1.pyCharm切换Python版本 : 在project Interpreter里面

IDLE的mac版快捷翻出上/下条命令 control + P(Previous) / control + N(Next)

2.py的数据类型

整型(不带0), 浮点(带0), e计法(1.5e2表示1.5的2次方), 布尔bool(True:1; False:0)

内置函数:

int('520') ==520 : str转换为整型

float(520) ==520.0 str或者整型转成浮点

str(520) =="520" 任何类型转成str

type() / isinstance() : 获取类型名

help(abs) //查看abs函数的帮助信息

3.运算符(+, -, *, %,

** : 幂运算3**5表示3的5次方

/ : 4/2 == 2.0

// : 4/4 == 2

三元运算符 small = x if x<y else y

4. 比较操作符

<, >, ==, !=, <=, >=

5. 逻辑操作符

or and not 返回True/False

6.assert 断言 assert 1>2 抛出异常

7. while循环/for循环

for 目标 in 表达式

循环体

range()

exp:

for i in range(0, 9) :

print(i)

8. 数组(可以加入任何数据类型)

array = [1, 1.2, "字符串", True]

array.append('fuck')

len(array)

array.extend(['me', 'he'])

array.insert(1, 'you')

array[0]
array.remove('he')
del array[1]
array.pop()
array[1:3] 数组分片,取出array[1], 出array[2]
array.reverse() 数组翻转
array.sort() 数组排序

9. 元组(不可改变的数组)
tuple1 = (1, 2, 3, 4, 5, 6)
tuple1 = (1,) //一个元素的元组

10. 字符串
str0.capitalize() //将首字母改成大写
str0.casefold() //所有字母变成小写
str0.count(substr) //substr在字符串出现的次数
str0.endwith()
'{0} 苦短, 我用{1}'.format('rensheng', 'python') //位置参数
'{a} 苦短, 我用{b}'.format(a='rensheng', b='python') //参数名赋值
字符串占位符写法 : print('my name is %s' %self.name)
11. list
list0 = list('a', 'b', 'c', 'd', 'e')
len(list0)
max(list0)
min(list0)
list1 = sorted(list0, lambda x : x['name'])
//sorted排序后原list不变需要返回, sort直接对原list排序

函数，对象，模块

def 定义函数
内部函数 闭包
lambda表达式(冒号前面是参数, 后面是要返回的表达式)
g = lambda x : 2 * x +1 //调用g(2)即可
g = lambda x, y : 2 * x + 3 * y //调用g(1, 2)即可

filter()过滤器 : 过滤掉非True的内容(0, False)
语法 : filter(function/None, iterable)
list(filter(lambda x : x%2, range(10))) //过滤掉0-9之间的偶数, 并将结果转成list
map()
语法 : map(function/None, iterable)
list(map(lambda x : x*2, range(10))) //将0-9之间的数按lambda/function加工后, 输出结果

递归(自己调用自己, Python默认调用最大100层) :
    汉诺塔, 树结构定义, 谢尔宾斯基三角形, 斐波那契数列

```
def fib(n) :
    if n==1 or n==2 :
        return 1
    else :
        return fib(n-2) + fib(n-1)
# n : 盘子数量; x:盘子的初始位置; y:需要借助的中间轴 z:盘子最后要到达的位置
def hanoi(n, x, y, z) :
    if n==1 :
        print(x, '->', z)
    else :
        hanoi(n-1, x, z, y)  #将x轴上的前n-1个盘子移动到y轴上
        print(x, '->', z) #将最底下的盘子移动到z轴上
        hanoi(n-1, y, x, z)  #将y轴上的盘子全部移动到z轴上
```

