develop_skills

发布 *2019.08.08*

captcha server developers

2021 年 05 月 21 日

topic

1 write_the_doc **3**

1.1 文档环境及其依赖 . . . 3
1.2 quickstart . . . 3
1.3 初始配置 . . . 5
1.4 修改配置 . . . 6
1.5 日常撰写文档流程 . . . 10
1.6 文档展示 . . . 11

2 unittest and pytest **13**

2.1 测试 . . . 13
2.2 pytest . . . 17

3 logging **25**

3.1 简介 . . . 25
3.2 logging 及其组件 . . . 25
3.3 日志模块详述 . . . 28
3.4 best practice . . . 33
3.5 引用 . . . 33

4 debug **35**

4.1 pdb(ipdb) . . . 35
4.2 gdb . . . 41

5 索引与高级查询 **45**

6 2018-05-25 gt-day 问题答案参考–pandas **47**

7 2018-05-25 gt-day 问题答案参考–Spark **55**

8 Indices and tables 63

Author captcha developers group

Date 2019-08-05T08:10:11.588674+08:00

CHAPTER 1

write_the_doc

Author RYefccd

Date 2019-08-08T08:26:11.588674+08:00

1.1 文档环境及其依赖

```
pip install sphinx
```

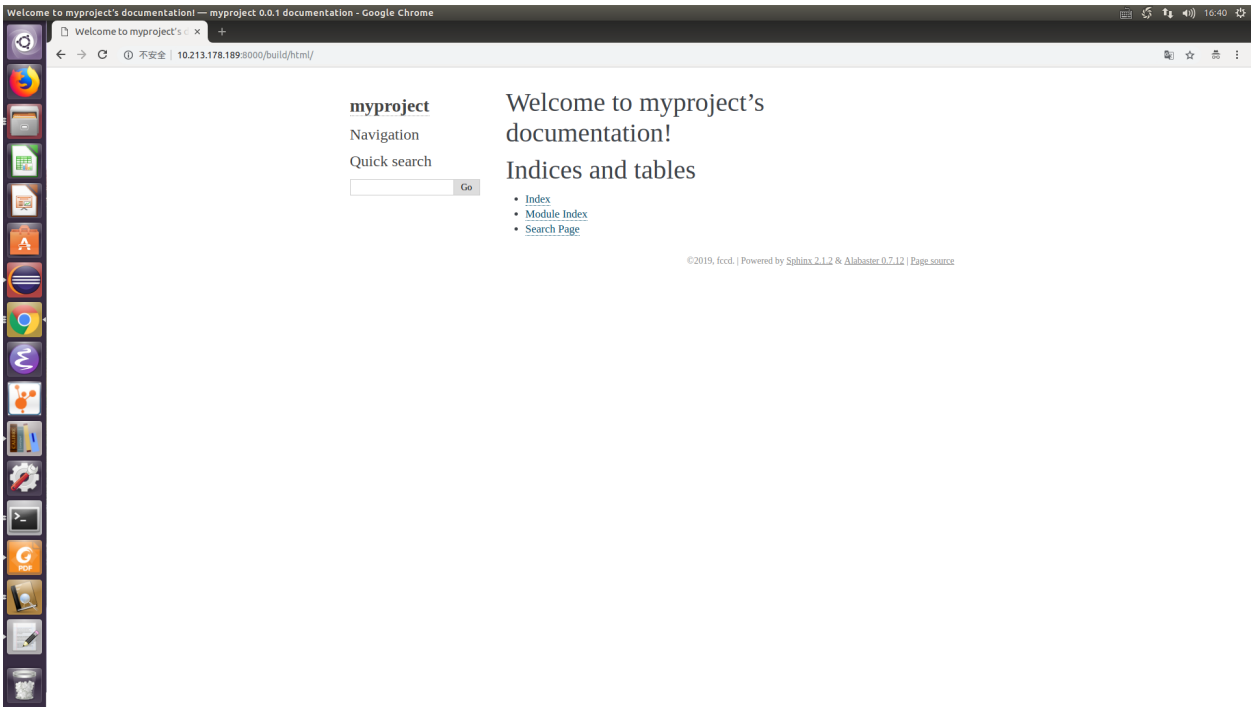
1.2 quickstart

```
(sphinx) ubuntu@pytorch:~/sphinx$ sphinx-quickstart demo/
Welcome to the Sphinx 2.1.2 quickstart utility.

Please enter values for the following settings (just press Enter to
accept a default value, if one is given in brackets).

Selected root path: demo/

You have two options for placing the build directory for Sphinx output.
Either, you use a directory "_build" within the root path, or you separate
"source" and "build" directories within the root path.
```

(下页继续)

(续上页)

```
> Separate source and build directories (y/n) [n]: y

The project name will occur in several places in the built documentation.
> Project name: myproject
> Author name(s): fccd
> Project release []: 0.0.1

If the documents are to be written in a language other than English,
you can select a language here by its language code. Sphinx will then
translate text that it generates into that language.

For a list of supported codes, see
https://www.sphinx-doc.org/en/master/usage/configuration.html#confval-language.
> Project language [en]:

Creating file demo/source/conf.py.
Creating file demo/source/index.rst.
Creating file demo/Makefile.
Creating file demo/make.bat.

Finished: An initial directory structure has been created.

You should now populate your master file demo/source/index.rst and create other␣
↪documentation
source files. Use the Makefile to build the docs, like so:
   make builder
where "builder" is one of the supported builders, e.g. html, latex or linkcheck.

(sphinx) ubuntu@pytorch:~/sphinx$ ls demo/
Makefile  build  make.bat  source

(sphinx) ubuntu@pytorch:~/sphinx/demo$ make html
Running Sphinx v2.1.2
loading pickled environment... done
building [mo]: targets for 0 po files that are out of date
building [html]: targets for 0 source files that are out of date
updating environment: 0 added, 0 changed, 0 removed
looking for now-outdated files... none found
no targets are out of date.
build succeeded.

The HTML pages are in build/html.
```



1.3 初始配置

列表 1: conf.py

```
# Configuration file for the Sphinx documentation builder.
#
# This file only contains a selection of the most common options. For a full
# list see the documentation:
# http://www.sphinx-doc.org/en/master/config

# -- Path setup --------------------------------------------------------------

# If extensions (or modules to document with autodoc) are in another directory,
# add these directories to sys.path here. If the directory is relative to the
# documentation root, use os.path.abspath to make it absolute, like shown here.
#
# import os
# import sys
# sys.path.insert(0, os.path.abspath('.'))


# -- Project information -----------------------------------------------------

```

(下页继续)

(续上页)

```
project = 'myproject'
copyright = '2019, fccd'
author = 'fccd'

# The full version, including alpha/beta/rc tags
release = '0.0.1'


# -- General configuration ---------------------------------------------------

# Add any Sphinx extension module names here, as strings. They can be
# extensions coming with Sphinx (named 'sphinx.ext.*') or your custom
# ones.
extensions = [
]

# Add any paths that contain templates here, relative to this directory.
templates_path = ['_templates']

# List of patterns, relative to source directory, that match files and
# directories to ignore when looking for source files.
# This pattern also affects html_static_path and html_extra_path.
exclude_patterns = []


# -- Options for HTML output -------------------------------------------------

# The theme to use for HTML and HTML Help pages.  See the documentation for
# a list of builtin themes.
#
html_theme = 'alabaster'

# Add any paths that contain custom static files (such as style sheets) here,
# relative to this directory. They are copied after the builtin static files,
# so a file named "default.css" will overwrite the builtin "default.css".
html_static_path = ['_static']
```

1.4 修改配置

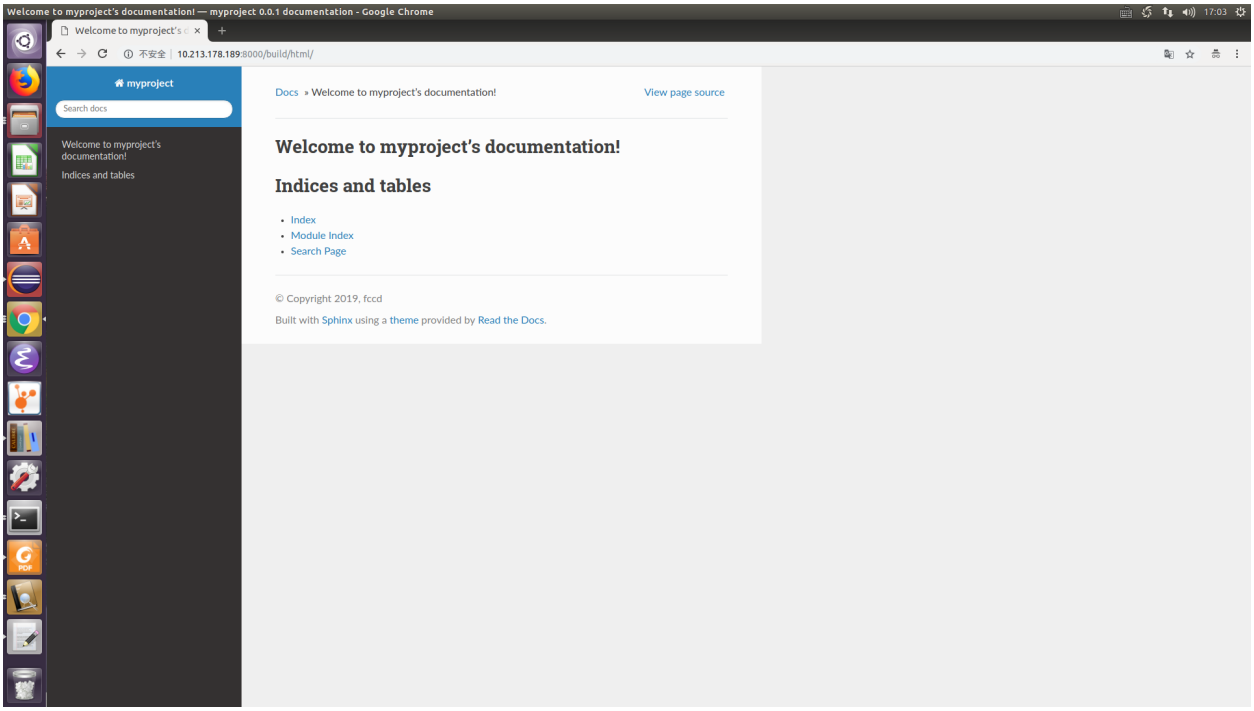
1.4.1 read the doc 风格文档

依赖:

```
pip install sphinx-rtd-theme
```

修改配置:

```
1
2 # The theme to use for HTML and HTML Help pages.  See the documentation for
3 # a list of builtin themes.
4 #
5 # html_theme = 'alabaster'
6 html_theme = 'sphinx_rtd_theme'
```



1.4.2 支持 markdown 格式

sphinx 默认支持 restructureText 格式, 如果需要支持 markdown 格式, 需要导入依赖和修改相关配置.

