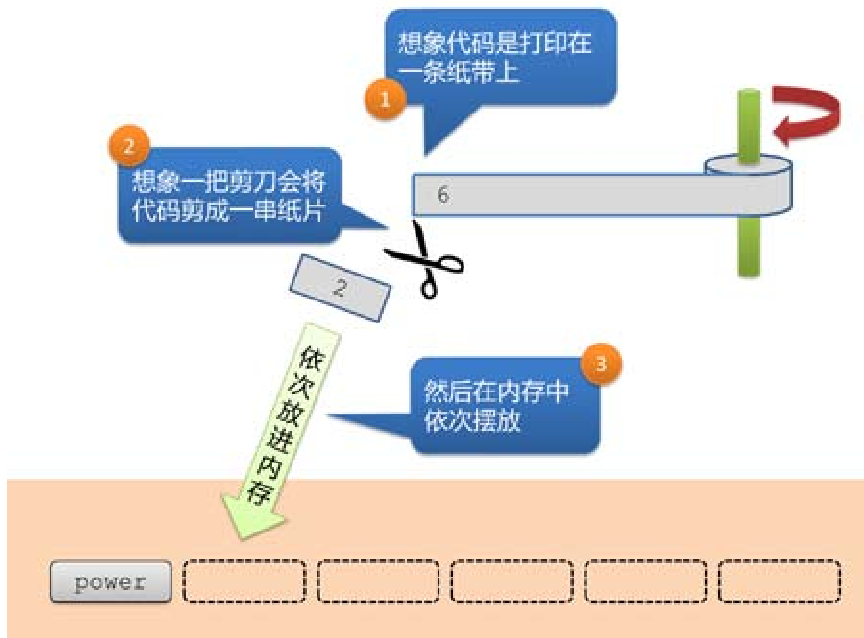
互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

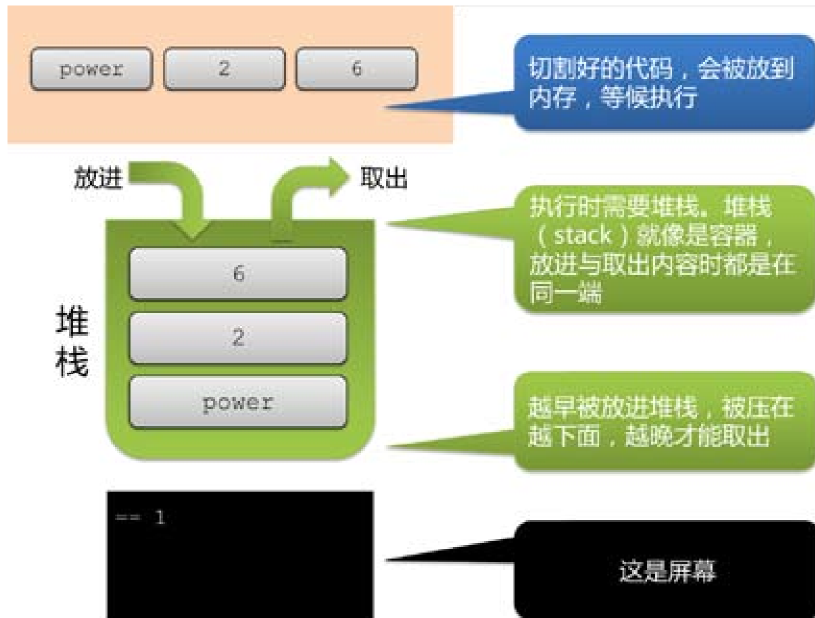
第5章 解释器原理

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章



文字解码完后，你可以想象有一条纸带，上面写着代码。从左侧向左拉动纸带，用剪刀将程序纸带剪成纸片，在内存中依次摆放这些纸片，然后才可以执行内存中的这些纸片。

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章



执行内存中代码的时侯，会用到一种名为堆栈（stack）的数据结构（也就是数据的组织处理方式）。堆栈像个容器，放东西与取东西都在同一端，越晚放进去的东西，越早被取出来（后进先出）。用通俗一点的比喻：堆栈就像是停车场，越早停进去的车，会停在越里面的位置，要等到比它晚进的车都开走之后，才能开走。