字典 (映射类型): 用大括号
    创建 : dict1 = {'lining':'一切皆有可能', 'nike':'just do it', 'adidas':'impossible is nothing'}
        dict2 = dict((('lining', '一切皆有可能'), ('nike', 'just do it'))) //dict()括号内只能有1个参数,
        dict3 = dict(lining='一切皆有可能', nike = 'just do it') //key不能加引号, 否则报错
        dict4 = dict.fromkeys((range(10)), 'number') //生成0-9共10个不同的key, value是相同的'number'
    访问 : dict1['nike']
        dict1.get(1)
        //打印key

```
        for key in dict4.keys() :
         print(key)
        //打印value
        for val in dict4.values() :
         print(val)
        //打印键值对
        for item in dict4.items() :
         print(item)
    清空字典  dict2.clear()
    浅拷贝 dict5 = dict4.copy()
    dict4.popitem()  //因为字典无顺序, 所以随机弹出
    dict4.pop(2)  //弹出key=2的

集合 : set1 = {1, 2, 3, 4, 5}  //元素不可重复, 且无序
       set2 = set((1, 2, 4))
       判断元素是否在set中: 1.for循环 2.元素in/not in set
      不可变集合 frozenset


文件  文件分两种格式二进制模式和文本模式

fo = open('/Users/chang/pytestio.txt')  //默认以读的方式打开
str = fo.read(10);  //读取10个字节, 不加数字则读取文件全部内容
str1 = fo.readline(10);  //读取一行内容
str2 = fo.readlines();  //读取全部内容, 并返回list
position = fo.tell();  //当前读取到的位置
position = fo.seek(0, 0);  //读取指针指向开头位置
fo.close()  //关闭文件

//以只读方式打开文件pytestio.txt, 如果文件不存在则报错
with open('/Users/chang/pytestio.txt', 'r') as fo :
    print(fo.read())

//以只写方式打开文件pytestio.txt, 如果文件不存在则创建文件, 存在则覆盖原文件
with open('/Users/chang/pytestio.txt', 'w') as f1 :
    f1.write('\n 6. who are you')

//以只写方式打开文件pytestio.txt, 并在文件末尾追加内容
with open('/Users/chang/pytestio.txt', 'a') as f2 :
```

```
f2.write('\n 6. who are you')
```

模块 即是文件xxx.py(相当于Integer.java/String.java)

每一个包目录下面都会有一个__init__.py的文件，这个文件是必须存在的，否则，Python就把这个目录当成普通目录，而不是一个包。__init__.py可以是空文件，也可以有Python代码，因为__init__.py本身就是一个模块

os模块的函数

```
getcwd()  //当前工作目录
chdir()   //改变当前工作目录
listdir()  //列出目录下所有文件
mkdir()  //创建单层目录
remove() //删除文件
rmdir() //删除空目录
rename()  //文件重命名
system()  //运行系统的shell(windows的cmd)
```

os.path模块

```
os.path.basename()  //返回文件名(去掉路径)
os.path.dirname()  //返回路径(去掉文件名)
        join()  //组合成一个路径
        split()  //分割文件名和路径
        splitext()  //分割文件路径文件和后缀
        getatime()/getctime()/getmtime()   //最近访问时间戳/创建时间戳/修改时间戳
        gettime()/getlocaltime() //将时间戳转成时间
        exists()  //判断路径或文件是否存在
        isabs()  //是否是绝对路径
        isdir()  //是否是目录
        isfile()  //是否是文件
```

pickle模块

```
city = open('city_data.pkl', 'wb')  //二进制形式新建文件
city_code_dict = {'beijing' : '010'}
pickle.dump(city_code_dict, 'city_data.pkl')  //将list,字典的数据写入city_data.pkl
pickle_file = open('city_data.pkl', 'rb')   //读模式打开文件
city = pickle.load(pickle_file)  //加载文件到city
```

```
1. #!/usr/bin/env python
```

```
2. # -*- coding: utf-8 -*-
3.
4. ' a test module '
5.
6. __author__ = 'Michael Liao'
7.
8. import sys
9. def test():
10. args = sys.argv
11. if len(args)==1:
12.     print 'Hello, world!'
13. elif len(args)==2:
14.     print 'Hello, %s!' % args[1]
15. else:
16.     print 'Too many arguments!'
17.
18. if __name__=='__main__':
19.     test()
```

第1行注释可以让这个hello.py文件直接在Unix/Linux/Mac上运行，
第2行注释表示.py文件本身使用标准UTF-8编码；
第4行是一个字符串，表示模块的文档注释，任何模块代码的第一个字符串都被视为模块的文档注释；
第6行使用__author__变量把作者写进去，这样当你公开源代码后别人就可以瞻仰你的大名；
以上就是Python模块的标准文件模板，当然也可以全部删掉不写，但是，按标准办事肯定没错。
后面开始就是真正的代码部分。
你可能注意到了，使用sys模块的第一步，就是导入该模块：
Python解释器把一个特殊变量__name__置为__main__，而如果在其他地方导入该hello模块时，if判断将失败，因此，这种if测试可以让一个模块通过命令行运行时执行一些额外的代码，最常见的就是运行测试。

python安装第三方库(两种方法)

1. pip install PIL
2. python3 MySQL-python install //系统有多个Python版本，指定Python3安装 //Python Imaging Library)