依赖:

```
pip install recommonmark
```

```
1 # markdown support
2 from recommonmark.parser import CommonMarkParser
3
4 source_parsers = {'.md': CommonMarkParser}
5 source_suffix = ['.rst', '.md']
```

1.4.3 支持 markdown table 格式

感谢胡达聪提供协助

sphinx 默认支持 restructureText 格式, 如果需要支持 markdown 格式, 需要导入依赖和修改相关配置. 目前 recommonmark 不支持 markdown table 的渲染, 如果需要支持, 请把 sphinx_markdown_tables 添加到 conf.py 配置文件中.

依赖:

```
pip instal sphinx-markdown-tables
```

```
extensions = [
 'sphinx_markdown_tables',
 ]
```

1.4.4 支持 ipynb(notebook) 文件格式

感谢胡达聪提供协助

依赖:

```
pip install nbsphinx
```

```
# Add any Sphinx extension module names here, as strings. They can be
# extensions coming with Sphinx (named 'sphinx.ext.*') or your custom
# ones.
extensions = [
'nbsphinx',
'sphinx.ext.mathjax',
]
```

1.4.5 支持中文搜索

感谢黄奇鹏提供配置

依赖:

```
pip install jieba
```

```
# 支持中文搜索
html_search_language = 'zh'
```

1.4.6 构建中文 pdf

https://docs.readthedocs.io/en/stable/guides/pdf-non-ascii-languages.html

For docs that are not written in Chinese or Japanese, and if your build fails from a Unicode error, then try `xelatex` as the `latex_engine` instead of the default `pdflatex` in your `conf.py`:

```
latex_engine = 'xelatex'
```

When Read the Docs detects that your documentation is in Chinese or Japanese, it automatically adds some defaults for you.

For *Chinese* projects, it appends to your `conf.py` these settings:

```
latex_engine = 'xelatex'
latex_use_xindy = False
latex_elements = {
    'preamble': '\\usepackage[UTF8]{ctex}\n',
}
```

1.4.7 autobuild

感谢曹佳豪提供此技巧分享

在撰写文档时, 每次想要看到效果都要执行 make html 才能看到渲染的 html 文档. 为了能够提升编辑文档的效率, 建议使用 autobuild 扩展. 一旦我们修改相关的文档和 conf.py 配置时, 就会自动触发构建.

依赖:

```
pip install sphinx-autobuild
```

使用:

```
sphinx-autobuild source build/html -H 0.0.0.0 -p 8000
最后在 http://localhost:8000 访问即可.
```

或者把这个放在 Makefile 中:

列表 2: Makefile

```
livehtml:
        mkdir -p $(BUILDDIR)/html/
```

```
make livehtml
```

1.5 日常撰写文档流程

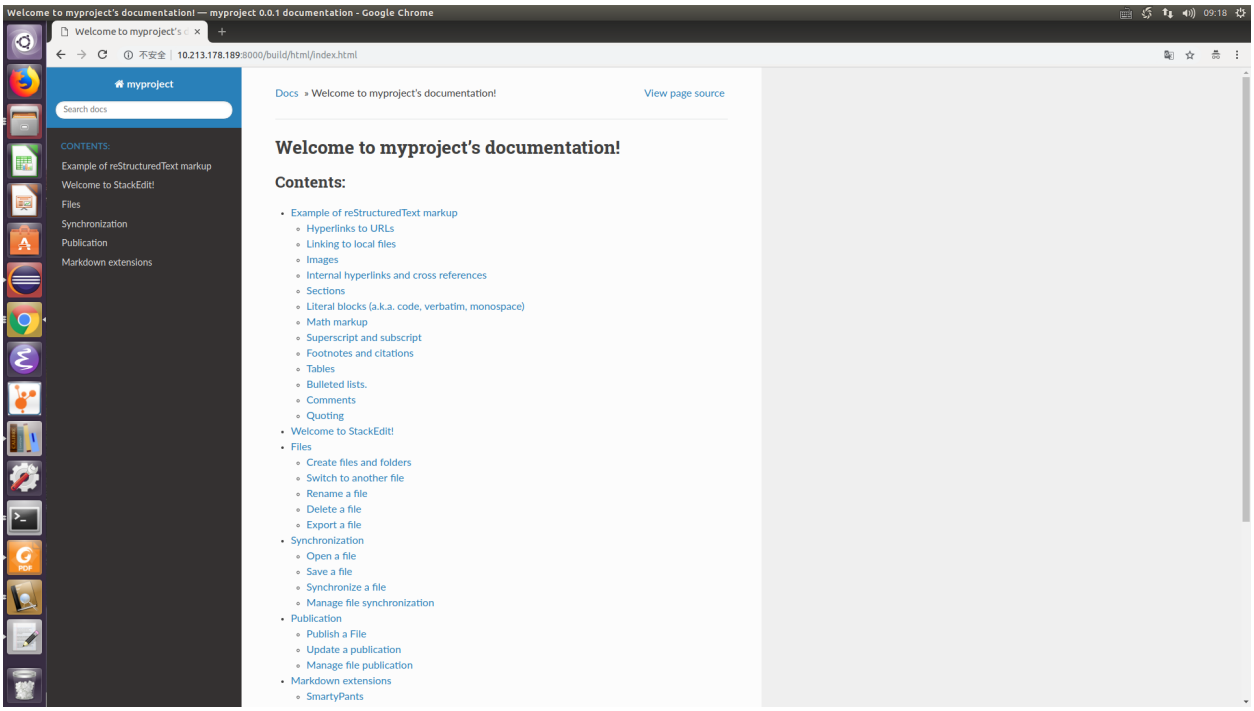
1. 开启自动文档构建

```
sphinx-autobuild source build/html -p 8000 -H 0.0.0.0
# 或者
make livehtml  # 参见上面的 Makefile 配置
```

2. 撰写文档

```
(sphinx) ubuntu@pytorch:~/sphinx/demo$ touch source/restructText_demo.rst
(sphinx) ubuntu@pytorch:~/sphinx/demo$ touch source/markdown_StackEdit.md
 ...
(sphinx) ubuntu@pytorch:~/sphinx/demo$ tree -l 2 .
.
├── Makefile
├── make.bat
└── source
    ├── _static
    ├── _templates
    ├── birthday-paradox.png
    ├── conf.py
    ├── index.rst
    ├── markdown_StackEdit.md
    └── restructText_demo.rst
```

3. 保存 (crtl+s) 触发自动构建
4. 在 http://localhost:8000/ 查看文档即可.



1.6 文档展示

rst 语法参考: https://3vshej.cn/rstSyntax/index.html

个人笔记: https://write-docs.readthedocs.io/en/latest/index.html

python-cookbook: https://python-cookbook-3rd-edition.readthedocs.io/zh_CN/latest/

CHAPTER 2

unittest and pytest

Author RYefccd

Date 2019-08-15T09:41:59.179550+08:00

2.1 测试

IEEE 定义: 使用人工或自动的手段来运行或测定某个软件系统的过程，其目的在于检验它是否满足规定的需求或弄清预期结果与实际结果之间的差别。

这个定义明确指出: 软件测试的目的是为了检验软件系统是否满足需求。

我们日常的测试希望保证新功能不会影响旧的逻辑，旧的代码逻辑修改不会干扰新的流程。加之我们的数据库中的配置项众多，接口测试难以覆盖我们新添加的代码，也难以确定对旧的功能的影响范围。针对这种情况，我们推进单元测试中的测试覆盖，来满足我们日益严峻的质量要求。

2.1.1 单元测试

以下面这段代码作为例子来说明不同的覆盖测试策略。

```
def my_demo_func(a, b):
    tmp = []
    if a > 6 and b > 9:
        tmp.append("F")
    else:
        tmp.append("T")
    print(tmp)
```

语句覆盖 (Statement Coverage)

Statement coverage is a white box test design technique which involves execution of all the `executable statements` in the source code at least once. It is used to calculate and measure the number of statements in the source code which can be executed given the requirements.

Statement coverage is used to derive scenario based upon the structure of the code under test.

$$\textbf{Statement Coverage} = \frac{\textit{Number of executed statements}}{\textit{Total number of statements}} \times 100$$

传入参数 a, b。观察语句覆盖情况。

- 假设给定 a=7, b=10, 代码执行覆盖如下:

```
def my_demo_func(a, b):
    tmp = []
    if a > 6 and b > 9:
        tmp.append("F")
    else:
        tmp.append("T")
    print(tmp)
```

Statement Coverage: 5/7 = 71%

What is covered by Statement Coverage?

