_elemental_attack
ice_elemental_attack_hit
ice_elemental_awake
ice_elemental_death
ki_eater_attack
ki_eater_attack_hit
ki_eater_awake
ki_eater_death
ki_gobbler_attack
ki_gobbler_attack_hit
ki_gobbler_awake
ki_gobbler_death
ki_slerper_attack
ki_slerper_attack_hit
ki_slerper_awake
ki_slerper_death
bell_ringer_attack
bell_ringer_attack_hit
cult_guard_attack
cult_guard_attack_hit
cult_marksman_attack
cult_marksman_attack_hit
cult_sorcerer_attack
cult_sorcerer_attack_hit
cultist_death
flying_serpent_loop
poison_adept_attack
poison_adept_attack_hit
serpent_attack
serpent_death
serpent_handler_release
serpent_hiss
serpent_slither
worm_rider_hop
booklet_open
booklet_turn_page
booklet_close
hammer_cast
hammer_hit
shield_dash
fissure_break_apart
fissure_unmake
mindstone_off
mindstone_on
triskelion_fuse
potion_berserk
potion_cleansing
potion_defensive
potion_experience
potion_healing
```

```
potion_invisibility
potion_lightning
potion_lucky
potion_strength
potion_vampiric
bronze_guardian_pulling_hammer
bronze_guardian_removing_hammer
bronze_guardian_turbine
bronze_guardian_ears_ring
bronze_guardian_fuse
bronze_guardian_attack4
yeti_attack
yeti_attack_flick
yeti_attack_hit
yeti_awake_blow
yeti_awake_explosion
yeti_awake_inhale
yeti_awake_lick
yeti_blow
yeti_blow_ice_wall
yeti_death
yeti_inhale
yeti_inhale_lick
nagaraja_awake_roar
nagaraja_awake_swallow
nagaraja_awake_tongue_1
nagaraja_awake_tongue_2
nagaraja_dead
nagaraja_poison_attack
nagaraja_poison_attack_hit
nagaraja_wail
nagaraja_wail_brick
nagaraja_attack_eat
nagaraja_attack_lick
nagaraja_attack_swallow
nagaraja_tongue_damaged
nagaraja_tongue_lift
nagaraja_tongue_smell
nagaraja_tongue_wrap
bearer3_talk
bearer_attack
bearer_attack_hit
bearer_death
bearer_stealing
bearer_super_attack
bearer_scream
bearer4_talk
bearer4_talk_evolving
bearer_evolving
elementalist_aether_attack
elementalist_aether_attack_hit
elementalist_aether_blink
elementalist_death
elementalist_fire_attack
elementalist_fire_attack_hit
elementalist_fire_blink
```

```
elementalist_ice_attack
elementalist_ice_attack_hit
elementalist_ice_blink
elementalist_poison_attack
elementalist_poison_attack_hit
elementalist_poison_blink
elementalist_vigor_attack
elementalist_vigor_attack_hit
elementalist_vigor_blink
bearer5_talk
elementalist_evolving
perfected_attack
perfected_attack_hit
perfected_death
perfected_defense
perfected_energy_ball
perfected_energy_ball_hit
perfected_summon
perfected_talk
epilogue_devolving
epliogue_player_evolves
epilogue_talk
devolved_talk
dysangelos_guidance
dysangelos_guidance_1
dysangelos_guidance_2
dysangelos_guidance_3
dysangelos_intro_talk
ranting_tree_talk_halt
ranting_tree_talk_again
ranting_tree_talk_how_dare
ranting_tree_talk_avenge
ranting_tree_talk_get_out
ranting_tree_talk_impressive
ranting_tree_talk_very_well
ranting_tree_talk_extra
hans_talk_intro
hans_talk_reward
scotty_a_pleasure
scotty_a_worthy_opponent
scotty_deuced
scotty_failte_back
scotty_getting_good
scotty_grr
scotty_guess_which
scotty_intro
scotty_lets_harden
scotty_make_ye_guess
scotty_noo_jist
scotty_perhaps_the_rules
scotty_pick_some_treasure
scotty_shall_we_up
scotty_we_have_wee_use
scotty_well_met
scotty_well_then
scotty_wizard
```

