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

# **my\_app -> view.py**

**def my\_fn(request): return HttpRespone(“text”)** функция берет запрос от пользователя и надает ответ. request - содержит запрос от пользователя , HttpResponse - дает ответ (нужно подключать через from django.http import HttpResponse

**my\_app -> urls.py**

# **my\_project -> urls.py**

**urlpatters = [ path ( “my\_app/” , my\_fn ) ]** добавление url запроса

**urlpatters = [ path ( “ ” , my\_fn ) ]** пустое поля означает главную страницу сайта без вложений

**urlpatters = [ path ( “ ” , include (my\_app.urls.py)) ]** импортируем все значения urlpatters из указаного файла в указаном приложении

# **----------------- Jinja -----------------**

# **Подключение**

**pip install Jinja** скачивание етого сторонего модуля

**import jinja2** подключение етого модуля к текущему файлу

**from jinja2 import Template , Environment** отдельное подключение составных частей

# **Создание Шаблона**

**my\_templ = Template(“text { { my\_value } } ” )** создание шаблона

**my\_templ = Template (“text { 5 \* 3 }” )** внутрь шаблона можна вписывать любой код python

**my\_var = ‘’’ {%raw%} text {{x}} text {%endraw%} ‘’’** ерканирование данных , виводяться данные не изменяя значений вложених переменных

**my\_var = Tepmlate (“{{ my\_value | e }}”)** екранирование кода html, на сайте выводиться как код

**↓ Математические Функции ↓**

**my\_data = “text {{my\_name | sum }}”** возвращает суму елементов переданой полседовательности

**my\_data = “text {{my\_name | sum (attribute = ‘my\_key’)}}”** можна дополнительно указывать атрибуты (для словаря)

**my\_data = “text {{ my\_name | max } } ”** возвращает максимальное значение из последовательности или строчку из таблици

**my\_data = “text {{ (my\_name | max(attribute = “my\_key”)). my\_value } } ”** выводи только указаное значение из вернувшейся строки

**my\_data = “text {{ (my\_name | radom} } ”** возвращает рандомное значение

**my\_data = “text {{ (my\_name | replace (‘value1’,’value2’)} } ”** изменяет значения

# **Операторы %%**

**my\_data = ‘’’ операторы’’’**

**{% for x in my\_list - %} {% - endfor - %}** применяет цикл for и берем значения из списка, который создан до етого. for x in my\_list **-** указывает что без пропусков выводить

**{% - if my\_value > 5 - %} {%endif%}** применяем логический оператор if

**{% - if my\_value > 5 - %} {%else%} {%endif%}**  конструкция if - else

**{% - if my\_value > 5 - %} {%elif my-value < 3 %} {%endif%}** конструкция if elif

**{%filter upper%} {{my\_name.my\_key}} {%endfilter%}** функция filter

↓ **Блоки** ↓

**{%block my\_name%} {%endblock%}** создание именнованых блоков

**{%extends ‘my\_file.html’%} {%block my\_name%} text {%endblock%}** в текущем файле береться весь функционал из my\_file.html и добавляеться блоками, котоыре будут ниже в текущем файле

**{%block my\_block scoped%} {%endblock%}** scoped позволяет обращаеться в нутри текущего блока к переменным из внешнего блока

**self.my\_block( )** возвращает содержимое блока, который лежит в текущем файле

**super( )** возвращает содержимое из базового файла (который наследуем) по названию блока

↓ **Макрос** ↓

**{%macro my\_macro(my\_value1 , my\_value2 = 0 )%}** создание макроса с параметрами , параметры могут быть по умолчанию

**{%endmacro%}**

**<p> {{ my\_macro(‘my\_username’) }}** вызов макроса и передача в него значений

↓ **Caller** ↓

{%- **macro** my\_macro(my\_value) -%}

**<ul>**

{%**for** u **in** list\_of\_user%}

**<li>** {{ u.name }} {{ **caller**(u) }}

внутри текущего макроса вызваеться через caller оперделеный макрос снизу

{%**endfor**%}

**</ul**

--------------------------------------------- {%- **endmacro** -%} -----------------------------------------------