1. Unused Statements
2. Dead Code
3. Unused Branches

判定覆盖 (Decision Coverage)

Decision coverage reports the true or false outcomes of each `Boolean expression`. In this coverage, expressions can sometimes get complicated. Therefore, it is very hard to achieve 100% coverage.

$$Decision\ Coverage = \frac{Number\ of\ Decision\ Outcomes\ Excercised}{Total\ Number\ of\ Decision\ Outcomes}$$

对 `a > 6 and b > 9` 整个 Boolean expression 构造整体表达式为真或者为假的判定逻辑。

- 假设给定 a=7, b=10, 代码执行覆盖如下: (`a > 6 and b > 9` is True)

```
def my_demo_func(a, b):
    tmp = []
    if a > 6 and b > 9:
        tmp.append("F")
    else:
        tmp.append("T")
    print(tmp)
```

Statement Coverage: 5/7 = 71%

- 假设给定 a=1, b=10, 代码执行覆盖如下:(`a > 6 and b > 9` is False)

```
def my_demo_func(a, b):
    tmp = []
    if a > 6 and b > 9:
        tmp.append("F")
    else:
        tmp.append("T")
    print(tmp)
```

Statement Coverage: 6/7 = 85%

分支覆盖 (Branch Coverage)

<table border="1" class="docutils"> <thead> <tr> <th>In the branch coverage, every outcome from a code module is tested. For example, if the outcomes are binary, you need to test both True and False outcomes.</th> </tr> </thead> <tbody> <tr> <td>It helps you to ensure that every possible branch from each decision condition is executed at least a single time.</td> </tr> <tr> <td></td> </tr> <tr> <td>By using Branch coverage method, you can also measure the fraction of independent code segments. It also helps you to find out which is sections of code don't have any branches.</td> </tr> <tr> <td></td> </tr> <tr> <td>The formula to calculate Branch Coverage:</td> </tr> </tbody> </table>

$$Branch\ Coverage = \frac{Number\ of\ Executed\ Branches}{Total\ Number\ of\ Branches}$$

分支覆盖就是构造的测试逻辑覆盖到了每一个条件判断的分支。(if-elif-else)
在这个例子中, 分支覆盖和上面的判定覆盖等价. 如果是有多个 if-elif-else 逻辑的话, 如下所示, 就有三个分支, 两个判定需要覆盖。

```
def my_demo_func(a, b):
    tmp = []
    if a > 6 and b > 9:
        tmp.append("F")
    elif: a > 2:
        pass
    else:
        tmp.append("T")
    print(tmp)
```

在实际测试中, 分支覆盖是我们最为关注的. 哪些分支没有被覆盖, 是因为什么原因没有被覆盖.....

Branch coverage Testing offers the following advantages:

- Allows you to validate-all the branches in the code
- Helps you to ensure that no branched lead to any abnormality of the program's operation
- Branch coverage method removes issues which happen because of statement coverage testing
- Allows you to find those areas which are not tested by other testing methods
- It allows you to find a quantitative measure of code coverage
- Branch coverage ignores branches inside the Boolean expressions

条件覆盖 (Condition Coverage)

Conditional coverage or expression coverage will reveal how the variables or subexpressions in the conditional statement are evaluated. In this coverage expressions with logical operands are only considered.

For example, if an expression has Boolean operations like AND, OR, XOR, which indicated total possibilities.

Conditional coverage offers better sensitivity to the control flow than decision coverage. Condition coverage does not give a guarantee about full decision coverage

The formula to calculate Condition Coverage:

$$Condition\ Coverage = \frac{Number\ of\ Executed\ Operands}{Total\ Number\ of\ Operands}$$

对于 `a > 6 and b > 9` 整个 Boolean expression, 我们有两个条件 a > 6 和 b > 9.

test	a > 6	b > 9
a=3, b=3	F	F
a=3, b=13	F	T
a=9, b=3	T	F
a=9, b=13	T	T

条件覆盖

2.2 pytest

- 方便的 assert 语句 (不需要记忆各种 self.assert* 断言函数)
- 自动发现测试模块和测试函数
- 模块化的 fixture, 可以更加容易组织测试结构。
- 兼容 unittest 测试用例, 无缝对接原有测试用例。

2.2.1 example

演示项目下载

测试项目结构如下:

```
(server18) ryefccd@fccd:~/workspace/pytest_demo$ tree -L 2
.
├── myproject
│   ├── handler.py
```

(下页继续)

(续上页)

```
|   ├── __init__.py
|   ├── mathexample.py
|   └── __pycache__
├── requirement_dev.txt
└── tests
    ├── __init__.py
    ├── __pycache__
    ├── test_math_opration.py
    └── test_tornado_client.py

4 directories, 7 files
```

普通模块

- 功能代码

列表 1: mathexample.py

```
'''
Created on 2019 年 8 月 15 日

@author: ryefccd
'''


def add_two(num1, num2):
    return num1 + num2


def sub_two(num1, num2):
    return num1 - num2
```

- 功能测试

列表 2: test_math_opration.py

```
'''
Created on 2019 年 8 月 15 日

@author: ryefccd
'''
import pytest
from myproject import mathexample
```

(下页继续)

(续上页)

```
8
9
10  @pytest.fixture(scope='module')
11  def resource_a_setup(request):
12      print('\nresources_a_setup()')
13
14      def resource_a_teardown():
15          print('\nresources_a_teardown()')
16      request.addfinalizer(resource_a_teardown)   # 可以在这里回收数据库连接
17
18      return 1234567
19
20
21  def test_add_two(resource_a_setup):
22      add = mathexample.add_two(resource_a_setup, 2)
23      assert add == 1234567 + 2
24
25
26  def test_sub_two():
27      substract = mathexample.sub_two(1, 2)
28      assert substract == -1
29      print("fccdny")
30
31
32  # @pytest.mark.skip(msg='failure')
33  def test_add_two_failure():
34      add = mathexample.add_two(1, 2)
35      assert add == 4
36
37
38  if __name__ == '__main__':
39      pass
```

- 执行测试

```
(server18) ryefccd@fccd:~/workspace/pytest_demo$ pytest tests/test_math_opration.
↪py
Test session starts (platform: linux, Python 3.5.2, pytest 5.0.1, pytest-sugar 0.
↪9.2)
rootdir: /home/ryefccd/workspace/pytest_demo
plugins: sugar-0.9.2, metadata-1.8.0, allure-pytest-2.7.1, xdist-1.29.0, cov-2.7.
↪1, forked-1.0.2, tornado-0.8.0, html-1.20.0
collecting ...
 tests/test_math_opration.py ✓✓                                                ␣
↪                                                                              ␣
↪                                    67% ███████
```

(下页继续)

(续上页)

```
_____________________________________________________________________________________________
↪test_add_two_failure          ____________________

    def test_add_two_failure():
        add = mathexample.add_two(1, 2)
>       assert add == 4
E       assert 3 == 4

 tests/test_math_opration.py:35: AssertionError

 tests/test_math_opration.py F                                                              [
↪                                                                                           [
↪                              100% ██████████

Results (0.12s):
       2 passed
       1 failed
         - tests/test_math_opration.py:33 test_add_two_failure
```

web 框架

依赖 pytest-tornado

```
pip install pytest-tornado
```

- 功能代码

列表 3: handler.py

```
'''
Created on 2018 年 8 月 19 日

@author: ryefccd
'''
import json
import asyncio

import aioredis
import tornado.web

from myproject.mathexample import add_two, sub_two
SERVER_REDIS_ADDRESS = ['192.168.1.200', 6379]

```

(下页继续)

(续上页)

```

class MainHandler(tornado.web.RequestHandler):
    def get(self):
        a = int(self.get_argument("a", "6"))
        b = int(self.get_argument("b", "2"))
        num_sum = add_two(a, b)
        num_delta = sub_two(a, b)
        res = {"sum": num_sum,
               "delta": num_delta,
               "a": a,
               "b": b}
        self.write(json.dumps(res))

application = tornado.web.Application([
    (r"/", MainHandler),
])


def init_app(ioloop, application):
    redis_conn = std_loop.run_until_complete(
        aioredis.create_redis_pool(SERVER_REDIS_ADDRESS, db=0))
    application.settings["REDIS_CONN"] = redis_conn


if __name__ == '__main__':
    std_loop = asyncio.get_event_loop()
    init_app(std_loop, application)
    http_server = tornado.httpserver.HTTPServer(application)
    http_server.listen(8989)
    std_loop.run_forever()
```

- 功能测试

列表 4: test_tornado_client.py

```
'''
Created on 2019 年 8 月 15 日

@author: ryefccd
'''
import json
import pytest
from myproject import handler

```

(下页继续)

(续上页)

```
10
11  @pytest.fixture
12  def app():
13      return handler.application
14
15
16  @pytest.fixture(scope="function")
17  def db_init(io_loop, app):
18      std_ioloop = io_loop.asyncio_loop
19      handler.init_app(io_loop, app)
20      # 填充数据库数据
21      yield
22      # 清楚数据库数据
23      conn = app.settings["REDIS_CONN"]
24      conn.close()
25      std_ioloop.run_until_complete(conn.wait_closed())
26
27
28  @pytest.mark.gen_test
29  def test_tornao_request_success(http_client, base_url):
30      url = base_url + "?a=7&b=2"
31      print("url:", url)
32      print("base_url:", base_url)
33      print("http_client:", http_client)
34      response = yield from http_client.fetch(url)
35
36      assert response.code == 200
37
38
39  @pytest.mark.gen_test
40  def test_tornao_request_fail(http_client, base_url):
41      url = base_url + "?a=7&b=2"
42      print("url:", url)
43      print("base_url:", base_url)
44      print("http_client:", http_client)
45      response = yield from http_client.fetch(url)
46      res = json.loads(response.body.decode())
47      print(res)
48      assert response.code == 200
49      assert res["sum"] == 10
50
51
52  if __name__ == '__main__':
53      pass
```

- 执行测试

```
(server18) ryefccd@fccd:~/workspace/pytest_demo$ pytest tests/test_tornado_client.
↪py
Test session starts (platform: linux, Python 3.5.2, pytest 5.0.1, pytest-sugar 0.
↪9.2)
rootdir: /home/ryefccd/workspace/pytest_demo
plugins: sugar-0.9.2, metadata-1.8.0, allure-pytest-2.7.1, xdist-1.29.0, cov-2.7.
↪1, forked-1.0.2, tornado-0.8.0, html-1.20.0
collecting ...
 tests/test_tornado_client.py ✓                                                 ␣
↪                                                                               ␣
↪                                50% ████▌

_____________________________________________________ test_tornao_request_fail␣
↪_____________________________________________________________________________

http_client = <tornado.simple_httpclient.SimpleAsyncHTTPClient object at␣
↪0x7f1393bfd358>, base_url = 'http://localhost:34575'

    @pytest.mark.gen_test
    def test_tornao_request_fail(http_client, base_url):
        url = base_url + "?a=7&b=2"
        print("url:", url)
        print("base_url:", base_url)
        print("http_client:", http_client)
        response = yield from http_client.fetch(url)
        res = json.loads(response.body.decode())
        print(res)
        assert response.code == 200
>       assert res["sum"] == 10
E       assert 9 == 10

tests/test_tornado_client.py:49: AssertionError
------------------------------------------------------------------- Captured␣
↪stdout call    ------------------------------------------------------------------
↪---------
url: http://localhost:34575?a=7&b=2
base_url: http://localhost:34575
http_client: <tornado.simple_httpclient.SimpleAsyncHTTPClient object at␣
↪0x7f1393bfd358>
{'a': 7, 'delta': 5, 'sum': 9, 'b': 2}

 tests/test_tornado_client.py ⨯                                                 ␣
↪                                                                               ␣
↪                               100% ██████████
```

(下页继续)

(续上页)

```
Results (0.16s):
       1 passed
       1 failed
         - tests/test_tornado_client.py:39 test_tornao_request_fail
```

2.2.2 pytest 使用技巧

```
pytest --help  # 查看帮助
```

- **-v** 详细的输出信息
- **-s** 不捕获标准输出 (测试用例中的 print 会打印出来)
- **-l** 当用例错误时, 打印测试函数内局部变量信息
- **-k EXPRESSION** 执行用例包含"EXPRESSION" 的用例
- **-x, --exitfirst** 当遇到错误时停止测试 (当维护很多测试用例时, 最迫切需要的功能)
- **--lf, --last-failed** 跑上一次错误的测试用例
- **--ff, --failed-first** 跑所有的用例, 但是优先上一次错误的用例
- **--pdb** 错误的测试用例陷入 pdb 调试环境

参考资料: pytest introduction

Written with StackEdit by RYefccd in 2019-11-27T14:00:58.752295+08:00.

