**Модуль 4: Основи Web**

**Заняття 1: Основи Web. Робота мережі**

Робота мережі полягає у передачі даних від одного комп'ютера до іншого. У цьому процесі можна виділити декілька окремих завдань:

* розпізнати дані;
* розбити дані на керовані блоки;
* додати інформацію до кожного блоку, щоб;
* вказати місце знаходження даних;
* вказати одержувача;
* додати інформацію синхронізації та інформацію для перевірки помилок;
* помістити дані в мережу та надіслати їх за заданою адресою.

Мережева операційна система при виконанні всіх завдань дотримується суворого набору процедур. Ці процедури називаються протоколами або правилами поведінки. Протоколи регламентують кожну мережеву операцію.

Стандартні протоколи дозволяють програмному та апаратному забезпеченню різних виробників нормально взаємодіяти. Існує стандарт – модель OSI.

**Модель OSI**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-04/module-04-01/intro#%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C-osi)У 1978 році International Standards Organization (ISO) випустила набір специфікацій, що описують архітектуру мережі з неоднорідними пристроями.

Вихідний документ стосувався відкритих систем, щоб усі вони могли використовувати однакові протоколи та стандарти для обміну інформацією.

В 1984 ISO випустила нову версію своєї моделі, названу еталонною моделлю взаємодії відкритих систем (Open System Interconnection reference model, OSI).

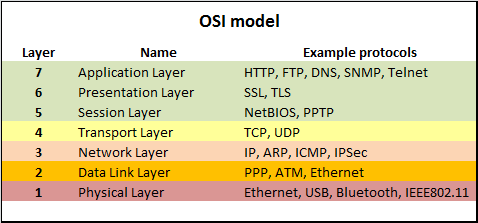
Версія 1984 стала міжнародним стандартом: її специфікації використовують виробники при розробці мережевих продуктів, вона є основою побудови різних мереж.

Ця модель — поширений метод опису мережевих середовищ. Будучи багаторівневою системою, вона відображає взаємодію програмного та апаратного забезпечення при здійсненні сеансу зв'язку, а також допомагає розв'язувати різноманітні проблеми.

**Багаторівнева архітектура**[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-04/module-04-01/intro#%D0%B1%D0%B0%D0%B3%D0%B0%D1%82%D0%BE%D1%80%D1%96%D0%B2%D0%BD%D0%B5%D0%B2%D0%B0-%D0%B0%D1%80%D1%85%D1%96%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)

У моделі OSI мережеві функції розподілені між сімома рівнями. Кожному рівню відповідають різні мережеві операції, обладнання та протоколи.

Нижче на малюнку представлена багаторівнева архітектура моделі OSI.



На кожному рівні виконуються певні мережеві функції, які взаємодіють із функціями сусідніх рівнів, що знаходяться вище і нижче. Наприклад, Сеансовий рівень повинен взаємодіяти лише з Представницьким та Транспортним рівнем тощо. Усі ці функції детально описані.

Нижні рівні – 1-й та 2-й – визначають фізичне середовище передачі даних та супутні завдання (такі як передача бітів даних через плату мережного адаптера та кабель). Найвищі рівні визначають, яким способом здійснюється доступ застосунків до послуг зв'язку. Що вищий рівень, то складніше завдання він вирішує.

Кожен рівень надає кілька послуг (тобто виконує кілька операцій), які готують дані для доставки мережею на інший комп'ютер. Рівні відокремлюються між собою межами — інтерфейсами. Всі запити від одного рівня до іншого передаються через інтерфейс. Кожен рівень використовує послуги нижчого рівня.

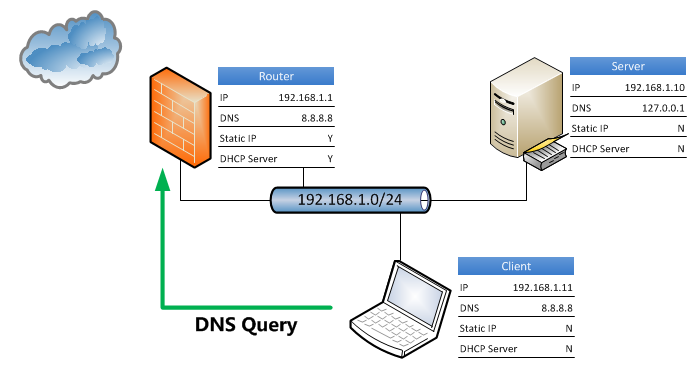
Докладніше про взаємодію рівнів моделі OSI перегляньте [додаткові матеріали](https://textbook.edu.goit.global/python-web-textbook/uk/docs/additional/web/osi) (https://textbook.edu.goit.global/python-web-textbook/uk/docs/additional/web/osi/)

**Як влаштований інтернет**

**Основні поняття**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-04/module-04-01/internet_02#%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D1%96-%D0%BF%D0%BE%D0%BD%D1%8F%D1%82%D1%82%D1%8F)**Хостинг** — послуга з надання обчислювальних потужностей для розміщення інформації на сервері, яка постійно знаходиться в мережі. Хостингом також називається послуга з розміщення обладнання клієнта на території провайдера із забезпеченням підключення його до каналів зв'язку з високою пропускною здатністю. Зазвичай хостинг входить до пакету з обслуговування сайту та вважається як мінімум послуга розміщення файлів сайту на сервері, на якому запущено ПЗ, необхідне для обробки запитів до цих файлів (веб-сервер). Як правило, до обслуговування вже входить надання місця для поштової кореспонденції, баз даних, DNS, файлового сховища на спеціально виділеному файл-сервері тощо, а також підтримка функціонування відповідних сервісів.

**Сервер DNS**— програмно-апаратний комплекс, що забезпечує трансляцію доменних імен на мережеві адреси (IP-адреси).



**Координатор** — уповноважена юридична особа, яка здійснює управління доменами .ua.

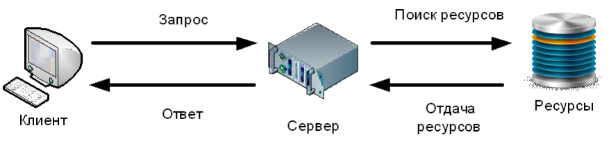
**Доменне ім'я** — символьне позначення, призначене для мережної адресації, в якій використовується система доменних імен.

**Стоп-лист** — перелік символьних позначень, реєстрація яких у якості доменних імен недоступна.

**Реєстратор** — юридична особа, акредитована Координатором для реєстрації доменних імен у доменах .ua

**Протокол HTTP**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-04/module-04-01/internet_02#%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB-http)Сервер та браузер спілкуються, надсилаючи один одному запити за особливим протоколом – HTTP (RFC-2616). З'єднання може ініціювати лише браузер. Він посилає серверу запит - показати такий-то файл.



**HTTP** є синхронним протоколом. Це означає, що клієнт надіслав запит серверу і поки чекає від нього відповідь, наступні запити надіслати не може. Також варто відзначити, що **HTTP** — протокол без стану. Тобто сервер не зберігає інформацію про користувача між запитами.

Протокол HTTP — це протокол передачі даних, спочатку призначений для передачі гіпертекстових документів (тобто документів, які можуть містити посилання, що дозволяють організувати перехід до інших документів).

По суті є дуже простим текстовим протоколом. Заснований на обробці спеціальних символів та ключових слів у повідомленнях.

Для збереження стану користувача в системі використовується механізм **cookie** або сесій.

**Запит відбувається у кілька етапів:**

* *DNS-запит* — пошук найближчого DNS-сервера, щоб перетворити URI (наприклад, google.com) на його числове представлення - IP-адреса (74.125.87.99, прим. - Отримано за допомогою команди ping). Це адреса і буде реальною адресою сайту в Інтернет.
* *з'єднання* — встановлення з'єднання з сервером за отриманою IP-адресою;
* *відправлення даних*;
* *очікування відповіді* - в цей момент чекаємо, доки пакети даних дійдуть до сервера, він їх обробить і відповідь повернеться назад;
* *отримання даних*.

Якщо якийсь елемент веб-сторінки розміщено на іншому хості, для запиту цього елемента встановлюється нове з'єднання, починаючи з DNS-запиту.

Основою HTTP є технологія "клієнт-сервер", тобто передбачається існування:

* Споживачів (клієнтів), які ініціюють з'єднання та надсилають запит;
* Постачальників (серверів), які очікують на з'єднання для отримання запиту, роблять необхідні дії та повертають назад повідомлення з результатом.

