

Лабораторная работа

Основы клиент-серверной технологии, обработка данных из формы

Задание 1. Установить FLASK и создать приложение «Hello, World!».

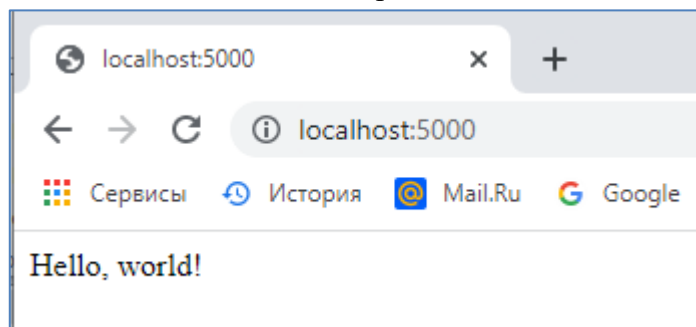


Рисунок 1. Страница приложения

Порядок выполнения работы

1. Установить FLASK как обычную библиотеку Python. Для этого:

- a. Найти папку, в которой размещена программа **pip** или **pip3**, например, это путь к папке на моем компьютере

C:\Users\OGP\AppData\Local\Programs\Python\Python39\Scripts
этот путь скопирован из командной строки папки Script (рисунок 2).

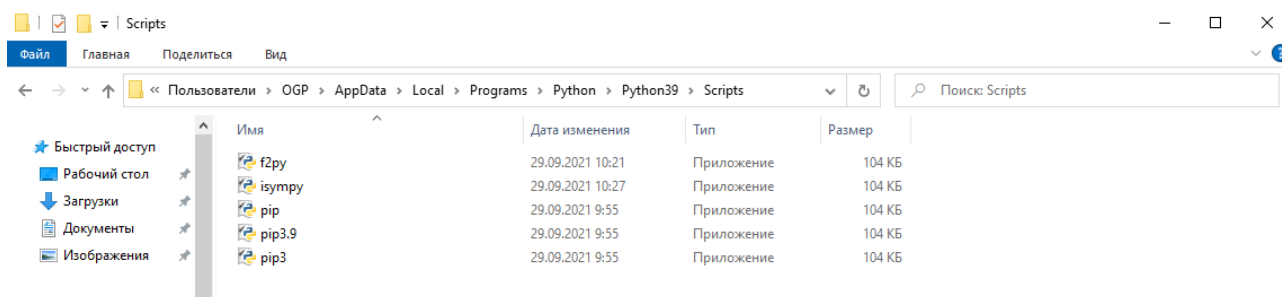


Рисунок 2. Папка, где расположена программа pip

- b. Открыть приложение «Командная строка»
c. Перейти в папку с pip с помощью команды **cd** (рисунок 3)
d. Инсталлировать нужный пакет с помощью команды:
`pip install имя_пакета`
желательно использовать последнюю версию pip, например, pip3:
`pip3 install имя_пакета`

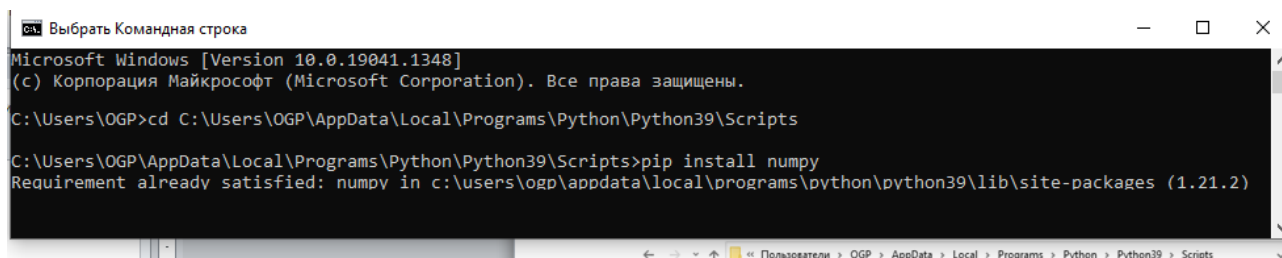


Рисунок 3. Использование командной строки для установки библиотек

Для установки FLASK использовать команду:

```
pip3 install flask
```

В результате в папке Script появится файл flask.

