argparse 模块

一个可执行文件或者脚本都可以接收参数。

```
$ ls -l /etc
/etc 是位置参数
-l 是短选项
```

如何把这些参数传递给程序呢?
从3.2开始Python提供了功能强大的参数分析的模块argparse。

参数分类

参数分为:

- 位置参数，参数放在那里，就要对应一个参数位置。例如/etc就是对应一个参数位置。
- 选项参数，必须通过前面是`-`的短选项或者`--`的长选项，然后后面的才算该选项的参数，当然选项后面也可以没有参数。

上例中，/etc对应的是位置参数，-l是选项参数。

`ls -alh src`

基本解析

先来一段最简单的程序

```
import argparse

parser = argparse.ArgumentParser() # 获得一个参数解析器
args = parser.parse_args()  # 分析参数
parser.print_help()  # 打印帮助
```

运行结果

```
$ python test.py -h
usage: test1.py [-h]

optional arguments:
  -h, --help  show this help message and exit
```

argparse不仅仅做了参数的定义和解析，还自动帮助生成了帮助信息。尤其是usage，可以看到现在定义的参数是否是自己想要的。

解析器的参数

参数名称	说明
prog	程序的名字，缺省使用 sys.argv[0] 的 basename
add_help	自动为解析器增加 `-h` 和 `--help` 选项，默认为True
description	为程序功能添加描述

`parser = argparse.ArgumentParser(prog='ls', add_help=True, description='list directory contents')`

```
$ python test.py --help
usage: ls [-h]

list directory contents

optional arguments:
  -h, --help  show this help message and exit
```

位置参数解析

ls 基本功能应该解决目录内容的打印。
打印的时候应该指定目录路径，需要位置参数。

```
import argparse

# 获得一个参数解析器
parser = argparse.ArgumentParser(prog='ls', add_help=True, description='list directory contents')
parser.add_argument('path')

args = parser.parse_args()  # 分析参数
parser.print_help()  # 打印帮助

# 运行结果，出现了错误，提示需要输入path对应的位置参数
usage: ls [-h] path
ls: error: the following arguments are required: path
```

程序定义为：
ls [-h] path
-h为帮助选项，可有可无
path为位置参数，必须提供

传参

```
parse_args(args=None, namespace=None)
```

args 参数列表，一个可迭代对象。内部会把可迭代对象转换成list。如果为None则使用命令行传入参数，非None则使用args参数的可迭代对象。

```
import argparse

# 获得一个参数解析器
```

```
parser = argparse.ArgumentParser(prog='ls', add_help=True, description='list directory contents')
parser.add_argument('path') # 位置参数

args = parser.parse_args(('/etc',))  # 分析参数，同时传入可迭代的参数
print(args, args.path) # 打印名词空间中收集的参数
parser.print_help()  # 打印帮助

运行结果
Namespace(path='/etc') /etc
usage: ls [-h] path

list directory contents

positional arguments:
  path

optional arguments:
  -h, --help  show this help message and exit
```

Namespace(path='/etc')里面的path参数存储在了一个Namespace对象内的属性上，可以通过Namespace对象属性来访问，例如args.path

非必须位置参数

上面的代码必须输入位置参数，否则会报错。

```
usage: ls [-h] path
ls: error: the following arguments are required: path
```

但有时候，ls命令不输入任何路径的话就表示列出当前目录的文件列表。

```
import argparse

# 获得一个参数解析器
parser = argparse.ArgumentParser(prog='ls', add_help=True, description='list directory contents')
parser.add_argument('path', nargs='?', default='.', help="path help") # 位置参数，可有可无，缺省值，帮助

args = parser.parse_args()  # 分析参数，同时传入可迭代的参数
print(args) # 打印名词空间中收集的参数
parser.print_help()  # 打印帮助

# 运行结果
Namespace(path='.')
usage: ls [-h] [path]

list directory contents

positional arguments:
  path        path help

optional arguments:
  -h, --help  show this help message and exit
```

