python开发培训

by 互二阅读-童浩

Jul 2019	Jul 2018	Change	Programming Language	Ratings	Change
1	1		Java	15.058%	-1.08%
2	2		C	14.211%	-0.45%
3	4	▲	Python	9.260%	+2.90%
4	3	▼	C++	6.705%	-0.91%
5	6	▲	C#	4.365%	+0.57%
6	5	▼	Visual Basic .NET	4.208%	-0.04%

Aug 2019	Aug 2018	Change	Programming Language	Ratings	Change
1	1		Java	16.028%	-0.85%
2	2		C	15.154%	+0.19%
3	4	▲	Python	10.020%	+3.03%
4	3	▼	C++	6.057%	-1.41%
5	6	▲	C#	3.842%	+0.30%

目录 / Contents

01 python基础

基本数据结构

字符串

模块

1 基本数据结构

python中的容器主要包括序列和映射和集合set

元组

tuple，元组是不可变序列

列表

list，列表是一种可变序列

字典

python中唯一的内置映射类型，由键值对组成

集合

Set中的元素是不可重复的

基本数据结构-序列

```
rst_tuple = (1, 2)
print rst_tuple == (1, 2)

rst_list = list()
rst_list.append("1")
rst_list.append("2")
print rst_list == ["1", "2"]
```

python内置了多种序列，包括主要的容器类型：列表，元组。

列表和元组的主要不同在于，列表是可以修改的，而元组不可以。

序列通用操作

索引

通过下标访问序列元素，正向从0开始，反向从-1开始

切片

通过下标访问特定范围的元素,中间用:隔开. 如: list[1:2]

相加

通过+拼接相同类型的序列.如[1, 2] + [3, 4]

相乘

通过*将序列与数字n相乘，表示将序列重复n次

成员资格

通过in来检查序列中是否存在某个元素.如:“x”in“xy”

其他：len，max，min

列表基本操作和方法

修改元素：

```
x = [1, 2, 3]
x[0] = 0
print x
```

删除元素：

```
x = [1, 2, 3]
del x[0]
print x
```

切片赋值：

```
x = [1, 2, 3]
x[1:] = [2, 4, 6, 8]
print x
```

```
x = [1, 2, 3]
x[1:1] = [4, 6, 8]
print x
```

```
x = [1, 2, 3]
x[1:] = []
print x
```

列表方法：append(x)，clear()，copy()，count(x)，extend(list)，index(x)，insert(x)，pop(x)，remove(x)，revert()，sort().

元组

元组是不可变的序列。

元组通常用作映射中的键或者集合set的成员。

只包含一个值的元组后面必须带上逗号，否则就不是元组。如(1,)和(1)

元组的创建与其元素的访问与其他序列相同。

基本数据结构-字典

字典由键及其相应的值组成，这种键-值对称为项（item）。

如：phonebook = {'Alice': '2341', 'Beth': '9102', 'Cecil': '3258'}

字典中的键必须是唯一的。

字典方法：clear()，copy()，fromkeys(list)，get(key)，items()，keys()，pop(x)，popitem()，setdefault(key, value)，update(list), values().

基本数据结构-集合

集合set中的元素不可重复。

```
a = {0, 1, 2, 3, 0, 1, 2, 3, 4, 5}
print a == {0, 1, 2, 3, 4, 5}
```

与字典一样，集合中的元素排列顺序是不确定的。

集合中的元素不可重复。

集合是可变的，而集合中的元素是不可变的，因此集合中不能包含另一个集合。

如果确实需要将一个集合作为另一个集合中的元素，可以使用frozenset，这是一个不可变集合。

字符串

字符串也是一种序列

所有序列通用操作对字符串同样适用。

除此之外，字符串还有很多方法。

.

