1. 虚拟环境：
   1. cmd转到目录后，virtualenv venv
   2. 激活虚拟环境：venv\Scripts\activate
2. Django模块：
   1. pip install Django==3.0.4
   2. 项目-应用
   3. MVC模型、MVT模型
   4. 创建项目：
      1. django-admin startproject 项目名 在目录下创建项目
      2. 进入项目名目录后，
      3. python manage.py runserver [端口号] 在本地ip的端口（默认8000）启动项目
   5. 创建应用：
      1. python manage.py startapp 应用名 创建应用
      2. 然后打开blog\setting.py
         1. 在INSTALLED\_APPS下添加 ’应用名’
   6. 数据模型：
      1. 打开 应用文件夹\models.py
      2. from django.db import models
      3. class 类名(models.Model):
         1. id = models.IntegerField(primary\_key=True) 整数字段，主键。
         2. user = models.ForeignKey(User, on\_delete=models.CASCADE)
            1. 设置外键，当User表的数据被删除时，此表数据也被删除。
         3. ……
      4. 字段类型：
         1. AutoField id自增字段 BinaryField 二进制源数据字段
         2. BooleanField 布尔值，应指明默认值 NullBooleanField 布尔值(可为None) CharField 字符串，指明max\_length值 TextField 文本域，用于大量文本
         3. DateField 日期字段，代表Python中datetime.date实例
         4. DateTimeField 时间字段，代表Python中datetime.datetime实例
         5. EmailField 邮件字段，是CharField的实现
         6. FileField 上传文件字段 ImageField 图片上传字段
         7. IntegerField 整数值字段 FloatField 浮点数值字段
         8. SlugField 只保存字母数字下划线和连接符，用于生成URL的短标签
         9. UUIDField 保存一般统一标识符的字段，代表Python中UUID的实例，建议-----------提供默认值
         10. ForeignKey 外键关系字段，需提供外键的模型参数。
         11. ManyToManyField 多对多关系字段，与上者类似
         12. OneToOneField 一对一关系字段，常用于扩展其他模型
      5. 类对应表，对象对应记录，对象属性对应字段。
      6. 数据库迁移：
         1. 打开blog\setting.py
            1. DATABASES改为
            2. DATABASES = {
            3. 'default': {
            4. 'ENGINE': 'django.db.backends.mysql',
            5. 'NAME': '数据库名',
            6. 'USER': '用户名',
            7. 'PASSWORD': '密码'
            8. }
            9. }
         2. 虚拟环境中安装pymysql
         3. 打开blog\\_\_init\_\_.py
            1. import pymysql
            2. pymysql.install\_as\_MySQLdb()
         4. 执行python manage.py makemigrations 创建数据表
         5. 执行 python manage.py migrate 数据库迁移
      7. 数据操作：
         1. 执行python manage.py shell 进入python命令行
         2. 执行from 应用.models import 类，类 导入类
         3. 新增数据：
            1. 变量 = 类.objects.create(字段名=值，字段。。。)
            2. 或
            3. 变量 = 类(字段名=值，。。。) 实例化类
            4. 变量.save() 同步到数据库
         4. 查询数据：
            1. for xx in 表.objects.all():
            2. print(xx . xxxxx)
            3. 或
            4. 表.objects.get(条件表达式，如id=1) 获取满足条件的一条记录
            5. 表.objects.first() 获取第一条
            6. 表.objects.filter(flag)[.order\_by(‘[-]id’)] 查找满足flag的所有记录，按id
            7. ----------------------------------升序排序，加负号表示降序排序

flag：字符字段\_ \_exact=’xx’ 字段必须与xx完全匹配

字符字段\_ \_iexact=’xx’ 字段必须与xx除大小写完全匹配

int字段\_ \_gt=1 字段大于1

int字段\_ \_lt=100 字段小于100

字符字段\_ \_contains=’xx’ 字段包含xx

字符字段\_ \_icontains=’xx’ 字段包含xx，不区分大小写

* + - 1. 修改数据：
         1. 变量=查询方法 将查询到的数据赋值给变量

查询到多个数据则需要使用for循环

* + - * 1. 变量.字段 = xx 对变量的字段赋值
        2. 变量.save() 同步到数据库
      1. 删除数据：
         1. 查询方法.delete()