Основним об'єктом маніпуляції в HTTP є ресурс, який вказує **URI**(Uniform Resource Identifier) ​​у запиті клієнта. Особливістю протоколу HTTP є можливість вказати в запиті та відповіді спосіб представлення одного і того самого ресурсу за різними параметрами: форматом, кодуванням, мовою тощо (зокрема для цього використовується HTTP-заголовок). Саме завдяки можливості зазначення способу кодування повідомлення, клієнт та сервер можуть обмінюватися двійковими даними, хоча цей протокол є текстовим.

**Що таке URI та URL?**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-04/module-04-01/internet_02#%D1%89%D0%BE-%D1%82%D0%B0%D0%BA%D0%B5-uri-%D1%82%D0%B0-url)Терміни URI та URL часто використовуються як синоніми, але це не зовсім одне й те саме.

**URI**

Являє собою ідентифікатор конкретного ресурсу. Як сторінку, книгу або документ.

**URL**

Представляє особливий тип ідентифікатора, який також говорить вам, як отримати до нього доступ

В принципі URI — це ширше поняття і включає URL. Якщо проводити якісь аналогії спрощення можна вважати, що URI — це щось на кшталт імені, а URL — саме ім'я та як до нього дістатися. Наприклад ISBN книги — це URI, а ось https://goit.ua — це URL, тут є ім'я і як до нього дійти - протокол. Тобто, якщо протокол (https, ftp тощо) або присутній, або мається на увазі для домену, ви повинні називати його URL-адресою, навіть якщо це також URI.

Існує наступна традиційна форма запису `URL:

<scheme>://[<login>[:<password>]@]<host>[:<port>]][/<path>][?<query>][#<fragment>]

Приклад:

http://user:p

assword@host.com:80/resourse/path/?query=name&ttt=123#hash

У цьому записі:

* *scheme* - Це мережевий протокол, за яким відбувається звернення до ресурсу.
* *login* - необов'язковий параметр ім'я користувача, яке використовується для доступу до ресурсу
* *password* - пароль вказаного користувача, якщо він необхідний
* *host* - повністю прописане доменне ім'я хоста в системі DNS (goit.ua) або IP-адреса хоста у формі чотирьох груп десяткових чисел, розділених крапками
* *port* - порт хоста для підключення, згадайте ми часто використовували http://localhost:3000, де 3000 — це і є порт. Виникає закономірне питання, чому порт не вказується, наприклад для URL у браузері. Просто браузер знає, що для протоколу http порт дорівнює 80, а для https він дорівнює 443 і підставляє його за нас
* *path* - уточнююча інформація про місце знаходження ресурсу.
* *query* - рядок запиту з переданими на сервер (методом GET) параметрами. Починається із символу ?, роздільник параметрів – знак &. Приклад: ?foo=123&baz=234&bar=value
* *fragment* - ідентифікатор із попереднім символом #. Часто його називають хеш - hash або якір. Якорем може бути вказаний заголовок всередині документа або атрибут id елемента. За таким посиланням браузер відкриє сторінку та перемістить вікно перегляду вказаного елемента. Наприклад, посилання на секцію контактів у лендінгу: https://example.com/#contact.

**Повідомлення по протоколу HTTP**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-04/module-04-01/internet_02#%D0%BF%D0%BE%D0%B2%D1%96%D0%B4%D0%BE%D0%BC%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F-%D0%BF%D0%BE-%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB%D1%83-http)Повідомлення бувають двох видів:

* запит, звернення від клієнта до сервера;
* відповідь.

Саме повідомлення в HTTP складається із двох частин:

* обов'язкова частина — заголовок;
* опціональна частина — тіло повідомлення.

У заголовку запиту обов'язково повинен міститися тип запиту. Основні типи, що використовуються в мережі:

* **POST** — створення нового ресурсу;
* **GET** — одержання ресурсу;
* **PUT** — зміна ресурсу;
* **PATCH** — часткова зміна ресурсу;
* **UPDATE** — повне оновлення вмісту ресурсу;
* **DELETE**— видалення ресурсу;
* **OPTION** — запит переліку підтримуваних методів.

**Розглянемо приклад заголовків запиту**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-04/module-04-01/internet_02#%D1%80%D0%BE%D0%B7%D0%B3%D0%BB%D1%8F%D0%BD%D0%B5%D0%BC%D0%BE-%D0%BF%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4-%D0%B7%D0%B0%D0%B3%D0%BE%D0%BB%D0%BE%D0%B2%D0%BA%D1%96%D0%B2-%D0%B7%D0%B0%D0%BF%D0%B8%D1%82%D1%83)Пакети даних у запитах та відповідях між клієнтом та сервером містять HTTP-заголовки:

GET /other-19 HTTP/1.1

Host: www.example.com

User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.0; ru; rv:1.9.1.5) Gecko/20091102 Firefox/3.5.5 (.NET CLR 3.5.30729)

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8

Accept-Language: ru,en-us;q=0.7,en;q=0.3

Accept-Encoding: gzip,deflate

Accept-Charset: windows-1251,utf-8;q=0.7,\*;q=0.7

Keep-Alive: 300

Connection: keep-alive

У першому рядку передається інформація про метод передачі даних (в цьому випадку GET), потім йде інформація про те, на який ресурс вони передаються, який необхідно повернути /other-19, і наприкінці версія протоколу HTTP/1.1. На наступному рядку передається інформація про хост, на який передається запит.

Далі у параметрі User-Agent клієнт передає інформацію про браузер. Потім у параметрах Accept, Accept-Language, Accept-Encoding, Accept-Charset передається інформація про кодування, з якою працює браузер, мову за замовчуванням для браузера та метод стиснення інформації. В кінці, у параметрі Connection слідує повідомлення серверу, що потрібно відкрити постійне з'єднання, а в параметрі Keep-Alive вказуємо на скільки (у секундах) ми відкриваємо з'єднання із сервером.

У свою чергу, сервер може надіслати відповідь з наступними заголовками:

HTTP/1.1 200 Ok

Server: nginx

Date: Mon, 29 Sep 2014 12:04:19 GMT

Content-Type: text/html; charset=UTF-8

Connection: close

Cache-Control: no-cache,no-store,max-age=0,must-revalidate

Expires: Mon, 29 Sep 2014 12:04:19 GMT

Last-Modified: Mon, 29 Sep 2014 12:04:19 GMT

Content-Encoding: gzip

Відповідь сервера складається з наступних параметрів:

1. У першому рядку сервер віддає інформацію про версію HTTP протоколу, статус та опис відповіді (усі статуси відповідей сервера, що використовуються, наведені [тут](https://en.wikipedia.org/wiki/List_of_HTTP_status_codes) ().
2. Параметр Server говорить про тип сервера.
3. Параметр Content-Type - типу вмісту, що віддається.
4. Параметри Cache-Control, Expires, Last-Modified - відповідають за кешування відповіді сервера.

**Коди стану відповідей сервера**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-04/module-04-01/internet_02#%D0%BA%D0%BE%D0%B4%D0%B8-%D1%81%D1%82%D0%B0%D0%BD%D1%83-%D0%B2%D1%96%D0%B4%D0%BF%D0%BE%D0%B2%D1%96%D0%B4%D0%B5%D0%B9-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D0%B0)Частина першого рядка відповіді сервера при запитах протоколу HTTP. Він є цілим числом з трьох десяткових цифр. Перша цифра вказує клас стану. За кодом відповіді зазвичай слідує відокремлена пробілом пояснювальна фраза англійською мовою, яка пояснює людині причину саме такої відповіді.

Подивитися, що вони означають можна [тут](https://en.wikipedia.org/wiki/List_of_HTTP_status_codes)

(https://en.wikipedia.org/wiki/List\_of\_HTTP\_status\_codes).

**CAUTION**

Відповідь на запит обов'язково повинен у заголовку повертати статус-код.

Сторінка [офіційної документації](https://docs.python.org/3.10/library/http.html?highlight=http)

(https://docs.python.org/3.10/library/http.html?highlight=http) по протоколу HTTP у Python містить основні статус-коди відповідей та перелік інструментів, за допомогою яких у Python реалізовано підтримку HTTP.

Найчастіше зустрічаються статус-коди:

* **200** — все добре;
* **301**— ресурс знаходиться за новою адресою;
* **404**— ресурс не знайдено;
* **400** — запит містить помилку.

