Прикладне програмування

ТЕМА 6. ПОБУДОВА ВЕБ-КЛІЄНТІВ

Веб

Всесвітня павутина (World Wide Web, далі – просто веб) є найбільш наповненою частиною мережі Інтернет.

Вона є настільки розповсюдженою, що багато хто вважає, що Інтернет – це і є веб, хоча це не так.

Веб базується на протоколі HTTP, який діє над TCP/IP. Але над TCP/IP діють і інші протоколи прикладного рівня: FTP, SMTP, POP3, IMAP тощо.

Веб став настільки популярним завдяки простим та інтуїтивно зрозумілим засобам навігації у мережі та величезній кількості документів, створених у веб.

Як і у будь-яких мережних застосуваннях, у веб є клієнти та сервери.

У даній темі ми розглянемо побудову веб клієнтів.

Ви вже маєте досвід використання веб-клієнтів, оскільки будь-який браузер є веб клієнтом.

Але час від часу виникає потреба у побудові специфічних веб-клієнтів замість використання стандартних браузерів.

Мова HTML

Своєю популярністю веб великою мірою завдячує мові HTML. HTML – це гіпертекстова мова розмітки документів (HyperText Markup Language).

HTML дозволяє зв’язувати багато документів у мережу (граф) шляхом застосування посилань між документами.

Якщо спочатку HTML передбачав тільки використання тексту, то зараз він дозволяє обробляти у документах інформацію різних типів, у тому числі, аудіо, зображення відео.

Структура документу HTML

Документ HTML складається з елементів.

Елемент – це частина тексту, обмежена початковим та кінцевим тегами.

Тег береться з обох боків у кутові дужки: < >.

Кожний тег має власне ім’я.

Кінцевий тег має таке ж ім’я, як і відповідний початковий, але відрізняється від початкового тим, що починається з косої риски ‘/’.

Деякі теги є тільки початковими.

Кожний тег може мати атрибути, які вказують після імені тегу.

Атрибут має фіксоване ім’я та довільне значення.

Між ім’ям та значенням ставиться знак «дорівнює» ‘=’, а саме значення береться з обох боків у лапки.

Атрибути визначають параметри, що дозволяють задати специфічну поведінку тегу.

Структура документу HTML.2

Документ також поділяється на заголовок документу (head) та тіло документу (body).

Окрім тегів та власне тексту документ може містити коментарі. Коментар позначається так: <!-- -->

Документ HTML повинен починатись описом версії HTML, що застосовується у документі.

Для HTML версії 4 опис може виглядати так:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

Для HTML версії 5 опис повинен виглядати так:

```
<!DOCTYPE html>
```

Основні теги HTML

Початковий тег	Кінцевий тег	Опис
<html>	</html>	Цей тег ставиться на початку документу після опису версії (кінцевий – у кінці документу), включає усі інші теги
<head>	</head>	Позначає область заголовку документу
<body>	</body>	Позначає область тіла документу
<title>	</title>	Назва документу, міститься у елементі <head>
** **		Абзац тексту документу. Може вказувати специфічне форматування абзацу
** **		Розрив рядка документу, не має кінцевого тегу
<h1>	</h1>	Заголовок першого рівня (найбільший)
<h2>	</h2>	Заголовок другого рівня (менший першого). Всього є 6 рівнів заголовків
<script>	</script>	Область, у якій розміщено програмний код однією з мов прграмування
<a>		Гіперпосилання на інший документ або місце у даному документі. Адреса посилання задається атрибутом href

URL

Одним з базових понять, що відносяться до веб, є поняття URL.

URL (Uniform Resource Locator) або єдиний визначник ресурсу – це рядок, що дозволяє однозначно визначити місцезнаходження ресурсу у мережі.

URL має таку структуру:

<схема>://<логін>:<пароль>@<хост>:<порт>/<URL-шлях>?<параметри>#<якір>

Схема – це спосіб підключення до ресурсу.

- Як правило, це назва протоколу. Значення: http, https, ftp тощо.
- Для того, щоб підключитись до локального файлу, вказують file.

Логін – це ідентифікатор користувача ресурсу.

Пароль – відповідний пароль доступу.

- Пару <логін>:<пароль> разом з @ вказують для аутентифікації користувача ресурсу з обмеженим доступом.

URL.2

Хост – це ім’я комп’ютера у мережі або IP-адреса. Порт – порт доступу.

- Для http стандартним є порт 80, для ftp – 20 або 21.
- Стандартні порти можна не вказувати.

URL-шлях – це шлях до ресурсу на визначеному комп’ютері через ‘/’.