CHAPTER 3

logging

3.1 简介

日志就是追踪在软件运行时产生的事件的方法. 一般由该软件的开发人员将日志记录调用添加到其代码中, 以指示已发生某些事件.

对于我们公司来说, 开发借助日志进行调试, 测试使用日志校验逻辑和功能, 运维监控日志提供预警, 数据依赖日志分析行为偏好.

作用如下:

- 信息搜集
- 故障排查
- 采样统计信息
- 审计

因此, 日志记录是非常有必要的. 作为开发者, 我们需要重视并做好日志记录过程.

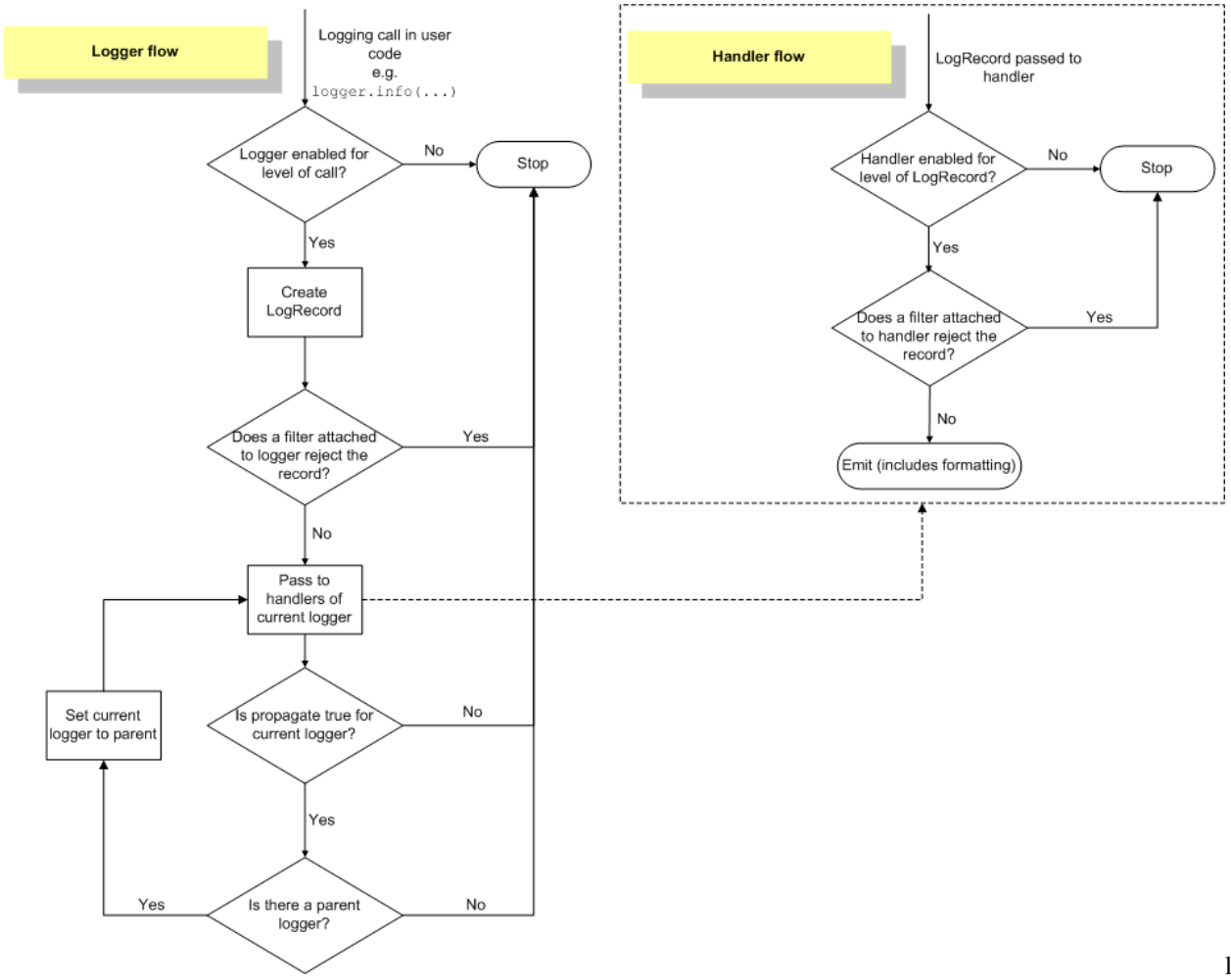
3.2 logging 及其组件

在 Python 中有一个标准的 logging 模块，我们可以使用它来进行标注的日志记录，利用它我们可以更方便地进行日志记录，同时还可以做更方便的级别区分以及一些额外日志信息的记录，如时间、运行模块信息等。

- Loggers

- Filters
- Handlers
- Formatters
- LogRecord

组件交互图:



logging flow

3.2.1 Logger

Logger 是用来执行日志流程的主类. 日志内容的产生, 转化, 过滤在此进行. Logger 默认是一个树型的继承体系. 默认就是的 logger 又叫做 root logger. 其他的 logger 均继承此 root logger. Logger 的层级通过”.” 来进行区分

如下, spam 是 spam.foo 的父 logger, 而 spam.foo 是 spam.foo.bar 的父 logger. (可以理解为一颗前缀树)

```
spam=logging.getLogger("spam")
spam_foo=logging.getLogger("spam.foo")
```

(下页继续)

(续上页)

```
spam_foo_bar=logging.getLogger("spam.foo.bar")

spam_foo_bar.info("the message 1")
spam.info("the message 2")
```

层级

Logger 只负责生成日志内容, 并传递至父类的各级 Logger 中. 以上面的代码举例子,

1. "the message 1" 会在四个 Logger 都生成一个消息记录 (LogRecord).
2. "the message 2" 会在两个 Logger 都生成一个消息记录 (LogRecord).

如果要阻止向父类 Logger 传递日志内容, 请把 **propagate** 属性置为 False.

level

Logger 会过滤掉小于当前等级的日志内容. 默认是日志级别是 **WARNING**.

3.2.2 Handlers

Logger 负责日志的产生, 而日志的最终输出就由 **Handlers** 负责进行. 下面是输出到文件的例子:

```
import logging
logger = logging.getLogger("example")
logger.setLevel(level=logging.INFO)
handler = logging.FileHandler('example.log')
formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')
handler.setFormatter(formatter)
logger.addHandler(handler)
logger.info('info logs')
logger.debug('debugging logs')
logger.warning('warning logs')
logger.info('final info logs')
```

```
# 日志输出

2019-11-26 16:39:35,005 - example - INFO - info logs
2019-11-26 16:39:35,005 - example - WARNING - warning logs
2019-11-26 16:39:35,005 - example - INFO - final info logs
```

logging:

- StreamHandler: 日志输出流, 可以到标准输出 (/dev/stdout), 标准错误输出 (/dev/stderr)

- FileHandler: 如上所示, 日志输出到文件.
- `NullHandler` : 空 Handler, 只记录日志, 不输出日志.

logging.handlers

- RotatingHandler: 按大小转储日志文件.
- TimeRotatingHandler: 按时间周期转储日志文件.
- SysLogHandler: 把日志输出到 syslog.
- SMTPHandler: 把日志输出至指定邮件地址.
- HTTPHandler: 通过 HTTP 协议输出日志.

除了这些常用的 Handler, 还有基于 TCP, UDP 的, 以及基于不同组件输出日志. 继承基类 Handler, 还可以实现更多的定制化的 Handler, 比如你可以实现一个 Handler 直接把日志输出至 kafka, 或者数据库等.

3.2.3 Formatters && LogRecord

控制日志的输出目的地是 **Handler**, 输出的格式就由 Formatters 指定. 上面例子中的格式如下:

```
# Format:
'%(asctime)s - %(name)s - %(levelname)s - %(message)s'
```

结果日志输出如下:

```
2019-11-26 16:39:35,005 - example - INFO - info logs
2019-11-26 16:39:35,005 - example - WARNING - warning logs
2019-11-26 16:39:35,005 - example - INFO - final info logs
```

%(asctime)s 输出本地时间%(name)s 输出 logger 的名字%(levelname)s 输出日志的级别%(message)s 输出实际的日志内容

Format 和 LogRecord 的属性 (Attribute) 是一一对应的. 请参考: https://docs.python.org/3/library/logging.html#logrecord-attributes

3.3 日志模块详述

3.3.1 使用 logging 替代 print

很多人习惯在开发时使用 print 替代日志, 然后再代码发布时移除这些 print 语句. 随着代码层级的加深和逻辑的复杂, 使用 print 很难满足多变的日志格式的需求, 甚至你可能忘了去掉这些 print 语句. 各种信息混在一起, 使得日志的排查愈加困难.

3.3.2 日志 (Logger) 是单实例的

同一个 Logger 在不同模块获得都是一个实例. 如下所示:

```
spam=logging.getLogger("spam")
```

如果在 a, b 模块都是用此语句获得 spam Logger, 则这是同一个实例.