```
scotty_wrong_choice
scotty_hell_be_back
scotty_out_of_treasure
scotty_there_he_is
nagaraja_choir
mushroom_boss_split
ant_hill
treasure_drop_common
treasure_drop_epic
treasure_drop_giant
treasure_drop_humble
treasure_drop_rare
dominotaur_awake
fire_geyser
ice_pillar
treasure_item_cyan
treasure_item_yellow
treasure_item_green
treasure_item_blue
treasure_item_red
treasure_item_rainbow
ki_slerper_walk
mindstone_found
ghost_tomb_death
perfected_fly_start
perfected_fly_loop
perfected_fly_end
shop_door_open
shop_door_enter
scorpion_death
bomb_cart_explosion
bomb_cart_fuse
bomb_cart_move
bronze_gate_close
poison_powerup
acronian_cultist_power_up
giant_ice_elemental_attack
scout_dialog
morel_punch
falling_stonefolk
scout_arrives
scout_leaves
scout_wing
scout_focus
dog_bark
frog
lost_item_boost
treasure_item_lost
blade_glow
blade_pallas_attack
blade_raise
blade_drag
auggie_voice
pallas_voice
quest_stone_jump
quest_stone_unlock
```

```
bardiche_cast
boo_voice
quarterstaff_cast
air_hiss
bang_go_forward
fire_orbs
open_note
talisman_reveal
fire_beast_1
fire_beast_2
uulaa_voice
mask_summon_1
mask_summon_2
```

·:·: 附录C.音乐 Music ·:·:

Stonescript可以根据自定义逻辑播放游戏中的音乐，可用曲目取决于游戏平台。

```
?loc.begin | loc.loop
  music.Play(temple_0)
```

以下是石头纪中可用音乐的完整列表：

```
Boiling Mine    灼热矿井
bronze_guardian_3
bronze_guardian_4
bronze_guardian_5
bronze_guardian_cyan
bronze_mine_0
bronze_mine_1
bronze_mine_2
bronze_mine_3
bronze_mine_4
bronze_mine_5
bronze_mine_cyan
slave_outro_climb
slave_outro_loop

Caves of Fear   恐怖洞窟
caustic_caves
spider_boss

Deadwood    枯木峡谷
cross_deadwood_river
cross_deadwood_wind
deadwood_0
deadwood_1
deadwood_2
deadwood_3
deadwood_4
deadwood_5
```

```
deadwood_cyan
tree_boss
waterfall_descent
poena

Haunted Halls   亡者之殿
skeleton_boss
undead_crypt_0
undead_crypt_1
undead_crypt_2
undead_crypt_3
undead_crypt_4
undead_crypt_5
undead_crypt_cyan
undead_crypt_intro

Icy Ridge   不融山
bridge_broken
bridge_crossing
icy_ridge_0
icy_ridge_1
icy_ridge_2
icy_ridge_3
icy_ridge_4
icy_ridge_5
icy_ridge_cyan
yeti

Mushroom Forest 蘑菇森林
fire_loop
fungus_forest_0
fungus_forest_1
fungus_forest_2
fungus_forest_3
fungus_forest_4
fungus_forest_5
fungus_forest_cyan
mushroom_boss
mushroom_boss_cyan
shop

Rocky Plateau   岩石高地
rocky_plateau_0
rocky_plateau_1
rocky_plateau_2
rocky_plateau_3
rocky_plateau_4
rocky_plateau_5
rocky_plateau_epilogue
rocky_plateau_fight
rocky_plateau_talk