**HTTP-сервер**

У Python для роботи з протоколом HTTP можна скористатися пакетом [http](https://docs.python.org/3.9/library/http.server.html) (https://docs.python.org/3.9/library/http.server.html), який реалізує дуже простий HTTP-сервер. Його не варто використовувати для "бойових" застосунків, але для навчання та розуміння основних механік роботи вебзастосунку його цілком достатньо.

from http.server import HTTPServer, BaseHTTPRequestHandler

class SimpleHTTPRequestHandler(BaseHTTPRequestHandler):

def do\_GET(self):

self.send\_response(200)

self.end\_headers()

self.wfile.write(b'Hello, world!')

def do\_POST(self):

pass

У цьому прикладі ми створили дуже простий HTTP сервер, який приймає запити за адресою / та на GET-запит відповідає рядком **'Hello, world!'**, а на POST-запит нічого не робить.

Клієнтська частина:

from http import client

h1 = client.HTTPConnection('localhost', 8001)

h1.request("GET", "/")

res = h1.getresponse()

print(res.status, res.reason)

data = res.read()

print(data)

Клієнт створюємо за допомогою класу**HTTPConnection** та вказуємо хост (адресу) та порт сервера. Клієнт робить GET-запит на **http://localhost:8001/**, викликаючи метод **getresponse**, та виводить у консоль тіло відповіді. Щоб перевірити, що відповідь містить тіло, ми можемо перевірити статус-код відповіді (**res.status**) та причину помилки (**res.reason**), якщо така була.

Щоб запустити сервер в роботу і не блокувати застосунок, запустимо його в окремому потоці:

from threading import Thread

from time import sleep

from http import client

from http.server import HTTPServer, BaseHTTPRequestHandler

class SimpleHTTPRequestHandler(BaseHTTPRequestHandler):

def do\_GET(self):

self.send\_response(200)

self.end\_headers()

self.wfile.write(b'Hello, world!')

def do\_POST(self):

pass

httpd = HTTPServer(('localhost', 8001), SimpleHTTPRequestHandler)

server = Thread(target=httpd.serve\_forever)

server.start()

sleep(0.5)

h1 = client.HTTPConnection('localhost', 8001)

h1.request("GET", "/")

res = h1.getresponse()

print(res.status, res.reason)

data = res.read()

print(data)

httpd.shutdown()

З неочевидного: ми вказали порт, на якому наш сервер очікує на з'єднання як **8001**. Якщо порт не вказати, то за замовчуванням буде використаний порт **80**, що потребує привілеїв адміністратора. У відповіді буде щось на зразок:

127.0.0.1 - - [04/Dec/2020 16:51:58] "GET / HTTP/1.1" 200 -

200 OK

b'Hello, world!'

**HTML**

**Структура документа**

**HTML** розшифровується як HyperText Markup Language (у перекладі означає Мова Розмітки ГіперТексту).

**HTML-документ** – це по суті звичайний текстовий файл, який містить текст, який нормально читається, і спеціальні команди розмітки (tags або теги), взяті у кутові дужки < і >. Теги мови HTML задають правила, за якими браузер виводить документ на екран: розміщення тексту у вікні, представлення графічних об'єктів (малюнків та зображень), а також виведення звукових, відеокліпів тощо. HTML теги зазвичай використовуються в парі, наприклад: <b></b>. Перший тег називається початковим тегом, а другий кінцевим тегом.

Текст, що знаходиться між початковим і кінцевим тегом, підпадає під "розмітку". Наприклад, <b>Привіт</b> буде показано браузером як **Привіт** (слово 'Привіт' написане жирним шрифтом).

Варто мати на увазі, що не всі теги сумісні з браузерами. Якщо браузер не розуміє тег, то він його просто ігнорує.

Найпростіший документ HTML – це наступна структура:

<!DOCTYPE html >

<html>

<head>

<title>

Назва

</title>

</head>

<body>

Текст сторінки

</body>

</html>

Файл: index.html

Всі **HTML5**документи повинні починатися з оголошення DOCTYPE.

Попередні версії HTML мали кілька типів DOCTYPE. **HTML5**має лише один:

<!DOCTYPE html>

Це оголошення переводить усі браузери в нормальний режим. Браузери, що не підтримують HTML5 в цьому режимі, будуть інтерпретувати старі теги та ігнорувати нові, які вони не підтримують.

Тег <html> є контейнером, який містить увесь вміст веб-сторінки, включаючи теги <head> та <body>.

Тег <head> призначений для зберігання інших елементів, метою яких є допомогти браузеру в роботі з даними. Також всередині контейнера <head> знаходяться метатеги, які використовуються для зберігання інформації, призначеної для браузерів та пошукових систем. Наприклад, механізми пошукових систем звертаються до метатегів для отримання опису сайту, ключових слів та інших даних.

Елементи, що знаходяться в секції head, не відображаються явно на сторінці і використовуються для службових функцій. У секції head можуть розташовуватися скрипти, інструкції про оформлення сторінки та різна мета інформація про цей**HTML**документ.

Метадані – це інформація про дані, яка знаходяться в**HTML** документі. Приклад метаданих: коротке кодування сторінки, опис змісту, ключові слова, ім'я автора, дата останньої модифікації.

Метадані не відображаються явно на сторінці, але використовуються браузерами та пошуковими системами.

В HTML метадані HTML документів визначаються за допомогою тегу <meta>. Тег <meta> завжди повинен розташовуватися у секції head.

Для того щоб вказати браузеру користувача, яке кодування використовується на цій сторінці, необхідно використовувати атрибут charset тегу meta, наприклад так <meta charset="UTF-8">.

Якщо явно не вказати кодування, браузер під час відображення сторінки буде визначати його автоматично. Якщо кодування, при цьому, буде визначено невірно, користувач побачить сторінку, яка містить безглузді символи, тому кодування обов'язково повинно вказуватися до кожного HTML документа.

Будь-який HTML документ має назву, взяту в теги <title> та </titIe>. За назвою документа HTML браузери можуть знайти інформацію, тому місце для назви завжди визначене - воно знаходиться вгорі екрана та окремо від вмісту документа. Максимальна довжина назви – 40 символів. Порожній заголовок, що не містить жодного символу, включаючи пробіл – не допустимий. Також заборонено включати в код два і більше елемента <title>, він повинен бути лише один.

Елемент title:

* Визначає заголовок вікна браузера;
* Використовується як заголовок сторінки в результатах видачі пошукових систем;
* Використовується як заголовок сторінки під час додавання сайту до обраного.

Тег <body> призначений для зберігання вмісту веб-сторінки, що відображається у вікні браузера. Інформацію, яку потрібно виводити в документі, варто розташовувати саме всередині контейнера body. До такої інформації належить текст, зображення, таблиці, списки та інші елементи. HTML елементом називається комбінація початкового тега, кінцевого та вмісту. Приклад HTML елемента:

<p> Це абзац. </p>

Більшість елементів можуть бути вкладені один в одного (тобто у вмісті одного елемента може знаходиться інший елемент).

HTML не чутливий до регістру, це означає, що тег <b> інтерпретуватиметься браузером так само, як і тег <B>.

Незважаючи на те, що HTML не наполягає на суворому синтаксисі, рекомендую заздалегідь напрацювати певні правила написання коду (наприклад, якщо Ви пишете теги в нижньому регістрі, то робіть так завжди).

HTML елементи можуть мати атрибути. Атрибути задаються в початковому тегу елементів і складаються з імені та значення, наприклад в атрибуті href="https://goit.ua/" href є ім'ям, а https://goit.ua/ – значенням.

**Текстові абзаци**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-04/module-04-01/html_04#%D1%82%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%96-%D0%B0%D0%B1%D0%B7%D0%B0%D1%86%D0%B8)Вивчимо основні теги для логічної розмітки тексту. Використовувати їх можна лише всередині тегу body. Почнемо з найпростішого тегу <p>, за допомогою якого створюються абзаци. За замовчуванням абзаци починаються з нового рядка та мають вертикальні відступи, якими можна управляти за допомогою стилів.

<p> Я текстовий абзац </p>

Для виділення тексту в HTML документах можуть бути використані такі теги: за допомогою тегу <b>, </b> можна встановити жирний шрифт, тег <i>, </i> встановлює похилий шрифт, але краще для оформлення завжди використовувати таблиці стилів CSS.

Допускається також використання тексту верхніх і нижніх індексів відповідно за допомогою тегів <sup>, </sup> і <sub>, </sub>.