3.3.3 在日志中记录相关异常栈

把异常栈记录到日志中:

```
import logging
import traceback
try:
  c = 7 / 0
except Exception as e:
    logging.error("Exception occurred:%s", traceback.format_exc())
```

请使用 pythonic 的做法:

```

import logging
try:
  c = 7 / 0
except Exception as e:
  logging.error("Exception occurred", exc_info=True)
```

3.3.4 线上环境使用日志转储

在生产环境中, 程序会一直产生日志, 为了防止磁盘被日志文件占满. 可以按照文件大小转储或时间转储日志文件, 这样可以节省维护的压力.

3.3.5 使用合理的日志级别

合理安排输出日志的级别, 避免日志洪流. 尽可能思考输出不同等级的日志. 便于后期不同需求的人来排查逻辑.

3.3.6 库程序使用 NullHandler

如果是提供别人使用的程序库, 一般会把 warnning 信息输出到 stderr(标准错误输出). 但是如果应用的标准错误输出有其他的用途, 不希望任何引用的第三方库把日志输出到标准错误输出时或者输出到其他的任何输出

位置, 那么可以使用一个占位符的” 空”Handler, 它接收到日志后什么也不做. 如下所示,

```
# foo.py
import logging
logging.getLogger('foo').addHandler(logging.NullHandler())
```

参考:https://docs.python.org/3/howto/logging.html#configuring-logging-for-a-library

It is strongly advised that you *do not add any handlers other than* `NullHandler` *to your library' s loggers.* This is because the configuration of handlers is the prerogative of the application developer who uses your library. The application developer knows their target audience and what handlers are most appropriate for their application: if you add handlers 'under the hood', you might well interfere with their ability to carry out unit tests and deliver logs which suit their requirements.

3.3.7 使用 name 作为 Logger 名字

这样可以使得 Logger 的名字和模块的名字统一. 可以用 Formatters 在日志中显示记录日志的模块, 行号. 便于定位程序逻辑的位置. 并且如上面的例子所示, 恰好也满足日志的继承层级,

```
spam_foo=logging.getLogger("spam.foo")
```

这样可以只给父类 Logger 设置 Handler, 那么模块中所有子子模块输出的日志都能输出.

3.3.8 在应用中使用 basicConfig 日志配置

在应用中有一个方便的初始化 root logger 的方法, **logging.basicConfig**. 可以在应用程序启动时调用此方法, 然后按照 logger 的继承体系, 就可以搜集到所有 logger 输出的日志信息.

如果 root logger 没有定义任何 handler, 日志函数 debug, info, warning, error, critical 将会自动调用此函数. 默认输出到标准错误 (stderr) 输出上.

```
import logging
logging.basicConfig(filename='example.log',level=logging.DEBUG)
```

3.3.9 多进程写日志

因为我们的线上服务 tornado 使用多进程启动. 一旦在开始时配置了日志, 又使用日志大小转储. 那么多个进程就会往一个日志文件里面写日志. 直到某一个进程写入一个新的日志记录时发现需要转储, 它便转储当前日志文件, 并重新创建一个新的日志文件. 但是其他进程并不知道, 仍然维持之前被重命名转储的那个文件 (linux 中不是靠文件名来识别文件, 而是文件句柄, 一个数字). 当另一个进程也写一条日志时, 发现也需要转储, 会重新转储当前日志, 并重新写新的日志. 结果就是会导致日志的混乱.

单进程写日志

代码如下:

```
import os
import sys
import logging
from logging.handlers import RotatingFileHandler
rotate_hd = RotatingFileHandler(filename="/tmp/whtest/test.log",
                                maxBytes=1*1024, backupCount=2)
f_format = logging.Formatter('%(asctime)s - %(name)s - %(process)d - %(message)s')
rotate_hd.setFormatter(f_format)
logging.basicConfig(handlers=[rotate_hd], level=logging.DEBUG)
logging.debug('This is a debug message')
logging.info('This is an info message')
logging.warning('This is a warning message')
logging.error('This is an error message')
logging.critical('This is a critical message')
pid = os.getpid()
for i in range(100):
    logging.info("info message")
```

结果如下:

```
ryefccd@fccd:/tmp/whtest$ ll
总用量 152
drwxrwxr-x  2 ryefccd ryefccd   4096 11 月 27 12:10 ./
drwxrwxrwt 15 root    root    135168 11 月 27 12:09 ../
-rw-rw-r--  1 ryefccd ryefccd    990 11 月 27 12:10 test.log
-rw-rw-r--  1 ryefccd ryefccd    990 11 月 27 12:10 test.log.1
-rw-rw-r--  1 ryefccd ryefccd    990 11 月 27 12:10 test.log.2
```

多进程写日志

代码如下:

```
import os
import sys
import logging
from logging.handlers import RotatingFileHandler
rotate_hd = RotatingFileHandler(filename="/tmp/whtest/test.log",
                                maxBytes=1*1024, backupCount=2)
f_format = logging.Formatter('%(asctime)s - %(name)s - %(process)d - %(message)s')
rotate_hd.setFormatter(f_format)
```

(下页继续)

(续上页)

```
logging.basicConfig(handlers=[rotate_hd], level=logging.DEBUG)
logging.debug('This is a debug message')
logging.info('This is an info message')
logging.warning('This is a warning message')
logging.error('This is an error message')
logging.critical('This is a critical message')

# pid = os.getpid()
for _ in range(3):
    pid = os.fork()
for i in range(100):
    logging.info("info message")
```

结果如下:

```
--- Logging error ---
Traceback (most recent call last):
  File "/home/ryefccd/python3.5/python3.5.2/lib/python3.5/logging/handlers.py", line␣
↪72, in emit
    self.doRollover()
  File "/home/ryefccd/python3.5/python3.5.2/lib/python3.5/logging/handlers.py", line␣
↪169, in doRollover
    os.rename(sfn, dfn)
FileNotFoundError: [Errno 2] No such file or directory: '/tmp/whtest/test.log.1' -> '/
↪tmp/whtest/test.log.2'
Call stack:
  File "logging_test.py", line 21, in <module>
    logging.info("info message")
Message: 'info message'
Arguments: ()
ryefccd@fccd:/tmp/whtest$ ll
总用量 152
drwxrwxr-x  2 ryefccd ryefccd   4096 11 月 27 12:20 ./
drwxrwxrwt 15 root    root    135168 11 月 27 12:19 ../
-rw-rw-r--  1 ryefccd ryefccd    540 11 月 27 12:20 test.log
-rw-rw-r--  1 ryefccd ryefccd    972 11 月 27 12:20 test.log.1
-rw-rw-r--  1 ryefccd ryefccd    540 11 月 27 12:20 test.log.2
```

最终方案

1. 最简单, 也是最直接可靠的办法就是一个进程写一个日志. 在 fork 之后初始化日志并更根据进程号写不同的日志文件.

2. RotatingFileHandler 进行转储时不仅判断大小还要判断当前的文件名. 优先判断当前文件的文件名, 如果

不是原始的文件名, 则修改为往最开始的文件名中写入日志.(没来得及做)

3. https://docs.python.org/3/howto/logging-cookbook.html#logging-to-a-single-file-from-multiple-processes

3.4 best practice

- 日志记录要有意义.
- 日志记录最好包含上下文信息.
- 日志要结构化以及设计不同的层级, 便于解析和阅读
- 日志不能包含太少或者太多的信息.
- 复杂的应用可以分不同的应用或者模块输出到不同的日志文件中.
- 让日志适应开发, 测试, 线上不同的环境, 方便不同人员通过日志协同.

3.5 引用

- Logging HOWTO
- Logging in Python
- Python logging tutorial
- Good logging practice in Python
- *Python Logging Basics*

Written with StackEdit.

CHAPTER 4

debug

4.1 pdb(ipdb)

4.1.1 demo

- start.py

```
import foo
import bar

tmp1 = "123"
tmp2 = "3f6"

print("tmp1 is num: %s by foo.is_num func" % foo.is_num(tmp1))
print("tmp2 is num: %s by foo.is_num func" % foo.is_num(tmp2))

print("tmp1 is num: %s by bar.is_num func" % bar.is_num(tmp1))
print("tmp2 is num: %s by bar.is_num func" % bar.is_num(tmp2))
```

- foo.py

```
def is_num(x):
    flag = x.isnumeric()
    return flag
```

- bar.py

```
def is_num(x):
    start = ord('0')
    end = ord('9')
    flag = True
    for i in x:
        num = ord(i)
        if not start <= num <= end:
            flag = False
            break
    return flag
```

4.1.2 introduction

- 进入调试的两种方法

1.

```
import pdb; pdb.set_trace()
```

1.

```
python -m pdb start.py
```

- use ipdb

```
pip install ipdb
python -m ipdb start.py
```

- printing expressions

```
> /home/ryefccd/env/server18/debug/start.py(6)<module>()
      5 tmp1 = "123"
----> 6 tmp2 = "3f6"
      7

ipdb> p tmp1
'123'
ipdb> p tmp2
*** NameError: name 'tmp2' is not defined
```

- stepping through code with `n` (next) and `s` (step)

```
ipdb> ll
      1
      2 import foo
      3 import bar
      4
      5 tmp1 = "123"
      6 tmp2 = "3f6"
      7
----> 8 print("tmp1 is num: %s by foo.is_num func" % foo.is_num(tmp1))
      9 print("tmp2 is num: %s by foo.is_num func" % foo.is_num(tmp2))
     10
     11 print("tmp1 is num: %s by bar.is_num func" % bar.is_num(tmp1))
     12 print("tmp2 is num: %s by bar.is_num func" % bar.is_num(tmp2))

ipdb> n
tmp1 is num: %s by foo.is_num func True
> /home/ryefccd/env/server18/debug/start.py(9)<module>()
      8 print("tmp1 is num: %s by foo.is_num func" % foo.is_num(tmp1))
----> 9 print("tmp2 is num: %s by foo.is_num func" % foo.is_num(tmp2))
     10

ipdb> s
--Call--
> /home/ryefccd/env/server18/debug/foo.py(1)is_num()
----> 1 def is_num(x):
      2     flag = x.isnumeric()
      3     return flag
```