Temple  神庙
nagaraja
temple_0
temple_1
```

```
temple_2
temple_3
temple_4
temple_5
temple_cyan

Events  事件
event_fall
event_hamartia
event_spring
event_stonejam
event_summer
event_winter

Other   其他
credits
main_menu
bone_factory
osteophone
uulaa
```

·:·: 附录D.环境音频循环 Ambient Loops ·:·:

Stonescript可以根据自定义逻辑播放多层背景环境音频

```
?loc.begin
  ambient.Stop()
  ambient.Add(ambient_crypt)
```

以下是石头纪中可用的环境音频的完整列表:

```
ambient_mines
ambient_caves
ambient_bronze_gate
ambient_deadwood
ambient_mushroom
ambient_bridge
ambient_icy
ambient_cave
ambient_rocky
ambient_temple
ambient_crypt
ambient_haunted_gate
ambient_pallas
waterfall_a
waterfall_b
waterfall_c
```

·:·: 附录E.中文/代码名称对照表 ·:·:

以下是石头纪中各装备、属性、魔法石、地点的中文和代码名称对照：

注意，只有在equip时能够带空格使用，其他场合例如在判断中，请将携带下划线 _

一些武器在基础名称上并无区别，但是带有特殊的武器标识符，例如附魔出技能的魔杖、长杖类武器，具体可以参考附录G.武器标识符。

基础装备

基础装备	
sword	剑
shield	盾
crossbow	弩
wand	魔杖
quarterstaff	铁头长杖

合成装备

合成装备	
long_sword	大剑
hammer	战锤
heavy_hammer	重锤
heavy_crossbow	重弩
repeating_crossbow	连弩
staff	长杖
dashing_shield	冲锋盾
compound_shield	复合盾
tower_shield	大盾
bardiche	巴迪什
socketed	已融合石头魔杖

属性

属性	
Poison	剧毒
Vigor	活力
AEther	以太

属性	
Fire	火焰
Ice	寒冰

道具

道具	
grappling_hook	钩爪
shovel	铲
hatchet	斧

石头

石头	
sight_stone	洞见石头
star_stone	星之石
xi_stone	气之石
xp_stone	经验石头
ouroboros_stone	衔尾蛇石头
quest_stone	任务石头
fissure_stone	裂隙石头
triskelion_stone	三曲腿石头
mind_stone	意念石头
moon_stone	月晷石头

遗物

遗物	
bashing_shield	冲撞盾牌
blade_of_god	堕神之剑
cult_mask	那伽面具
skeleton_arm	骷髅手
fire_talisman	火焰护符
aether_talisman	以太护符
Crusader's Shield(暂定)	十字军盾牌

召唤物

召唤物	
cinderwisp	煤灰精灵（火焰精灵）
voidweaver	织虚/虚空编制者（以太精灵）

地名

地名	
rocky_plateau	岩石高地
deadwood_valley	枯木峡谷
caustic_caves	恐怖洞窟
waterfall	枯木瀑布
fungus_forest	蘑菇森林
fungus_forest_boss	愤怒蘑菇
uulaa_shop	温泉店
undead_crypt_intro	亡者之门
undead_crypt	亡者之殿
undead_crypt_boss	无皮巨人
cross_deadwood_river	枯木之河对岸
bronze_mine	灼热矿井
bronze_guardian_harder	青铜守卫
icy_ridge	不融山
cross_bridge	过桥
temple	神庙
nagaraja	蛇怪之间

敌人或NPC

敌人或NPC	
岩石高地	
acronian_scout	卫城斥候
acronian_soldier	卫城士兵
acronian_warcaster	卫城战术师
dysangelos_beare	持石狄斯安琪洛斯

敌人或NPC	
dysangelos_elementalist	混元之体
dysangelos_perfected	狄斯安琪洛斯完全体
枯木峡谷	
huge_mosquito	巨型蚊子
flesh_scarab	石龟子
wasp	黄蜂
wasp_nest	蜂巢
tree_boss	琉刻的残体"苦痛之木"
poena	惩戒夫人珀伊纳
恐怖洞窟	
scorpion	碎膝蝎
small_bat	舐伤蝠
medium_bat	吸血蝠
spider_eggs	不是鱼子酱
small_spider	剥皮蛛
tiny_spider	啮甲蛛
cool_bat	悬蝠
spider_boss	蜘蛛怪布洛希
蘑菇森林	
collossal_snail	巨型蜗牛
epic_snail	超大蜗牛
ant	蚂蚁
ant_hill	蚁丘
fluff	毛毛
mushroom_boss	愤怒蘑菇
mushroom_boss_fat	孢兄羊肚君
mushroom_boss_skinny	孢妹金针姑
温泉店	
uulaa	乌拉
亡者之殿	

敌人或NPC	
small_skeleton	骷髅喽啰