2. Создать папку для разработки приложения Lab_6_0. В эту папку необходимо скопировать файл flask.

3. Настроить и запустить локальный сервер flask, для этого в **Командной строке**:

a. перейти в папку Lab_6_0 (с помощью команды cd);

b. запустить последовательность команд:

```
set FLASK_APP=app
set FLASK_ENV=development
flask run
```

4. В папке Lab_6_0 создать файл app.py со следующим содержимым:

```
from flask import Flask

app = Flask(__name__)

@app.route('/')

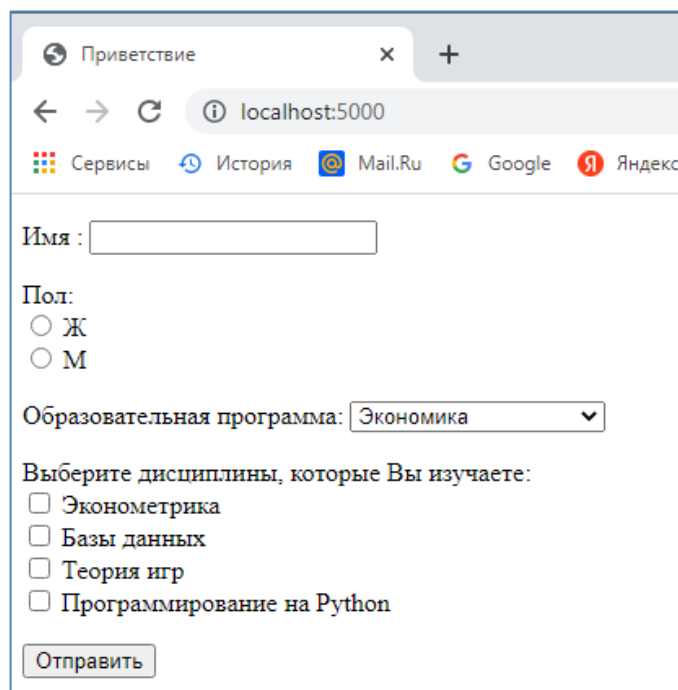
def index():
    return "Hello, world!"
```

5. В браузере набрать

http://localhost:5000

В результате на страницу браузера будет выведено сообщение Hello, world! (рисунок 1).

Задание 2. Создать приложение «Приветствие». Пользователю предлагается форма для заполнения (рисунок 4). После того, как пользователь заполнит поля формы и нажмет кнопку **Отправить**, должна отобразиться страница с приветствием, показанная на рисунке 5.



Приветствие

← → ↻ ⓘ localhost:5000

Сервисы История Mail.Ru Google Яндекс

Имя:

Пол:

☐ Ж

☐ М

Образовательная программа: Экономика ▼

Выберите дисциплины, которые Вы изучаете:

☐ Эконометрика

☐ Базы данных

☐ Теория игр

☐ Программирование на Python

Рисунок 4. Форма для заполнения

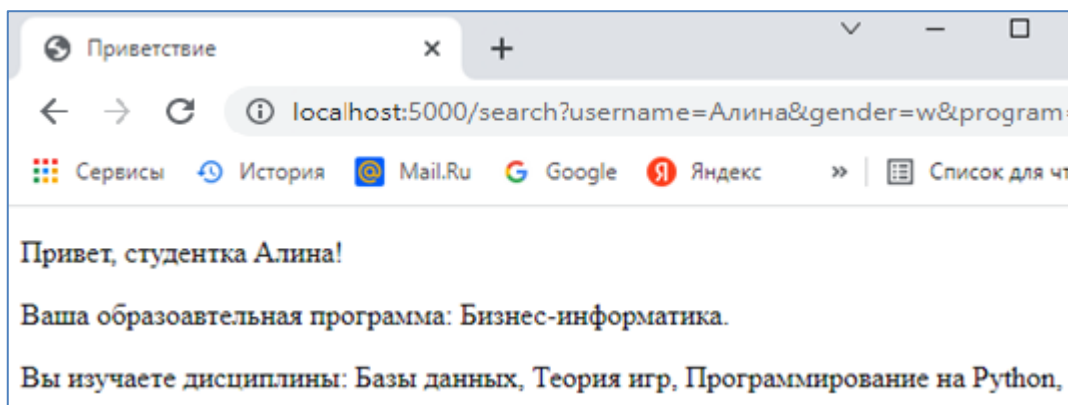


Рисунок 5. Страница с приветствием

Структура приложения

Приложение будет состоять из двух страниц. Главная страница `index` используется для отображения формы, страница `hello` выводит приветствие пользователю на экран.