字符串

center()

方法center通过在两边添加填充字符（默认为空格）让字符串居中。

find()

方法find在字符串中查找子串。如果找到，就返回子串的第一个字符的索引，否则返回-1。

join()

join是一个非常重要的字符串方法，其作用与split相反，用于合并序列的元素。

replace()

方法replace将指定子串都替换为另一个字符串，并返回替换后的结果。

strip()

方法strip将字符串开头和末尾的空白（但不包括中间的空白）删除，并返回删除后的结果。

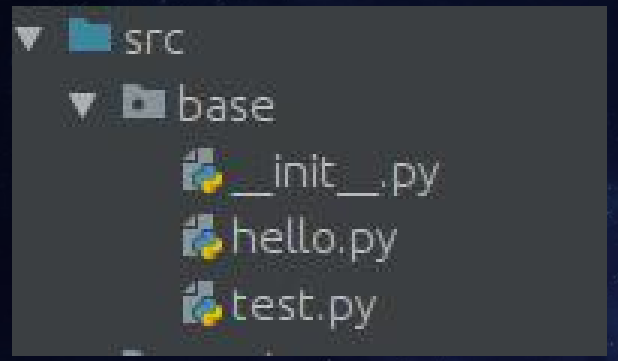
translate()

方法translate与replace一样替换字符串的特定部分，但它能同时替换多个字符。

模块

模块定义

任何Python程序都可作为模块导入。假设你编写了一段代码，并将其保存在文件hello.py中，这个文件的名称（不包括扩展名.py）将成为模块的名称。



包

为组织模块，可将其编组为包（package）。包其实就是另一种模块，但有趣的是它们可包含其他模块。模块存储在扩展名为.py的文件中，而包则是一个目录。要被Python视为包，目录必须包含文件__init__.py。

02

基本标准库

Remember what should be remembered, and forget what should be forgotten.

2 基本标准库

基本标准库

sys让你能够访问与Python解释器紧密相关的变量和函数.

sys

变量sys.argv包含传递给Python解释器的参数，其中包括脚本名。 例如：$python test.py this is a test ，则sys.argv是一个长度为5的列表。

映射sys.modules将模块名映射到模块（仅限于当前已导入的模块）。它是一个全局字典，该字典在python启动后就加载到内存中。每当导入新的模块时，sys.modules都将记录这些模块。

变量sys.path是一个字符串列表，其中的每个字符串都是一个目录名，执行import语句时将在这些目录中查找模块。

变量sys.platform（一个字符串）是运行解释器的“平台”名称。这可能是表示操作系统的名称（如sunos5或win32），也可能是表示其他平台类型（如Java虚拟机）的名称（如java1.4.0）——如果你运行的是Jython。

变量sys.stdin、sys.stdout和sys.stderr是类似于文件的流对象，表示标准的UNIX概念：标准输入、标准输出和标准错误。简单地说，Python从sys.stdin获取输入（例如，用于input中），并将输出打印到sys.stdout。

基本标准库

os让你能够访问多个操作系统服务.