large_skeleton	骷髅头头
ghost	嘘灵
large_ghost	大嘘
ghost_tomb	坟墓
big_tomb	不散阴魂
skeleton_boss	无皮巨人帕拉斯
skeleton_boss_stage_2	无足巨人帕拉斯
small_skeleton_minion	骷髅喽啰(boss战)
灼热矿井	
slave_master	马面监工
big_slave_master	牛头监工
mine_walker	矿井机器人
fire_elemental	火元素
fire_geyser	烈火温泉
bomb_cart	炸弹推车
bronze_guardian	青铜守卫
不融山	
ki_eater	食气
ki_slerper	摄气
ki_gobbler	吞气
ice_elemental	冰元素
ice_elemental_elite	精英冰元素
ice_pillar	冰柱
giant_ice_elemental	巨型冰元素
ice_wall	冰墙
yeti	利姆尼尔
神庙	
cult_guard	蛇卫兵
cult_marksman	蛇教射手

敌人或NPC	
cult_sorcerer	邪术师
ground_serpent	地蛇
poison_adept	毒使
serpent_handler	驯蛇高手
heavy_hitter	撞钟人
flying_serpent	假发强盗
worm_rider	爬虫骑士
acronian_cultist	卫城信徒
nagaraja	吞炬蛇神那伽王

部分相关内容请参照5.搜索过滤器/关键词 Search Filters

·:·: 附录F.增益、减益属性对照表Buff/Debuff ·:·:

以下是石头纪中各种buff和debuff的标准名称，如需要通过buffs.string等相似系统变量判断，可以使用其中部分单词，具体使用情况请按实际进行：

代码名	中文
player's buff	**玩家增益效果**
∞:debuff_damage	剧毒p，增加伤害
·:smite	堕神之剑技能，重击
≡:pick_pocket	骷髅手普攻，扒窃
o:bardiche_buff_aoe_chance	巴迪什技能，单体攻击限制
o:bardiche_buff_crit_chance	巴迪什技能，100%暴击
o:bardiche_buff_crit_mult	巴迪什技能，增加暴击倍数
o:bardiche_buff_move_speed	巴迪什、重锤技能，短暂增加移速
o:quarterstaff_buff_attack_speed	铁头长杖技能，短暂增加攻击速度
o:quarterstaff_buff_stun	铁头长杖技能，短暂造成眩晕
@:experience	经验药水，增加获得的经验和气
o:strength	怪力药水，击晕与破甲
!:lucky_crit	暴击药水，必定暴击

代码名	中文
×:lucky_mult	暴击药水，暴击倍率增加1.8×
W:vampiric	吸血药水，20%吸血
O:berserk	狂暴药水，攻速+15
?:invisibility	隐身药水，完全闪避
player's debuff	**玩家减益效果**
∞:debuff_duration_damage	受到剧毒敌人攻击
∞:dysangelos_debuff_damage	大眼混元之体剧毒攻击
❄:debuff_chill	大眼混元之体寒冰攻击，惩戒夫人反射
φ:debuff_dot	大眼混元之体火焰攻击，惩戒夫人反射
❄:debuff_yeti_chill	雪怪利姆尼尔雪球，减攻速
❄:debuff_move_speed	雪怪利姆尼尔吹风，减移速
♥:puff_debuff_damage	毛毛爆炸，禁止回血
*:pallas_phase2_debuff	骷髅帕拉斯二阶段的致盲，减射程
o:stun	愤怒蘑菇攻击或惩戒夫人反射造成的眩晕
∞:debuff_damage	惩戒夫人反射
i:ignition	惩戒夫人反射煤灰精灵技能
*:unstable	惩戒夫人反射织虚者技能
foe's buff:	**敌人增益效果**
♥:buff_protection	大眼混元之体的免疫异常
∞:spider_buff_damage	蜘蛛布洛希和惩戒夫人的伤害增加
!:poena_crit	惩戒夫人受到暴击，必定暴击
×:poena_mult	惩戒夫人受到暴击，暴击倍率
♀:poena_mirror	惩戒夫人，buff反射
adaptive_defense	大眼狄斯安琪洛斯完全体的属性抗性
♥:adaptive_defense_vigor	对活力抗性
❄:adaptive_defense_ice	对冰霜抗性
φ:adaptive_defense_fire	对火焰抗性
∞:adaptive_defense_poison	对剧毒抗性

代码名	中文
*:adaptive_defense_aether	对以太抗性
foe's debuff:	**敌人减益效果**
∞:debuff_damage	剧毒P，减少伤害
❄:debuff_chill	寒冰I或i，减少攻速
φ:debuff_dot	火焰F，持续掉血
φ:debuff_dot_2	火焰f，持续掉血
stun	冲撞盾、铁头长杖技能，怪力药水，重锤、大锤普攻
"o:stun""♥:stun""φ:stun""❄:stun""*:stun""∞:stun"	取决于武器属性
∞:debuff_feeble	邪教面具技能，虚弱
×:debuff_armor_fatigue	重锤技能，减少护甲回复
i:ignition	煤灰精灵技能，按层数持续掉血
*:unstable	织虚者技能，使敌人有概率消解

·:·:· 附录G.武器标识符一览 ·:·:·

以下是石头纪中各种武器标识符的格式及含义，可以使用各种标识符精确的定义武器或物品。 **（待补充）**