За допомогою HTML тегу <pre> можна відобразити передформатований текст. Все, що знаходиться всередині тега <pre> буде відображено точно так, як написано. Браузер не видаляє пробіли та символи перенесення рядка.

Встановлення кольорів у HTML документі здійснюється за допомогою таблиці стилів **CSS.**

Вибір кольору можна здійснювати різними способами: наприклад, зазначенням імені або визначенням **RGB**номера обраного кольору. Підтримуються наступні імена кольорів: AQUA, BLACK, BLUE, FUCHSIA, GRAY, GREEN, LIME, MAROON, NAVY, OLIVE, PURPLE, RED, SILVER, TEAL, WHITE, YELLOW.

Номер кольору RGB задається трьома двоцифровими шістнадцятковими числами, причому кожне число з інтервалу 00 - FF і визначає інтенсивність відповідного кольору. Наприклад, номер кольору #FF0000 відповідає червоному кольору, оскільки має максимальну інтенсивність для червоного, а зелений та блакитний мають значення, рівні нулю. Відповідно, номер #00FF00 кодує зелений колір та номер #0000FF - блакитний.

В HTML5 з'явився новий тег <mark>, який позначає виділений текст. Іноді під час роботи з об'ємними текстами ми використовуємо маркер, щоб виділяти ключові слова, ідеї чи щось інше, на що варто звернути увагу. Таке саме призначення й у тегу <mark>. У сучасних браузерах текст всередині <mark> підсвічується жовтим фоном.

**Заголовки та підзаголовки**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-04/module-04-01/html_04#%D0%B7%D0%B0%D0%B3%D0%BE%D0%BB%D0%BE%D0%B2%D0%BA%D0%B8-%D1%82%D0%B0-%D0%BF%D1%96%D0%B4%D0%B7%D0%B0%D0%B3%D0%BE%D0%BB%D0%BE%D0%B2%D0%BA%D0%B8)Для створення структури великих текстів зазвичай використовуються заголовки. У текстових редакторах можна виділити частину тексту, знайти пункт «Заголовок» потрібного рівня меню, і застосувати його.

Перш ніж залишитися на сторінці і приступити до її прочитання, користувачі, зазвичай, швидко пробігають очима її вмістом, перевіряючи, чи містить вона інформацію, яка їх цікавить. Зазвичай заголовки – це перше (і часто єдине), на що вони звертають увагу, тому неправильне використання заголовків може призвести до втрати відвідувачів сторінки. В першу чергу заголовки повинні коротко і точно описувати вміст, який вони озаглавлюють. Найважливіша інформація сторінки повинна розташовуватися під заголовками більшого розміру, а найменш важлива – під заголовками меншого розміру.

У мові HTML дял виділення заголовків передбачено ціле сімейство тегів: від <h1> до <h6>. Тег <h1> означає найважливіший заголовок (заголовок верхнього рівня), а тег <h6> позначає підзаголовок найнижчого рівня. Насправді рідко зустрічаються тексти, у яких зустрічаються підзаголовки нижче третього рівня. Тому, тегами заголовків, що найчастіше використовуються, є: <h1>, <h2> і <h3>.

Варто зазначити, що пошукові системи надають особливого значення заголовкам, тому необхідно вчитися правильно їх використовувати.

**Посилання**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-04/module-04-01/html_04#%D0%BF%D0%BE%D1%81%D0%B8%D0%BB%D0%B0%D0%BD%D0%BD%D1%8F)Найважливішою властивістю HTML є посилання, які дозволяють зв'язати текст або зображення з іншими гіпертекстовими документами. Текст, як правило, виділяється кольором та оформляється підкресленням, для чого використовується тег <а>, структура якого наступна:

<a href=”filename”>текст\_посилання</a>

Тут filename - ім'я файлу або адреса Internet, на яку необхідно послатися, а текст-посилання - текст гіпертекстового посилання, яке буде безпосередньо показано в HTML документі. Наприклад, гіпертекстове посилання

<а href=”my\_work.html”>Views</a>

посилається на документ my\_work.html, утворюючи гіпертекстове посилання у вигляді слова Views. Якщо документ, який формує посилання, що знаходиться в іншій папці, що стосується web-сайту, то таке посилання називається відносним.

Наприклад:

<а href=”photo/my\_photo.html”>Мій фотоальбом</а>

посилається на файл my\_photo.html, розташований у папці photo, вкладеній у поточну, та утворює посилання у вигляді тексту Мій фотоальбом. Якщо необхідно послатися на ресурс Internet, розташований на віддаленому сервері, або вказати в посиланні повне ім'я файлу і шлях до файлу, то використовують абсолютні посилання. Структура такого тега аналогічна, тільки він формується на основі повного шляху до ресурсу у вигляді протокол:/URL/шлях. Наприклад:

<а href=”https://www.google.com”>Пошук в інтернет</а>

В HTML ділять гіперпосилання так само на два види:

* Зовнішні гіперпосилання переміщують користувача, який натиснув на них на інший HTML документ. Це ті, що ми розглянули вище.
* Внутрішні гіперпосилання (якорі) переміщують користувача на попередньо створену закладку в документі, в якому вони визначені.

<!-- Створення гіперпосилання на закладку -->

<a href="#bookmark"> Текст посилання </a>

<!-- Створення закладки -->

<div id="bookmark"> Текст закладки.</div>

**Зображення**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-04/module-04-01/html_04#%D0%B7%D0%BE%D0%B1%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%BD%D1%8F)У документах HTML можуть використовуватись зображення та графіка, для чого використовується тег <img>. Допускається використання файлів у форматі PNG, GIF або JPG/JPEG. Наступний приклад демонструє вставку в документ JPG файлу:

<img src="pic.jpg" width=45 height=50 alt="Photo">

Тут атрибут src= визначає **URL** адресу графічного файлу. У наведеному прикладі файл буде розміщено в області шириною 45 та висотою 50 пікселів відповідно. Якщо розміри, вказані атрибутами height= (висота) та width= (ширина), не збігаються з розмірами графічного файлу, останній масштабується. Рекомендується для графічних файлів завжди задавати їх розміри в таблицях стилів CSS, замість атрибутів height та width.

Атрибут alt= вказує, що саме потрібно підставити на місце зображення, якщо браузер не показує графічні файли або внаслідок повільної швидкості з'єднання файл ще не отримано. Цей атрибут є обов'язковим з точки зору семантики і валідатор видаватиме помилку.

Зображення може бути засобом задавання та управління вибором гіперпосилань в HTML документі, для чого на тег img повинно вказувати гіперпосилання, яка визначається тегом а. Наприклад

<а href="index.htm">

<img src="logo.jpg" width="43" height="35" alt="На головну сторінку">

</a>

**Списки**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-04/module-04-01/html_04#%D1%81%D0%BF%D0%B8%D1%81%D0%BA%D0%B8)Мова HTML має можливості для створення різних списків та переліків. Для їх створення можуть використовуватись теги ul та ol, а елементи списку відзначаються тегом li, при цьому допускаються вкладені списки будь-якої глибини. Розглянемо наступний приклад ненумерованого списку та використання тегів ul і li

<ul>

<li> червоний</li>

<li> помаранчевий</li>

<li> жовтий</li>

<li> зелений</li>

<li> блакитний</li>

<li> синій</li>

<li> фіолетовий</li>

</ul>

Нумерований список створюють, використовуючи теги ol та li. Наприклад список:

<ol>

<li>елемент перший</li>

<li>елемент другий</li>

<li>елемент третій</li>

</ol>

За замовчуванням нумерація задається арабськими цифрами, починаючи з одиниці. Використовуючи атрибути тега ol, можна змінити стиль оформлення списку. Атрибут type= визначає стиль нумерації (літери або цифри), при цьому допускаються наступні його значення:

* type=A - використовувати великі латинські літери;
* type=a - використовувати маленькі літери;
* type=I - використовувати великі римські цифри;
* type=i - використовувати маленькі римські цифри;
* type=1 - використовувати арабські цифри

Атрибут start= визначає початкове значення списку (десяткове число), наприклад start=5.

І, нарешті, розглянемо приклад вкладеного списку, в якому можна використовувати теги ul, ol і li

<ul>

<li>Перший розділ</li>

<ul>

<li>Перший підрозділ першого розділу</li>

<li>Другий підрозділ першого розділу

<ol type="a">

<li>Перший список</li>

<li>Другий список</li>

</ol>

</li>

<li>Третій підрозділ першого розділу</li>

</ul>

<li>Другий розділ</li>

</ul>

**Коментарі**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-04/module-04-01/html_04#%D0%BA%D0%BE%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%80%D1%96)

Тег додає коментар до коду документа. Текст коментаря не відображається на сторінці. Дозволяється всередину коментаря додавати інші теги, вкладені коментарі (коли один коментар розташований всередині іншого) неприпустимі.