os

os.environ包含环境变量的映射。这个映射也可用于修改环境变量，但并非所有的平台都支持这样做。

os.system用于运行外部程序。例如：os.system(“mkdir ~/a”)，成功执行则返回0。

os.sep表示路径名中的分隔符，如“/”。

os.linesep表示行分隔符。如”\n”,”\r\n”等。

os.pathsep表示分割不同路径的分隔符，如“:”或“;”。

基本标准库

time

time包含用于获取当前时间、操作时间和日期、从字符串中读取日期、将日期格式化为字符串的函数。

time.time返回当前的国际标准时间，以从新纪元开始的秒数表示。

time.localtime将一个实数（从新纪元开始后的秒数）转换为日期元组（本地时间）。如果要转换为国际标准时间，应使用gmtime。

time.strftime可以将日期元组按照转换成制定格式的字符串。

time.mktime将日期元组转换为从新纪元后的秒数，这与localtime的功能相反。

time.sleep让解释器等待指定的秒数。

还有两个较新的与时间相关的模块：datetime和timeit。前者提供了日期和时间算术支持，而后者可帮助你计算代码段的执行时间。

基本标准库

json用于处理json字符串，可以将pathon值与json字符串相互转换。

json

json.dumps将python对象编码成json字符串。

json.loads将已编码的json字符串解码成python对象。

基本标准库

datetime支持特殊的日期和时间对象，并让你能够以各种方式创建和合并这些对象。相比于模块time，模块datetime的接口在很多方面都更加直观。

枚举类型是一种只有少数几个可能取值的类型。如果你在使用Python时需要这样的类型，要用到这个模块。

logging模块用于打印日志。

单该模块为数学数据提供了计算数学统计量的函数。

03 魔法方法、迭代器

3 魔法方法

什么叫魔法方法

特殊方法

在Python中，有些方法名称很特别，开头和结尾都是两个下划线。如__init__。这样的拼写表示该方法有特殊意义，正常情况下不要主动在程序中创建这样的方法。

如果对象实现了这些方法，它们将在特定情况下（具体是哪种情况取决于方法的名称）被Python调用，而几乎不需要直接调用。

魔法方法

__init__

构造器，当一个实例被创建的时候调用的初始化方法。

__del__

析构器，当一个实例被销毁的时候调用的方法。

__getitem__

定义获取容器中指定元素的行为。

__setitem__

定义设置容器中指定元素的行为。

__delitem__

定义删除容器中指定元素的行为。

__len__

定义当被 len() 调用时的行为。

魔法方法

__getattribute__

当属性被访问时自动调用。

__getattr__

当属性被访问而对象没有这样的属性时自动被调用。

__setattr__

给属性赋值时自动调用。

__delattr__

删除属性时自动调用。

迭代器

实现了__iter__方法的对象是可迭代的，实现了__next__（或next）的对象是迭代器

迭代：当我们使用一个循环来遍历某个对象时，这个过程就叫迭代。

```
class Fibs:
    def __init__(self):
        self.a = 0
        self.b = 1

    def next(self):
        self.a, self.b = self.b, self.a + self.b
        return self.a

    def __iter__(self):
        return self


if __name__ == '__main__':
    for x in Fibs():
        print x
```

迭代器

生成器是一种使用普通函数语法定义的迭代器

包含yield语句的函数都被称为生成器。生成器的行为与普通函数截然不同。差别在于，生成器不是使用return返回一个值，而是可以生成多个值，每次一个。每次使用yield生成一个值后，函数都将冻结，即在此停止执行，等待被重新唤醒。被重新唤醒后，函数将从停止的地方开始继续执行。

```
class Fibs:
    def __init__(self):
        self.a = 1
        self.b = 1

    def generate(self, num):
        while self.a < num:
            yield self.a
            self.a, self.b = self.b, self.a + self.b


if __name__ == '__main__':
    for x in Fibs().generate(100):
        print x
```

迭代器

生成器被调用时不会执行函数体内的代码，而是返回一个迭代器。每次请求值时，都将执行生成器的代码，直到遇到yield或return。yield意味着应生成一个值，而return意味着生成器应停止执行。

换而言之，生成器由两个单独的部分组成：生成器的函数和生成器的迭代器。生成器的函数是由def语句定义的，其中包含yield。生成器的迭代器是这个函数返回的结果。用不太准确的话说，这两个实体通常被视为一个，通称为生成器。