```
equip sword ice -long +1
```

标识符	说明和例子
Ice / Fire / Poison / AEther / Vigor / Stone	武器的符文属性，没有符文的基本是Stone
A/D/aX/ax/dX/dx	武器的符文效果 A和a代表强或弱的符文防御效果（遇到克制属性的敌人时候加盾） D和d代表强或弱的符文攻击效果（对克制属性敌人造成额外伤害） X和x代表符文的属性特殊效果： X为攻击敌人生效，不同属性分别为I / F / P / U / L x为被敌人攻击或敌人交战时被动生效，不同属性分别为i / f / p / u / h

标识符	说明和例子
*0 / *0*	武器的等级，范围是 *0 至 *10， *10的完整写法是 *10*max*
+1	武器的附魔，范围是 +0 至 +21
socket / -socket socketed / -socketed	武器上是否合成有石头魔杖（符文孔）
long / -long	大剑具有long标识符，-long则用于长剑（小剑）
lost / -lost	是否是失落武器
hidden / -hidden	是否具有隐藏附魔技能
shiny / -shiny	是否是闪光皮肤武器
gold / -gold golden / golden	是否为黄金皮肤武器
prismatic / -prismatic	是否为光彩皮肤武器

·:·: 附录H.特殊修改器一览 ·:·:

在弱点活动或绿星关卡中，可能会遇到具有特殊修改器敌人，其具有不同的特别效果。修改器的效果数值有可能会随精英怪等级发生变化，相关的打印函数见encounter遭遇。