//MySQL的驱动:MySQL-python
//用于科学计算的NumPy库：numpy
//用于生成文本的模板工具:Jinja2

模块搜索路径

导入第三方模块并重命名(import sys as s)
默认情况下，Python解释器会搜索当前目录、所有已安装的内置模块和第三方模块，搜索路径存放在sys模块的path变量中，如果我们要添加自己的搜索目录，有两种方法：
一是直接修改sys.path(运行时修改，运行结束后失效)，添加要搜索的目录：

```
import sys;
sys.path.append('/Users/chang/my_py_scripts')
```

第二种方法是设置环境变量PYTHONPATH，该环境变量的内容会被自动添加到模块搜索路径中。设置方式与设置Path环境变量类似。注意只需要添加你自己的搜索路径，Python自己本身的搜索路径不受影响。

类

```
class Student(object):
    def __init__(self, name, score):
        self.name = name
        self.score = score
```

__init__方法(构造方法)的第一个参数永远是self，表示创建的实例本身，因此，在__init__方法内部，就可以把各种属性绑定到self，因为self就指向创建的实例本身。
有了__init__方法，在创建实例的时候，就不能传入空的参数了，必须传入与__init__方法匹配的参数，但self不需要传，Python解释器自己会把实例变量传进去，和普通的函数相比，在类中定义的函数第一个参数永远是实例变量self，并且调用时，不用传递该参数。

如果要让内部属性不被外部访问，可以把属性的名称前加上两个下划线__，在Python中，实例的变量名如果以__开头，就变成了一个私有变量（private）
在Python中，变量名类似__xxx__的，也就是以双下划线开头，并且以双下划线结尾的，是特殊变量，特殊变量是可以直接访问的，不是private变量，所以，不能用__name__、__score__这样的变量名。

图形化界面 esaygui

```
继承 class son(father)  //son类继承father
    Son类继承Father类并重写__init__方法时，
```

需要手动调用Father.__init__(self)或使用super().__init__()

公用 :
私有 : 变量名/函数名前加两个下划线即可{__name}
访问私有变量的方法 p.Person__name

issubclass(sonClassInstance, fatherClassInstance) //检查是否是子类
isinstance(sonClassInstance, fatherClassInstance)

hasattr(classInstance, attrname) //测试是否有指定属性
getattr(classInstance, attrname, 错误提示)
setattr(classInstance, attrname, attrval)

property(fget=None, fset=None, fdel=None, doc=None)

python魔法方法
__new__() : Python程序第一个被调用的方法
__init__() : 一定返回None
__del__() : 对象销毁时这个方法自动被调用(没有引用指向对象时会自动调用垃圾回收)
__add__() :加
__sub__() :减
__str__() : 相当于Java的toString方法, print(Student('Michael'))时调用
__repr__() : 相当于Java的toString方法, Student('Michael')时调用,
一个只重写__str__(), 然后__repr__() = __str__()
__iter__() : 如果一个类想被用于for ... in循环，类似list或tuple那样，
就必须实现一个__iter__()方法，该方法返回一个迭代对象，
然后，Python的for循环就会不断调用该迭代对象的__next__()方法拿到循环的
下一个值，直到遇到StopIteration错误时退出循环。
例如：

```
class Fib(object):
    def __init__(self):
        self.a, self.b = 0, 1 # 初始化两个计数器a，b
    def __iter__(self):
        return self # 实例本身就是迭代对象，故返回自己
    def __next__(self):
        self.a, self.b = self.b, self.a + self.b # 计算下一个值
        if self.a > 100000: # 退出循环的条件
            raise StopIteration();
        return self.a # 返回下一个值
```

__getattr__() : 当调用不存在的属性变量时，Python解释器会
试图调用__getattr__(self, attrName)来尝试获得属性

```python
class Student(object):
    @property
    def birth(self):   # birth读
        return self._birth

    @birth.setter  # birth写
    def birth(self, value):
        self._birth = value

    @property
    def age(self): # age读
        return 2014 - self._birth

s.birth = 18 # OK，实际转化为s.set_birth(18)
s.birth  # OK，实际转化为s.get_birth()
```

python内置模块

import datetime datetime.datetime.now() //两个datetime

collections

- namedtuple：二维tuple
- deque：实现插入和删除操作的双向列表
- defaultdict：key不存在时，返回一个默认值
- OrderedDict：Key排序后的dict
- Counter 是一个简单的计数器

base64

hashlib

itertools

- chain()：可以把一组迭代对象串联起来，形成一个更大的迭代器
- groupby()：把迭代器中相邻的重复元素挑出来放在一起

解析html的模块

1. HTMLParser
2. SGMLParser
3. Beautiful Soup

urllib是Python3.x标准库的一部分,包含urllib.request, urllib.error, urllib.parse, urlli.robotparser四个子模块