为什么需要堆栈？因为程序在执行的过程中，有时候需要把某些事暂时保留不做（因为条件不成熟），等到后面的事做完才能回头去做之前保留的事，这时候用堆栈是最适合的。

从下一页开始连续有好几个范例，你会看到这些范例在执行的过程中堆栈如何变化，而且屏幕上的输出如何变化。通过这些范例，你应该可以理解解释器的运行原理。

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

对于 abs -1 这段代码来说，会被剪成两个值：abs 与 -1，然后开始执行。

在状态 A，堆栈与屏幕都是空的。然后把 abs 放进堆栈中，abs 是求绝对值的函数，它后面需要跟着一个数字，但堆栈中目前只有 abs 自己，所以执行不了，这是状态 B。

到了状态 C，-1 也被放进堆栈。有了这一个参数，abs 终于可以计算了，计算方式是把这两个值都从堆栈中取出，计算之后得到的值是 1，再把 1 放回堆栈，现在是状态 D。

内存中的代码已经执行完毕，堆栈中也没有任何进一步的计算要进行，这表示程序要结束了。结束时，堆栈中剩下的 1 就是返回值。所以在状态 E 中，我们看到堆栈被清空，但屏幕上出现 == 1。

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

对于 power 2 6 这段代码来说，会被剪成 power、2、6 这三个值，然后开始执行。

在状态 A，堆栈与屏幕都是空的。然后把 power 放进堆栈中，成为状态 B。power 是求幂的函数，它后面需要跟着两个数字，分别是底数与指数，所以目前计算不了。状态 C，把 2 放进堆栈，依然无法执行。

状态 D，把 6 放进堆栈，现在 power 的两个参数都到齐了，可以计算了。方式是把三个值都取出来计算，结果得到 64，再把结果放回堆栈，进入状态 E。

内存中的代码已经执行完毕，堆栈中也没有任何进一步的计算要进行。这表示程序要结束了。结束时，堆栈中剩下的 64 就是返回值。所以在状态 F 中，我们看到堆栈被清空，但屏幕上出现 == 64。

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

对于 print add 3 -4 这段代码来说，会被剪成 print、add、3、-4 这四个值，然后开始执行。

先把 print 放进堆栈，进入状态 A。print 需要一个参数，所以此时无法计算。再把 add 放进堆栈，进入状态 B。add 现在无法当做 print 的参数，因为 add 自己就是一个函数，必须等 add 计算完毕的值才能当 print 参数。add 需要两个参数。

3 与 -4 被依次放进堆栈，进入状态 C。这个时候，add 的两个参数已经到齐，可以计算了。把这三个值取出来，计算之后得到结果 -1，把 -1 放回堆栈，现在是状态 D。

这个时候，print 需要的参数（-1）已经出现了，取出这两个值，计算（执行）的结果是屏幕上出现 -1。print 是没有返回值的。没有返回值也就是说返回值是一个特殊值 unset（未设）。把 unset 放进堆栈中。现在状态是 E。

内存中的代码已经执行完毕，堆栈中也没有任何进一步的计算要进行。这表示程序要结束了。结束时，堆栈中剩下的特殊值 unset 就是返回值，返回值为特殊值 unset 相当于没有返回值。所以在状态 F 中，我们看到堆栈被清空，屏幕中没有出现 ==。

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

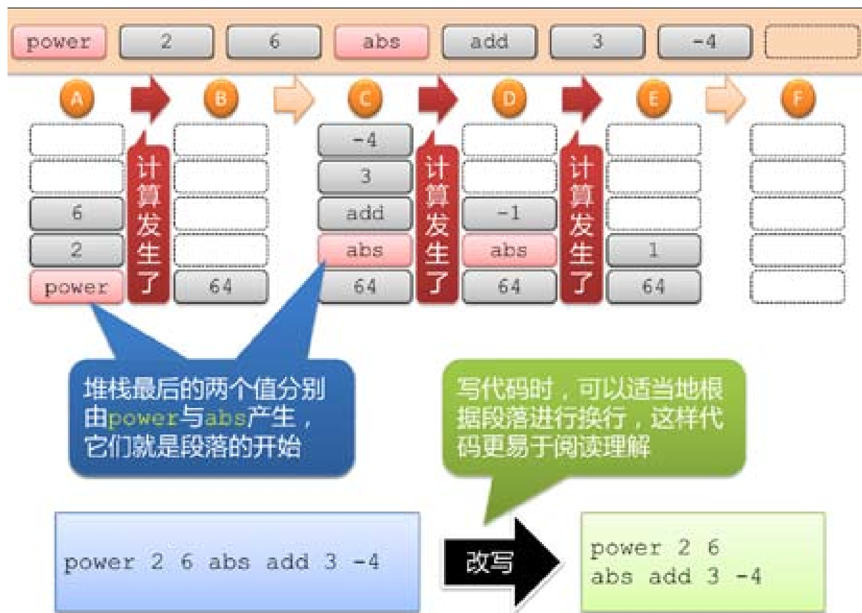
对于 `power 2 6 abs add 3 -4` 这段代码来说，会被剪成 `power`、`2`、`6`、`abs`、`add`、`3`、`-4` 这 7 个值，然后开始执行。

