Tedu.cn
达内教育

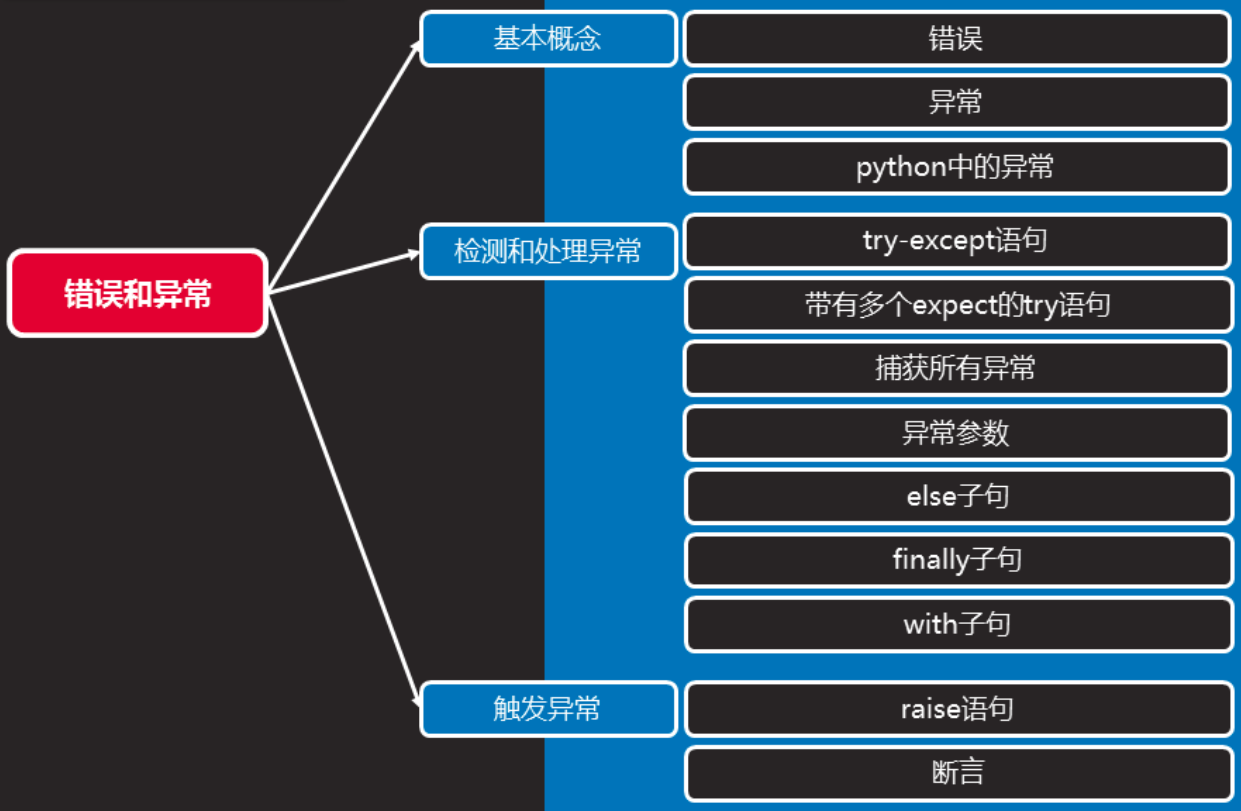
Python运维开发

NSD PYTHON DAY03

内容

上午	09:00 ~ 09:30	作业讲解和回顾
	09:30 ~ 10:20	错误和异常
	10:30 ~ 11:20	re模块
	11:30 ~ 12:20	
下午	14:00 ~ 14:50	多线程
	15:00 ~ 15:50	
	16:00 ~ 16:50	paramiko模块
	17:00 ~ 17:30	总结和答疑

错误和异常



基本概念

Tedu.cn
达内教育

错误

- 从软件方面来说，错误是语法或是逻辑上的
  - 语法错误指示软件的结构上有错误，导致不能被解释器解释或编译器无法编译。这些错误必须在程序执行前纠正
  - 逻辑错误可能是由于不完整或是不合法的输入所致。还可能是逻辑无法生成、计算，或是输出结果需要的过程无法执行

