

python

The Python logo, consisting of two interlocking snakes, one blue and one yellow, is positioned below the word "python".

```
import turtle
turtle.setup(650,350,200,200)
turtle.penup()
turtle.fd(-250)
turtle.pendown()
turtle.pensize(25)
turtle.pencolor("purple")

for i in range(4):
    turtle.circle(40, 80)
    turtle.circle(-40, 80)
    turtle.fd(40)
    turtle.circle(16, 180)
    turtle.fd(40 * 2/3)
```

实例4: 文本进度条



嵩 天
北京理工大学





"文本进度条"问题分析

文本进度条

用过计算机的都见过

- 进度条什么原理呢？



75%



需求分析

文本进度条

- 采用字符串方式打印可以动态变化的文本进度条
- 进度条需要能在一行中逐渐变化

问题分析

如何获得文本进度条的变化时间？

- 采用sleep()模拟一个持续的进度
- 似乎不那么难



"文本进度条"简单的开始

简单的开始

#TextProBarV1.py

```
import time
```

```
scale = 10
```

```
print("-----执行开始-----")
```

```
for i in range(scale+1):
```

```
    a = '*' * i
```

```
    b = '.' * (scale - i)
```

```
    c = (i/scale)*100
```

```
    print("{:^3.0f}%[{}->{}]" .format(c,a,b))
```

```
    time.sleep(0.1)
```

```
print("-----执行结束-----")
```

```
-----执行开始-----  
0 %[->.....]  
10 %[*->.....]  
20 %[**->.....]  
30 %[***->.....]  
40 %[****->.....]  
50 %[*****->.....]  
60 %[*****->.....]  
70 %[*****->.....]  
80 %[*****->.....]  
90 %[*****->.....]  
100%[*****->.....]  
-----执行结束-----
```




"文本进度条"单行动态刷新

单行动态刷新

刷新的关键是 `\r`

- 刷新的本质是：用之后打印的字符覆盖之前的字符
- 不能换行： `print()` 需要被控制
- 要能回退：打印后光标退回到之前的位置 `\r`

单行动态刷新

#TextProBarV2.py

import time

for i in range(101):

print("\r{:3}%".format(i), end="")

time.sleep(0.1)

0%	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%	11%	12%	13%	14%	15%	16%	17%	18%	19%
20%	21%	22%	23%	24%	25%	26%	27%	28%	29%	30%	31%	32%	33%	34%	35%	36%	37%	38%	39%
40%	41%	42%	43%	44%	45%	46%	47%	48%	49%	50%	51%	52%	53%	54%	55%	56%	57%	58%	59%
60%	61%	62%	63%	64%	65%	66%	67%	68%	69%	70%	71%	72%	73%	74%	75%	76%	77%	78%	79%
80%	81%	82%	83%	84%	85%	86%	87%	88%	89%	90%	91%	92%	93%	94%	95%	96%	97%	98%	99%

IDLE屏蔽了\r功能

单行动态刷新

```
#TextProBarV2.py
```

```
import time
```

```
for i in range(101):
```

```
    print("\r{:3}%".format(i), end="")
```

```
    time.sleep(0.1)
```

```
D:\PYECourse>python TextProBarV2.py  
44%_
```

命令行执行



"文本进度条"实例完整效果

完整效果

```
#TextProBarV3.py
```

```
import time
```

```
scale = 50
```

```
print("执行开始".center(scale//2, "-"))
```

```
start = time.perf_counter()
```

```
for i in range(scale+1):
```

```
    a = '*' * i
```

```
    b = '.' * (scale - i)
```

```
    c = (i/scale)*100
```

```
    dur = time.perf_counter() - start
```

```
    print("\r{:^3.0f}%[{}->{}]{:.2f}s".format(c,a,b,dur),end='')
```

```
    time.sleep(0.1)
```

```
print("\n"+"执行结束".center(scale//2, '-'))
```

```
D:\PYECourse>python TextProBar.py
```

```
-----执行开始-----
```

```
100%[*****->]5.02s
```

```
-----执行结束-----
```

准备好电脑，与老师一起编码吧！



"文本进度条"举一反三

#TextProBarV3.py

```
import time
scale = 50
print("执行开始".center(scale//2, "-"))
start = time.perf_counter()
for i in range(scale+1):
    a = '*' * i
    b = '.' * (scale - i)
    c = (i/scale)*100
    dur = time.perf_counter() - start
    print("\r{:^3.0f}%[{}->{}]{:.2f}s".format(c,a,b,dur),end='')
    time.sleep(0.1)
print("\n"+"执行结束".center(scale//2, '-'))
```

