

# ІНФОРМАТИКА ТА ПРОГРАМУВАННЯ

---

Тема 27 Побудова веб-  
серверів

# Веб-сервери

- **Веб-сервер** – це сервер, що забезпечує обробку запитів клієнтів у веб-мережі.
- Веб-сервер здійснює спілкування з клієнтами згідно протоколу HTTP (або HTTPS).
- Функції серверу включають:
  - Приймання HTTP-запитів від клієнтів
  - Обробку запитів
  - Надання відповідей клієнтам у вигляді HTML-сторінок

# Веб-сервери у Python

- У Python класи, які створюють веб-сервер та забезпечують його роботу, знаходяться у модулі `http.server`.
- Для створення серверу застосовують клас `HTTPServer`, який є нащадком класу `TCPServer`, розглянутого нами у темі «Загальна будова глобальних мереж».
- `HTTPServer`, як і `TCPServer`, при створенні об'єкту приймає два параметри: кортеж (адреса, порт) та клас, який забезпечує обробку HTTP- запитів.
- У модулі `http.server` є три стандартних класи для обробки запитів:
  - `BaseHTTPRequestHandler`
  - `SimpleHTTPRequestHandler`
  - `CGIHTTPRequestHandler`
- Клас `BaseHTTPRequestHandler` не обробляє запитів, а тільки визначає інтерфейс.
- Цей клас містить ряд полів для збереження або зміни інформації про запит.
- Класи `SimpleHTTPRequestHandler` та `CGIHTTPRequestHandler` є його нащадками.

# Веб-сервери у Python.2

- Клас SimpleHTTPRequestHandler обробляє запити клієнта та повертає список файлів з поточного каталогу або один файл (якщо цей файл може прочитати веб-клієнт).
- Клас CGIHTTPRequestHandler обробляє запити клієнта та запускає сценарій (програму) для обробки запиту.
- Два останніх класи містять методи для обробки запитів HEAD, GET, POST.
- Ці методи називаються do\_HEAD(), do\_GET(), do\_POST() та можуть бути перевизначені класами-нащадками.
- Щоб запустити сервер, достатньо його імпортувати та виконати команду:

```
from http.server import *
```

```
HTTPServer((HOST, PORT),  
           SimpleHTTPRequestHandler).serve_forever()
```

# CGI

- **CGI** (Common Gateway Interface або загальний шлюзовий інтерфейс) – це стандартний протокол зв'язку веб-серверів з програмами, які у динаміці будують відповіді на запити клієнтів.
- Ці програми також називають **CGI-сценаріями**.
- Усі сучасні веб-сервери підтримують стандарт CGI.
- Тому відповідні сценарії можуть працювати у різних серверних середовищах.
- Python також є мовою, яка застосовується для побудови CGI-сценаріїв.
- CGI-сценарій отримує запит веб-клієнта, аналізує та обробляє його, а також будує відповідь клієнту, яку передає веб-серверу (а той, у свою чергу, - веб-клієнту).

# Передача параметрів у веб-запитах

- CGI-сценарії, що будують сторінки відповіді у динаміці, базують ці відповіді на параметрах, отриманих від клієнта у запитах.
- Спосіб передачі цих параметрів залежить від методу запиту (GET або POST).
- У запитах GET параметри передаються як частина URL, а у запитах POST – окремо.
- Частіше за все, для передачі параметрів використовують поля HTML-форм.

# HTML-форми

- Ми вже розглядали HTML-форми у темі «Побудова веб-клієнтів». Зараз розглянемо їх докладніше.
- Форма починається HTML-тегом `<form>`.
- Окрім текстової частини, яка відображається на сторінці, форма містить поля для введення або редагування даних.
- Такі поля (окрім списків) позначаються тегом `<input>`.
- Цей тег дозволяє розміщувати на сторінці елементи інтерфейсу користувача подібно тому, як це робиться у графічному інтерфейсі.
- Для визначення типу елемента інтерфейсу використовують атрибут `type` тегу `<input>`.
- Значення цього атрибуту наведено у таблиці:

# HTML-форми.2

Тип	Опис
<b>button</b>	Кнопка.
<b>checkbox</b>	Кнопки вибору. Дозволяють вибрати один з двох варіантів.
<b>file</b>	Поле для введення імені файлу, який пересилається на сервер.
<b>hidden</b>	Приховане поле. Воно ніяк не відображується на веб-сторінці.
<b>image</b>	Поле з зображенням. При натисненні на рисунок дані форми відправляються на сервер.
<b>password</b>	Поле для введення паролю. Усі символи зображуються зірочками.
<b>radio</b>	Радіокнопки. Використовуються, коли треба вибрати один варіант з декількох.
<b>reset</b>	Кнопка для повернення даних форми у початкове значення.
<b>submit</b>	Кнопка для відправки даних форми на сервер.
<b>text</b>	Текстове поле. Призначено для введення символів за допомогою клавіатури.



# HTML-форми.3

- Списки позначаються тегом `<select>`, а окремі елементи списків, - тегом `<option>`
- Атрибут `name` визначає ім'я поля, а атрибут `value`, - значення поля.
- Для текстових полів `value` – це початкове значення, а для кнопок вибору, радіокнопок та елементів списку – значення, яке буде передано серверу при виборі відповідної кнопки (елементу списку).
- Група кнопок вибору (`checkbox`) мають однакове ім'я (`name`) та повертають список вибраних кнопок (значень).
- Метод запиту для форми (`GET` або `POST`) визначається атрибутом `method` тегу `<form>`.
- Атрибут `action` задає CGI-сценарій, який буде використано сервером для обробки форми.
- Значення цього атрибуту буде додано до URL під час передачі запиту від клієнта до сервера.

# HTML-форми.4

- Приклад HTML-форми:

```
<form method=POST action="t27_23_quiz_login.py">
```

```
<p>
```

```
    Логін <input type=text name=login value="">
```

```
</p>
```

```
<p>
```

```
    Пароль <input type=password name=pass value="">
```

```
</p>
```

```
<p>
```

```
    <input type=submit value="Увійти">
```

```
</p>
```

```
</form>
```

# Обробка даних HTML-форм у Python

- Для обробки даних, що надходять у запитах від HTML-форм, у Python використовують модуль `cgi`.
- У цьому модулі, зокрема є клас `FieldStorage`, який надає можливість прочитати значення, що передані з HTML-форми у елементах інтерфейсу.
- Для створення об'єкту цього класу у CGI-сценарії, який запускається для обробки даних форми, треба викликати конструктор:

**`form = cgi.FieldStorage()`**

- Після цього до `form` можна звертатись як до словника, вказуючи в якості ключа значення атрибуту `name` HTML-форми. Наприклад,

**`login = form["login"].value`**

повертає значення поля "login".

# Обробка даних HTML-форм у Python.2

- Інший спосіб отримати значення полів форми – це виклик методів `getfirst` або `getlist` класу `FieldStorage`.
- Цей спосіб є більш надійним, оскільки значенням поля форми може бути не тільки рядок, але й список значень (наприклад, у випадку групи кнопок вибору).
- Виклик

**`login = form.getfirst("login", "")`**

повертає значення поля "login".

- При цьому, якщо значення є списком, то повертає перший елемент цього списку.
- Якщо ж такого поля у формі немає, то повертає порожній рядок "".
- Виклик

**`lst = form.getlist("username")`**

повертає список значень поля "username".

- Якщо ж такого поля у формі немає, то повертає порожній список.

# Формування відповідей сервера у Python

- CGI-сценарії формують HTML-сторінки для відправки відповіді клієнту.
- Ці сторінки можуть повністю будуватись у Python.
- Інший шлях – підготувати заготовки у «майже готових» HTML-файлах, прочитати їх та вставити потрібні дані у визначені місця.
- Побудована сторінка – це довгий рядок, розділений символами ‘\n’.
- Щоб відправити сторінку на сервер, використовують добре відому стандартну функцію print.
- У CGI-сценарії print не виводить дані на екран, а відправляє їх веб-серверу, який, у свою чергу, передає сторінку клієнту.