知识讲解

异常

知识讲解

- 当python检测到一个错误时，解释器就会指出当前流已经无法继续执行下去，这时候就出现了异常
- 异常是因为程序出现了错误而在正常控制流以外采取的行为
- 这个行为又分为两个阶段：
  - 首先是引起异常发生的错误
  - 然后是检测（和采取可能的措施）阶段

python中的异常

知识讲解

- 当程序运行时，因为遇到未解的错误而导致中止运行，便会出现traceback消息，打印异常

异常	描　述
NameError	未声明/初始化对象
IndexError	序列中没有没有此索引
SyntaxError	语法错误
KeyboardInterrupt	用户中断执行
EOFError	没有内建输入，到达EOF标记
IOError	输入/输出操作失败

Tedu.cn 达内教育

检测和处理异常

Tedu.cn 达内教育

try-except语句

知识讲解

- 定义了进行异常监控的一段代码，并且提供了处理异常的机制

```
try:
    try_suite #监控这里的异常
except Exception[, reason]:
    except_suite #异常处理代码
```

```
>>> try:
...         f = open('foo.txt')
... except IOError:
...         print 'No such file'
...
No such file
```

带有多个expect的try语句

Tedu.cn 达内教育

知识讲解

- 可以把多个except语句连接在一起，处理一个try块中可能发生的多种异常

```
>>> try:
...         data = int(raw_input('input a number: '))
... except KeyboardInterrupt:
...         print 'user cancelled'
... except ValueError:
...         print 'you must input a number!'
...
input a number: hello
you must input a number!
```

案例1：简化除法判断

Tedu.cn 达内教育

课堂练习

- 编写除法程序
  1. 提示用户输入一个数字作为除数
  2. 如果用户按下Ctrl+C或Ctrl+D则退出程序
  3. 如果用户输入非数字字符，提示用户应该输入数字
  4. 如果用户输入0，提示用户0不能作为除数

捕获所有异常

知识讲解

- 如果出现的异常没有出现在指定要捕获的异常列表中，程序仍然会中断，可以使用
- 在异常继承的树结构中，BaseException是在最顶层的，所以使用它可以捕获任意类型的异常

```
>>> try:
...     data = int(raw_input('input a number: '))
... except BaseException:
...     print '\nsome error'
...
input a number:      [Ctrl + C]
some error
```

异常参数

知识讲解

- 异常也可以有参数，异常引发后它会被传递给异常处理器
- 当异常被引发后参数是作为附加帮助信息传递给异常处理器的

```
>>> try:
...         data = 10 / 0
... except ZeroDivisionError, e:
...         print 'Error:', e
...
Error: integer division or modulo by zero
```

else子句

知识讲解

- 在try范围中没有异常被检测到时，执行else子句
- 在else范围中的任何代码运行前，try范围中的所有代码必须完全成功

```
>>> try:
...         res = 10 / int(raw_input('input a number: '))
... except BaseException, e:
...         print 'Error:', e
... else:
...         print res
...
input a number: 5
2
```

finally子句

知识讲解

- finally子句是无论异常是否发生，是否捕捉都会执行的一段代码
- 如果打开文件后，因为发生异常导致文件没有关闭，可能会发生数据损坏。使用finally可以保证文件总是能正常的关闭

```
try:
    try:
        ccfile = open('carddata.txt', 'r')
        txns = ccfile.readlines()
    except IOError:
        log.write('no txns this month\n')
finally:
    if ccfile:
    ccfile.close()
```

with子句

Tedu.cn 达内教育

知识讲解

- with语句是用来简化代码的
- 在将打开文件的操作放在with语句中，代码块结束后，文件将自动关闭

```
>>> with open('foo.py') as f:
...         data = f.readlines()
...
>>> f.closed
True
```

Tedu.cn 达内教育

触发异常

raise语句

知识讲解

- 要想引发异常，最简单的形式就是输入关键字raise，后面跟要引发的异常的名称
- 执行raise语句时，Python会创建指定的异常类的一个对象
- raise语句还可指定对异常对象进行初始化的参数