<!-- текст коментаря -->

За допомогою коментарів Ви можете залишати в коді пояснювальні нотатки.

* Коментарі повністю ігноруються браузером;
* Коментарі не відображаються під час перегляду сторінки.
* Коментарі можуть використовуватися під час відлагодження коду, якщо Ви не впевнені, чи хочете бачити цей заголовок, абзац тощо в підсумковій версії сторінки, Ви можете взяти код в тег <!-- --> на час прийняття рішення.

**Спецсимволи HTML**

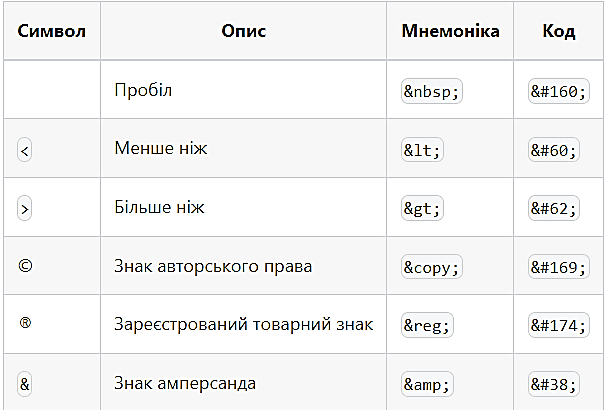
[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-04/module-04-01/html_04#%D1%81%D0%BF%D0%B5%D1%86%D1%81%D0%B8%D0%BC%D0%B2%D0%BE%D0%BB%D0%B8-html)Крім тегів у мові HTML використовується спеціальний керуючий символ & - амперсанд. Цей символ використовується для виведення спеціальних символів та символів із розширеної кодової таблиці, які не можна ввести з клавіатури.

Наприклад, виведення самого символу амперсанд & здійснюється за допомогою послідовності символів &аmp; для виведення кутових дужок використовуються &lt; для < та &gt; для >, а, скажімо, символ з номером 182 з кодової таблиці, може бути заданий послідовністю &#182;.

Спецсимвол, який найчастіше використовується, - це нерозривний пробіл (його мнемоніка &nbsp;).

Цей символ використовується для контролю перенесення рядка (після цього символу автоматичне перенесення рядка неможливе) і для вставки в текст пробілів, що йдуть підряд (за замовчуванням, якщо Ви вставите в текст 5 пробілів, що йдуть поспіль, браузер виріже 4 і відобразить лише один).

Мнемоніки та коди спецсимволів, що часто використовуються



**Семантичні теги**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-04/module-04-01/html_04#%D1%81%D0%B5%D0%BC%D0%B0%D0%BD%D1%82%D0%B8%D1%87%D0%BD%D1%96-%D1%82%D0%B5%D0%B3%D0%B8)Завдяки використанню CSS можна створювати сторінки з добре зрозумілою для користувачів візуальною структурою, але чи будуть ці сторінки також зрозумілі для пошукових систем або браузерів?

Наприклад, як пошуковий робот може відрізнити вміст документа від навігаційного меню, якщо вони розмічені за допомогою однакових div елементів?

Для того, щоб вирішити цю проблему, в HTML5 були введені семантичні теги. За допомогою семантичних тегів Ви можете зробити сторінки сайтів зрозумілішими для пошукових систем та браузерів.

* <footer> Визначає футер.
* <header> Визначає заголовний блок сайту.
* <nav> Визначає навігаційне меню.

За допомогою тегу <section> Ви можете групувати логічно пов'язаний вміст у документі.

Якщо логічно пов'язаний вміст є автономним (може використовуватись в інших документах, незалежно від решти вмісту на сторінці) необхідно використовувати замість <section> тег <article>.

Тег <aside> використовується для виділення елементів, які не є частиною вмісту, але безпосередньо пов'язані з ним. Цим тегом можуть виділятися: цитати, додаткова інформація до статті, словник із термінами, список посилань тощо. Як уже говорилося вище за допомогою тегу <mark> Ви можете виділити "важливу" частину тексту.

У журналах та газетах ілюстрації часто супроводжуються підписами. В HTML4 неможливо було створювати підписи, не вдаючись до використання CSS.

В HTML5 цю проблему вирішено додаванням нових тегів: <figure> та <figcaption>.

<figure>

<img src='foto-1387634.jpg' width='300' height='230'/>

<figcaption>Ми з Танею на відпочинку, 'Чорне море'</figcaption>

</figure>

**HTML Форма**

Функціональність сайтів, які надають користувачеві можливість введення даних та отримання результатів їх обробки, забезпечуються використанням програм, що працюють на стороні сервера - серверних програм. Ці програми обробляють отримані від відвідувача Web-сайту дані та видають результат у вигляді звичайної Web-сторінки. Саме для них в HTML передбачена можливість створення Web-форм та елементів управління – щоб відвідувач міг ввести дані, які потім обробить серверний застосунок

Основна схема роботи серверного застосунку

1. Відвідувач вводить в елементи управління, які розташовані у Web-формі на Web-сторінці, необхідні дані.
2. Ввівши дані, відвідувач натискає розташовану в тій самій Web-формі спеціальну кнопку - кнопку надсилання даних (Send, Ok, тощо).
3. Web-форма кодує введені в неї дані та відправляє їх серверному застосунку, розташованому по зазначеному URL.
4. Web-сервер перехоплює надіслані дані, запускає серверний застосунок та передає дані йому.
5. Серверний застосунок обробляє отримані дані.
6. Серверний застосунок формує відповідь (можливо, це Web-сторінка) з результатами обробки даних відвідувача та передає її Web-серверу.
7. Web-сервер отримує сформовану серверним застосунком відповідь та надсилає її відвідувачу.

Форма призначена для обміну даними між користувачем та сервером. Сфера застосування форм не обмежена відправленням даних на сервер, за допомогою клієнтських скриптів можна отримати доступ до будь-якого елементу форми, змінювати його та застосовувати на свій розсуд.

Для створення інтерактивних html документів використовують форми, які заповнюються і визначаються тегом form. В одному html документі може бути визначено кілька форм для заповнення, але одночасно на сервер може бути відправлена ​​тільки одна форма.

Тому дані форм повинні бути незалежні один від одного і вкладені теги form не допускаються. Формат тегу form наступний:

<form action="url" method="GET/POST">

...

</form>

Тут використовуються наступні атрибути:

* action=url - URL-адреса сервера запитів, куди буде надіслано зміст форми після підтвердження, якщо це поле відсутнє, буде використана URL-адреса поточного документа.
* method=GET/POST - HTTP метод, що використовується для надсилання змісту заповненої форми на сервер. При цьому можливі наступні варіанти:
* GET - це метод за замовчуванням, який призводить до додавання вмісту заповненої форми до URL;
* POST - при використанні цього методу вміст заповненої форми пересилається не як частина URL, а як вміст тіла запиту.
* ENCTYPE= задає тип кодування вмісту заповненої форми. Цей атрибут діє тільки коли використовується метод POST.

Коли форма відправляється на сервер, управління даними передається програмі, заданій атрибутом action тегу <form>. Попередньо браузер готує інформацію у вигляді пари `ім'я=значення', де ім'я визначається атрибутом name тегу <input>, а значення введене користувачем або встановлене в полі форми за замовчуванням.

Всередину контейнера <form> поміщаються інші теги, при цьому, сама форма жодним чином не відображається на веб-сторінці, тільки її елементи та результати вкладених тегів.

**Теги форми**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-04/module-04-01/html-form_05#%D1%82%D0%B5%D0%B3%D0%B8-%D1%84%D0%BE%D1%80%D0%BC%D0%B8)Всередині тегу form можуть знаходитися будь-які теги, крім іншого тегу form.

Для встановлення інтерфейсних елементів всередині form використовуються теги input, select, textarea та інші. Давайте їх розглянемо.

Для того щоб ввести інформацію про користувача (логін, пароль, дату народження тощо), використовують елемент введення <input>. Його основний атрибут – type, визначає, що ми будемо вводити і як.