Логика работы приложения для главной страницы показана на рисунке 6а.

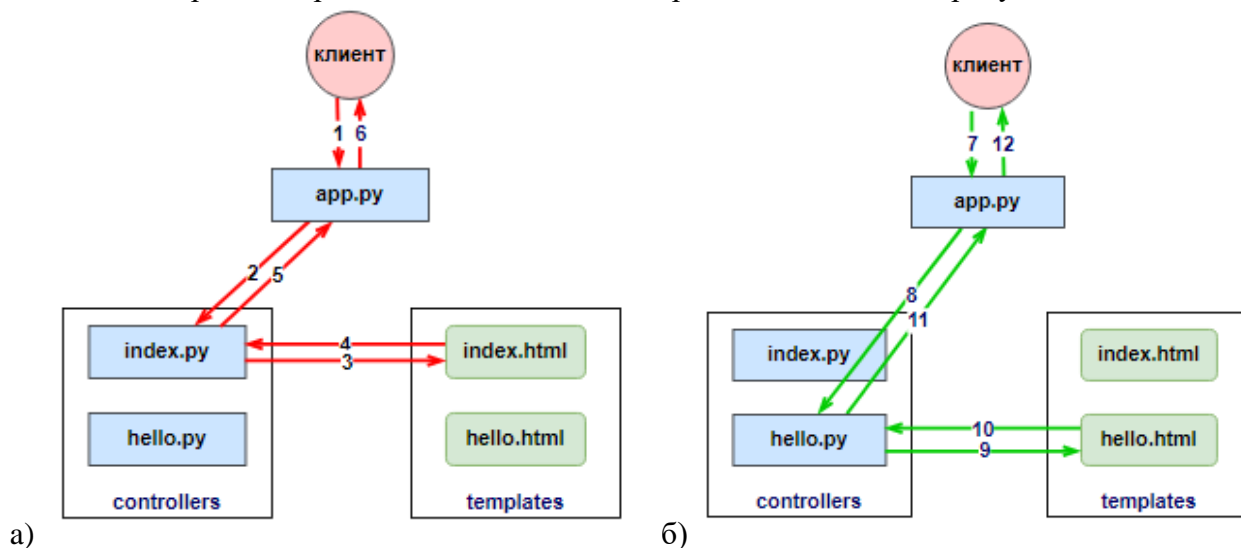


Рисунок 6. Структура приложения

1. В браузере пользователь вызывает **главную страницу** приложения (с пустой формой для заполнения, рисунок 4). Запрос передается в управляющую программу приложения `app.py`, которая создает экземпляр приложения Flask.

2. Программа `app.py` передает запрос пользователя контроллеру главной страницы `index.py`.

3. Контроллер выполняет обработку запроса, формирует необходимые данные и запускает генерацию шаблона страницы `index.html`.

4. Сгенерированная страница `index.html` передается обратно в контроллер `index.py`.

5. Контроллер `index.py` возвращает полученную html-страницу в `app.py`.

6. Управляющая программа отправляет страницу `index.html` в браузер пользователя.

На рисунке 6б показана логика работы приложения после того, как пользователь заполнит форму и нажмет кнопку **Отправить**.

Порядок выполнения работы

1. Создать папку **Lab_6**. В нее скопировать файл с библиотекой **flask**. В этой папке создать вложенные папки:

- `templates`, в которой будут храниться шаблоны html-страниц;
- `controllers` – для программ-контроллеров;
- `static` – для рисунков и css-стилей.