{%- **call**(user) my\_macro(my\_value) -%}

**<ul>**

**<li>** age : {{ user.old }}

**<li>** weight: {{ user.weight }}

**</ul>**

{%- **endcall** -%}

↓ **Подключение Файла как Шаблона** ↓

**{% include ‘name.html’%}** позволяет подлючать другие файлы как шаблоны в текущий

**{% include [ ‘name1.html’ , ’name2.html’ ] %}** подключение сразу нескольких файлов

**{% include [ ‘name1.html’ , ’name2.html’ ] ignore missing%}** проигнорируте если файл не будет найден

**{% import ‘name.html’ as new\_name%}** импортирование модуля и дача ему псевдонима

**{{ new\_name. my\_macro(‘’) }}** обращается к модулю в том же файле, где и импортировали

**{% from ‘name\_html’ import my\_macro%}** берем из модуля только определеный макрос

# **Работа с Шаблоном**

**my\_templ.render( my\_value = my\_var** или **my\_value = “text”** или **my\_value = my\_object )** подстановка значения в шаблон, также можна вписывать в шаблон свойства из обьекта

# **Загрузчики**

**FileSystemLoader(“my\_folder”)** етот класс позволяет подключать шаблоны из файлов жостокого диска текущего устройства. Нужно указывать папку в которой лежит шаблон, если в текущем каталоге тогда пустота (“ ”)

**FunctionLoader(my\_fn)** берет шаблон из функции. Нужно передавать ссылку на функцию , а не работу функции( )

↓ **Как Работают** ↓

**a = FileSystemLoader(“my\_folder”)** берем папку с документами, где есть шаблон

**b = Environment(loader = a)** загружаме все папк в главный класс Environment

**c = b.get\_template(“my\_name.html”)** берем указаный файл как целый шаблон

**c.render(name = my\_value)** начинаем использовать

# **---------------- SQLite ----------------**

# **Подключение**

**import sqlite3** подлючения модуля для работы с базами данных

**sqlite3.connect(“my\_db”)** подключение уже существующей базы данных (база лежит в однаковом каталоге). Если нету, тогда создаеться

**sqllite3.close()** закрытия базы данных после работы, прекащает работу базы

# **Типы Данных**

**NULL = NULL** значение NULL, не пустая строка

**INTEGER = 1 , 2 ,3** целое число

**REAL = 1.2 , 5.3 , 7.1** числа с плавающей точкой

**TEXT = “name”** строковый тип данных

**BLOB = “123.png”** для хранения изображений

# **Работа с Cursor()**

**my\_cur = slqlite.connect(“my\_db”). cursor( )** позволяет работать с базой данных через переменую **my\_cur.execute(“”” ”””)** действия с базой данных -> создание таблици , их изменения или запросы (делает только одно действие)

**for x in my\_list: cur.execute(“INSERT INTO my\_tab VALUES (NULL , ? , ?)”, x )** заполнениея таблци с помощью цикла for и списка кортежей, который был заполнен до етого

**cur.executemany(“INSERT INTO my\_tab VALUES (NULL , ? , ?)”, my\_list )** функция переберает указаныи список кортжей и заполняет ими таблицу

**my\_cur.execute(“UPDATE my\_tab SET my\_col = : my\_value WHERE my\_col > 1”, {“my\_value”: 0 })** изменение значений строк в таблице

**my\_cur.execute(“INSERT INTO my\_tab VALUES (NULL,?)”, ( slqlite3Binary(my\_imd), ) )** запись изображения в строку базы

**my\_cur.executescript(“”” UPDATE; DELETE”””)** выполнение сразу нескольких действий

**my\_cur.lastrowid** возвращает id последнего елемента который добавляли в таблицу

**my\_cur.fetchall( )** возвращает все строки из базы данных (перед етим нужно прописывать запрос в cur.execute(“SELECT \* FROM my\_tab”) )

**my\_cur.fetchmany(3)** возвращает указаное количество строк из базы данных

**my\_cur.fetchone()** возвращает первую строку из базы данных

# **Работа с Самой Базой**

**my\_connect = slqlite3.connect(“test”)** подключение или создания базы данных на диске устройства

**my\_connect = sqlite.connect(“ :memory: ”)** подключение или создание базы данных в ОЗУ устройства

**my\_connect.commit( )** переносит зделаные изменения в базу данных

**my\_connect.rollback()** при возникновении ошибки откатывает изменения до написанной команды BEGIN;

**my\_connect.row\_factory = sqlite3.Row** теперь каждая строка в таблице ето обьект, разширивает функционал взаемодейсвия

**sqlite3.Binary(my\_img)** повзращает картинку в бинарном виде для записи в строку базы

**my\_con.iterdump()** возвращает всю таблицу в виде кода .sql (нужно проходиться циклом for)

# **----------------- 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

# **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 елементво (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

# **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