**Input**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-04/module-04-01/html-form_05#input)Тег input використовується для встановлення простого елемента введення, при цьому можуть бути використані атрибути:

* accept– встановлює фільтр на типи файлів, які ви можете надіслати через поле завантаження файлів;
* alt – альтернативний текст для кнопки із зображенням;
* autocomplete – вмикає або вимикає автозаповнення, відповідно набуває значення on і off;
* checked – попередньо активований перемикач або прапорець;
* disabled – блокує доступ та зміну елемента;
* list – вказує на список варіантів, які можна вибирати під час введення тексту;
* max – максимально допустиме значення для введення числа або дати;
* maxlength – максимально допустима кількість символів, дозволених у тексті;
* min - мінімально допустиме значення для введення числа або дати;
* multiple – дозволяє завантажити кілька файлів одночасно;
* name – задає ім'я поля, призначене для того, щоб обробник форми міг його ідентифікувати;
* pattern – встановлює шаблон введення;
* placeholder – виводить текст-підказку
* readonly – встановлює, що поле не може змінюватись користувачем;
* required – зазначає, що поле є обов'язковим для заповнення;
* size – задає ширину текстового поля;
* step – задає крок збільшення для числових полів;
* tabindex – визначає порядок переходу між елементами за допомогою кнопки Tab;
* type – повідомляє браузеру, до якого типу належить елемент форми.
* value – визначає значення елемента.

Значення передостаннього атрибуту type у списку дуже великі і важливі, оскільки задають поведінку при введенні в input



Дуже часто ми стикаємося з необхідністю вибору. Наприклад, ми вибираємо, чи згодні з ліцензійною угодою, ми вибираємо, які сторінки відображати у меню соц. мережі, застосунку...

Коли потрібно вибрати кілька елементів із багатьох, використовують checkbox – поле з квадратиком, куди встановлюється галочка. Створюється таке поле так

<input id="checkbox1" type="checkbox" value="Англійська"/> Англійська <br/>

value – значення, яке Ви обрали. Використовується для обробки у скриптах. Поряд із полем пишеться те слово, яке відповідає елементу вибору, і ми хочемо відобразити на сторінці в браузері.

Іноді ми можемо вибирати лише один елемент зі списку запропонованих альтернатив. Наприклад, ми обираємо стать: чоловіча/жіноча. Такий вибір нам допоможуть організувати радіокнопки. Створюються вони так:

<input type="radio" name="gender" value="чоловічий"/> чоловічий

<input type="radio" name="gender" value="жіночий"/> жіночий

Зверніть увагу, що імена name – однакові. Це забезпечує те, що ви зможете вибрати (одночасно) лише один з елементів.

Іноді дуже зручно задавати альтернативи у вигляді розкривного списку. На деяких сайтах так організовано вибір країни, внз, віку. Встановлення списку альтернатив зручно для обробки, оскільки користувач правильно введе дані. Використання розкривного списку значно економить місце на формі.

**Select**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-04/module-04-01/html-form_05#select)Тег select призначений для створення списків у формі, при цьому всередині дозволена лише послідовність тегів option, за кожним з яких слідує деяка кількість простого тексту.

Атрибути тегу select наступні:

* name=ідентифікатор - символьне ім'я для елемента select, за яким він ідентифікується;
* size=n - якщо значення дорівнює 1 або якщо цей атрибут відсутній, то елемент select буде представлений як розкривне меню, якщо size=2 або більше, елемент буде представлений як вікно вибору, а значення визначатиме, скільки елементів списку буде видно;
* multiple - якщо цей атрибут є, то допускається множинний вибір зі списку.

Створити список можна так:

<select id="select1">

<option>Елемент списку</option>

...

</select>

де select – сам список, option – елемент списку.

Можна також організувати вибір із заданого списку: datalist. Це називається альтернатива і її можна задавати за допомогою списку варіантів. Виглядає це так:

<input type="text" name="team" id="favorite\_dish" list="dish\_list">

<datalist id="dish\_list">

<option>Борщ</option>

<option>Суп</option>

<option>Харчо</option>

<option>Холодник</option>

</datalist>

Зверніть увагу на те, що атрибут list елемента input та атрибут id елемента <datalist> містять однакове значення dish\_list. Це їх і пов'язує.

**Button**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-04/module-04-01/html-form_05#button)Тег button створює на веб-сторінці кнопки і за своєю дією нагадує результат, що отримується за допомогою тегу <input> (з атрибутом type="button | reset | submit"). На відміну від цього тегу, <button> пропонує розширені можливості створення кнопок. Наприклад, на такій кнопці можна розміщувати будь-які елементи HTML, включно і зображення. Використовуючи стилі, можна визначити вигляд кнопки шляхом зміни шрифту, кольору фону, розмірів та інших параметрів.

Теоретично тег <button> повинен розташовуватися всередині форми, яка встановлюється елементом <form>. Проте браузери не виводять повідомлення про помилку і коректно працюють із тегом <button>, якщо він зустрічається самостійно. Однак, якщо необхідно результат натискання на кнопку відправити на сервер, поміщати <button> у контейнер <form> обов'язково.

**Textarea**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-04/module-04-01/html-form_05#textarea)І, нарешті, тег textarea може бути використаний для розташування багаторядкового поля введення з необов'язковим вмістом у формі.

Атрибути тегу textarea наступні:

* name= - символьне ім'я поля введення;
* rows= - кількість рядків у полі введення, тобто висота поля;
* cols= - кількість стовпців у полі введення, тобто ширина поля.

Об'єкт textarea має смуги прокручування, тому може бути введена будь-яка кількість тексту. Зміст за замовчуванням повинен бути виключно **ASCII**текстом, при цьому символи перенесення рядка сприймаються.

Приклад нескладної форми

<form method="POST" name="student" action='/students'>

<label for="firstname"> Enter your name

</br>

<input type="text" id="firstname" name="firstname" size="45"/>

</label>

</br></br>

<input type="radio" checked name="job"/> <span>University student</span>

</br>

<input type="radio" name="job"/> <span>Other</span>

</br></br>

<button type="submit"> Submit</button>

<button type="reset"> Reset</button>

</form>

Примітка: </br> тег перенесення рядка

﻿

**Стилізація HTML**

Питання стилізації веб-додатків стосується веб-розробки, але це розділ Front-end розробки. Back-end розробник (зона відповідальності мови Python) не зобов'язаний вміти виконувати стилізацію html сторінок, але мати хоча б загальне уявлення повинен.

**WIKI**

**Front-end** — презентаційна частина інформаційної або програмної системи, її інтерфейс користувача та пов'язані з ним компоненти; застосовується у співвідношенні з базисною частиною системи, її внутрішньою реалізацією, яка називається в цьому випадку **Back-end**.

**CSS** – це спеціальна мова стилю сторінок, що використовується для опису їхнього зовнішнього вигляду. Самі ж сторінки написані мовами розмітки даних. CSS є основною технологією всесвітньої павутини, поряд з HTML і JavaScript.

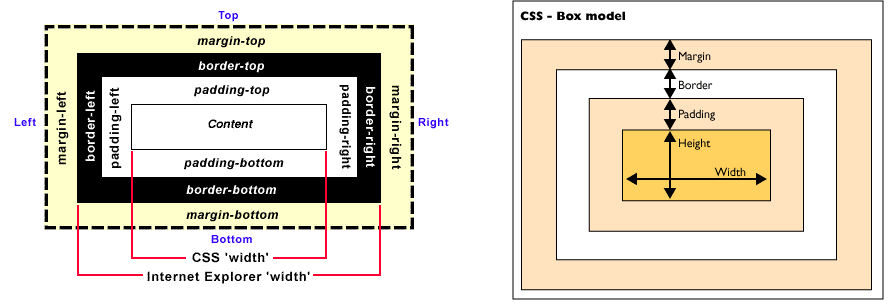
**Блокові та рядкові елементи в CSS**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-04/module-04-01/layout_06#%D0%B1%D0%BB%D0%BE%D0%BA%D0%BE%D0%B2%D1%96-%D1%82%D0%B0-%D1%80%D1%8F%D0%B4%D0%BA%D0%BE%D0%B2%D1%96-%D0%B5%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D0%B8-%D0%B2-css)**Блок** — це звичайна прямокутна область, що має ряд властивостей, таких як: рамка, поля та відступи. Вмістом блоку може бути будь-що: частина тексту, зображення, відеоролик, список, форма для заповнення, меню навігації тощо.

**Рамка (border)** — це контур, для якого можна задати такі характеристики як товщина, колір та тип (пунктирна, суцільна, точкова). На відміну від таблиць, вона необов'язкова.