2. В папке `Lab_6` создать главную программу приложения `app.py`:

```
from flask import Flask

app = Flask(__name__)

# здесь должны импортироваться все программы-контроллеры,
# размещенные в папке controllers
import controllers.index
import controllers.hello
```

2. В папке `controllers` создать программу `index.py` для генерации страницы с формой `index.html`.

```
from app import app
from flask import render_template

@app.route('/', methods=['GET'])

def index():
    # выводим форму
    html = render_template('index.html')
    return html
```

3. В папке `templates` создать шаблон `index.html`, в котором разместить форму с текстовым полем (`username`) и кнопкой **Отправить**. Поля формы будем обрабатывать с помощью метода GET, данные из формы будут отправляться в программу `hello.py`.

```
<!DOCTYPE HTML>
<html>
  <head>
    <title> Приветствие </title>
  </head>
  <body>
    <form action="{{ url_for('hello') }}" method="get">
      <p>Имя: <input type = text name=username></p>
      <p><input type=submit value=Отправить></p>
    </form>
  </body>
</html>
```

4. В браузере запустить локальный сервер (`localhost:5000`), который по умолчанию выводит страницу `index.html`, результат показан на рисунке 7

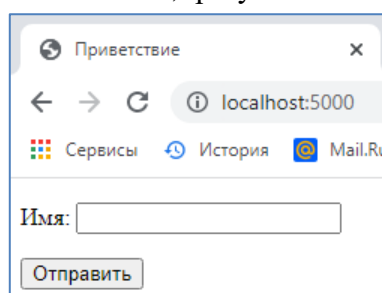


Рисунок 7. Страница с формой

2. Если в поле формы ввести имя и нажать кнопку **Отправить**, то будет выведена ошибка, так как программа `hello.py` для обработки данных из формы еще не создана. В папке `controllers` необходимо создать программу `hello.py`, которая будет обрабатывать данные формы, а также генерировать страницу с приветствием `hello.html`.

```
from app import app
from flask import render_template, request

@app.route('/hello', methods=['GET'])
def hello():
    # для каждого передаваемого параметра формы нужно установить
    # значение по умолчанию, на случай если пользователь ничего не введет
    name = ""

    # получаем параметр из формы
    name = request.values.get('username')

    html = render_template(
        'hello.html',
        name = name
    )
    return html
```

3. В папке `templates` создать шаблон `hello.html`, в котором будет выводиться приветствие пользователю, в качестве параметра в шаблон передается имя пользователя `name`:

```
<!DOCTYPE HTML>
<html>
  <head>
    <title> Приветствие </title>
  </head>
  <body>
    <p>
      Привет, {{name}}!
    </p>
  </body>
```

В результате, (предварительно обновить страницу в браузере) после того, как будет введено имя в поле и нажата кнопка **Отправить**, на странице отобразится приветствие пользователю (рисунок 8).



Рисунок 8. Ввод данных в форму и ответ сервера

4. Добавить в форму (файл `index.html`) группу из двух переключателей (`gender`) для выбора пола (рисунок 9а):

```
<body>
  <form action="{{ url_for('hello') }}" method="get">
    <p>Имя: <input type = text name=username></p>
    <p>Пол:<br>
      <input type = radio name=gender value=w> Ж <br>
      <input type = radio name=gender value=m> М
```

```

        </p>
        <p><input type=submit value=Отправить></p>
    </form>
</body>

```

5. Добавить обработку переключателей в `hello.py`, передать в шаблон параметры `name`, `genre`:

```

def hello():
    # для каждого передаваемого параметра формы нужно установить
    # значение по умолчанию, на случай если пользователь ничего не введет
    name = ""
    gender = ""

    name = request.values.get('username')
    gender = request.values.get('gender')

    # генерируем страницу, передаем в шаблон введенные
    # пользователем параметры
    html = render_template('hello.html',
                           name = name,
                           gender = gender
                           )

    return html

```

6. Исправить шаблон `hello.html`, если пользователь выбрал женский пол, то вывести обращение «студентка», если выбран мужской пол – вывести обращение «студент»:

```

<body>
<p>
    Привет,
    {% if gender == "m" %}
        студент
    {% elif gender == "w" %}
        студентка
    {% endif %}
    {{name}}!
</p>
</body>

```

В результате, после того, как будет введено имя в поле, выбран пол и нажата кнопка **Отправить**, на странице отобразится новое приветствие пользователю (рисунок 9).