# Приклад: CGI-сценарій обчислення заданого числа Фібоначчі (Версії 1 та 2)

- Скласти CGI-сценарій обчислення заданого числа Фібоначчі.
- Програма використовує простий локальний сервер, який для обробки запитів використовує клас `CGIHTTPRequestHandler`.
- Сервер налаштовано на порт 8000.
- У версії 1 програми у браузері треба відкрити HTML-файл `fib.html`.
- У ньому вказано форму та сценарій обробки `t27_11_fib_web_v1.py`, який отримує введений номер та повертає результат.
- Версія 2 програми буде HTML-сторінку не тільки для введення даних, але й для показу результату.
- Таким чином, можна багаторазово виконувати обчислення.
- Для запуску сценарію з використанням локального веб-сервера, який приєднується до порту 8000, треба у вікні браузера набрати `http://localhost:8000/cgi-bin/t27_11_fib_web_v2.py`.

# Мінімальні URL

- Для вказання сценарію, що буде обробляти HTML-форму, у атрибуті action тегу <form> можна використовувати абсолютний шлях до відповідного файлу сценарію або відносний шлях.
- Відносний шлях інтерпретується веб-браузером як шлях, починаючи з поточного каталогу, а поточним вважається каталог, який був відкритий востаннє.
- Так, замість
- [http://localhost:8000/cgi-bin/t27\\_11\\_fib\\_web\\_v3.py](http://localhost:8000/cgi-bin/t27_11_fib_web_v3.py)
- можна використовувати
- [t27\\_11\\_fib\\_web\\_v3.py](#)
- Мінімальні URL дають змогу переносити програмний код та дані на інші веб-сервери без зміни самих програм.

# Табличне розташування елементів HTML-сторінки

- Табличне розташування елементів у HTML-сторінках застосовують для вирівнювання цих елементів.
- У цьому табличне розташування дещо нагадує менеджер розміщення Grid з tkinter, який ми розглядали у темі «Графічний інтерфейс».
- Щоб створити таблицю, використовують тег `<table>`.
- Для позначення нового рядка таблиці, - тег `<tr>` (без кінцевого тегу), а для нової клітинки у рядку, - тег `<td>`.
- Атрибут `align` тегів `<table>` та `<td>` вказує вирівнювання відповідно всієї таблиці та тексту у клітинці.
- Можливі значення цього атрибуту: `left` (ліворуч), `center` (по центру), `right` (праворуч), `justify` (ліворуч та праворуч).
- Атрибут `colspan` тегу `<td>` вказує об'єднання декількох клітинок таблиці в одну. Наприклад, `colspan=2`.



## Приклад: CGI-сценарій обчислення заданого числа Фібоначчі (Версія 3)

- Скласти CGI-сценарій обчислення заданого числа Фібоначчі.
- Програма використовує простий локальний сервер, який для обробки запитів використовує клас `CGIHTTPRequestHandler`.
- Сервер налаштовано на порт 8000.
- Версія 3 програми будує HTML-сторінку не тільки для введення даних, але й для показу результату.
- Також у цій версії використовуються мінімальні URL та табличне розташування елементів.

# HTTP – протокол без збереження стану

- Кажуть, що HTTP – це протокол без збереження стану (stateless).
- Це означає, що кожен запит веб-клієнта трактується сервером як новий, у якому нічого не відомо про можливі попередні запити цього ж клієнта.
- Така побудова протоколу HTTP робить його простішим та більш надійним, але має свої «мінуси».
- Зокрема, при обробці запитів CGI-сценаріями неможливо зберегти, наприклад, значення глобальних змінних між окремими запитами, хоча часто це є необхідним.

# Передача інформації між запитами у НТТР

- Для передачі інформації між запитами у НТТР використовують декілька підходів:
  - Передача у параметрах адресного рядка запиту
  - Передача у прихованих полях форм
  - Передача шляхом використання “куків”
  - Використання серверних баз даних
- Усі підходи базуються на тому, що CGI-сценарій у процесі динамічної побудови веб-сторінки включає до неї додаткову інформацію, яка потім повертається браузером у наступному запиті.
- Передача у параметрах адресного рядка запиту полягає у тому, що до цього рядка включаються параметри, які мають спеціальні значення.
- Наприклад, у сторінці може бути посилання, яке використовує тег `<a>`. У це посилання додають необхідні параметри.
- Наприклад,

```
<a href=question.html?user=user1>
```