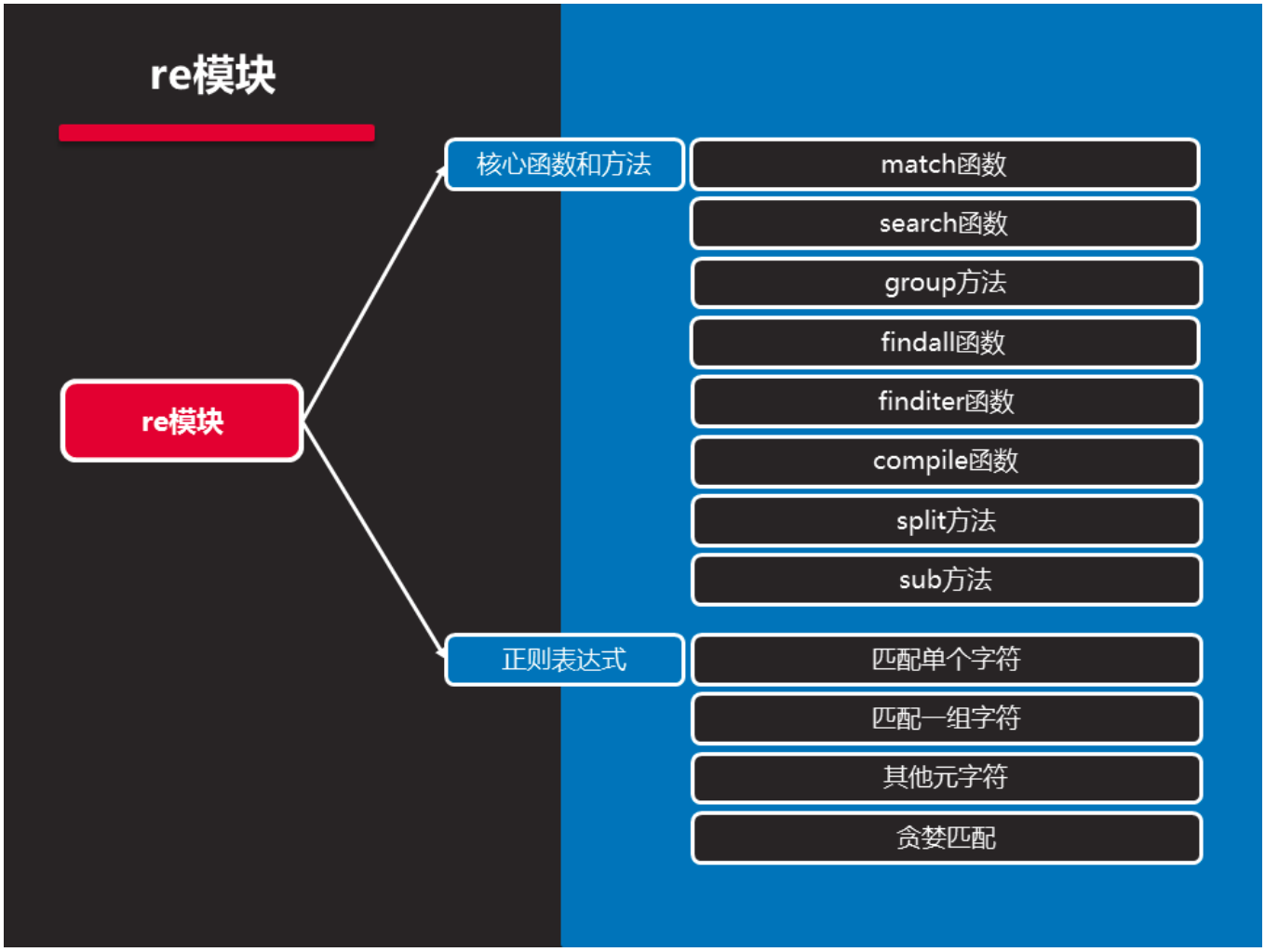
```
>>> number = 3
>>> try:
...         if number < 10:
...             raise ValueError, 'Number is too smalle.'
...  except ValueError,e:
...     print 'Error:',  e
...
Error: Number is too smalle.
```