а)
 Пол:
☒ Ж
☐ М

б)

Привет, студентка Алина!

Рисунок 9. Ввод данных в форму и ответ сервера

7. В папке `Lab_6` создать программу `constants.py`, в которой будут храниться списки с информацией, необходимой для нашего приложения:

```

# список образовательных программ

```

```

programs = ["Экономика", "Бизнес-информатика", "Туризм"]
# список дисциплин
subjects = ["Эконометрика", "Базы данных", "Теория игр",
            "Программирование на Python"]

```

8. В программу `index.py` добавить передачу параметра со списком программ в шаблон, также передать функцию для вычисления длины списка:

```

html = render_template(
    'index.html',
    program_list = constants.programs,
    len = len
)

```

9. В форму (файл `index.html`) добавить поле со списком (`program`) для выбора образовательной программы, на которой учится студент (рисунок 9а). Список будет формироваться на основе параметра `program_list`, атрибут `value` установим равным индексу образовательной программы в списке `program_list`.

```

<body>
  <form action="{{ url_for('hello') }}" method="get">
    ...
  </p>
  Образовательная программа:
  <select name=program>
    {% for i in range(len(program_list)) %}
      <option value={{i}}>{{program_list[i]}}</option>
    {% endfor %}
  </select>
</p>

  <p><input type=submit value=Отправить></p>
</form>
</body>

```

10. Добавить обработку поля со списком в `hello.py`. Передать в шаблон параметр с названием выбранной образовательной программы, а также список со всеми образовательными программами и функцию для вычисления длины списка. Указать, что список берется из файла `constants.py`:

```

import constants

from app import app
from flask import render_template, request

@app.route('/hello', methods=['GET'])
def hello():
    # для каждого передаваемого параметра формы нужно задать
    # значение по умолчанию, на случай если пользователь ничего не введет
    name = ""
    gender = ""
    program_id = 0

    name = request.values.get('username')
    gender = request.values.get('gender')
    program_id = request.values.get('program')

    html = render_template(
        'hello.html',
        name = name,
        gender = gender,

```

```

        program = constants.programs[int(program_id)],
        program_list = constants.programs,
        len = len
    )
    return html

```

11. Расширить шаблон `hello.html`, вывести программу, которую выбрал пользователь:

```

<body>
...
<p>
    Ваша образовательная программа: {{program}}.
</p>
</body>

```

В результате, после того, как будет введено имя в поле, выбран пол, выбрана образовательная программа и нажата кнопка **Отправить**, на странице отобразится новое приветствие пользователю (рисунок 10).

a)

Имя:
 Пол:
☒ Ж
☐ М
 Образовательная программа:

б)

Привет, студентка Алина!
 Ваша образовательная программа: Бизнес-информатика.

Рисунок 10. Ввод данных в форму и ответ сервера

12. В программу `index.py` добавить передачу параметра со списком дисциплин в шаблон:

```

html = render_template(
    'index.html',
    program_list = constants.programs,
    subject_list = constants.subjects,
    len = len
)

```

13. В форму (файл `index.html`) добавить группу переключателей `checkbox` (имя `subject[]`) для выбора нескольких дисциплин, которые изучает студент (рисунок 11 а). Переключатели будут формироваться на основе параметра `subject_list`, атрибут `value` установим равным индексу дисциплины в списке.

```

<body>
<form action="{{ url_for('hello') }}" method="get">
...
<p>
    Выберите дисциплины, которые Вы изучаете:<br>
    {% for i in range(len(subject_list)) %}
        <input type = checkbox name=subject[] value={{i}}>
        {{subject_list[i]}}<br>
    
```



```

        {% endfor %}
    </p>
    <p><input type=submit value=Отправить></p>
</form>
</body>