1. urllib.request == python2.x的urllib2
2. urllib.parse == python2.x的urlparse

urljoin、

```
>>> urlparse.urljoin('http://www.oschina.com/tieba','index.php')
>>> 'http://www.oschina.com/index.php'
>>> urlparse.urljoin('http://www.oschina.com/tieba/','index.php')
>>> 'http://www.oschina.com/tieba/index.php'
```

urlsplit(返回一个包含5个字符串项目的元组：协议、位置、路径、查询、片段。

allow_fragments为False时，该元组的组后一个项目总是空，

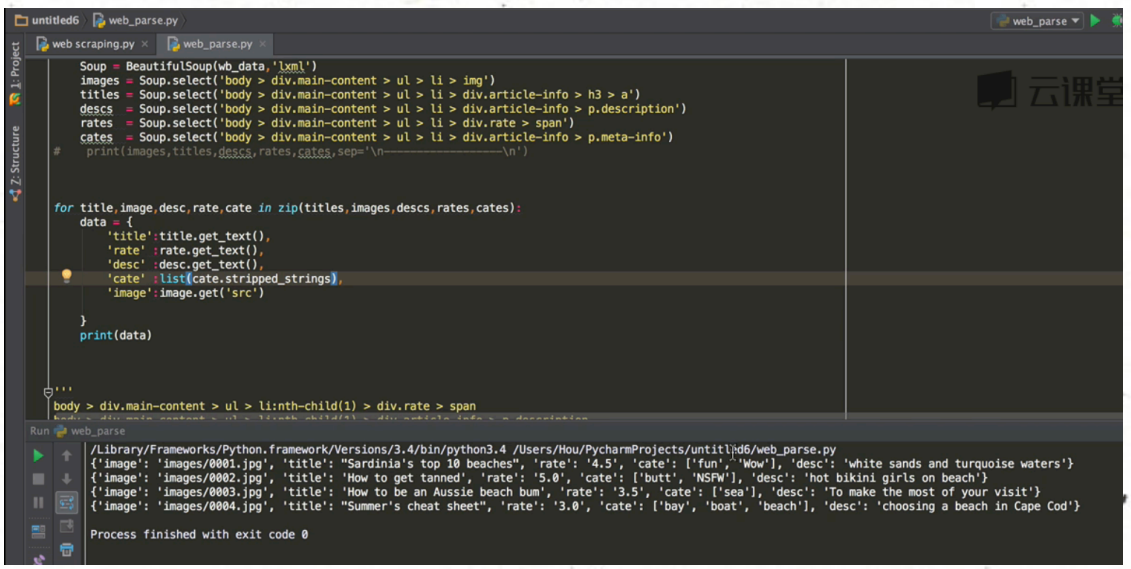
不管urlstring有没有片段，省略项目的也是空。urlsplit()和urlparse()差不多。

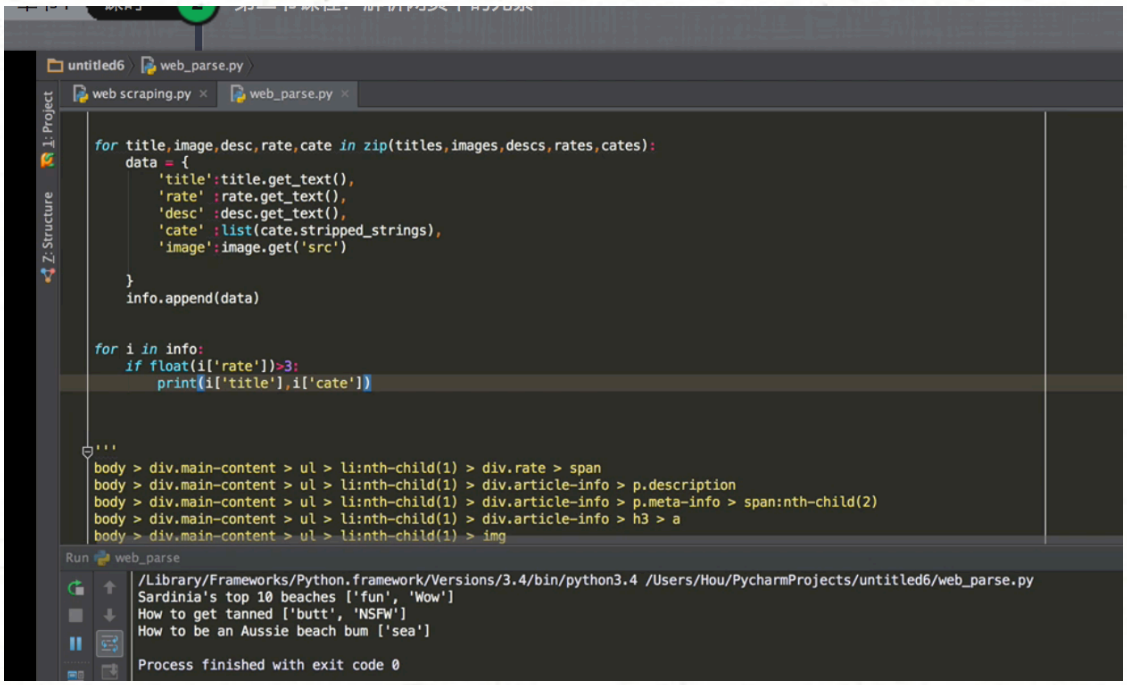
不过它不切分URL的参数。)

```
>>> url=urlparse.urlparse('http://www.baidu.com/index.php?username=guol')
>>> print url
ParseResult(scheme='http', netloc='www.baidu.com', path='/index.php',
params='', query='username=guol', fragment='')
>>> url=urlparse.urlsplit('http://www.baidu.com/index.php?username=guol')
>>> print url
SplitResult(scheme='http', netloc='www.baidu.com', path='/index.php',
query='username=guol', fragment='')
```