举一反三



PY01B15 天志

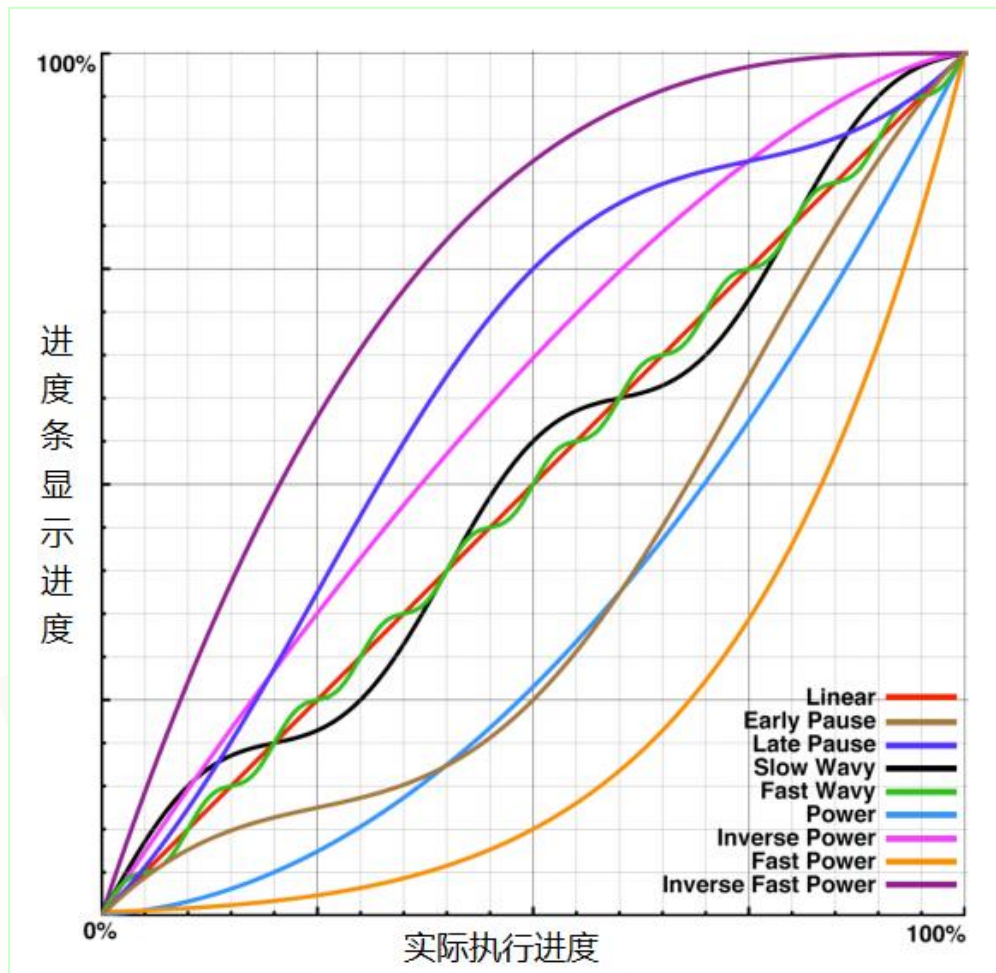
计算问题扩展

- 文本进度条程序使用了perf_counter()计时
- 计时方法适合各类需要统计时间的计算问题
- 例如：比较不同算法时间、统计程序运行时间

举一反三

进度条应用

- 在任何运行时间需要较长的程序中增加进度条
- 在任何希望提高用户体验的应用中增加进度条
- 进度条是人机交互的纽带之一



举一反三

文本进度条的不同设计函数

设计名称	趋势	设计函数
Linear	Constant	$f(x) = x$
Early Pause	Speeds up	$f(x) = x + (1 - \sin(x \cdot \pi \cdot 2 + \pi/2)) / -8$
Late Pause	Slows down	$f(x) = x + (1 - \sin(x \cdot \pi \cdot 2 + \pi/2)) / 8$
Slow Wavy	Constant	$f(x) = x + \sin(x \cdot \pi \cdot 5) / 20$
Fast Wavy	Constant	$f(x) = x + \sin(x \cdot \pi \cdot 20) / 80$

举一反三

文本进度条的不同设计函数

设计名称	趋势	设计函数
Power	Speeds up	$f(x) = (x + (1-x) * 0.03)^2$
Inverse Power	Slows down	$f(x) = 1 + (1-x)^{1.5} * -1$
Fast Power	Speeds up	$f(x) = (x + (1-x)/2)^8$
Inverse Fast Power	Slows down	$f(x) = 1 + (1-x)^3 * -1$