04 常用组件

4 常用组件

sqlalchemy

sqlAlchemy是python中最著名的ORM(Object Relationship Mapping)框架。

pymongo

pymongo用于在python中连接mongodb数据库。

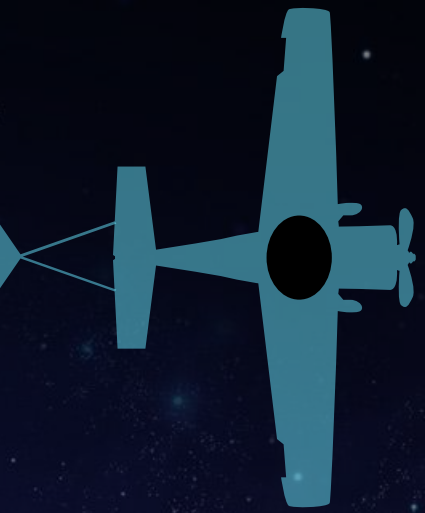
sqlalchemy

创建session

```
engine = create_engine('mysql://[user]:[password]@[host]:[port]/[dbname]?charset=utf8')
Session = sessionmaker(bind=engine)  # 创建Session类
session = Session()
```

新增

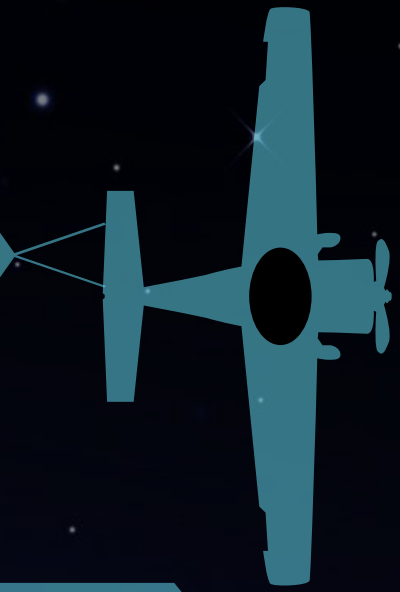
```
session.add(Person(name='jack'))  # 新增单条数据
session.add_all([Person(name='mike'), Person(name='siri')])  # 新增多条数据
session.commit()
```



sqlalchemy

查询

```
session.query(Person).limit(100).all()
session.query(Person).filter(Person.name == 'jack').all()    # 按条件查询
session.query(Person).first()    # 查询第一条
```



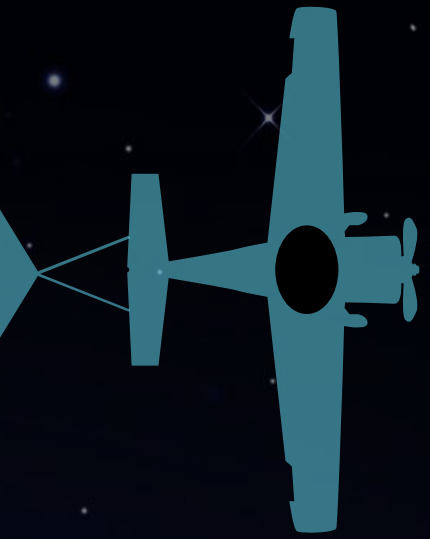
排序

```
session.query(Person).order_by(-Person.id).limit(10).all()
session.query(Person).order_by(desc(Person.id)).limit(10).all()
```

sqlalchemy

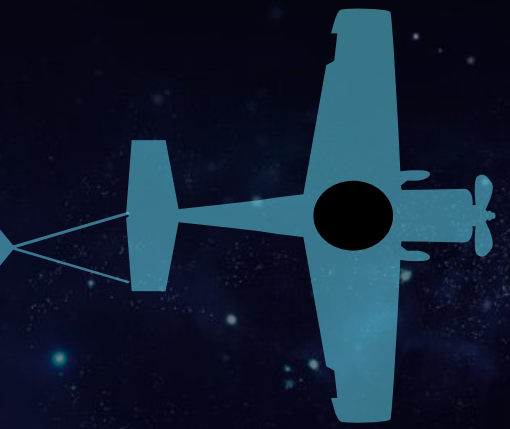
AND、OR查询

```
query.filter(Person(Person.id == 1, Person.name == 'jack')).all()
query.filter(and_(Person.id == 1, Person.name == 'jack')).all()
query.filter(Person.id == 1).filter(Person.name == 'jack').all()
query.filter(or_(Person.id == 1, Person.id == 2)).all()
```