```

14. Добавить обработку группы переключателей в `hello.py`, передать в шаблон соответствующий параметр, а также список с дисциплинами:

```

def hello():
    # для каждого передаваемого параметра формы нужно задать
    # значение по умолчанию, на случай если пользователь ничего не введет
    name = ""
    gender = ""
    program_id = 0
    # список из номеров выбранных пользователем дисциплин
    subject_id = []
    # список из выбранных пользователем дисциплин
    subjects_select = []

    name = request.values.get('username')
    gender = request.values.get('gender')
    program_id = request.values.get('program')
    subject_id = request.values.getlist('subject[]')
    # формируем список из выбранных пользователем дисциплин
    subjects_select = [constants.subjects[int(i)] for i in subject_id]

    html = render_template(
        'hello.html',
        name = name,
        gender = gender,
        program = constants.programs[int(program_id)],
        program_list = constants.programs,
        len = len,
        subjects_select = subjects_select,
        subject_list = constants.subjects
    )
    return html

```

15. Исправить шаблон `hello.html`, вывести дисциплины, которые выбрал пользователь:

```

<body>
...
<p>
    Вы изучаете дисциплины:
    {% for sub in subjects_select %}
        {{sub}},
    {% endfor %}
</p>
</body>

```

В результате, после того, как будет введено имя в поле, выбран пол, выбрана образовательная программа, отмечены изучаемые дисциплины и нажата кнопка **Отправить**, на странице отобразится приветствие пользователю (рисунок 11).

Имя:

Пол:
☒ Ж
☐ М

Образовательная программа:

Выберите дисциплины, которые Вы изучаете:

☐ Эконометрика
☒ Базы данных
☒ Теория игр
☒ Программирование на Python

а)

Привет, студентка Алина!

Ваша образовательная программа: Бизнес-информатика.

Вы изучаете дисциплины: Базы данных, Теория игр, Программирование на Python.

б)

Рисунок 11. Ввод данных в форму и ответ сервера

Самостоятельные задания

1. Исправить шаблон приложения `hello.html` так, чтобы:

- если пользователь не выбрал имя, вывести «Введите имя, пожалуйста!» (рисунок 11а);
- если пользователь не выбрал дисциплины - вывести: «Вы не изучаете никаких дисциплин» (рисунок 12б);
- если дисциплины выбраны – после последней дисциплины поставить точку (рисунок 11в).

а)

Введите имя, пожалуйста!

б)

Привет, Алина!

Ваша образовательная программа: Экономика.

Вы не изучаете никаких дисциплин.

в)

Привет, студентка Алина!

Ваша образовательная программа: Бизнес-информатика.

Вы изучаете дисциплины: Базы данных, Теория игр, Программирование на Python.

Рисунок 12. Варианты ответа сервера

2. Добавить в форму еще одну позицию:

«Отметьте олимпиады и конкурсы, в которых Вы участвовали:»

Реализовать возможность выбрать несколько мероприятий.

В приветствии перечислить конкурсы и олимпиады, в которых участвовал студент.

Если нигде не участвовал – вывести соответствующее сообщение.

Задание 3. Добавить новые возможности в приложение «Приветствие». На странице с приветствием реализовать возможность перейти по ссылке на новую страницу для каждой дисциплины и просмотреть информацию об этой дисциплине (рисунок 13а). На новой странице показать картинку и краткое описание дисциплины. После клика на «Вернуться», должна загрузиться исходная форма с полями для заполнения (рисунок 13б).