**Поля (padding)** — відокремлюють вміст блоку від його рамки, щоб текст, наприклад, не ліпився тісно до стінок блоку.

**Відступи (margin)**— це порожній простір між різними блоками або між блоком і стінками сторінки сайту. Вони дозволяють розташувати блоки на заданій відстані один до одного.

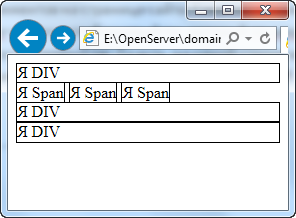


На відміну від стандартних HTML-тегів до свого вмісту (p - до абзаців, а - до посилань, img - до зображень), тег div є, по суті, нейтральним. Тобто йому все одно, що містити, хоч усе разом. Зазвичай тег div використовують для встановлення великих функціональних областей на сторінці, таких як: шапка (в HTML5 це тег header), блок навігації (в HTML5 це тег nav), блок(и) основного вмісту, футер (в HTML5 це тег footer) або підвал по-нашому тощо.

Але він може застосовуватися і до дрібніших деталей сайту. І він також є парним: <div></div> Щоб розрізняти однотипні теги, для них були придумані спеціальні атрибути. Найчастіше використовують два види: ідентифікатори (id) та класи (class).

id - атрибут, що дозволяє надати тегу унікальний набір властивостей, тобто такий, що використовується на сторінці сайту тільки один раз. class — атрибут, який дозволяє той самий набір властивостей задати кільком елементам на сторінці сайту.

Тег span, як і div, є нейтральним. Він може застосовуватися до будь-якого елемента або групи елементів на сторінці сайту. Головна відмінність його від діва – порядок розміщення на сторінці. Якщо div — це чисто блоковий тег, який за замовчуванням не терпить сусідства з іншим блоком, то span — це рядковий тег. Тобто, на одному рядку може розміщуватися поспіль кілька "спанів", тоді як "діви" прагнуть залізти один під інший.



Блокові елементи характеризуються тим, що займають всю доступну ширину, висота елемента визначається його вмістом, та він завжди починається з нового рядка.

Рядкові називаються такі елементи веб-сторінки, які є безпосередньою частиною іншого елемента, наприклад, абзацу тексту. Здебільшого вони використовуються для зміни вигляду тексту або його логічного виділення.

Рядкові елементи можуть містити лише дані або інші рядкові елементи, а в блокові допустимо вкладати інші блокові елементи, рядкові елементи, а також дані. Іншими словами, рядкові елементи жодним чином не можуть зберігати блокові елементи (в HTML5 в рядковий елемент посилання <a> можна вкладати блокові елементи).

Блокові елементи завжди починаються з нового рядка, а рядкові таким способом не акцентуються.

Блокові елементи займають всю доступну ширину, наприклад, вікна браузера, а ширина рядкових елементів дорівнює їх вмісту плюс значенню відступів, полів і меж.

**Управління розмірами контейнерів**[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-04/module-04-01/layout_06" \l "%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%96%D0%BD%D0%BD%D1%8F-%D1%80%D0%BE%D0%B7%D0%BC%D1%96%D1%80%D0%B0%D0%BC%D0%B8-%D0%BA%D0%BE%D0%BD%D1%82%D0%B5%D0%B9%D0%BD%D0%B5%D1%80%D1%96%D0%B2" \t "_blank)

Тег <div> – контейнер, що представляє прямокутну область. Значення висоти та ширини цієї області визначаються такими стандартними атрибутами стилів, як:

* висота
* min-height – задає мінімальну висоту елемента;
* height – задає висоту елемента;
* max-height – задає максимально можливу висоту елемента;
* ширина
* min-width – задає мінімальну ширину елемента;
* width – задає ширину елемента
* max-width – задає максимально можливу ширину елемента.

Також допустиме використання наступних значень атрибутів управління розмірами:

* auto – розмір встановлюється, залежно від розмірів контенту;
* inherit – значення наслідується від батьківського елемента.

**Управління розміщенням контейнерів**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-04/module-04-01/layout_06#%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%96%D0%BD%D0%BD%D1%8F-%D1%80%D0%BE%D0%B7%D0%BC%D1%96%D1%89%D0%B5%D0%BD%D0%BD%D1%8F%D0%BC-%D0%BA%D0%BE%D0%BD%D1%82%D0%B5%D0%B9%D0%BD%D0%B5%D1%80%D1%96%D0%B2)Для управління розміщенням елементів використовуються сітки (Grid) та флексбокси (Flexbox)

Для структурної розмітки html сторінки використовують технологію CSS Grid

Скрінкаст по технології [**CSS Grid**](https://www.youtube.com/playlist?list=PLNkWIWHIRwMHlq6yOP65F_rNH5wID1U21) (*Прим.*: російською)

https://www.youtube.com/playlist?list=PLNkWIWHIRwMHlq6yOP65F\_rNH5wID1U21

[Гра](http://cssgridgarden.com/) (https://cssgridgarden.com/) для швидкого занурення у технологію CSS Grid.

Також можна використовувати специфікацію **CSS Flexible Box Layout**

**Module** (або спрощено Flexbox), яка дозволяє контролювати розмір, порядок та вирівнювання елементів за кількома осями, розподілом вільного місця між елементами та багато іншого.

Скрінкаст по технології [CSS Flexbox](https://www.youtube.com/watch?v=O-ytfplFQ3c&list=PLNkWIWHIRwMG0EUBS8rvTRVNL9IcxcawW) (*Прим.*: російською)

(https://www.youtube.com/watch?v=O-ytfplFQ3c&list=PLNkWIWHIRwMG0EUBS8rvTRVNL9IcxcawW)

Є дві гри, щоб зрозуміти та засвоїти властивості flexbox.

1. [FLEXBOX FROGGY](http://flexboxfroggy.com/#uk) (https://flexboxfroggy.com/#uk)
2. [Flexbox Defense](http://www.flexboxdefense.com/) (http://www.flexboxdefense.com/)

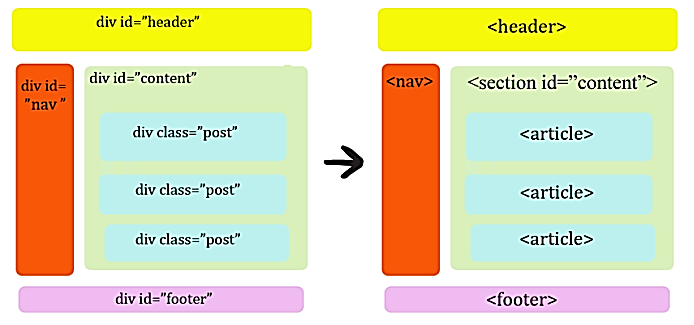
І невелика [шпаргалка](https://codepen.io/enxaneta/full/adLPwv/) (https://codepen.io/enxaneta/full/adLPwv/), щоб розуміти основні моменти.

**Семантика верстки**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-04/module-04-01/layout_06#%D1%81%D0%B5%D0%BC%D0%B0%D0%BD%D1%82%D0%B8%D0%BA%D0%B0-%D0%B2%D0%B5%D1%80%D1%81%D1%82%D0%BA%D0%B8)

Часто можна почути, що верстка повинна бути семантичною. В HTML5 з'явилося кілька нових тегів, які покликані замінити блоки div, що часто застосовуються.

На відміну від універсального div, нові теги несуть семантичне навантаження, та суворе визначення для кожного блоку його місця та ролі:



* <header> </header> — Визначає область «шапки» сайту з логотипом, первинною навігацією тощо;
* <footer> </footer> — Визначає «підвал», колонтитул веб-сторінки сайту або розділу, в якому зазвичай розміщується додаткова інформація;
* <section> </section> — Визначає блок, який слугує для групування однотипних об'єктів, або для поділу тексту на розділи;
* <article> </article> — Визначає автономну частину сторінки, це може бути повідомлення форуму, журнал або газетна статтю, запис у блозі тощо;
* <nav> </nav> — Визначає область навігації, як правило, список посилань;

Семантична верстка - це верстка з правильним використанням HTML-тегів. З використанням їх за призначенням, як їх і задумували розробники мови HTML та веб-стандартів. Наприклад, тег <p> - це абзац, і не варто ним розмічати рядки веб-форм. А <b> - це просто виділення тексту жирним, а зовсім не заголовок.