可以看出path也变成**可选**的位置参数，没有提供就使用默认值`.点号`表示当前路径。

- help 表示帮助文档中这个参数的描述
- nargs 表示这个参数接收结果参数
  - ? 表示可有可无
  - + 表示至少一个
  - * 可以任意个
  - 数字表示必须是指定数目个
- default 表示如果不提供该参数，就使用这个值。一般和？、*配合，因为它们都可以不提供位置参数，不提供就用缺省值

选项参数

-l的实现

`parser.add_argument('-l')`就增加了选项参数，参数定义为
`ls [-h][-l L] [path]`
和我们要的形式有一点出入，我们期望的是`[-l]`，怎么解决？

nargs能够解决吗？
parser.add_argument('-l', nargs='?')
`ls [-h][-l [L]] [path]`
-l还不是可选参数。

那么，直接把nargs=0，意思就是让这个选项接收0个参数，如下
`parser.add_argument('-l', nargs=0)`
结果，抛出异常

> raise ValueError('nargs for store actions must be > 0; if you '
> ValueError: nargs for store actions must be > 0; if you have nothing to store, actions such as store true or store const may be more appropriate

看来nargs是控制位置参数和选项参数的，不能影响选项参数的参数。

为了这个问题，使用**action参数**
`parser.add_argument('-l', action='store_true')`
看到命令定义变成了`ls [-h] [-l] [path]`

提供-l选项，例如
`ls -l`得到Namespace(l=True, path='.')，提供-l值是True
`ls`得到Namespace(l=False, path='.')，未提供-l值是False
这样同True、False来判断用户是否提供了该选项

```
parser.add_argument('-l', action='store_const', const = 20)
# 提供-l选项，属性值为20；否则，对应值为None
```

-a的实现

```
parser.add_argument('-a', '--all', action='store_true') # 长短选项同时给
```

属性名称

参数都是Namespace对象的属性，如果想指定这些属性名，可以使用dest。

```
parser.add_argument('-l', action='store_true', dest='longfmt')
```

代码

```
import argparse

# 获得一个参数解析器
parser = argparse.ArgumentParser(prog='ls', add_help=True, description='list directory contents')
parser.add_argument('path', nargs='?', default='.', help="directory") # 位置参数，可有可无，缺省值，帮助
parser.add_argument('-l', action='store_true', dest='longfmt', help='use a long listing format')
parser.add_argument('-a', '--all', action='store_true', help='show all files, do not ignore entries starting with .')

args = parser.parse_args()  # 分析参数，同时传入可迭代的参数
print(args) # 打印名词空间中收集的参数
parser.print_help()  # 打印帮助

# 运行结果
Namespace(all=False, longfmt=False, path='.')
usage: ls [-h] [-l] [-a] [path]

list directory contents

positional arguments:
  path        directory

optional arguments:
  -h, --help  show this help message and exit
  -l          use a long listing format
  -a, --all   show all files, do not ignore entries starting with .

# parser.parse_args('-l -a /tmp'.split())运行结果
Namespace(all=True, longfmt=True, path='/tmp')
```

作业

实现ls命令功能，实现-l、-a和--all、-h选项

- 实现显示路径下的文件列表
- -a和-all 显示包含.开头的文件
- -l 详细列表显示
- -h 和-l配合，人性化显示文件大小，例如1K、1G、1T等，可以认为1G=1000M

- 类型字符
  - c 字符
  - d 目录
  - 普通文件
  - l 软链接
  - b 块设备
  - s socket文件
  - p pipe文件，即FIFO

```
参看Linux、Unix命令ls -lah

-rw-rw-r--   1     python python     5      Oct 25 00:07       test4
mode        硬链接   属主     属组        字节     时间                文件名

按照文件名排序输出，可以和ls的顺序不一样，但要求文件名排序
要求详细列表显示时，时间可以按照“年-月-日 时:分:秒” 格式显示，例如2015-06-17 17:05:00
```