# Відправка даних для запиту POST.2

- Передача у прихованих полях форм полягає у включенні до HTML-форм прихованих полів, що мають тип hidden.
- Значення цих полів і містять необхідну інформацію. Наприклад,  
`<input type=hidden name=user value="user1">`
- «Куки» (cookies – печеньки) – це текстові дані, які браузер може записати на диск клієнтського комп'ютера.
- Передача шляхом використання «куків» полягає у записі цих даних протягом одного запиту та їх читанні під час підготовки іншого запиту.
- Використання серверних баз даних дозволяє зберігати дані від одного запиту до іншого.
- У цій моделі CGI-сценарій звертається до бази даних та отримує значення необхідних параметрів.
- Якщо значення змінюються, то сценарій записує змінені значення до бази даних.

# Приклад: Проходження тестів

- Скласти програму, яка підтримує проходження тестів у веб-браузері.
- Тести записано у файлі MS Excel.
- Програма повинна забезпечити введення імені та паролю користувача (також зберігаються у файлі MS Excel), вибір теми тесту, передачу питань, отримання та аналіз відповідей, а також показ результату тесту (кількості балів).
- Тест може містити питання трьох типів:
  - Так або ні
  - З вибором одного варіанту відповіді
  - З вибором декількох варіантів відповіді
- За кожне питання, на яке дано правильну відповідь, нараховується визначена кількість балів.
- Для реалізації опишемо CGI-сценарії та класи для читання та обробки тестів.

# Проходження тестів. Файл з тестами

- Фрагмент файлу MS Excel, що містить тести, вказано нижче:

Ітератори та генератори		
	Ітератором у називається	Onlyone 2
		об'єкт, який здатен повертати всі елементи послідовності x тільки у порядку їх слідування
		об'єкт, який здатен повертати по черзі всі елементи x у деякому порядку та фіксувати момент завершення елементів x True

## Проходження тестів. Файл з тестами.2

- У першому рядку та у першому стовпчику вказано тему тесту.
- Починаючи з другого рядка та у другому стовпчику містяться питання.
- Для кожного питання у третьому стовпчику вказано його тип, а у четвертому, - кількість балів.
- Для кожного питання вказують також варіанти відповідей.
- Вони містяться у третьому стовпчику, починаючи з наступного рядка за рядком з питанням.
- Біля правильних відповідей у четвертому стовпчику вказують True.

# Проходження тестів. Класи обробки тестів. Клас QuizSuite

- Для обробки тестів опишемо класи для набору тестів, одного тесту, одного питання.
- Для відповідей та результатів використаємо іменовані кортежі Answer та Result.
- Клас QuizSuite виконує дії над набором тестів. Клас містить поля:
  - self.IO - об'єкт класу введення тестів та виведення результатів
  - self.quizzes - список тестів (об'єктів класу Quiz)
  - self.results - список результатів (кортежів Result)
- Клас QuizSuite також включає методи:
  - \_\_init\_\_ - конструктор
  - \_\_str\_\_ - представлення об'єкту класу у вигляді рядка
  - getthemes – отримати список тем тестів
  - getusers – отримати список користувачів та їх паролів
  - getquiz – отримати тест за темою
  - writeresult – зберегти результат проходження тесту



# Проходження тестів. Клас Quiz

- Клас Quiz реалізує один тест. Клас містить поля:
  - `self.master` - набір тестів (об'єкт)
  - `self.text` - назва тесту
  - `self.questions` - список запитань тесту (об'єктів класу Question)
- Методи класу:
  - `__init__` - конструктор
  - `__str__` - представлення об'єкту класу у вигляді рядка
  - `assess` – оцінити пройдений тест
  - `writeresult` – зберегти результат проходження тесту

# Проходження тестів. Клас Question

- Клас Question реалізує одне питання тесту.. Клас містить поля:
  - `self.master` - тест (об'єкт)
  - `self.text` - текст питання
  - `self.type` - тип питання (так/ні - YESNO, з одним варіантом - ONLYONE, з декількома варіантами - SEVERAL)
  - `self.points` - кількість балів за питання
  - `self.answers` - список запитань тесту (кортежів Answer)
- Методи класу:
  - `__init__` - конструктор
  - `__str__` - представлення об'єкту класу у вигляді рядка
  - `assess` – оцінити одне питання тесту