使用text自定义sql

```
query.filter(text(“id” > 1)).all()
query.filter(text(“id > :id”)).params(id = 1).all()
query.from_statement(text(“select * from person where name = :name”)).params(name='jack').all()
```



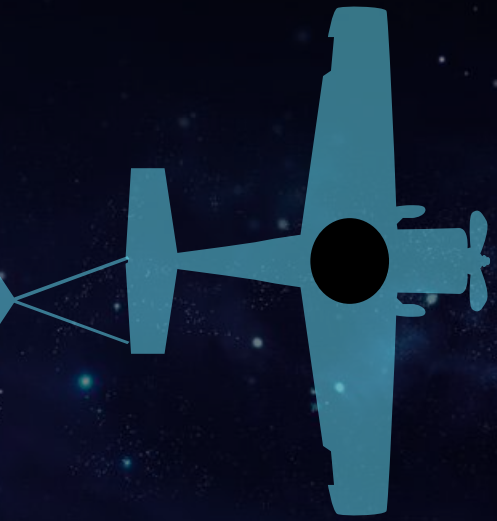
sqlalchemy

COUNT和GROUP

```
query.filter(Person.id > 1).count()
session.query(func.count(Person.id)).scalar()
session.query(func.count('*')).select_from(Person).scalar()
session.query(func.count(Person.name), Person.name).group_by(Person.name).all()
```

连表查询和统计

```
sql_from = Person.__table__.join(Address.__table__, Person.id == Address.person_id)
sql = select([func.count()]).select_from(sql_from).where(Person.name == 'jack')
count = session.execute(sql).scalar()
docs = session.query(Person.id, Person.name, Address.name).select_from(sql_from).filter(Person.name ==
'jack').all()
```



pymongo

创建连接client

要连接上mongoDB，通过pymongo库里的MongoClient来实现。传入host和port两个参数，或者直接传入一个连接字符串也行，如下所示：

client = pymongo.MongoClient(host='localhost', port=27017) 或 client = MongoClient('mongodb://localhost:27017/')

获取db和collection

client.mytest和client['mytest']这两种方式是一样的效果。同样地，指定操作集合时，db.person和db['person']也是一样的。

db = client.mytest

collection = db.person

insert

获取到collection之后，可以直接通过调用insert函数来插入数据。在Mongo3.x版本中官方已经不推荐使用insert()方法，而是替换成了insert_one()和insert_many()，用于插入单条记录和多条记录。

insert方法返回的结果是标识id（_id）列表，而insert_one方法返回的是一个InsertOneResult对象，通过inserted_id属性获取_id。

pymongo

find

find()方法返回一个生成器对象，需要进行遍历才能得到对应的dict对象。而find_one()方法直接返回的就是一个dict对象。

update

collection.update(condition, newValue)

condition和newValue均为字典类型，格式如：{'key' : 'value'}。前者表示更新条件，后者为更新值。更新值中可以带上$set，如collection.update(condition, {'$set': newValue})。不使用$set时，会使用当前更新的字典对象替换原对象，新对象中不含而旧对象中存在的字段会消失。使用$set时，则只更新新对象中存在的字段，旧对象的其他字段保持不变。

update也是官方不推荐使用的方法，推荐的是update_one和update_many，它们的第二个参数必须使用$set作为字典的键名，不允许直接传入修改后的新字典对象。

delete

删除操作通过remove函数来进行，在参数中指定删除条件。remove返回结果是一个dict对象：{'ok': 1, 'n': 1}，表示执行成功，删除条数为1。

同样地，官方推荐方法为delete_one()和delete_many()，返回结果是DeleteResult对象，其中的deleted_count属性为删除的数据条数。