**Viewport**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-04/module-04-01/layout_06#viewport)Вебзастосунку можуть відкриватися на пристроях з різною роздільною здатністю екрану. **Viewport** - це видима користувачеві область веб-сторінки. Тобто, це те, що може побачити користувач, не вдаючись до прокручування.

Розміри цієї області визначаються розміром екрана пристрою.

Метатег viewport був розроблений компанією Apple для того, щоб вказувати браузерам на те, в якому масштабі необхідно відображати користувачеві видиму область веб-сторінки. Іншими словами viewport призначений для того, щоб веб-сторінки відображалися (виглядали) правильно (коректно) на смартфонах, планшетах та інших пристроях з високою щільністю пікселів (> 200ppi). Цей метатег призначений здебільшого для адаптивних сайтів, але за допомогою нього можна покращити відображення веб-сторінок, що мають фіксовану або гнучку розмітку. Розглянемо використання метатегу viewport для адаптивних сайтів. Увімкнення підтримки тегу <meta>``viewport для адаптивних сайтів здійснюється за допомогою додавання всього одного рядка в розділ head веб-сторінки:

<meta name="viewport" content="width=device-width, initial-scale=1">

Атрибут name призначений для того, щоб вказати браузеру, яку інформацію про сторінку хочемо йому повідомити. У цьому випадку ця інформація стосується viewport. Контент (вміст) цих відомостей вказується як значення атрибуту content за допомогою пар ключ-значення, розділених між собою комами. Для адаптивного дизайну значення атрибуту content viewport повинно визначатися двома параметрами:

width=device-width, initial-scale=1

Перший параметр width=device-width відповідає за те, щоб ширина видимої області веб-сторінки дорівнювала CSS ширині пристрою (device-width). Ця ширина (CSS) - не фізична роздільна здатність екрану. Це деяка величина, яка залежить від роздільної здатності екрану. Вона призначена для того, щоб мобільний адаптивний дизайн сайту відображався на всіх пристроях однаково, незалежно від їхньої щільності пікселів екрану. Другий параметр initial-scale - встановлює початковий масштаб веб-сторінки. Значення 1 означає те, що масштаб дорівнює 100%.

Якщо сайт не має адаптивного дизайну, то його відображення на екрані смартфона теж можна поліпшити. Можна зробити так, щоб ширина сторінки масштабувалася під ширину пристрою. Здійснюється це також за допомогою встановлення параметру width значення device-width. Тобто, для неадаптивних сайтів у розділ head необхідно додати наступний рядок:

<meta name="viewport" content="width=device-width">

**Скидання початкових стилів**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-04/module-04-01/layout_06#%D1%81%D0%BA%D0%B8%D0%B4%D0%B0%D0%BD%D0%BD%D1%8F-%D0%BF%D0%BE%D1%87%D0%B0%D1%82%D0%BA%D0%BE%D0%B2%D0%B8%D1%85-%D1%81%D1%82%D0%B8%D0%BB%D1%96%D0%B2)Якщо ми створимо сторінку на «голому» HTML без оформлення та стилів, браузер все одно відобразить вміст тегу <h1> великим і жирним, <h2> - трохи меншим розміром, виділить текст у тезі <i> курсивом, <u> зробить підкресленим, а <b>

* жирним.

Відбудеться так тому, що кожен браузер має за замовчуванням якийсь набір базових стилів, які він застосовує до сторінки за замовчуванням. І річ у тому, що у різних браузерах ці правила трохи відрізняються. Років 10 тому ці відмінності були кардинальними, і дуже впадали в очі. Зараз вони мінімальні, але все ж таки є.

Щоб усунути ці відмінності, і зробити за замовчуванням відображення сторінки у всіх браузерах однаковим, використовуються спеціальні .css файли. Наприклад [**normalize.css**](https://necolas.github.io/normalize.css/) (<https://necolas.github.io/normalize.css/>) або[**sanitize.css**](https://csstools.github.io/sanitize.css/) (https://csstools.github.io/sanitize.css/). Це забезпечує для HTML-елементів найкращу кросбраузерність в стилях за замовчуванням.

Але загалом для верстки простіше використовувати готові інструменти на кшталт [**Bootstrap**](https://getbootstrap.com/) (<https://getbootstrap.com/>)— вільний набір інструментів для створення сайтів та вебзастосунків. Включає HTML і CSS шаблони оформлення для типографіки, веб-форм, кнопок, міток, блоків навігації та інших компонентів веб-інтерфейсів, включаючи JavaScript розширення.

**Шаблонізатор Jinja**

Створення чистих HTML-документів не є частим завданням у Python. Набагато частіше доводиться мати справу з попередньо сформованим HTML-документом, куди потрібно як у шаблон підставити потрібні змінні. Для цього дуже добре підходить пакет Jinja.

Встановимо Jinja:

pip install jinja2

**Jinja** — це пакет для рендерингу шаблонів. Він має свою власну мета-мову, за допомогою якої він розуміє, як саме потрібно сформувати з шаблону готовий документ. Підтримує передачу Python-об'єктів, цикли, оператори розгалуження if.

Насправді немає жодної прив'язки саме до HTML. За замовчуванням шаблон сприймає:

* код всередині**{{ }}** як Python код, який потрібно виконати і результат перетворити на рядок, і підставити.
* код всередині **{% %}** як вираз**for** або **if**, які впливають на те, як саме потрібно рендерити шаблон до виразу **{% endfor %}** або**{% endif %}**.

from jinja2 import Template

name = 'Bill'

age = 28

tm = Template("My name is {{ name }} and I am {{ age }}")

msg = tm.render(name=name, age=age)

print(msg) # My name is Bill and I am 28

Щоб відрендерити шаблон, у нього потрібно викликати метод **render** та передати туди набір іменованих аргументів. Важливо, щоб імена аргументів у шаблоні і ті, які передали, збігалися. Ще один важливий момент, Jinja сприймає звернення до атрибутів елемента через крапку всередині шаблону та звернення до елементів словника через квадратні дужки як одне й те саме:

from jinja2 import Template

persons = [

{'name': 'Andrej', 'age': 34},

{'name': 'Mark', 'age': 17},

{'name': 'Thomas', 'age': 44},

{'name': 'Lucy', 'age': 14},

{'name': 'Robert', 'age': 23},

{'name': 'Dragomir', 'age': 54}

]

rows\_tmp = Template("""{% for person in persons -%}

{{ person.name }} {{ person.age }}

{% endfor %}""")

print(rows\_tmp.render(persons=persons))

Andrej 34

Mark 17

Thomas 44

Lucy 14

Robert 23

Dragomir 54

Але найчастіше ми виконуємо підстановку якихось даних в HTML документі. Наприклад, вказаний список person необхідно вивести списком на сторінці html. Основним компонентом Jinja є клас Environment.

Коли шаблони зберігаються у файлах, ми створюємо середовище Jinja за допомогою FileSystemLoader. Туди ми можемо передати шлях, який вказує на папку наших шаблонів. Тепер, замість передачі рядка, ми завантажуємо файл persons.html у якості шаблону.

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport"

content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Document</title>

</head>

<body>

<ul>

{% for person in persons -%}

<li>{{ person.name }} {{ person.age }}</li>

{% endfor %}

</ul>

</body>

</html>

Як тільки ваш шаблон завантажено, ви можете використовувати його знову і знову, щоб заповнити його вмістом. Наприклад, за допомогою наступного скрипту.

from jinja2 import Environment, FileSystemLoader

env = Environment(loader=FileSystemLoader('.'))

template = env.get\_template("persons.html")

persons = [

{'name': 'Andrej', 'age': 34},

{'name': 'Mark', 'age': 17},

{'name': 'Thomas', 'age': 44},

{'name': 'Lucy', 'age': 14},

{'name': 'Robert', 'age': 23},

{'name': 'Dragomir', 'age': 54}

]

output = template.render(persons=persons, )

with open("new\_persons.html", "w", encoding='utf-8') as fh:

fh.write(output)

При його виконанні ми отримаємо html-файл з уже готовими підставленими даними.

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport"

content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Document</title>

</head>

<body>

<ul>

<li>Andrej 34</li>

<li>Mark 17</li>

<li>Thomas 44</li>

<li>Lucy 14</li>

<li>Robert 23</li>

<li>Dragomir 54</li>

</ul>

</body>

</html>

Шаблони є важливим компонентом повнофункціональної веб-розробки. За допомогою Jinja можна створювати багаті, повноцінні шаблони, які забезпечують інтерфейс веб-застосунків на Python. Але в основному вони використовуються з веб-фреймворками.