把 `power`、`2`、`6` 依次放进堆栈，进入状态 A。终于可以计算了，计算之后进入状态 B。堆栈中已经没有任何进一步的计算要进行，但内存中还有后续的代码，所以尚未结束。

把内存中的 `abs`、`add`、`3`、`-4` 依次放进堆栈，进入状态 C。终于可以计算了，取出最上面的三个值，计算之后把结果 `-1` 放回堆栈，进入状态 D。状态 D 也可以计算，取出两个值，计算结果 `1` 放回堆栈，进入状态 E。

内存中的代码已经执行完毕，堆栈中也没有任何进一步的计算要进行。这表示程序要结束了。结束时，堆栈中若有多个值，最上层的值就是返回值。所以在状态 F 中，我们看到堆栈被清空，屏幕中出现 `== 1`，而 `64` 直接被扔了。

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

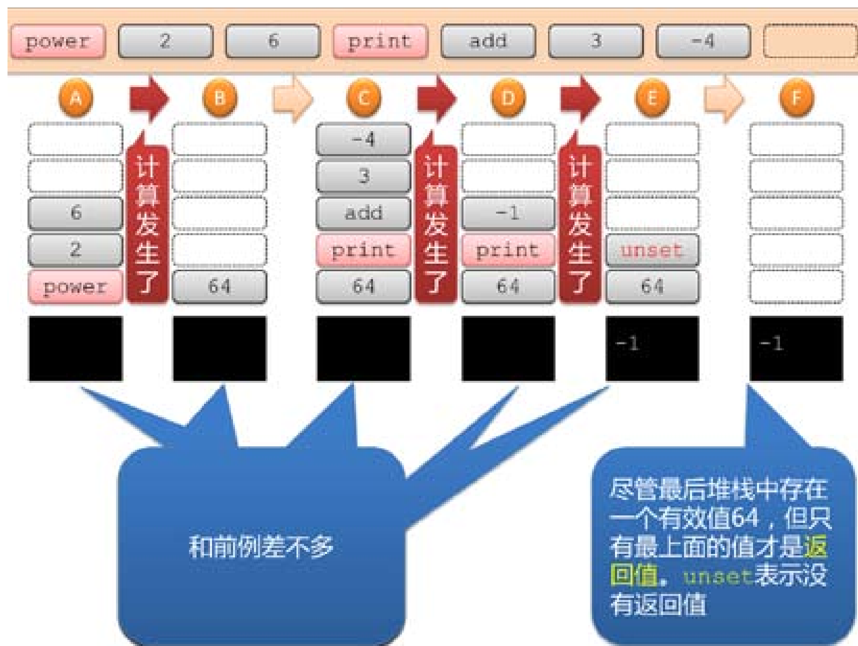


这个例子其实有两个段落，所以最后堆栈会有两个值。段落之间彼此独立，因此写代码时，我们喜欢让段落之间换行，可以帮助阅读理解。

每次发生计算，其实就是找到一个段落。堆栈最后的值分别由 power 与 abs 产生，所以它们两个就是段落的起点。

既然每次发生计算就是一个段落，那么 add 应该也是段落的起点。但因为 add 只是 abs 的一部分（也就是说它是 abs 段落的子段落），而且 add 没有兄弟段落（power 与 abs 在这里就是兄弟关系，是平级的），再说 abs 太简单，让 abs 和 add 摆一起比较好。把 add 段落拆出去没有太大必要。

互动出版网
China-Pub.com
专业图书网上第一专营店
www.china-pub.com
独家提供样章



最后一个例子。对于 power 2 6 print add 3 -4 这段代码来说，这个例子和前一个例子差不多，只是 abs 换成了 print。这导致状态 E 的堆栈中有两个值，而最上面的值是特殊值 unset。

尽管最后堆栈中存在一个有效值 64，但只有最上面的值才是返回值。unset 表示没有返回值。

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

☑ 我知道解释器为何需要堆栈

如果你确定做到了，到本篇开始“学习目标”处打钩。