- using breakpoints

```
ipdb> w
  /home/ryefccd/python3.5/python3.5.2/lib/python3.5/bdb.py(431)run()
    430         try:
--> 431             exec(cmd, globals, locals)
    432         except BdbQuit:

  <string>(1)<module>()

> /home/ryefccd/env/server18/debug/start.py(11)<module>()
     10
---> 11 print("tmp1 is num: %s by bar.is_num func" % bar.is_num(tmp1))
     12 print("tmp2 is num: %s by bar.is_num func" % bar.is_num(tmp2))

ipdb> ll
```

(下页继续)

(续上页)

```
      ...
      8 print("tmp1 is num: %s by foo.is_num func" % foo.is_num(tmp1))
      9 print("tmp2 is num: %s by foo.is_num func" % foo.is_num(tmp2))
     10
---> 11 print("tmp1 is num: %s by bar.is_num func" % bar.is_num(tmp1))
     12 print("tmp2 is num: %s by bar.is_num func" % bar.is_num(tmp2))

ipdb> b bar:3
Breakpoint 1 at /home/ryefccd/env/server18/debug/bar.py:3
ipdb> c
> /home/ryefccd/env/server18/debug/bar.py(3)is_num()
      2     start = ord('0')
1---> 3     end = ord('9')
      4     flag = True
```

- continuing execution with `unt` (until)

```
ipdb> ll
      1 def is_num(x):
      2     start = ord('0')
1---> 3     end = ord('9')
      4     flag = True
      5     for i in x:
      6         num = ord(i)
      7         if not start <= num <= end:
      8             flag = False
      9             break
     10     return flag

ipdb> l

ipdb> unt 8
> /home/ryefccd/env/server18/debug/bar.py(10)is_num()
      8             flag = False
      9             break
---> 10     return flag

ipdb> p flag
True
```

- displaying expressions

```
> /home/ryefccd/env/server18/debug/bar.py(3)is_num()
      2     start = ord('0')
```

(下页继续)

(续上页)

```
1---> 3     end = ord('9')
      4     flag = True

ipdb> ll
      1 def is_num(x):
      2     start = ord('0')
1---> 3     end = ord('9')
      4     flag = True
      5     for i in x:
      6         num = ord(i)
      7         if not start <= num <= end:
      8             flag = False
      9             break
     10     return flag

ipdb> display i, num
display i, num: ** raised NameError: name 'i' is not defined **
ipdb> b 7
Breakpoint 2 at /home/ryefccd/env/server18/debug/bar.py:7
ipdb> b
Num Type         Disp Enb   Where
1   breakpoint   keep yes   at /home/ryefccd/env/server18/debug/bar.py:3
        breakpoint already hit 1 time
2   breakpoint   keep yes   at /home/ryefccd/env/server18/debug/bar.py:7
ipdb> c
> /home/ryefccd/env/server18/debug/bar.py(7)is_num()
      6         num = ord(i)
2---> 7         if not start <= num <= end:
      8             flag = False

display i, num: ('1', 49)  [old: ** raised NameError: name 'i' is not defined **]
ipdb> c
> /home/ryefccd/env/server18/debug/bar.py(7)is_num()
      6         num = ord(i)
2---> 7         if not start <= num <= end:
      8             flag = False

display i, num: ('2', 50)  [old: ('1', 49)]
ipdb> c
> /home/ryefccd/env/server18/debug/bar.py(7)is_num()
      6         num = ord(i)
2---> 7         if not start <= num <= end:
      8             flag = False
```

(下页继续)

(续上页)

```
display i, num: ('3', 51)  [old: ('2', 50)]
```

- finding the caller of a function(where)

```
ipdb> w
  /home/ryefccd/python3.5/python3.5.2/lib/python3.5/bdb.py(431)run()
    430         try:
--> 431             exec(cmd, globals, locals)
    432         except BdbQuit:

  <string>(1)<module>()

  /home/ryefccd/env/server18/debug/start.py(11)<module>()
     10
---> 11 print("tmp1 is num: %s by bar.is_num func" % bar.is_num(tmp1))
     12 print("tmp2 is num: %s by bar.is_num func" % bar.is_num(tmp2))

> /home/ryefccd/env/server18/debug/bar.py(7)is_num()
      6         num = ord(i)
2---> 7         if not start <= num <= end:
      8             flag = False
```

4.1.3 cheatsheet

帮助

- Use **h**(elp) or ? to list all commands.

控制

- **n**(ext) -> Continue execution until the next line in the current function is reached or it returns.
- **s**(tep) -> Execute the current line, stop at the first possible occasion (either in a function that is called or on the next line in the current function).
- **r**(eturn) -> Continue execution until the current function returns.
- **u**(p) and **d**(own) -> Move the current frame count (default one) levels up/down in the stack trace (to an older/newer frame).
- **c**(ont(inue)) can be useful if you have multiple breakpoints, it continues execution until a next breakpoint is encountered.
- **unt**(il) [lineno] -> Without argument, continue execution until the line with a number greater than the current one is reached. -> useful to get out of a for loop.

- **b**(reak) [lineno] and **cl**(ear) to set / clear a break point in the current file (it even accepts a condition).

打印上下文

- **l**(ist) -> List source code for the current file. Without arguments, list 11 lines around the current line or continue the previous listing.
- **w**(here) -> Print a stack trace, with the most recent frame at the bottom. An arrow indicates the current frame, which determines the context of most commands. -> handy for web frameworks
- **bt** -> Get a stack trace of the functions that have been called so far.
- **pp** expression -> Like the p command, except the value of the expression is pretty-printed using the pprint module -> very useful for nested data structures.

4.1.4 相比 gdb 的缺陷

不能附加在一个已经开始运行的进程中.

4.1.5 参考资料

https://realpython.com/python-debugging-pdb/#displaying-expressions

4.2 gdb

gdb 是 gnu 自由软件维护的一个调试工具. 支持多种语言的调试.

4.2.1 使用场景

There are types of bugs that are difficult to debug from within Python:

- segfaults (not uncaught Python exceptions)
- hung processes (in cases where you can't get a Python traceback or debug with pdb)
- out of control daemon processes

4.2.2 安装

- ubuntu

```
sudo apt-get install gdb python3-dbg
```

- centos

```
sudo yum install gdb python-debuginfo
```

4.2.3 使用

实际情况中, 一般是在上面所说的异常时通过下面的命令进入进程.

```
gdb python3 -p  进程号
```

注意, 这个 python 一定是要上面装了调试信息的那个 Python 解释器.

首先开启进程

```
ryefccd@fccd:~/env/server18/debug$ python3.4 start.py
tmp1 is num: True by foo.is_num func
tmp2 is num: False by foo.is_num func
```

然后附加到进程进行调试

```
ryefccd@fccd:~/env/server18/debug$ ps -ef f|grep python3.4
ryefccd  18600 17607  0 17:02 pts/39   S+     0:00          |   |   \_ grep --
↪color=auto python3.4
ryefccd  18477 24591  0 17:01 pts/41   S+     0:00          |       \_ python3.4␣
↪start.py
ryefccd@fccd:~/env/server18/debug$ sudo gdb  python3.4 -p  18477
GNU gdb (Ubuntu 7.7.1-0ubuntu5~14.04.3) 7.7.1
Copyright (C) 2014 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from python3.4...Reading symbols from /usr/lib/debug//usr/bin/python3.
↪4m...done.
done.
Attaching to program: /usr/bin/python3.4, process 18477
Reading symbols from /lib/x86_64-linux-gnu/libpthread.so.0...Reading symbols from /
↪usr/lib/debug//lib/x86_64-linux-gnu/libpthread-2.19.so...done.
```

(下页继续)

(续上页)

```
done.
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Loaded symbols for /lib/x86_64-linux-gnu/libpthread.so.0
Reading symbols from /lib/x86_64-linux-gnu/libc.so.6...Reading symbols from /usr/lib/
↪debug//lib/x86_64-linux-gnu/libc-2.19.so...done.
done.
Loaded symbols for /lib/x86_64-linux-gnu/libc.so.6
Reading symbols from /lib/x86_64-linux-gnu/libdl.so.2...Reading symbols from /usr/lib/
↪debug//lib/x86_64-linux-gnu/libdl-2.19.so...done.
done.
Loaded symbols for /lib/x86_64-linux-gnu/libdl.so.2
Reading symbols from /lib/x86_64-linux-gnu/libutil.so.1...Reading symbols from /usr/
↪lib/debug//lib/x86_64-linux-gnu/libutil-2.19.so...done.
done.
Loaded symbols for /lib/x86_64-linux-gnu/libutil.so.1
Reading symbols from /lib/x86_64-linux-gnu/libexpat.so.1...(no debugging symbols␣
↪found)...done.
Loaded symbols for /lib/x86_64-linux-gnu/libexpat.so.1
Reading symbols from /lib/x86_64-linux-gnu/libz.so.1...(no debugging symbols found)...
↪done.
Loaded symbols for /lib/x86_64-linux-gnu/libz.so.1
Reading symbols from /lib/x86_64-linux-gnu/libm.so.6...Reading symbols from /usr/lib/
↪debug//lib/x86_64-linux-gnu/libm-2.19.so...done.
done.
Loaded symbols for /lib/x86_64-linux-gnu/libm.so.6
Reading symbols from /lib64/ld-linux-x86-64.so.2...Reading symbols from /usr/lib/
↪debug//lib/x86_64-linux-gnu/ld-2.19.so...done.
done.
Loaded symbols for /lib64/ld-linux-x86-64.so.2
0x00007f26034c28f3 in __select_nocancel () at ../sysdeps/unix/syscall-template.S:81
81        ../sysdeps/unix/syscall-template.S: 没有那个文件或目录.
(gdb) py-bt
Traceback (most recent call first):
  File "/home/ryefccd/env/server18/debug/bar.py", line 6, in is_num
    time.sleep(30)
  File "start.py", line 11, in <module>
    print("tmp1 is num: %s by bar.is_num func" % bar.is_num(tmp1))
(gdb) py-locals
x = '123'
start = 48
end = 57
flag = True
time = <module at remote 0x7f2603a5c548>
```

4.2.4 参考链接

- https://www.wzdftpd.net/blog/python-scripts-in-gdb.html
- https://sourceware.org/gdb/wiki/PythonGdbTutorial

CHAPTER 5

索引与高级查询

这部分内容由曹佳豪已经由佳豪分享. 之后会整理成 rst 文档, 便于后续的维护和更新.