# Проходження тестів. Абстрактний клас TestIO

- Абстрактний клас TestIO грає роль кореневого класу для визначення інтерфейсу читання/запису наборів тестів.
- У даному прикладі набір тестів записано у файлі MS Excel, але можуть бути й інші формати запису тестів.
- Клас TestIO містить поля:
  - `self.quissuite` - тести (об'єкт класу QuisSuite)
  - `self.urn` - розташування ресурсу з тестами (файл або база даних)
  - `self.params` - додаткові параметри для читання тестів
  - `self.users` - список користувачів - кортежів (користувач, пароль)
  - `self.results` = список результатів - кортежів Result
- Клас має конструктор, що присвоює початкові значення полям, а також визначає абстрактні методи `read` (читати набір тестів), `writeresult` (зберегти результати) та властивість `users` (список користувачів разом з паролями).

# Проходження тестів Клас TestExcelIO

- Клас TestExcelIO реалізує інтерфейс читання/запису набору тестів з файлу MS Excel.
- Окрім реалізації абстрактних методів та властивостей класу TestIO, клас також містить поля:
  - self.wb - робоча книга MS Excel з тестами
  - self.resurn - розташування ресурсу з результатами (текстовий файл)
- Клас має внутрішні методи \_readws (прочитати аркуш робочої книги) та \_readquiz (прочитати один тест).
- Основна частина модуля містить програмний код для перевірки правильності читання тестів.

# Проходження тестів. Реалізація

- Для реалізації програми проходження тестів, окрім згаданих класів, використовуються HTML-файли:
  - `quiz_login.html` – файл входу до системи (введення логіну та паролю)
  - `quiz_themes.html` – заготовка для виведення тем тестів
  - `quiz_question.html` – заготовка для виведення чергового запитання та відповідей
  - `quiz_result.html` – заготовка для виведення результатів проходження тесту
- Також використовуються модулі Python (CGI-сценарії):
  - `t27_23_quiz_login` – сценарій входу у систему
  - `t27_24_quiz_question` – сценарій обробки питання
- Для передачі інформації між запитаними слугують приховані поля форм.
- Передається інформація про логін користувача (`user`), тему тесту (`theme`), номер питання (`quest_no`), відповіді на запитання (`reply_str`).
- `reply_str` – це рядок, який містить відповіді на окремі питання, розділені `','`.
- Відповідь на окреме питання – це рядок з нулів та одиниць, розділених комами.
- Сумарна кількість нулів та одиниць дорівнює кількості відповідей.
- Одиниці ставляться для вибраних відповідей.

# Проходження тестів. CGI-сценарій входу у систему

- Для початку проходження тесту треба у браузері відкрити файл `quiz_login.html`.
- Він містить форму для введення логіну та паролю.
- Для аналізу даних форми викликається CGI-сценарій `t27_23_quiz_login.py`.
- Цей сценарій читає набір тестів та аналізує логін та пароль.
- Якщо вони правильні, то повертає сторінку з темами тестів для вибору теми.
- Якщо неправильні, - виводить сторінку з повідомленням про помилку.
- Сценарій також задіює стандартний модуль `cgitb` для обробки помилок під час роботи сценарію.
- Для використання цього модуля треба написати
- **`import cgitb`**
- **`cgitb.enable()`**
- Якщо виникає помилка, то модуль `cgitb` повертає її опис клієнту.
- Цей опис відображається у браузері.

# Проходження тестів. CGI-сценарій обробки питання

- Сценарій `t27_24_quiz_question.py` викликається після вибору теми тесту та після відповіді на кожне запитання.
- У будь-якому випадку сценарій читає набір тестів та аналізує отримані поля форми.
- Після вибору теми сценарій просто показує перше питання разом з варіантами відповідей.
- Після отримання відповіді на питання сценарій формує рядок з відповідями для передачі у наступне питання або оцінювання.
- Якщо питання закінчились, то сценарій оцінює весь тест, зберігає результат а також показує результат проходження тесту клієнту.