断言

知识讲解

- 断言是一句必须等价于布尔值为真的判定
- 此外，发生异常也意味着表达式为假

```
>>> number = 3
>>> try:
...     assert number > 10, 'number is too small'
... except AssertionError, e:
...     print 'Error:', e
...
Error: number is too small
```



Tedu.cn
达内教育

核心函数和方法

match函数

Tedu.cn 达内教育

知识讲解

- 尝试用正则表达式模式从字符串的开头匹配，如果匹配成功，则返回一个匹配对象；否则返回None

```
>>> import re
>>> m = re.match('foo', 'food')        #成功匹配
>>> print m
<_sre.SRE_Match object at 0x7f2fa3f6bac0>
>>>
>>> m = re.match('foo', 'seafood')     #未能匹配
>>> print m
None
```

search函数

Tedu.cn 达内教育

知识讲解

- 在字符串中查找正则表达式模式的第一次出现，如果匹配成功，则返回一个匹配对象；否则返回None

```
>>> import re
>>> m = re.search('foo', 'food')
>>> print m
<_sre.SRE_Match object at 0x7f2fa3f6bac0>
>>>
>>> m = re.search('foo', 'seafood')    #可以匹配在字符中间的模式
>>> print m
<_sre.SRE_Match object at 0x7f2fa3f6bb28>
```

group方法

Tedu.cn 达内教育

知识讲解

- 使用match或search匹配成功后，返回的匹配对象可以通过group方法获得匹配内容

```
>>> import re
>>> m = re.match('foo', 'food')
>>> print m.group()
foo

>>> m = re.search('foo', 'seafood')
>>> m.group()
'foo'
```

findall函数

Tedu.cn 达内教育

知识讲解

- 在字符串中查找正则表达式模式的所有（非重复）出现；返回一个匹配对象的列表

```
>>> import re
>>> m = re.search('foo', 'seafood is food')
>>> print m.group()   #search只匹配模式的第一次出现
foo
>>>
>>> m = re.findall('foo', 'seafood is food') #可以匹配全部匹配的出现
>>> print m
['foo', 'foo']
```

finditer函数

知识讲解

- 和findall()函数有相同的功能，但返回的不是列表而是迭代器；对于每个匹配，该迭代器返回一个匹配对象

```
>>> import re
>>> m = re.finditer('foo', 'seafood is food')
>>> for item in m:
...     print item.group()
...
foo
foo
```

compile函数

知识讲解

- 对正则表达式模式进行编译，返回一个正则表达式对象
- 不是必须要用这种方式，但是在大量匹配的情况下，可以提升效率

```
>>> import re
>>> patt = re.compile('foo')
>>> m = patt.match('food')
>>> print m.group()
foo
```

split方法

Tedu.cn 达内教育

知识讲解

- 根据正则表达式中的分隔符把字符分割为一个列表，并返回成功匹配的列表
- 字符串也有类似的方法，但是正则表达式更加灵活

```
>>> import re     #使用 . 和 - 作为字符串的分隔符
>>> mylist = re.split('\.|-', 'hello-world.data')
>>> print mylist
['hello', 'world', 'data']
```

sub方法

Tedu.cn 达内教育

知识讲解

- 把字符串中所有匹配正则表达式的地方替换成新的字符串

```
>>> import re
>>> m = re.sub('X', 'Mr. Smith', 'attn: X\nDear X')
>>> print m
attn: Mr. Smith
Dear Mr. Smith
```

Tedu.cn 达内教育

正则表达式

Tedu.cn 达内教育

匹配单个字符

知识讲解

记号	说 明
.	匹配任意字符（换行符除外）
[...x-y...]	匹配字符组里的任意字符
[^...x-y...]	匹配不在字符组里的任意字符
\d	匹配任意数字，与[0-9]同义
\w	匹配任意数字字母字符，与[0-9a-zA-Z_]同义
\s	匹配空白字符，与[\r\v\f\t\n]同义