CHAPTER 6

2018-05-25 gt-day 问题答案参考-pandas

[1]:
```
import pandas as pd
```

[4]:
```
import io

datas_csv = io.StringIO("""
学年, 学号, 科目, 分数
2011-2012,1,Chinese,97
2011-2012,2,Chinese,80
2011-2012,3,Chinese,85
2011-2012,4,Chinese,86
2011-2012,5,Chinese,91
2011-2012,6,Chinese,79
2011-2012,7,Chinese,91
2011-2012,1,English,70
2011-2012,2,English,80
2011-2012,3,English,94
2011-2012,4,English,72
2011-2012,5,English,94
2011-2012,6,English,96
2011-2012,7,English,77
2011-2012,1,math,72
2011-2012,2,math,90
2011-2012,3,math,89
2011-2012,5,math,72
```

(下页继续)

(续上页)

```
2011-2012,6,math,91
2011-2012,7,math,66
2011-2012,4,math,60
2012-2013,6,Chinese,95
2012-2013,1,Chinese,85
2012-2013,7,Chinese,83
2012-2013,3,Chinese,78
2012-2013,2,Chinese,76
2012-2013,5,Chinese,76
2012-2013,4,Chinese,72
2012-2013,5,English,95
2012-2013,2,English,91
2012-2013,6,English,90
2012-2013,4,English,86
2012-2013,3,English,78
2012-2013,1,English,72
2012-2013,7,English,72
2012-2013,5,math,94
2012-2013,7,math,94
2012-2013,3,math,92
2012-2013,4,math,89
2012-2013,1,math,88
2012-2013,2,math,76
2012-2013,6,math,70
""")
```

[5]:
```
df = pd.read_csv(datas_csv)
df
```

[5]:

	学年	学号	科目	分数
0	2011-2012	1	Chinese	97
1	2011-2012	2	Chinese	80
2	2011-2012	3	Chinese	85
3	2011-2012	4	Chinese	86
4	2011-2012	5	Chinese	91
5	2011-2012	6	Chinese	79
6	2011-2012	7	Chinese	91
7	2011-2012	1	English	70
8	2011-2012	2	English	80
9	2011-2012	3	English	94
10	2011-2012	4	English	72
11	2011-2012	5	English	94
12	2011-2012	6	English	96
13	2011-2012	7	English	77

(下页继续)

(续上页)

```
14  2011-2012   1     math  72
15  2011-2012   2     math  90
16  2011-2012   3     math  89
17  2011-2012   5     math  72
18  2011-2012   6     math  91
19  2011-2012   7     math  66
20  2011-2012   4     math  60
21  2012-2013   6  Chinese  95
22  2012-2013   1  Chinese  85
23  2012-2013   7  Chinese  83
24  2012-2013   3  Chinese  78
25  2012-2013   2  Chinese  76
26  2012-2013   5  Chinese  76
27  2012-2013   4  Chinese  72
28  2012-2013   5  English  95
29  2012-2013   2  English  91
30  2012-2013   6  English  90
31  2012-2013   4  English  86
32  2012-2013   3  English  78
33  2012-2013   1  English  72
34  2012-2013   7  English  72
35  2012-2013   5     math  94
36  2012-2013   7     math  94
37  2012-2013   3     math  92
38  2012-2013   4     math  89
39  2012-2013   1     math  88
40  2012-2013   2     math  76
41  2012-2013   6     math  70
```

[3]:
```
# 问题 1:   展示用户每一门课程的历史最高分数 (长格式展示)
df.groupby([' 学号', ' 科目'])[' 分数'].max()
```

[3]:
```
学号  科目
1   Chinese    97
    English    72
    math       88
2   Chinese    80
    English    91
    math       90
3   Chinese    85
    English    94
    math       92
4   Chinese    86
    English    86
```

(下页继续)

(续上页)

```
    math       89
5   Chinese    91
    English    95
    math       94
6   Chinese    95
    English    96
    math       91
7   Chinese    91
    English    77
    math       94
Name: 分数, dtype: int64
```

[4]:
```
# 问题 1:  展示用户每一门课程的历史最高分数 (宽格式 (透视) 展示)
pd.pivot_table(df, index=' 学号', columns=' 科目', values=' 分数', aggfunc=max)
```

[4]:
```
科目  Chinese  English  math
学号
1         97       72    88
2         80       91    90
3         85       94    92
4         86       86    89
5         91       95    94
6         95       96    91
7         91       77    94
```

[5]:
```
# 问题 2:  以学年, 学号, 科目排序
df.sort_values([' 学年', ' 学号', ' 科目'])
```

[5]:
```
            学年  学号        科目  分数
0   2011-2012   1  Chinese  97
7   2011-2012   1  English  70
14  2011-2012   1     math  72
1   2011-2012   2  Chinese  80
8   2011-2012   2  English  80
15  2011-2012   2     math  90
2   2011-2012   3  Chinese  85
9   2011-2012   3  English  94
16  2011-2012   3     math  89
3   2011-2012   4  Chinese  86
10  2011-2012   4  English  72
20  2011-2012   4     math  60
4   2011-2012   5  Chinese  91
11  2011-2012   5  English  94
17  2011-2012   5     math  72
```

(下页继续)

(续上页)

```
5   2011-2012   6  Chinese  79
12  2011-2012   6  English  96
18  2011-2012   6     math  91
6   2011-2012   7  Chinese  91
13  2011-2012   7  English  77
19  2011-2012   7     math  66
22  2012-2013   1  Chinese  85
33  2012-2013   1  English  72
39  2012-2013   1     math  88
25  2012-2013   2  Chinese  76
29  2012-2013   2  English  91
40  2012-2013   2     math  76
24  2012-2013   3  Chinese  78
32  2012-2013   3  English  78
37  2012-2013   3     math  92
27  2012-2013   4  Chinese  72
31  2012-2013   4  English  86
38  2012-2013   4     math  89
26  2012-2013   5  Chinese  76
28  2012-2013   5  English  95
35  2012-2013   5     math  94
21  2012-2013   6  Chinese  95
30  2012-2013   6  English  90
41  2012-2013   6     math  70
23  2012-2013   7  Chinese  83
34  2012-2013   7  English  72
36  2012-2013   7     math  94
```

[6]:
```
# 问题 3: 以学年,  科目, 学号排序
df.sort_values([' 学年', ' 科目', ' 学号'])
```

[6]:
```
           学年  学号        科目  分数
0   2011-2012   1  Chinese  97
1   2011-2012   2  Chinese  80
2   2011-2012   3  Chinese  85
3   2011-2012   4  Chinese  86
4   2011-2012   5  Chinese  91
5   2011-2012   6  Chinese  79
6   2011-2012   7  Chinese  91
7   2011-2012   1  English  70
8   2011-2012   2  English  80
9   2011-2012   3  English  94
10  2011-2012   4  English  72
11  2011-2012   5  English  94
```

(下页继续)

(续上页)

```
12  2011-2012   6  English  96
13  2011-2012   7  English  77
14  2011-2012   1     math  72
15  2011-2012   2     math  90
16  2011-2012   3     math  89
20  2011-2012   4     math  60
17  2011-2012   5     math  72
18  2011-2012   6     math  91
19  2011-2012   7     math  66
22  2012-2013   1  Chinese  85
25  2012-2013   2  Chinese  76
24  2012-2013   3  Chinese  78
27  2012-2013   4  Chinese  72
26  2012-2013   5  Chinese  76
21  2012-2013   6  Chinese  95
23  2012-2013   7  Chinese  83
33  2012-2013   1  English  72
29  2012-2013   2  English  91
32  2012-2013   3  English  78
31  2012-2013   4  English  86
28  2012-2013   5  English  95
30  2012-2013   6  English  90
34  2012-2013   7  English  72
39  2012-2013   1     math  88
40  2012-2013   2     math  76
37  2012-2013   3     math  92
38  2012-2013   4     math  89
35  2012-2013   5     math  94
41  2012-2013   6     math  70
36  2012-2013   7     math  94
```

[7]:
```
# 问题 4: 以学年,  科目, 分数排序
df.sort_values([' 学年', ' 科目', ' 分数'])
```

[7]:
```
           学年  学号       科目  分数
5   2011-2012   6  Chinese  79
1   2011-2012   2  Chinese  80
2   2011-2012   3  Chinese  85
3   2011-2012   4  Chinese  86
4   2011-2012   5  Chinese  91
6   2011-2012   7  Chinese  91
0   2011-2012   1  Chinese  97
7   2011-2012   1  English  70
10  2011-2012   4  English  72
```

(下页继续)

(续上页)

```
13  2011-2012   7  English  77
8   2011-2012   2  English  80
9   2011-2012   3  English  94
11  2011-2012   5  English  94
12  2011-2012   6  English  96
20  2011-2012   4     math  60
19  2011-2012   7     math  66
14  2011-2012   1     math  72
17  2011-2012   5     math  72
16  2011-2012   3     math  89
15  2011-2012   2     math  90
18  2011-2012   6     math  91
27  2012-2013   4  Chinese  72
25  2012-2013   2  Chinese  76
26  2012-2013   5  Chinese  76
24  2012-2013   3  Chinese  78
23  2012-2013   7  Chinese  83
22  2012-2013   1  Chinese  85
21  2012-2013   6  Chinese  95
33  2012-2013   1  English  72
34  2012-2013   7  English  72
32  2012-2013   3  English  78
31  2012-2013   4  English  86
30  2012-2013   6  English  90
29  2012-2013   2  English  91
28  2012-2013   5  English  95
41  2012-2013   6     math  70
40  2012-2013   2     math  76
39  2012-2013   1     math  88
38  2012-2013   4     math  89
37  2012-2013   3     math  92
35  2012-2013   5     math  94
36  2012-2013   7     math  94
```

[ ]:

CHAPTER 7

2018-05-25 gt-day 问题答案参考-Spark

[1]:
```
from pyspark.sql import SparkSession
from pyspark.sql import functions as F
```