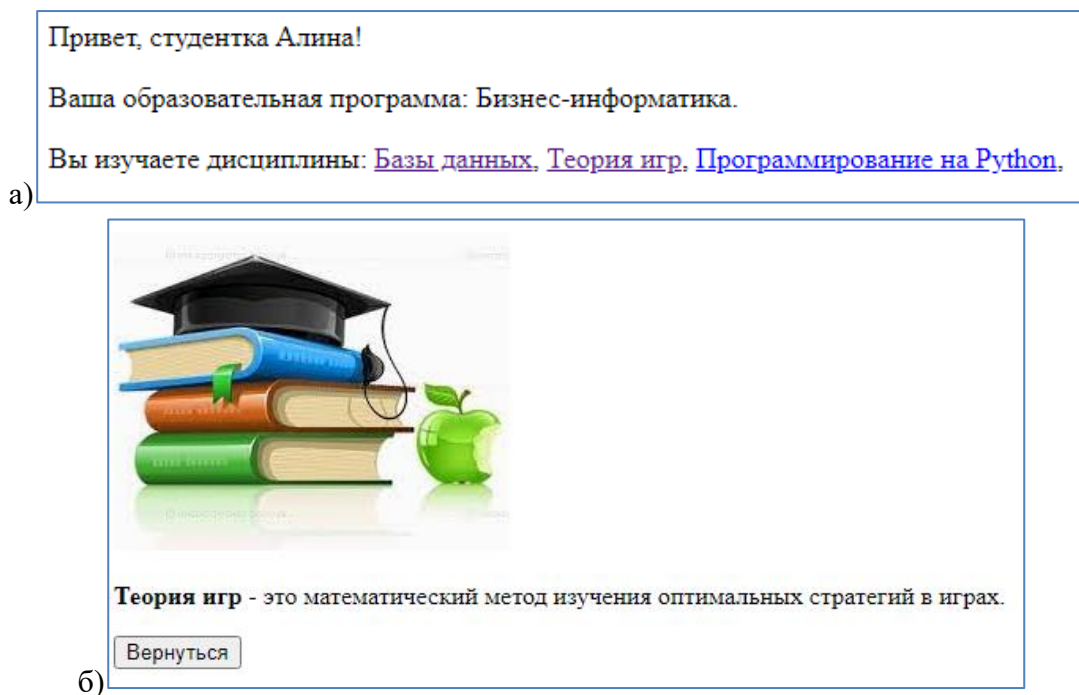


Рисунок 13. Страницы со списком дисциплин и описанием дисциплины

Порядок выполнения работы

1. В папке `static` создать папку `images`. В папке `images` разместить произвольный рисунок, который должен называться `book.jpg`.

2. В файл `constants.py` добавить словарь с описанием дисциплин:

```
# словарь с описанием дисциплины
subject_dict = {
    "Эконометрика": "наука, изучающая количественные и качественные экономические взаимосвязи с помощью статистических и других математических методов и моделей.",
    "Базы данных": "упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе.",
    "Теория игр": "математический метод изучения оптимальных стратегий в играх.",
    "Программирование на Python": "высокоуровневый язык программирования общего назначения, который используется в том числе и для разработки веб-приложений."
}
```

3. В папке `controllers` создать новый файл `subject.py`, в котором генерируется страница с информацией о дисциплине. В качестве параметра в соответствующую функцию передается название дисциплины (`sub`). Шаблон страницы называется `subject.html`, в качестве параметров в шаблон передается название дисциплины и ее описание.

```
import constants

from app import app
from flask import render_template, request

@app.route('/subject/<sub>')

def subject(sub):
    html = render_template(
        'subject.html',
        sub = sub,
        discription = constants.subject_dict[sub]
    )
    return html
```

4. В файле `app.py` добавить новый контроллер:

```
import controllers.subject
```

5. В папке `templates` создать шаблон `subject.html`, в который вставить картинку, абзац с описанием дисциплины и кнопку для возврата на предыдущую страницу:

```
<!DOCTYPE HTML>
<html>
  <head>
    <title> Дисциплина</title>
  </head>
  <body>
    
    <p><b>{{sub}}</b> - это {{discription}}</p>
    <p><input name=action onclick="history.back()"
      type=submit value="Вернуться"/>
    </p>
  </body>
</html>
```

6. Исправить шаблон `hello.html`, в котором для каждой дисциплины добавить ссылку `href` на функцию `subject` с параметром название дисциплины `sub`.

```
<p>
  Вы изучаете дисциплины:
  {% for sub in subjects_select %}
    <a href={{ url_for('subject', sub=sub) }}> {{sub}}</a>,
  {% endfor %}
</p>
```

В результате, после того, как будет введено имя в поле, выбран пол, выбрана образовательная программа, отмечены изучаемые дисциплины и нажата кнопка **Отправить**, на странице отобразится приветствие пользователю. Затем можно кликнуть по дисциплине (рисунок 13а) – будет осуществлен переход на новую страницу (рисунок 13б). При нажатии на «Вернуться» должна загрузиться предыдущая страница (рисунок 13а).

Самостоятельное задание

Для «Конкурсов и олимпиад» реализовать просмотр информации о них на отдельной странице. Страница должна загружаться по клику на названии мероприятия.