```
>`0,1,当前的修改器为 @encounter.eliteMod@
```

代码名	名称	说明
各地点普遍刷新		
berserker	狂暴	+75%攻击伤害，+5攻击速度，+2移动速度
blessed	受佑	免疫暴击
spawn_hydra	海德拉	死亡时会分裂成两个，子体的生命值是母体最大生命值的一半
metamorph	变化	敌人的元素属性被另一个随机替换，获得与新元素对应的能力
monarch	君主	只有一个君主，但其最大生命值和最大护甲值+300%，+1移动速度，+2攻击伤害，完全免疫消解
mundane	平庸	属性弱点移除
plated	重甲	最大生命值-50%，额外获得等于原始最大生命值200%的护甲
regen	恢复	每秒恢复{5×level}点生命值
stoic	坚忍	免疫第一次受到的伤害

代码名	名称	说明
tenacious	坚韧	免疫减益效果
枯木峡谷 刷新		
granite	花岗岩	攻击具有800%的攻击伤害，非暴击伤害降为0，并受到800%的暴击伤害，移速减半
reflective	反弹	获得增益Mirror镜像
spawn_wasp	黄蜂爆炸	在死亡时生成黄蜂
恐怖洞窟 刷新		
venom	毒物	攻击会造成“虚弱”，伤害-2，攻击一定暴击
spawn_spider	蜘蛛爆炸	死亡时生成剥皮蛛
蘑菇森林 刷新		
spore	孢子	当精英敌人被击败时，周围的敌人会被完全治愈
titan	泰坦	+150%最大生命值
亡者之殿 刷新		
rebirth	重生	受到致命攻击时，生命值和护甲会恢复到初始状态
last_stand	背水一战	受到致命攻击时，敌人会固定为1hp并获得无敌状态；2秒后自我消解
灼热矿井 刷新		
enfused	爆炸	被击败或获得火焰减益(dot)时爆炸，爆炸伤害30点
lucky	幸运	攻击有100%的暴击几率和4倍的暴击伤害
不融山 刷新		
vampire	吸血	攻击会窃取所造成伤害的ki气，并治愈4倍伤害值的生命值
spawn_pillar	永冻	死亡时生成冰柱
spawn_elemental	元素爆炸	死亡时生成冰元素
神庙 刷新		
false_god	伪神	获得无敌状态，10秒后自我消解；受到的击晕时间和击退距离翻倍
persistence	不懈	精英敌人施加的增益和减益持续时间延长60秒

·:·: 附录I.药剂信息 ·:·:

以下是石头纪中各种药剂的名称、翻译，以及效果，代码指令详细请参考 `item.potion`，`activate potion`，`brew (ingredients)` 的用法，各个药水的buff效果名称详见附录F 。

英文名称	中文名称	配方	效果
Strength	怪力	stone 石头	攻击具有眩晕效果（15帧/次），并且对敌人护甲造成三倍伤害，持续300帧
Experience	经验	wood 木头	每击败一名敌人，额外获得1点经验和1点气，持续900帧
Healing	恢复	tar 焦油	完全恢复生命
Lightning	闪电	bronze 青铜	对敌人造成200点伤害（极限距离为69）
Vampiric	吸血	tar + bronze 焦油+青铜	对敌人造成伤害时，恢复等同于伤害值20%的生命，持续600帧
Cleansing	净化	wood + tar 木头+焦油	恢复一半生命，并移除自身所有负面效果
Berserk	狂暴	wood + bronze 木头+青铜	攻击速度+15，持续300帧
Lucky	幸运	stone + bronze 石头+青铜	暴击率提升至100%，暴击伤害+1.8倍，持续180帧
Invisibility	隐形	stone + wood 石头+木头	闪避率提升至100%，持续450帧
Defensive	防御	stone + tar 石头+焦油	恢复一半生命，并获得等同于自身生命上限的护甲（可突破上限）