[2]:
```
spark = SparkSession.builder.getOrCreate()
```

[3]:
```
# 读取源数据
df = spark.read.csv('/Users/cjh/Downloads/demo_data.csv', header=True)
df.show(n=100)
```

```
+---------+----+-------+----+
|     学年 | 学号 |   科目 | 分数 |
+---------+----+-------+----+
|2011-2012|   1|Chinese|  97|
|2011-2012|   2|Chinese|  80|
|2011-2012|   3|Chinese|  85|
|2011-2012|   4|Chinese|  86|
|2011-2012|   5|Chinese|  91|
|2011-2012|   6|Chinese|  79|
|2011-2012|   7|Chinese|  91|
|2011-2012|   1|English|  70|
|2011-2012|   2|English|  80|
|2011-2012|   3|English|  94|
|2011-2012|   4|English|  72|
|2011-2012|   5|English|  94|
```

(下页继续)

(续上页)

```
|2011-2012|   6|English|  96|
|2011-2012|   7|English|  77|
|2011-2012|   1|   math|  72|
|2011-2012|   2|   math|  90|
|2011-2012|   3|   math|  89|
|2011-2012|   5|   math|  72|
|2011-2012|   6|   math|  91|
|2011-2012|   7|   math|  66|
|2011-2012|   4|   math|  60|
|2012-2013|   6|Chinese|  95|
|2012-2013|   1|Chinese|  85|
|2012-2013|   7|Chinese|  83|
|2012-2013|   3|Chinese|  78|
|2012-2013|   2|Chinese|  76|
|2012-2013|   5|Chinese|  76|
|2012-2013|   4|Chinese|  72|
|2012-2013|   5|English|  95|
|2012-2013|   2|English|  91|
|2012-2013|   6|English|  90|
|2012-2013|   4|English|  86|
|2012-2013|   3|English|  78|
|2012-2013|   1|English|  72|
|2012-2013|   7|English|  72|
|2012-2013|   5|   math|  94|
|2012-2013|   7|   math|  94|
|2012-2013|   3|   math|  92|
|2012-2013|   4|   math|  89|
|2012-2013|   1|   math|  88|
|2012-2013|   2|   math|  76|
|2012-2013|   6|   math|  70|
+---------+----+-------+----+
```

[4]:
```
# spark Dataframe 的排序默认为升序

# 学生每一门课程的历史最高分数 (长格式)
df.groupBy(' 学号', ' 科目').agg(F.max(' 分数').alias(' 最高分')).sort(' 学号').
↪show(n=100)
```

```
+----+-------+------+
| 学号 |   科目 | 最高分 |
+----+-------+------+
|   1|English|    72|
|   1|   math|    88|
```

(下页继续)

(续上页)

```
|   1|Chinese|    97|
|   2|Chinese|    80|
|   2|English|    91|
|   2|   math|    90|
|   3|   math|    92|
|   3|Chinese|    85|
|   3|English|    94|
|   4|English|    86|
|   4|   math|    89|
|   4|Chinese|    86|
|   5|Chinese|    91|
|   5|   math|    94|
|   5|English|    95|
|   6|Chinese|    95|
|   6|English|    96|
|   6|   math|    91|
|   7|Chinese|    91|
|   7|English|    77|
|   7|   math|    94|
+----+-------+------+
```

[5]:
```
# 学生每一门课程的历史最高分数 (宽格式)
df.groupBy(' 学号').pivot(' 科目', values=['Chinese', 'English', 'math']).agg(F.max(' 分数')).sort(' 学号').show()
```

```
+----+-------+-------+----+
| 学号 |Chinese|English|math|
+----+-------+-------+----+
|   1|     97|     72|  88|
|   2|     80|     91|  90|
|   3|     85|     94|  92|
|   4|     86|     86|  89|
|   5|     91|     95|  94|
|   6|     95|     96|  91|
|   7|     91|     77|  94|
+----+-------+-------+----+
```

[6]:
```
# 以学年, 学号, 科目排序
df.sort(' 学年', ' 学号', ' 科目').show(n=100)
```

```
+---------+----+-------+----+
|     学年 | 学号 |   科目 | 分数 |
```

(下页继续)

(续上页)

```
+---------+----+-------+----+
|2011-2012|   1|Chinese|  97|
|2011-2012|   1|English|  70|
|2011-2012|   1|   math|  72|
|2011-2012|   2|Chinese|  80|
|2011-2012|   2|English|  80|
|2011-2012|   2|   math|  90|
|2011-2012|   3|Chinese|  85|
|2011-2012|   3|English|  94|
|2011-2012|   3|   math|  89|
|2011-2012|   4|Chinese|  86|
|2011-2012|   4|English|  72|
|2011-2012|   4|   math|  60|
|2011-2012|   5|Chinese|  91|
|2011-2012|   5|English|  94|
|2011-2012|   5|   math|  72|
|2011-2012|   6|Chinese|  79|
|2011-2012|   6|English|  96|
|2011-2012|   6|   math|  91|
|2011-2012|   7|Chinese|  91|
|2011-2012|   7|English|  77|
|2011-2012|   7|   math|  66|
|2012-2013|   1|Chinese|  85|
|2012-2013|   1|English|  72|
|2012-2013|   1|   math|  88|
|2012-2013|   2|Chinese|  76|
|2012-2013|   2|English|  91|
|2012-2013|   2|   math|  76|
|2012-2013|   3|Chinese|  78|
|2012-2013|   3|English|  78|
|2012-2013|   3|   math|  92|
|2012-2013|   4|Chinese|  72|
|2012-2013|   4|English|  86|
|2012-2013|   4|   math|  89|
|2012-2013|   5|Chinese|  76|
|2012-2013|   5|English|  95|
|2012-2013|   5|   math|  94|
|2012-2013|   6|Chinese|  95|
|2012-2013|   6|English|  90|
|2012-2013|   6|   math|  70|
|2012-2013|   7|Chinese|  83|
|2012-2013|   7|English|  72|
|2012-2013|   7|   math|  94|
```

(下页继续)

(续上页)

```
+---------+----+-------+----+
```

[7]:
```
# 以学年,  科目, 学号排序
df.sort(' 学年', ' 科目', ' 学号').show(n=100)
```

```
+---------+----+-------+----+
|     学年 | 学号 |   科目 | 分数 |
+---------+----+-------+----+
|2011-2012|   1|Chinese|  97|
|2011-2012|   2|Chinese|  80|
|2011-2012|   3|Chinese|  85|
|2011-2012|   4|Chinese|  86|
|2011-2012|   5|Chinese|  91|
|2011-2012|   6|Chinese|  79|
|2011-2012|   7|Chinese|  91|
|2011-2012|   1|English|  70|
|2011-2012|   2|English|  80|
|2011-2012|   3|English|  94|
|2011-2012|   4|English|  72|
|2011-2012|   5|English|  94|
|2011-2012|   6|English|  96|
|2011-2012|   7|English|  77|
|2011-2012|   1|   math|  72|
|2011-2012|   2|   math|  90|
|2011-2012|   3|   math|  89|
|2011-2012|   4|   math|  60|
|2011-2012|   5|   math|  72|
|2011-2012|   6|   math|  91|
|2011-2012|   7|   math|  66|
|2012-2013|   1|Chinese|  85|
|2012-2013|   2|Chinese|  76|
|2012-2013|   3|Chinese|  78|
|2012-2013|   4|Chinese|  72|
|2012-2013|   5|Chinese|  76|
|2012-2013|   6|Chinese|  95|
|2012-2013|   7|Chinese|  83|
|2012-2013|   1|English|  72|
|2012-2013|   2|English|  91|
|2012-2013|   3|English|  78|
|2012-2013|   4|English|  86|
|2012-2013|   5|English|  95|
|2012-2013|   6|English|  90|
|2012-2013|   7|English|  72|
```

(下页继续)

(续上页)

```
|2012-2013|   1|   math|  88|
|2012-2013|   2|   math|  76|
|2012-2013|   3|   math|  92|
|2012-2013|   4|   math|  89|
|2012-2013|   5|   math|  94|
|2012-2013|   6|   math|  70|
|2012-2013|   7|   math|  94|
+---------+----+-------+----+
```

[28]:
```
# 以学年,  科目, 分数
df.sort(' 学年', ' 科目', ' 分数').show(n=100)
```

```
+---------+----+-------+----+
|     学年 | 学号 |   科目 | 分数 |
+---------+----+-------+----+
|2011-2012|   6|Chinese|  79|
|2011-2012|   2|Chinese|  80|
|2011-2012|   3|Chinese|  85|
|2011-2012|   4|Chinese|  86|
|2011-2012|   5|Chinese|  91|
|2011-2012|   7|Chinese|  91|
|2011-2012|   1|Chinese|  97|
|2011-2012|   1|English|  70|
|2011-2012|   4|English|  72|
|2011-2012|   7|English|  77|
|2011-2012|   2|English|  80|
|2011-2012|   3|English|  94|
|2011-2012|   5|English|  94|
|2011-2012|   6|English|  96|
|2011-2012|   4|   math|  60|
|2011-2012|   7|   math|  66|
|2011-2012|   5|   math|  72|
|2011-2012|   1|   math|  72|
|2011-2012|   3|   math|  89|
|2011-2012|   2|   math|  90|
|2011-2012|   6|   math|  91|
|2012-2013|   4|Chinese|  72|
|2012-2013|   2|Chinese|  76|
|2012-2013|   5|Chinese|  76|
|2012-2013|   3|Chinese|  78|
|2012-2013|   7|Chinese|  83|
|2012-2013|   1|Chinese|  85|
|2012-2013|   6|Chinese|  95|
```

(下页继续)

(续上页)

```
|2012-2013|   1|English|  72|
|2012-2013|   7|English|  72|
|2012-2013|   3|English|  78|
|2012-2013|   4|English|  86|
|2012-2013|   6|English|  90|
|2012-2013|   2|English|  91|
|2012-2013|   5|English|  95|
|2012-2013|   6|   math|  70|
|2012-2013|   2|   math|  76|
|2012-2013|   1|   math|  88|
|2012-2013|   4|   math|  89|
|2012-2013|   3|   math|  92|
|2012-2013|   5|   math|  94|
|2012-2013|   7|   math|  94|
+---------+----+-------+----+
```

[ ]:

CHAPTER 8

Indices and tables

- genindex
- modindex
- search