匹配一组字符

知识讲解

记号	说 明
literal	匹配字符串的值
re1|re2	匹配正则表达式re1或re2
*	匹配前面出现的正则表达式零次或多次
+	匹配前面出现的正则表达式一次或多次
?	匹配前面出现的正则表达式零次或一次
{M, N}	匹配前面出现的正则表达式至少M次最多N次

其他元字符

知识讲解

记号	说 明
^	匹配字符串的开始
$	匹配字符串的结尾
\b	匹配单词的边界
()	对正则表达式分组
\nn	匹配已保存的子组

贪婪匹配

Tedu.cn 达内教育

知识讲解

- *、+和?都是贪婪匹配操作符，在其后加上?可以取消其贪婪匹配行为
- 正则表达式匹配对象通过groups函数获取子组

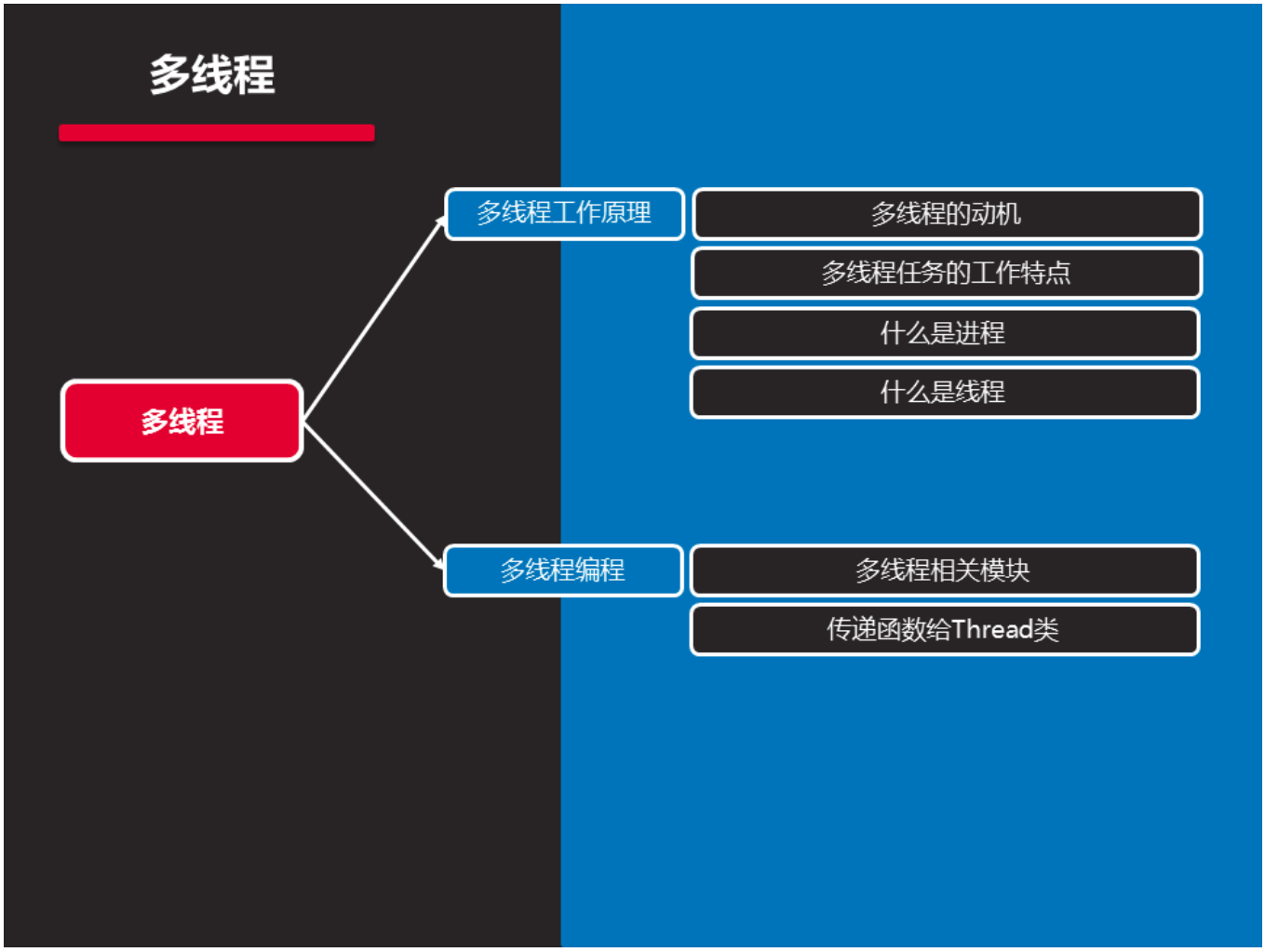
```
>>> data = 'My phone number is: 150888899999'
>>> m = re.search('.+(\d+)', data)
>>> print m.groups()
('9',)
>>>
>>> m = re.search('.+?(\d+)', data)
>>> m.groups()
('150888899999',)
```

