НАВИГАТОР ПО ШПАРГАЛКАМ ГРАФИЧЕСКАЯ ШПАРГАЛКА

Python Basics | HTML | Flask Basic | Шаблоны | Создание API | CSS

Python Basics

Переменные и типы данных

Основные типы данных

```
boolean = True/False
integer = 10
float = 10.01
string = "123abc"
list = [ value1, value2, ... ]
dict = { key1: value1, key2: value2, ... }
```

Операции с числами

+	сложение
-	вычитание
*	умножение
/	деление
**	возведение в степень
%	модуль
//	целочисленное деление

Типы и преоб

str(a)	'5
int(a)	5
float(a)	5
bool(a)	T

Вывод

Простой вывод

```
print("Hello world!")
```

Hello world с переменной

```
msg = "Hello world!"
print(msg)
```

Ввод

По умолчанию всегда строка

Подсказка для ввода строки

```
name = input("What's your name? ")
print("Hello, " + name + "!")
```

Подсказка для ввода чисел

```
age = input("How old are you? ")
age = int(age)

pi = input("What's the value of pi? ")
pi = float(pi)
```

Строки

Операции со строками

my_string = "Awesome"	Объявляет строку
my_string + "skypro"	Прибавляет строку к строке
my_string * 2	Повторяет строку дважды
's' in my_string	Проверяет вхождение подстроки

Слайсы в строках

my_string[0]	Возвращает символ на позиции i
my_string[1:3]	Выбирает элементы на позиции 1 и до 3
my_string[1:]	Выбирает элементы на позиции 1 и дальше

Методы строк

string.upper()	Переводит в верхний регис
string.lower()	Переводит в нижний регист
string.count(x)	Сколько раз встречается x
string.find(x)	Позиция первого вхождени
string.replace(x,y)	Заменить в строке x на y
string.strip(x)	returns a list of values determ
string.join(L)	склеить коллекцию L с помо
"Привет {} !".format("Вася")	Заменяет {} на Вас

Условия

If условие

```
if age >= 18:
    print("You can vote!")
```

If - Elif - Else условие

```
if age < 4:
    ticket_price = 0
elif age < 18:
    ticket_price = 10
else:
    ticket_price = 15

    game_active =
    can_edit = False
True
```

Операторы сравнения

==	равно
!=	отличается
>	больше
<	меньше
>=	больше или равно
<=	меньше или равно

Логические операторы

and	логическое И
or	логическое ИЛИ
not	логическое НЕ

Условия со списками

```
if <значение> in <список>:
    <блок кода>
if <значение> not in <список>:
    <блок кода>
```

Списки

Операции над списками

list = []	создать пустой список

Методы списков

list.append(x)	Добавляет значение x в коне

list[i] = x	записывает x на позиции i
list[i]	возвращает элемент на позиции i
list[-1]	возвращает последний элемент
del list[i]	удаляет элемент с индексом i

Переменные и многоуровневые списки

```
a = "is"; b = "nice"; my_list = [a, b]
my_list2 = [[4,5,6,7],[3,4,5,6]]
```

Слайсы

my_list[1:3]	Выбирает элементы на позиции 1 и до 3
my_list[1:]	Выбирает элементы на позиции 1 и дальше
my_list[:3]	Выбирает элементы на позиции до 3
my_list[:]	Создает копию списка

list.extend(L)	Добавляет список L в конец
list.insert(i,x)	Добавить x на позицию i
list.copy()	Создает копию списка
list.index(x)	Возвращает позицию элеме
list.count(x)	Сколько в списке элементов
list.sort()	Сортирует элементы (есть ог
list.reverse()	Инвертирует порядок элеме
list.remove(x)	Удаляет элементы, равные x
list.clear()	Удаляет все элементы списк
list.pop(i)	Удаляет элемент на позиции возвращает его значение

Циклы

Перебор списка

```
i = 0
while i > 5:
    i++
```

```
for item in ["a", "b", "c"]
    print(item)
```

Диапазон

```
for item in range (0, 5):
    print(item)
```

range(5)	Возвращает последовательность от 0 до
range(2000,2018)	Возвращает последовательность от 200
range(0,11,2)	Возвращает последовательность от 0 до с шагом в 2
range(0,-10,-1)	Возвращает последовательность от 0 до с шагом в -1

Словари

Простой словарь

```
alien = { "color": "green", "points": 5 }
```

Получение значения

```
print("The alien's color is " + alien["color"])
```

Добавление новой пары

```
alien["x_position"] = 0
```

Методы словарей

dict.keys()	Возвращает список ключей
dict.values()	Возвращает список значений
dict.items()	Возвращает пару (key,value)
dict.get(k)	Получает значение по ключу k
dict.pop(k)	Удаляет элемент с ключом i, возвращает его значение
dict.update(D)	Добавляет пары (D) в словарь
dict.clear()	Удаляет все пар из словаря
dict.copy()	Создает копию словаря

