# **--------------- Django - CMD ---------------**

# **Основы**

**python -m venv name\_venv** создание виртуального окружения python

**pip list** показывает все модули установление в вирутальном окружении

**pip freeze или pip freeze > req.txt** показывает все сторонние моудли,которые мы сами установили в вирутальном окружении

**pip install -r req.txt** скачиваение всех зависимостей из указаногго файла

**python -m pip install --upgrade pip** обновление покатного менеджера pip до последней версии

**django-admin** выводит все команды django

**django-admin check** проверка всего проекта на ошибки

**python manage.py shell** оболочка проекта

**python manage.py createsuperuser** создание одного администратора

# **Базы Данных ( Модели )**

**python manage.py makemigrations** создание файла миграции(определяет структура базы данных)

**python manage.py sqlmigrate** отображает в терминале инструкции SQL, которые мы зделали

**python manage.py migrate** применение изменений в базе данных

**python manage.py shell\_plus --print-sql** в етой оболоче будут показываться SQL инструкции, которые мы зделали в командной строке

# **Отношения Таблиц друг к другу**

**my\_column = models.ForeignKey(“Other\_Table”, on\_delete = models.CASCAD , related\_name = “name\_manager”)** отношение многие к одному

**my\_column = models.ManyToMany(“Other\_Table”, on\_delete = models.PROTECT , related\_name = “name\_manager” )** отношение многи ко многим

**my\_column = models.OneToOne(“Other\_Table” , on\_delete.SETNULL , related\_name = “name\_manager”)** отношение один к одному

**Other\_Table** название другой таблици в которую будут вставляться значения в текущую таблицу

**on\_delete** указывает что будет на место значение в текущей таблице, если удаляем значение во второй таблице, на которую ссылаеться текущая

**related\_name** указывает названия менеджера для обращения в другой таблице из текущей

# **Записи в Базе Данных**

**my\_var = ClassBase (my\_property = “text”)** создание записи в таблице

**my\_var.save( )** сохранение записи в таблице

**BaseClass.objects.create(my\_column = “text”)** создание и применение записи в базу данных без использования переменной

**my\_var.update(my\_column= 1)** изменение значений колонки в записи в указаной колонке

**my\_var.delete( )** удаление записи из таблици

**my\_var = \_** нижняя черта содержит последний созданый обьект

**my\_var.pk** или **my\_var.id** выводит значение primary key (id) записи в базе

**ClassBase.objects.all()** выводит все записи из указаной базы данных

**ClassBase.objects.all()[ 0 ]** или **ClassBase.objects.all([ : 3 ])** возвращает записи по индексу или по срезу

**ClassBase.objects.filter(my\_column= “text”)** возвращает запись из базы данных только если виражение возвращает True

**my\_var.set ( my\_var2 )** или **my\_var.set ( [my\_var1 , my\_var2] )** изменение пустых строк на значение в третьей таблице, отношение многие к многим

**my\_var.add(my\_var2)** добавление значения в третьей таблице, отношения многие ко многим

**my\_var1.remove(my\_var2)** удаление значение в третьей таблице, отношения многие к многим

# **Основные Операторы**

**ClassBase.objects.filter( Q(pk\_id = 5) | Q(value = True))** обьект Q позволяет работат с условиями указывая свой оператор | (OR) &(AND) . ~Q(pk\_id = 3 ) означает NOT

**ClassBase.objects.first**(last)**()** возвращает первый или последнеий запись из базы данных

**ClassBase.objects.earliest(latest)(“my\_column”)** возвращает первую или последнюю запись в базе по указаному столбцу

**ClassBase.objects.get\_previous\_by\_my\_column ()** (get\_next\_by\_my\_column ()) возвращает предидущий или следующий леемент в базе по указаному столбцу

**my\_var.my\_manager** обращение от значения из текущей таблици в таблицу, где может находиться

**my\_var.my\_manager.exists()** проверят на существование в другой таблице

**my\_var.my\_manager.count()** проверят запись на количество в другой таблице

**ClassBase.objects.order\_by(“my\_column”)** вывод отсортированых записей в базе данных

**ClassBase.objects.filter(pk\_\_gt = F(“my\_column”))** позволяет обращаться к значения колонки по ее названию (from django.db.models import F )