Empty	空瓶子		

·:·: 附录J.隐藏附魔效果 ·:·:

从邪神面具实装后，武器装备中逐渐添加了各种隐藏附魔效果，以下是石头纪中各种隐藏附魔的效果。一般隐藏附魔效果会随着**附魔等级的提升而改变**，具体数值在下表中标出，附魔等级用 enLv 表示。

装备	隐藏附魔效果
邪神面具 cult mask	延长时间：你最先获得的增益每过 $\{ 22 - \mathrm{enLv} \}$ 帧就会延长1帧
弩 / 连弩	穿透：具备 $\{ \mathrm{enLv} \}$ 的穿透效果
属性锤	溅射：有 $\{ 20 + 4 \times (\mathrm{enLv} - 1) \}$ % 的概率溅射
符文 / 符文武器	增加生命上限，其中效果最好的是符文，增加 $\{ \lceil 7.5 \times (\mathrm{enLv} - 1) \rceil \}$ 的生命上限

装备	隐藏附魔效果
灾厄魔杖 Calamity Wand	技能：- [14]，令附近敌人当前生命减少 { $10 + 1.5 \times \mathrm{enLv}$ } % ，CD：900帧
爆炸魔杖 Explosive Wand	技能：- [4]，造成爆炸，造成{ $40 + 10 \times \mathrm{enLv}$ }点伤害 ，CD：900帧
冰霜魔杖 Frost Wand	技能：- [3]，产生寒风令敌人眩晕{ $\lceil 90 + 4.5 \times (\mathrm{enLv} - 1) \rceil$ }帧，CD：900帧
疫病魔杖 Plague Wand	技能：- [6]，产生毒雾造成{ $\lfloor \frac{\mathrm{enLv}-1}{4} \rfloor + 1$ }层虚弱，虚弱减益持续90帧，CD：900帧 注：该效果为debuff_damage，与dP武器效果相同且相互覆盖，且层数均为显示欺诈，实际只有1层。
荡涤魔杖 Reset Wand	技能：- [1]，令附近所有敌人清除身上可以清除的减益和增益 ，CD：{ $900 - 15 \times \mathrm{enLv}$ }帧
引力魔杖 wand stone hidden	技能：- [0]，强迫附近敌人与你站成一排，CD：{ $900 - 30 \times (\mathrm{enLv} - 1)$ }帧
延展长杖 Gravity Wand	技能：- [2]，攻击距离增长{} （最远22） ，CD：帧
炼狱长杖 staff fire hidden	技能：- [5]，地狱之火环绕，攻击速度加快 {} ， 获得灼烧增益效果 ，CD：帧
永恒长杖 staff ice hidden	技能：- [1]，将自己塞入冰箱，免疫伤害且1秒内不能进行动作 ，CD：帧
狂战长杖 staff poison hidden	技能：- [7]，进入狂暴状态，令你受到和获得的伤害全部增加{}% ，CD：帧
防阻长杖 staff vigor hidden	技能：- [4]，令你在{}秒内受到的第一个减益效果无效 ，CD：帧
灵敏长杖 staff stone hidden	技能：- [3]，退后闪避，CD：{}帧

（部分装备未完全实装，在实装后补充相关数据）

版权所有： Martian Rex, Inc. 2020

翻译：逃税奸商

版本：v4.10.1 - 2025/02/02

翻译更新时间：2025/02/05

感谢up主 福阿月 所进行的翻译工作，本文档在他的基础上完成（节省了大量复制粘贴的时间）。

感谢Okamiroy，LUMEN，别看我只是zhz，xx，星海之城，你所未知的风，电子e⁻，狮子，offy，∴，阿库娅的风 等人以及曾经与未来游玩这个游戏的中文玩家对Stonescript的指教与孜孜不倦的追求。（欢迎署名）

未来翻译更新计划：

附录（音乐效果）的翻译