# WSGI

- **WSGI** (Web Server Gateway Interface або шлюзовий інтерфейс веб сервера, читається «віски») – це засоби зв'язку веб серверів з програмами у Python.
- Сервер, який підтримує WSGI, при надходженні кожного запиту від клієнта викликає у підключеній до нього Python-програмі функцію `application`.
- Цій функції передається словник з параметрами (`environ`) та ім'я функції, яку треба викликати для передачі клієнту (`start_response`).
- Функція повинна обробити запит та надати у відповідь HTML-сторінку.
- При цьому, результат функції повинен належати типу, що ітерується, а кожний елемент цього типу повинен бути рядком байтів.
- Часто в якості результату `application` вказують список з одного рядка байтів.
- Стандартний шаблон функції `application` виглядає так:



# WSGI.2

```
def application(environ, start_response):  
    # обробити запит, використавши environ  
  
    # нехай сторінка відповіді у рядку body  
  
    # якщо все нормально, надіслати заголовки  
    status = '200 OK'  
    response_headers = [('Content-type','text/plain')]  
    start_response(status, response_headers)  
  
    # інакше надіслати повідомлення про помилку  
  
    # повернути результат  
    return [bytes(body, encoding='utf-8')]
```

# WSGI-сервер

- Для використання WSGI цей інтерфейс повинен підтримувати веб сервер.
- На щастя, більшість сучасних веб серверів мають підтримку WSGI.
- У Python також є веб сервер з підтримкою WSGI.
- Він міститься у модулі `wsgiref.simple_server`.
- Для запуску сервера треба виконати, наприклад, такі рядки програми:

```
from wsgiref.simple_server import make_server  
httpd = make_server('localhost', 8051, application)  
httpd.serve_forever()
```

- Це означає запуск сервера на локальному комп'ютері, порт 8051.

# Обробка параметрів запиту у WSGI

- У WSGI нескладно обробити параметри, які передаються у словнику `environ`.
- Взагалі, `environ` містить багато параметрів, які можна побачити, якщо повернути рядки `environ` в якості результату запиту до сервера.
- Для обробки результатів нас будуть цікавити параметри з іменами “`PATH_INFO`”, “`REQUEST_METHOD`” та “`wsgi.input`”.
- Параметр “`PATH_INFO`” – це частина URL, що містить шлях до ресурсу на сервері (або сценарій обробки HTML-форми).
- Параметр “`REQUEST_METHOD`” повертає метод запиту: GET, POST або інший.
- Параметр “`wsgi.input`” містить дані полів форми (запит POST) або параметрів рядка URL (запит GET).
- Його можна розібрати на складові частини за допомогою вже відомого класу `FieldStorage` з модуля `cgi`. Виклик

**`form = cgi.FieldStorage(fp=environ['wsgi.input'], environ=environ)`**

повертає об'єкт `form`, який можна в подальшому обробляти так, як ми це розглядали у випадку CGI-сценаріїв.

# Приклад: Обчислення заданого числа Фібоначчі у WSGI

- Скласти програму обчислення заданого числа Фібоначчі з використанням WSGI.
- Програма використовує локальний WSGI-сервер, який налаштовано на порт 8051.
- Програма містить функцію `application`, яка будує HTML-сторінку не тільки для введення даних, але й для показу результату.
- Цю ж програму можна використати у зв'язці з будь-яким іншим WSGI-сервером без зміни програмного коду.

# Використання класів та збереження стану у WSGI

- Замість функції `application` можна використовувати клас.
- Цей клас може мати довільне ім'я, але повинен реалізовувати спеціальний метод `__call__` з параметрами `environ` та `start_response`.
- Метод `__call__` має робити те ж саме, що й функція `application`.
- При створенні об'єкту цього класу слід присвоїти створений об'єкт змінній з ім'ям `application`.
- Тоді при виклику `application` буде викликатись метод `__call__` описаного класу.
- Використання класів дозволяє зберігати певні дані між викликам `application` (наприклад, у полях класу), тобто між запитам клієнта.
- Таким чином, застосування WSGI дає можливість збереження стану поточного з'єднання з клієнтом.
- Це з'єднання ще називають сеансом (`session`).

