

# 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)
```

## 模块2: time库的使用

---



嵩 天  
北京理工大学





# time库基本介绍

# time库概述

time库是Python中处理时间的标准库

- 计算机时间的表达
- 提供获取系统时间并格式化输出功能
- 提供系统级精确计时功能，用于程序性能分析

```
import time
```

```
time.<b>()
```

# time库概述

## time库包括三类函数

- 时间获取: `time()` `ctime()` `gmtime()`
- 时间格式化: `strftime()` `strptime()`
- 程序计时: `sleep()`, `perf_counter()`



# 时间获取

# 时间获取

函数	描述
time()	获取当前时间戳，即计算机内部时间值，浮点数 <code>&gt;&gt;&gt;time.time()</code> <code>1516939876.6022282</code>
ctime()	获取当前时间并以易读方式表示，返回字符串 <code>&gt;&gt;&gt;time.ctime()</code> <code>'Fri Jan 26 12:11:16 2018'</code>

# 时间获取

函数	描述
gmtime()	<p>获取当前时间，表示为计算机可处理的时间格式</p> <pre>&gt;&gt;&gt;time.gmtime()  time.struct_time(tm_year=2018, tm_mon=1, tm_mday=26, tm_hour=4, tm_min=11, tm_sec=16, tm_wday=4, tm_yday=26, tm_isdst=0)</pre>





# 时间格式化

# 时间格式化

将时间以合理的方式展示出来

- 格式化：类似字符串格式化，需要有展示模板
- 展示模板由特定的格式化控制符组成
- `strftime()`方法

# 时间格式化

函数	描述
strftime(tpl, ts)	<p>tpl是格式化模板字符串，用来定义输出效果</p> <p>ts是计算机内部时间类型变量</p> <pre>&gt;&gt;&gt;t = time.gmtime() &gt;&gt;&gt;time.strftime("%Y-%m-%d %H:%M:%S",t) '2018-01-26 12:55:20'</pre>

# 格式化控制符

格式化字符串	日期/时间说明	值范围和实例
%Y	年份	0000~9999, 例如: 1900
%m	月份	01~12, 例如: 10
%B	月份名称	January~December, 例如: April
%b	月份名称缩写	Jan~Dec, 例如: Apr
%d	日期	01~31, 例如: 25
%A	星期	Monday~Sunday, 例如: Wednesday

# 格式化控制符

格式化字符串	日期/时间说明	值范围和实例
%a	星期缩写	Mon~Sun, 例如: Wed
%H	小时 (24h制)	00~23, 例如: 12
%I	小时 (12h制)	01~12, 例如: 7
%p	上/下午	AM, PM, 例如: PM
%M	分钟	00~59, 例如: 26
%S	秒	00~59, 例如: 26

# 时间格式化

```
>>>t = time.gmtime()
```

```
>>>time.strftime("%Y-%m-%d %H:%M:%S",t)
```

  
'2018-01-26 12:55:20'

  
>>>timeStr = '2018-01-26 12:55:20'

```
>>>time.strptime(timeStr, "%Y-%m-%d %H:%M:%S")
```

# 时间格式化

函数	描述
strptime(str, tpl)	<p>str是字符串形式的时间值</p> <p>tpl是格式化模板字符串，用来定义输入效果</p> <pre>&gt;&gt;&gt;timeStr = '2018-01-26 12:55:20' &gt;&gt;&gt;time.strptime(timeStr, "%Y-%m-%d %H:%M:%S") time.struct_time(tm_year=2018, tm_mon=1, tm_mday=26, tm_hour=4, tm_min=11, tm_sec=16, tm_wday=4, tm_yday=26, tm_isdst=0)</pre>



# 程序计时应用



# 程序计时

## 程序计时应用广泛

- 程序计时指测量起止动作所经历时间的过程
- 测量时间: `perf_counter()`
- 产生时间: `sleep()`

# 程序计时

函数	描述
perf_counter()	<p>返回一个CPU级别的精确时间计数值，单位为秒 由于这个计数值起点不确定，连续调用差值才有意义</p> <pre>&gt;&gt;&gt;start = time.perf_counter() 318.66599499718114 &gt;&gt;&gt;end = time.perf_counter() 341.3905185375658 &gt;&gt;&gt;end - start 22.724523540384666</pre>

# 程序计时

函数	描述
sleep(s)	<p>s拟休眠的时间，单位是秒，可以是浮点数</p> <pre>&gt;&gt;&gt;def wait():     time.sleep(3.3)</pre> <p>&gt;&gt;&gt;wait()     #程序将等待3.3秒后再退出</p>



# 小花絮

# 如何使用Python官方文档?

<https://docs.python.org/zh-cn/3/>

- 3.7.3版本开始, Python官方文档有了中文版, 快去看看, 能看英文版更好
- 鉴于官方文档并非教程, 而是技术手册, 可以阅读但请注意:
  - **不建议**初学者阅读, 技术手册中包含较多背景知识, 阅读要求较高
  - **不建议**作为教程学习, 官方文档未考虑认知规律, 缺少实例, 跟学进展会比较慢
  - **建议**作为某些疑惑内容深入理解和查阅的工具手册, 与字典用法相似