05 Web框架

5 Web框架

常用的web框架

Django：Python 界最全能的 web 开发框架， 各种功能完备，可维护性和开发速度都很好。常有人说 Django 慢，其实主要慢在 Django ORM 与数据库的交互上，所以是否选用 Django，取决于项目对数据库交互的要求以及各种优化。而对于 Django 的同步特性导致吞吐量小的问题，其实可以通过 Celery 等方式解决。

Tornado：天生异步，性能强悍是 Tornado 的名片，然而 Tornado 相比 Django 来说，是较为原始的框架，很多内容需要自己去处理。当然，随着项目越来越大，框架能够提供的功能占比越来越小，更多的内容需要自定义实现，同时大项目往往需要性能的保证，这时候 Tornado 就是比较好的选择。Tornado项目代表：知乎。

Flask：微框架的典范，号称 Python 代码写得最好的项目之一。Flask灵活、轻便、安全且容易上手。 Flask框架的主要特征是核心构成比较简单，但具有很强的扩展性和兼容性，程序员可以使用Python语言快速实现一个网站或Web服务。一般情况下，它不会指定数据库和模板引擎等对象，用户可以根据需要自己选择各种数据库。

Web框架

Hello World

```
import  tornado.ioloop
import  tornado.web

class MainHandler(tornado.web.RequestHandler):
    def get(self):
        self.write("Hello, world")])

if  __name__ == "__main__":
    application = tornado.web.Application([(r"/", MainHandler)])
    application.listen(8888)
    tornado.ioloop.IOLoop.instance().start()
```

首先，生成一个Apptioncation实例，在构造函数中传入一个列表类型的参数（列表元素是由url路径和处理类组成的元组）。

然后，调用Application的listen方法监听服务器端口，事实上该方法会先创建一个HTTPServer实例，然后在HTTPServer实例上进行监听。

最后，执行IOLoop类的实例方法 start()，即可启动服务器。

请求方法和参数处理

Tornado支持任何合法的HTTP请求(GET、POST、PUT、DELETE、HEAD、OPTIONS)，只要在RequestHandler类中使用同名的方法即可。

熟悉http协议的都知道，http请求报文中包含请求方法的名称，因此很容易判断出http请求交由哪一个方法进行处理。

不管是get请求还是post请求，请求的参数都可以通过get_argument(“xxx”)来进行解析。举例如下：

```
class IndexHandler(tornado.web.RequestHandler):
    def get(self):
        greeting = self.get_argument('greeting', 'Hello')
        self.write(greeting + ',friendly user!')
```

HTTP状态码

写HTTP响应的方式是：使用write方法。

Http响应报文中则需要返回状态码，而状态码可以直接通过RequestHandler类的set_status()方法显式地设置。
当然，正常情况下，tornado会自动设置HTTP状态码，如200表示成功，400表示错误的请求，405表示不被允许的请求等。如果你想使用自己的方法代替默认的错误响应，可以通过重写write_error来实现。举例如下：

```
class IndexHandler(tornado.web.RequestHandler):
    def get(self):
        greeting = self.get_argument('greeting', 'Hello')
        self.write(greeting + ',friendly user!')

    def write_error(self, status_code, **kwargs):
        self.write("You caused a %d error." % status_code)
```

当尝试发起一个post请求时， 会输出：You caused a 405 error.

如果不复写write_error方法，则默认输出：405: Method Not Allowed。

作业

1. info = 'abc'
   info[2] = 'd'
为什么会报错？

2. 怎样将上题中的info字符串变成“abd”？

3. 输入1个数，判断其是否是素数。

4. 创建一个mysql表：user，含有id，name，age字段。插入10条记录，查询出所有记录并输出为json格式。最后更新id为5的记录。

5. 输出字符串格式的当前时间，格式为“年-月-日 时:分:秒”；将字符串格式的当前时间转化为long型的时间戳。


Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehen-derit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
EPS 10
ABSTRACT GRAPHIC
vector Illustration

THANK YOU