Операции над словарями

dict = {}	Объявляет пустой словарь
dict[k] = x	Присваивает значение по кл
dict[k]	Получает значение по ключу
del dict[k]	Удаляет пару по ключу k

Итерация по парам ключ-значение

```
fav_number = {"eric": 17, "ever": 4}
for name, number in fav_number.items():
    print(name + " loves " + str(number))
```

Итерация по всем ключам

```
fav_number = {"eric": 17, "ever": 4}
for name in fav_number.keys():
    print(name + " loves a number")
```

Итерация по значениям

```
fav_number = {"eric": 17, "ever": 4}
for name in fav_number.values():
    print(str(number) + ' is a favorite')
```

оформление)

orme'>Раскрась меня</h1>

ько классов

ton red'>Раскрась меня</h1>

ля оформления

Блочный (большой)

ran> Строчный (маленький)

HTML

Настраиваем | Показываем

Минимальный документ

```
<!DOCTYPE html>
<html lang=ru>
<head>
    <meta charset="UTF-8">
    <title>Название вкладки</title>
</head>
<body>
    <h1> Красивое</h1>
    <p> Для начала норм</p>
</body>
</html>
```

Заголовки

Шесть разных вариантов

```
<h1> Самый крутой </h1>
<h6> Самый мелкий </h6>
```

Парарафы

```
<p> Я параграф текста </p>
<p> Я тоже параграф текста </p>
```

Форматирование

in.css' rel='stylesheet' />

Списки

```
<ul>
  <li>раз</li>
  <li>раз</li>
  <li>раз</li>
</ul>
```

```
<ol>
  <li>раз</li>
  <li>раз</li>
  <li>раз</li>
</ol>
```

Таблицы

```
<table>
  <tr> <td> 1 </td> <td> 2 </td></tr>
  <tr> <td> 3 </td> <td> 4 </td></tr>
</table>
```

Семантическая разметка

<main>	Блок основного контента
<header>	Блок шапочки
<footer>	Блок подвальчика
<nav>	Блок навигации
<section>	Блок контента с заголовком
<article>	Блок статьи, например
<aside>	Сайдбар (боковая колонка)

Правила

- ⚠ Проверяем парность
- ⚠ Закрываем кавычки у атрибутов
- ⚠ Следим за отступами
- ⚠ Проверяем тип ссылок у ссылок и картинок
- ⚠ Весь текст контента храним в парарафах

** ** Полужирненький

** ** Курсивненький

<u> </u> С подчеркиванием

<s> </s> С зачеркиванием

Картинки

Внутренняя

```
<img src="/внутренняя.png" />
```

Внешняя

```
<img src="http://site.com/внешняя.png" />
```

Разделители

**
 Перенос строки**

<hr /> Горизонтальный разделитель

Спецсимволы

ru.wikipedia.org/wiki/Мнемоники_в_HTML

Грамматика

Парные теги

<открыли> написали **</закрыли>**

Одиночные теги

<открылизакрыли />
<открылизакрыли>

Теги с атрибутами

<открылизакрыли **свойство="значение"** />

<открыли **свойство="значение"**> ага </закрыли>

<открыли **инструкция**> ага </закрыли>

CSS

Подключение стилей

```
<link rel="stylesheet" href="css/style.css">
```

Селектор выбирает теги

Точка с запятой разделяет правила

```
h1 {
  свойство : значение ;
  свойство : значение ;
}
```

Двоеточие отделяет свойство от значения

Селекторы

```
<h2>Заголовок тут</h2>
```

```
h2 { color: red; }
```

```
<div class="myclass">
  Текст тут
</div>
```

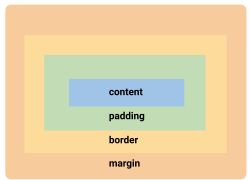
```
.class {
  font-weight: bold;
}
```

Цвет и фон

background-color: fuchsia;	Фон любого блока
color: #fc3;	Цвет текста

Шрифты

font-family: Arial	Какой шрифт
font-size: 20px	Какого размера



Размеры

width: 200px;	Ширина
height: 200px;	Высота

Границы и рамки

border-width: 1px;	Толщина
border-style: solid / dotted / dashed;	Стиль линии
border-color: red;	Цвет линии

Или

border: 1px solid red;

Отступы и поля

margin-left: 10px;	Поля снаружи
margin-right: 10px;	
margin-top: 10px;	
margin-bottom: 10px;	

text-decoration: none / underline / Подчеркнем?

text-align: left / right / center Какое выравнивание?

padding-left: 10px;
padding-right: 10px;
padding-top: 10px;
padding-bottom: 10px;

Отступы внутри

Считать отступы частью ширины
box-sizing: border-box;

Flask Basics

Простое приложение

Установка

```
$ pip install Flask
```

Минимальное приложение

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello():
    return 'Hello, World!'

app.run('0.0.0.0', 8000)
```

Рендер шаблона

Без данных

```
from flask import Flask, render_template

app = Flask (__name__)

@app.route('/')
def get_page():
    return render_template ('index.html')
```

С данными

```
from flask import Flask , render_template

app = Flask(__name__)

@app.route("/")
def get_page():
    title = "Страничка про нас"
    return render_template("index.html", title=title)
```

Получение JSON

Пример запроса

```
POST /gojson

{
"title": "Изучаем Flask",
"weeks": 4
}
```

```
from flask import Flask, request

...