**my\_var.my\_column = F(“my\_column”) + 1** увеличение текущего значения колонки записи на 1 (можно также уменьшать)

**my\_var = ClassBase.objects.all().annotate(my\_column = Value(my\_value))** возвращает такую же таблице, но с добавленой колонкой и указаными внутри ее значениями

**ClassBase.objects.aggregate(Min(“my\_column”))** позволяет работать с агрегирующими функциями такими как Count , Min , Max , Avg , Sum . Count – возвращает количество записей в указаной колонке. **values(“my\_column1” , “my\_column2”)** будет возвращать в выборке только указанные колонки

**ClassBase.objects.values(“name1\_\_name2”)** выводиться значения из таблици на которую ссылаеться текущая. name1 – название колонки в текущей таблице , name2 – название колонки в другой таблице на которую ссылаеться текущая таблица

**ClassBase.objects.all().annotate(new\_column=True)**  временно добавляет новую колонку в указаную таблицу из указаным значение True

**ClassBase.objects.annotate(Count(“name\_manager”))** групировка GROUP BY по всем колонкам текущей таблици

# **Import**

from . берем из текущей директории

from django.shortcuts import **render**

from django.contrib import **admin**

from django.urls import **path** , **re\_path** , **include**

from django.urls import **reverse** , **reverse\_lazy**

from django.db import **models**

from django.views.generic.base import **TemplateView ,**

from django.views.generic.list import **ListView**

from django.views.generic.detail import **DetailView**

from django.views.generic.edit import **FormView , CreateView , UpdateView , DeleteView**

from django.contrib.auth.decorators import **login\_required**

from django.contrib.auth.mixins import **LoginRequiredMixin**

from django.contrib.auth.views import **LoginView** , **LogoutView , PasswordChangeView , PasswordChangeDoneView , PasswordResetView , PasswordResetDoneView , PasswordResetConfirmView , PasswordResetCompleteView**

from django.contrib.auth import **authenticate, login, logout , get\_user\_model**

from django.contrib.auth.forms import **AuthenticationForm , UserCreationForm**

from django.contrib.auth.backends import **BaseBackend**

from django.core.validators import **MinLengthValidator, MaxLengthValidator**

# **----------------- HTML ----------------**

# **Фундамент**

**<! DOCTYPE html>** указываем последнюю версию html

**<html>** внутри храниться весь сайт

**<head></head>** внутри храняться характеристики для сайта, для посикових роботов браузера

**<body></body>** внутри храниться инофрмация которая отображаеться пользователю

**</html>** обозначает конец сайта

# **Внутри Head**

**<title>/** изменяет название вкладки сайта

**<meta charset = “utf-8”>** устанавливаем кодировку символов для сайта

**<meta name = “author” content = “max”>** указываем автора для сайта , content всегда указывает значение. Вместо атора можно подставлять descriptions или keywords

**<link rel=”stylesheet” href=”my\_file.css” >** подключение файла CSS my\_file.css к html странице

**<script src = “my\_file.js”>/** подключение файла JS my\_file.js к html странице

**<style> h1 { color : red } /**  подключение CSS внутри html документа

# **Теги Контейнеры**

**<header>/** внутри блока храниться шапка сайта

**<main>/** внутри блока храниться основная часть сайта

**<aside>/** внутри блока храниться боковая часть сайта

**<footer>/** внутри блока хранитсья подвал сайта

**<div class = “my\_name”>/** блочный тег контейнер в который помещаем участки кода, групировка елементов сайта. Каждый новый контейнер div с абзаца. Нужно прописывать атрбуты class чтоб работать с ними в CSS

**<span class = “my\_name”>/** строчный тег конейнер. Каждый новый контейнер span идет в строку. Нужно использовать внутри div для работы CSS как с отдельным елементом

