

实例 3：文本进度条

进度条以动态方式实时显示计算机处理任务时的进度，它一般由已完成任务量与剩余未完成任务量的大小组成。本实例要求编写程序，实现图 1 所示的进度条动态显示的效果。

```
=====开始下载=====
64% [*****.....]
```

(a) 下载中

```
=====开始下载=====
100% [*****]
=====下载完成=====
```

(b) 下载完成

图 1 文本进度条

实例目标

- 掌握变量与 `print()` 函数的使用
- 了解类型转换、模块导入、`for` 循环、字符串格式化输出

实例分析

在本实例中可以将进度条拆分为百分比、已完成下载量、未完成下载量、显示输出四部分，其中百分比使用已完成下载量除以下载总量乘 100% 所得，已完成下载量使用符号 “*” 表示，未完成下载量使用符号 “.” 来表示，显示输出通过 `print()` 函数与 `format()` 函数将计算结果根据指定格式输出。

进度条中的下载总量可以设定为 50，使用 `for` 循环遍历 `range()` 函数生成显示下载总量的整数序列，使用 `print()` 函数与 `format()` 函数将它们进行格式化输出；进度条的实时刷新可以使用 `\r` 来完成，`\r` 可以将输出的内容返回到第一个指针，后面的内容将会覆盖掉前面的内容，便可以完成实时刷新的效果；最后使用 `time` 模块中的 `sleep()` 方法控制进度条下载的速度。根据以上分析可整理出以下实现思路：

- (1) 导入 `time` 模块
- (2) 设定下载总量
- (3) 设定 `for` 循环的次数
- (4) 在 `for` 循环中分别计算已完成下载量、未完成下载量、百分比
- (5) 在 `for` 循环中对已完成下载量、未完成下载、百分比进行格式化输出
- (6) 设置进度条下载速度

代码实现

```
import time          # 导入time模块
incomplete_sign = 50 # 下载总量
print('='*23+'开始下载'+'='*25)
for i in range(incomplete_sign + 1):
    completed = "*" * i # 已完成下载量
    incomplete = "." * (incomplete_sign - i) # 未完成下载量
    percentage = (i / incomplete_sign) * 100 # 百分比
    print("\r{:.0f}%[{}{}]" .format(percentage, completed, incomplete),
          end="")

    time.sleep(0.5)
print("\n" + '='*23+'下载完成'+'='*25)
```

首先使用 `import` 语句导入 `time` 模块，然后将设定的下载总量赋值给变量 `incomplete_sign`，通过 `for` 循环遍历 `range()` 函数生成的整数序列。需要注意的是，当 `range()` 函数没有指定其实数字时，生成的整数序列从 0 开始，当进度条下载量为 0% 时，需要使用 50 个 “.” 表示，因此生成的整数序列为 `incomplete_sign+1`，接着分别计算已完成下载量，未完成下载量、百分比。在使用 `print()` 函数输出时，需要注意最后使用 `end=""` 替换末尾的换行符，这样每次刷新的进度条都只在一行中显示。

代码测试

运行代码，控制台输出结果如下（下载中）：

```
=====开始下载=====
56%[*****.....]
```

运行代码，控制台输出结果如下（下载完成）：

```
=====开始下载=====
100%[*****]
=====下载完成=====
```