删除无法恢复，定义数据模型时添加status字段，标记是否可用。

* + - 1. 外键：表示该条数据指向哪条数据
         1. 查询方法.外键名 返回所指数据的对象
         2. 表. 查询方法. 子数据所在表\_set. all() 返回指向查询到数据的子数据
  1. 管理后台：
     1. python manage.py createsuperuser 创建超管
     2. 启动服务后
     3. 网址/admin 进入登录页面
     4. 导入数据模型：
        1. 应用\admin.py 打开管理文件
        2. 绑定模型类到管理类：
           1. from 应用.models import 模型类,模型类
           2. class 类名Admin(admin.ModelAdmin):
           3. list\_display = (字段们) 添加显示列表
           4. list\_filter = (字段们) 添加过滤器
           5. search\_fields = ([字段们]) 添加查找器
           6. admin.site.register(模型类, 类名Admin)
        3. 在管理后台界面管理
  2. 路由：
     1. blog\urls.py -> urlpatterns
        1. 添加path(‘abcdef/’, 函数名) 当接收到网址/abcdef时，执行对应函数
           1. 函数定义在同目录下的views.py中
           2. 对应函数需接收参数request
        2. path(‘abcdef/<格式转换类型:变量名>/’, 函数名) 执行时向函数传递变量
           1. 格式转换类型（默认为str）:

str：匹配除分隔符（/）外的非空字符

int：匹配自然数

slug：匹配字母、数字、横杠、下划线的字符串

uuid：匹配UUID

path：匹配任何非空字符串

* + - 1. from django.urls import re\_path
      2. re\_path(r‘^xx/(?P<变量名>正则表达式)/$’, 函数名) 以正则表达式匹配URL
    1. 包含路由：打包添加路由
       1. blog\urls.py ->
          1. from django.urls import include

urlpatterns中添加path(‘xx/’, include(‘包名.模块名’))

* + - 1. 包名\模块名.py ->
         1. from django.urls import path
         2. urlpatterns= [
         3. path(’dd/’, 函数)。。。]

当匹配到网址/xx/dd时执行函数

* 1. 视图：xx\View.py ->
     1. FBV:
        1. from django.http import HttpResponse
        2. def fun(request):
           1. ...
           2. html = HTML代码，如”<html><body>65468453</body></html>”
           3. return HttpResponse(html)
     2. CBV:
        1. from django.views import View
        2. class XXX(View):
           1. def get(self,response):

pass

* + - * 1. def post(self, response):

pass

* + - 1. xx\urls.py -> urlpatterns
         1. path(‘’, views.XXX.as\_view()) 匹配到URL时执行get方法
  1. 模板：
     1. xx\views.py ->
        1. from django.shrotcus import render
        2. def 函数(request):
           1. return render(request,’xx.html’,{‘变量’:’值’}) 返回同级目录templates下
           2. --------------的xx.html，将变量传递，在html文件中用{{变量}}接收
     2. html ->
        1. 若传递值为可遍历对象：
           1. {% for xx in 值 %}

{{值.username}}

* + - * 1. {% endfor %}
      1. 使用表格模板：
         1. Bootstrap网站找到代码，并添加css文件到head标签
      2. 外键处理：
         1. 表. 外键. 属性
      3. 时间格式处理：
         1. {{ 表. 时间 | date:’Y-m-d’}} 使用过滤器实现功能
      4. 结构语句：
         1. {% if 条件表达式 %}
         2. {% endif %}
      5. 模板继承：
         1. 创建父模板.html，body标签内部：

重合代码

{% block content %}

{% endblock %}

* + - * 1. 子模板：

{% extends “父模板.html” %}

{% block content %}

不同代码

{% endblock %}

* + - * 1. 相当于把子模板block中的内容放到父模板的block位置