案例2：分析apache访问日志

Tedu.cn 达内教育

课堂练习

- 编写一个apche日志分析脚本
  1. 统计每个客户端访问apache服务器的次数
  2. 将统计信息通过字典的方式显示出来
  3. 分别统计客户端是Firefox和MSIE的访问次数
  4. 分别使用函数式编程的方式实现



Tedu.cn
达内教育

多线程工作原理

多线程的动机

Tedu.cn 达内教育

知识讲解

- 在多线程（MT）编程出现之前，电脑程序的运行由一个执行序列组成，执行序列按顺序在主机的中央处理器（CPU）中运行
- 无论是任务本身要求顺序执行还是整个程序是由多个子任务组成，程序都是按这种方式执行的
- 即使子任务相互独立，互相无关（即，一个子任务的结果不影响其它子任务的结果）时也是这样
- 如果并行运行这些相互独立的子任务可以大幅度地提升整个任务的效率

多线程任务的工作特点

Tedu.cn 达内教育

知识讲解

- 它们本质上就是异步的，需要有多个并发事务
- 各个事务的运行顺序可以是不确定的，随机的，不可预测的
- 这样的编程任务可以被分成多个执行流，每个流都有一个要完成的目标
- 根据应用的不同，这些子任务可能都要计算出一个中间结果，用于合并得到最后的结果

什么是进程

Tedu.cn 达内教育

知识讲解

- 计算机程序只不过是磁盘中可执行的、二进制（或其它类型）的数据
- 进程（有时被称为重量级进程）是程序的一次执行
- 每个进程都有自己的地址空间、内存以及其它记录其运行轨迹的辅助数据
- 操作系统管理在其上运行的所有进程，并为这些进程公平地分配时间

什么是线程

Tedu.cn 达内教育

知识讲解

- 线程（有时被称为轻量级进程）跟进程有些相似。不同的是，所有的线程运行在同一个进程中，共享相同的运行环境
- 线程有开始，顺序执行和结束三部分
- 线程的运行可能被抢占（中断），或暂时的被挂起（也叫睡眠），让其它的线程运行，这叫做让步
- 一个进程中的各个线程之间共享同一片数据空间，所以线程之间可以比进程之间更方便地共享数据以及相互通讯

什么是线程（续1）

知识讲解

- 线程一般都是并发执行的，正是由于这种并行和数据共享的机制使得多个任务的合作变为可能
- 需要注意的是，在单CPU 的系统中，真正的并发是不可能的，每个线程会被安排成每次只运行一小会，然后就把CPU 让出来，让其它的线程去运行

Tedu.cn 达内教育

多线程编程

多线程相关模块

Tedu.cn 达内教育

知识讲解

- thread和threading模块允许程序员创建和管理线程
- thread模块提供了基本的线程和锁的支持，而threading提供了更高级别、功能更强的线程管理功能
- 推荐使用更高级别的threading模块
- 只建议那些有经验的专家在想访问线程的底层结构的时候，才使用thread模块