# **Общие Атрибуты Для Тегов**

**title = “text”**  выводиться подсказка значения title при наведении мишкой на етот обьект

**class = “my\_name”** определяет имена для елементов (для взаемодействия с CSS)

**id = “my\_name”** позволяет установить уникальных идентификатор для елемента. Нельзя повторять значение id на одной html странице

**hidden**  скрывает елемент на странице

# **Текст**

**<h1** или **h2,3,4,5,6>/** теги заголовков с текстом

**<p>/** абзац с текстом

**<strong>/** или **<b>/** делает текст жирным

**<em>/** или **<i>/** делает текст курсивом

**<sub>/** или **<sup>/** делает текаст ниже или више основного ряда

**<br>** перенос текста на новую строку

**<hr>** перенос на новую строку с горизонтальной линией

**<abbr title = “name”>/** аббревиатура, при наведении мишки выводиться значение title

**<adress>/** для указания адресса

**<pre>/** сохраняет все отступы и проблемы при воводе на странице

**<code>/** внутри записываеться код языка програмирования

**<h1 style = “color: red ” >** применение стилизации css к тегу html

# **Блоки**

**<a href=”www.site” target = “\_parent”>/** создание ссылки , targer указывает как будем открывать ссылку

**<a name=”my\_name”>/ <a href= “#my\_name”>/** ссылка якорь, переход на указаную часть текущего сайта

**<a href=”mailto : name@gmail.com”>** переход на написание письма на указаную почту

**<img src = “photo.png” width=”100px” height = “50%” alt = “text”>** вставляет указаную картинку по названию, атрибутами можна регулировать размер, alt показывает текст который выводить при незагрузке картинки

**<progress max = “100” value=”30”>/** прогресс бар, max - указывает максимальное значение , value - указывает текущее значение

**<details> <summary>text</summary> </details>** текст при нажатии на который выпадает другой текст

# **Таблици**

**<table>** создание таблици , начало таблици

**<thead></thead> <tbody></body>** **<tfoot></tfoot>** разделение таблици на три части

**<tr> <th>my\_col1</th> <th>col2</th> </tr>** tr - создает рядок в таблице, th - создает верхньюю главную колонку , td - создает простые колонки уже ниже

**</table>** завершение таблици

# **Списки**

**<ol start = “3” type = “a ” > <li> text </li> </ol>** создание пронумерованого списка, start с какого числа начинает отсчет , type тип нумерации

**<ul type = “square”> <li> text </li> </ul>** создание непронумерованого списка , type указывает символы слева от елеметов списка

# **Формы та Поля для Ввода**

**<form action = ”new.html” method=”get”>/** контейнер для форм та поллей для ввода, action - указыват куда будуть отправление данные после ввода пользвателем , method - указываем как способом будут переданы данные. Все что ниже помещаеться туда

**<label for = “my\_class” >/** подпись над вводом текста, for - указываеть с каким полем надпись работает

**<input type = “text , submit , checkbox , color , button , reset ”>** поле для ввода текста, нажатии кнопки отправки , квадрат с галочкой , ввод цвета , кнопку . Тип submit - отправляет данные на сервер , button - просто кнопка которая ничего не делает , reset - удаляет данные из всех полей ввода указаныне пользователем

**<input placeholder = “my\_prompt” , value = “my\_value”>** placeholder - отобращаеться подксказки внупри поля для ввода текста , value - значения по умолчанию в поле

**<input name=”my\_name”>** можна обращаться к значению которое ввел пользователь в других языках

**<textarea>/** создает поля для ввода большого количества текста

**<button type = “submit , reset , button”>/** создание кнопки , атрибут type из поля input

**<select> <option>text</option> </select>** выпадающий список

**<select multiple size = “3” >/** multiple - можно выбирать несколько значений списка одновременно, size - количество елементов спискка, которое будет показано пользователю

**<option selected disabled >/** атрибуты для выпадающего списка, selected - ето по умолчанию выбрано ето значение, disabled - нельзя выбрать ето значение