# Приклад: Проходження тестів у WSGI

- Скласти програму, яка підтримує проходження тестів у веб-браузері.
- Тести записано у файлі MS Excel.
- Програма повинна забезпечити введення імені та паролю користувача (також зберігаються у файлі MS Excel), вибір теми тесту, передачу питань, отримання та аналіз відповідей, а також показ результату тесту (кількості балів).
- На цей раз використаємо WSGI для розв'язання даної задачі.
- Опишемо класи QuizApplication та QuizSession.

# Клас QuizApplication

- Клас QuizApplication призначено для організації проходження тесту. Клас має поля:
  - `self.path` - шлях до даного модуля від модуля, який його імпортує
  - `self.last_id` - номер останньої започаткованої сесії
  - `self.sessions` - словник сесій (об'єктів класу QuizSession)
  - `self.suite` - набір тестів
  - `self.commands` - словник команд з HTML-файлів та функцій їх обробки
- Клас також містить конструктор та методи `start`, `login`, `theme`, `question`.
- Ці методи обробляють рядки, які отримують у URL від клієнта.
- Ці рядки ми будемо інтерпретувати як команди та викликати відповідний метод обробки (`start` – почати роботу, `login` – аутентифікувати користувача, `theme` – вибрати тему тесту, `question` – сформулювати чергове питання тесту).
- Метод `__call__` реалізує інтерфейс WSGI.
- Внутрішні методи `_check_user`, `_show_themes`, `_show_error`, `_show_question`, `show_result`, `_get_reply` практично не відрізняються від відповідних функцій, які ми розглядали у варіанті з використанням CGI.

# Клас QuizSession

- Клас QuizSession призначено для фіксації та зміни стану одного сеансу роботи одного користувача.
- Клас містить поля:
  - self.app - клас, що містить даний (QuizApplication)
  - self.sid - номер сесії
  - self.user - користувач, який проходить тест
  - self.replies - список списків наданих відповідей
  - self.quest\_no - номер питання
  - self.theme - тема тесту
  - self.quiz – тест
- Окрім полів, клас містить тільки конструктор



# Проходження тестів у WSGI. Реалізація

- Окрім згаданих класів QuizApplication та QuizSession, реалізація прикладу у WSGI включає HTML-файли:
  - w\_quiz\_login.html
  - w\_quiz\_themes.html
  - w\_quiz\_question.html
  - quiz\_result.html
- Файли, що починаються з “w\_” відрізняються від відповідних файлів, розглянутих у прикладі для CGI, тим, що у формах тільки одне приховане поле --sid (номер сеансу) – та замість імен сценаріїв у атрибуті action вказано рядки, які інтерпретуються у QuizApplication як команди.
- Головний модуль створює об’єкт application та запускає сервер.
- Для початку проходження тестів слід запустити сервер та набрати адресу сервера у браузері. Для локального сервера це – `http://localhost:8051`

# Резюме

- Ми розглянули:
  1. Веб-сервери у Python
  2. CGI
  3. Передача параметрів у веб-запитах
  4. HTML-форми
  5. Обробка даних HTML-форм у Python
  6. Формування відповідей сервера у Python
  7. Табличне розташування елементів HTML-сторінки
  8. HTTP – протокол без збереження стану
  9. Передача інформації між запитом у HTTP
  10. WSGI
  11. Обробка параметрів запиту у WSGI
  12. Використання класів та збереження стану у WSGI

# Де прочитати

1. Peter Norton, Alex Samuel, David Aitel та інші - Beginning Python
2. Magnus Lie Hetland - Beginning Python from Novice to Professional, 2nd ed – 2008
3. Mark Lutz - Programming Python. 4<sup>th</sup> Edition - 2011
4. Прохоренко Н.А. - Python 3 и PyQt. Разработка приложений – 2012
5. Mark Pilgrim - Dive into Python, Version 5.4 - 2004
6. Jim Knowlton - Python Create Modify Reuse – 2008
7. John Goerzen -Foundations of Python Network Programming. – 2004
8. <http://wsgi.tutorial.codepoint.net/>
9. <http://archimedeanco.com/wsgi-tutorial/>
10. <http://citforum.ru/programming/python/wsgi/>
11. [http://htmlbook.name/index/uchebnik\\_html/0-4](http://htmlbook.name/index/uchebnik_html/0-4)
12. [https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D1%8B\\_HTML](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D1%8B_HTML)
13. <http://html5book.ru/html-tags/>