传递函数给Thread类

Tedu.cn 达内教育

知识讲解

- 多线程编程有多种方法，传递函数给threading模块的Thread类是介绍的第一种方法
- Thread对象使用start()方法开始线程的执行，使用join()方法挂起程序，直到线程结束

```
#!/usr/bin/env python
import threading
import time
nums = [4, 2]

def loop(nloop, nsec):    #定义函数，打印运行的起止时间
    print 'start loop %d, at %s' % (nloop, time.ctime())
    time.sleep(nsec)
    print 'loop %d done at %s' % (nloop, time.ctime())
```

传递函数给Thread类（续1）

知识讲解

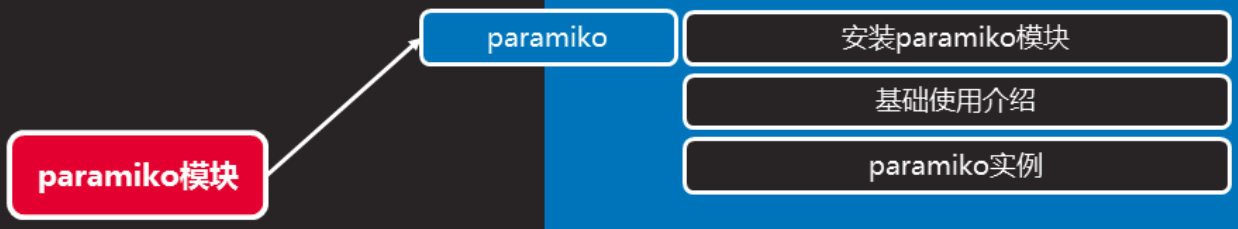
```
def main():
    print 'starting at: %s' % time.ctime()
    threads = []
    for i in range(2):    #创建两个线程，放入列表
        t = threading.Thread(target = loop, args = (0, nums[i]))
        threads.append(t)

for i in range(2):
        threads[i].start()    #同时运行两个线程
    for i in range(2):
        threads[i].join()     #主程序挂起，直到所有线程结束
    print 'all Done at %s' % time.ctime()

if __name__ == '__main__':
    main()
```

案例3：扫描存活主机

课堂练习

- 编写扫描存活主机的脚本
  1. 调用系统的ping命令进行扫描
  2. 扫描教室中所有存活的主机
  3. 采用多线程的方式编写

paramiko模块



Tedu.cn
达内教育

paramiko

安装paramiko模块

Tedu.cn 达内教育

知识讲解

- 安装常用的rpm包

```
# yum  install  -y  gcc  gcc-c++  python-devel
```

- 解压paramiko压缩包

```
# tar  xzf  paramiko-1.15.4.tar.gz
```

- 进入解压目录后安装

```
# python  setup.py  install
```

基础使用介绍

Tedu.cn 达内教育

知识讲解

- SSHClient
  - 创建用于连接ssh服务器的实例

```
>>>  ssh = paramiko.SSHClient()
```

- paramiko.AutoAddPolicy
  - 设置自动添加主机密钥
- ssh.connect
  - 连接ssh服务器
- ssh.exec_comand
  - 在ssh服务器上执行指定命令

paramiko实例

知识讲解

- 编写用于实现ssh访问的脚本
  - 创建SSHClient实例
  - 设置添加主机密钥策略
  - 连接ssh服务器
  - 执行指定命令
  - 在shell命令行中接受用于连接远程服务器的密码以及在远程主机上执行的命令

案例4：利用多线程实现ssh并发访问

课堂练习

- 编写脚本程序
  1. 在文件中取出所有远程主机IP地址
  2. 在shell命令行中接受远程服务器IP地址文件、远程服务器密码以及在远程主机上执行的命令
  3. 通过多线程实现在所有的远程服务器上并发执行命令

总结和答疑

Tedu.cn
达内教育

IDE没有语法提示

问题现象

- 安装好paramiko模块后，通过pycharm编写代码，但是pycharm无法实现语法提示

知识讲解

故障分析及排除

- 原因分析
  - pycharm在起动程序时搜索模块
  - 新安装的模块，pycharm不能及时加载
- 解决办法
  - 关闭IDE，重新打开

知识讲解