@app.route('/gojson')
def json_data():
    title = request.json.get('title')
    weeks = request.json.get('weeks')
    return f'{title} {weeks}'
```

Эндпоинт без параметра

```
@app.route('/')
def main():
    return 'Здесь будет главная'
```

Эндпоинт с переменной

```
@app.route('/numbers/')
def working_with_numbers(num):
    return 'Вы выбрали номер {num}'
```

Эндпоинт с квери параметром

abc.com/?s=greeting=hello

```
from flask import Flask, request

...

@app.route('/search')

def search():
    query = request.values.get("s")
    return f"Выполняем поиск по {query}"
```

abc.com/?from=3&to=6

```
from flask import Flask, request

...

@app.route('/filter')

def search():
    from = request.values.get("from")
    to = request.values.get("to")
    return f"Фильтруем {from} + {to}"
```

Возвращение JSON

```
from flask import Flask, jsonify

...

@app.route('/gojson')
def json_data():
    data = {'title': 'Изучаем Flask', 'weeks': 4}
    return jsonify(data)
```

Content-Type: application/json — Возвращает

```
{
"title": "Изучаем Flask",
"weeks": 4
}
```

Flask шаблоны

Jinja 2

Рендер шаблона с данными

```
from flask import Flask , render_template

app = Flask(__name__)

@app.route("/")
def get_page():
    title = "Страничка про нас"
    return render_template("index.html", title=title)
```

Вывод переменной в шаблон

```
# templates/hello.html

<title>Привет, {{ name }}!</title>

<h1>Привет, {{ name }}!</h1>
```

Вы

```
# t

<h

<p

<p
```

Условия в шаблоне

```
{% if is_in_store %}
    Этот продукт доступен
{% endif %}

{% if is_in_store %}
    Этот продукт доступен
{% else %}
    Этот продукт недоступен
{% endif %}
```

Циклы в шаблоне

```
{% for item in items %}
    <li>{{ item.href }}</li>
{% endfor %}

{% for item in navigation %}
    <a href="{{ item.href }}">{{ item.caption }}</a>
{% endfor %}
```

Вы

```
# t

<h

<p

<p
```

Сериализация

Установка

```
pip3 install marshmallow
```

Импорт

```
from marshmallow import Schema, fields
```

Выпол

Одну з

```
user = d
book_sc
json_str
```

Стартовый код

```
# Импорт
from marshmallow import Schema, fields

# Это модель, в ней мы ничего не меняем
class Book(db.Model):
    __tablename__ = 'books'
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String)


# Описать сложную модель в виде класса схемы
class BookSchema(Schema):
    id = fields.Int(dump_only=True)
    name = fields.Str()

book = Book(id=1, name="Kolobok", author="people")

# Создать экземпляр схемы
book_schema = BookSchema()

# Выполнить метод схемы
json_string = book_schema.dumps(user))

print(json.string)
```

Схема

```
class BookSchema(Schema):          Наследуем от Schema
    # список полей
    name = fields.Str()            Указываем поля
```

Типы полей для сериализации

```
a = fields.Str()                   Строка
a = fields.Int()                   Число
a = fields.Float()                 Число с точкой
a = fields.Decimal()               Десятичное число
a = fields.Bool()                  Булев тип
```

Одну з

```
user = d
book_sc
book_da
```

Список

```
user = d
book_sc
book_da
```

Список

```
user = d
book_sc
book_da
```

Создание API

Обработка GET запроса

Одна запись

```
@app.route('/users/<int:id>')
def get_users(uid):
    data = {'name': 'Ivan'}
    return jsonify(data)
```

Обработка ошибок

```
@app.route('/hello')                   Указываем верный код
def hello():
    if ...:
        return {"error": "something s wrong"}, 400
```

HTTP(

```
200: Ok
201: Cr
204: Nc
300: Mu
```


Несколько записей

```
@app.route('/users')
def get_users():
    data = [{'name': 'Ivan'} , {'name': 'Daria'}]
    return jsonify(data)
```

```
301: Mu
400: Ba
401: Ur
403: Fo
404: Nc
405: Mu
500: Int
503: Se
```

Обработка POST запроса

```
@app.route('/test', methods=['POST'])
def add_user():
    data = request.json
    print("Данные обрабатываются ", data )
    return jsonify({"status":"Данные
обрабатываются"})
```