urlparse

```
>>> url=urlparse.urlparse('http://www.baidu.com/index.php?username=guol')
>>> print(url)
ParseResult(scheme='http', netloc='www.baidu.com', path='/index.php',
params='', query='username=guol', fragment='')
>>> print(url.netloc)
>>> www.baidu.com
```


untitled6
web_parse.py
web scraping.py
web_parse.py
Soup = BeautifulSoup(wb_data,'lxml')
images = Soup.select('body > div.main-content > ul > li > img')
titles = Soup.select('body > div.main-content > ul > li > div.article-info > h3 > a')
descs = Soup.select('body > div.main-content > ul > li > div.article-info > p.description')
rates = Soup.select('body > div.main-content > ul > li > div.rate > span')
cates = Soup.select('body > div.main-content > ul > li > div.article-info > p.meta-info')
# print(images,titles,descs,rates,cates,sep='\n----------\n')
for title,image,desc,rate,cate in zip(titles,images,descs,rates,cates):
data = {
'title':title.get_text(),
'rate' :rate.get_text(),
'desc' :desc.get_text(),
'cate' :list(cate.stripped_strings),
'image':image.get('src')
}
print(data)
'''
body > div.main-content > ul > li:nth-child(1) > div.rate > span
Run web_parse
/Library/Frameworks/Python.framework/Versions/3.4/bin/python3.4 /Users/Hou/PycharmProjects/untitled6/web_parse.py
{'image': 'images/0001.jpg', 'title': "Sardinia's top 10 beaches", 'rate': '4.5', 'cate': ['fun', 'Wow'], 'desc': 'white sands and turquoise waters'}
{'image': 'images/0002.jpg', 'title': 'How to get tanned', 'rate': '5.0', 'cate': ['butt', 'NSFW'], 'desc': 'hot bikini girls on beach'}
{'image': 'images/0003.jpg', 'title': 'How to be an Aussie beach bum', 'rate': '3.5', 'cate': ['sea'], 'desc': 'To make the most of your visit'}
{'image': 'images/0004.jpg', 'title': "Summer's cheat sheet", 'rate': '3.0', 'cate': ['bay', 'boat', 'beach'], 'desc': 'choosing a beach in Cape Cod'}
Process finished with exit code 0


untitled6
web_parse.py
web scraping.py
web_parse.py
for title,image,desc,rate,cate in zip(titles,images,descs,rates,cates):
data = {
'title':title.get_text(),
'rate' :rate.get_text(),
'desc' :desc.get_text(),
'cate' :list(cate.stripped_strings),
'image':image.get('src')
}
info.append(data)
for i in info:
if float(i['rate'])>3:
print(i['title'],i['cate'])
'''
body > div.main-content > ul > li:nth-child(1) > div.rate > span
body > div.main-content > ul > li:nth-child(1) > div.article-info > p.description
body > div.main-content > ul > li:nth-child(1) > div.article-info > p.meta-info > span:nth-child(2)
body > div.main-content > ul > li:nth-child(1) > div.article-info > h3 > a
body > div.main-content > ul > li:nth-child(1) > img
Run web_parse
/Library/Frameworks/Python.framework/Versions/3.4/bin/python3.4 /Users/Hou/PycharmProjects/untitled6/web_parse.py
Sardinia's top 10 beaches ['fun', 'Wow']
How to get tanned ['butt', 'NSFW']
How to be an Aussie beach bum ['sea']
Process finished with exit code 0