Параметри визначають запит, який спрямовано до ресурсу.

Якір вказує місце у документі, до якого слід перейти після його відкриття, або надає додаткові параметри.

URL.3

Приклади URL:

http://eu.univ.kiev.ua/

https://en.wikipedia.org/wiki/Python_(programming_language)

https://www.google.com.ua/webhp?sourceid=chrome-instant&ion=1&espv=2&ie=UTF-8#q=python

file:///C:/obv/Lect_Python2/T21/stud.txt

Веб-клієнти у Python. Пакет urllib

Засоби побудови веб-клієнтів у Python зібрано у пакеті urllib.

Пакет urllib містить, крім інших, модулі urllib.request для відкриття та читання з URL а також urllib.parse для аналізу рядка, у якому записано URL.

Щоб відкрити ресурс за заданим URL, використовують функцію urlopen:

```
response = urllib.request.urlopen(url)
```

Ця функція повертає об’єкт класу HTTPResponse, який дозволяє читати дані ресурсу так, як із файлу, відкритого для читання у нетекстовому режимі.

Приклад: відкриття сторінки у мережі (версія 1)

Відкрити веб-сторінку з заданим URL. Прочитати та показати всі рядки html-файлу.

Кодування символів html-файлів. Визначення кодування

Текст у html-файлі може містити не ASCII символи.

Тоді виникає питання щодо стандарту кодування символів у такому файлі.

Знати кодування необхідно для того, щоб правильно перетворити рядок байтів, який повертає інструкція читання з html-файлу, у рядок символів.

Для визначення кодування можна використати метод getheaders об’єкту класу HTTPResponse, що повертає функція urlopen:

```
response.getheaders()
```

Цей метод повертає список кортежів заголовків html-файлу.

Заголовки містяться у тегах елементу head.

Кожний кортеж складається з назви заголовку та його значення.

Кодування символів html-файлів. Визначення кодування.2

Для визначення кодування файлу використовують заголовок 'Content-Type', що визначає тип ресурсу та, можливо, кодування символів.

Кодування символів – це значення атрибуту charset, тобто, відповідний підрядок повинен мати вигляд charset=<стандарт_кодування>.

Об’єкт класу HTTPResponse також дозволяє перевірити стан виконання запиту до ресурсу:

response.status

Стан – це натуральне число.

Якщо стан дорівнює 200, це означає, що сторінку відкрито успішно.

Один зі станів 4xx або 5xx (xx – дві цифри) свідчить про помилку.

Приклад: відкриття сторінки у мережі (версія 2)

Відкрити веб-сторінку з заданим URL. Прочитати та показати всі рядки html-файлу.

Визначити кодування файлу та стан виконання запиту.

Кодування файлу визначає функція getencoding, яка аналізує заголовок 'Content-Type', використовуючи регулярний вираз.

Завантаження файлів з мережі

Для завантаження з заданого URL та збереження файлу на локальному комп'ютері можна використати функцію urlretrieve

result = urllib.request.urlretrieve(url, filename)

- де filename – ім'я локального файлу.

Якщо filename не вказано, то створюється тимчасовий файл на локальному комп'ютері.

Функція urlretrieve повертає кортеж, що складається з двох елементів: імені файлу, у який збережено ресурс з URL, а також заголовків.

Приклад: завантаження файлу з мережі

Завантажити з сайту файл з описом однієї з тем даного курсу.

Матеріали лекцій містяться на сайті http://matfiz.univ.kiev.ua/ у вигляді файлів у форматі .pdf.

Програма запитує (або отримує з командного рядка) ім’я локального файлу, у який треба зберегти матеріали, та номер теми, після чого зберігає дані та відкриває файл за допомогою функції os.startfile(filename).

Протокол HTTP

Спілкування між серверами та клієнтами у веб здійснюється згідно з протоколом HTTP. У цьому протоколі визначено декілька методів запиту до ресурсу у мережі:

Метод	Опис
GET	Метод GET використовується для отримання інформації з даного сервера, використовуючи заданий URL. Запити, що використовують GET, повинні тільки отримувати дані і не повинні мати ніякого іншого впливу на дані.
HEAD	Те ж, що й GET, але повертає тільки стан та розділи заголовка.
POST	Запит POST використовується для відправки даних на сервер, наприклад, інформації про клієнтів, завантаження файлів і т.д. за допомогою HTML-форм.
PUT	Замінює всі поточні представлення цільового ресурсу з завантаженого контенту.
DELETE	Видаляє всі поточні представлення цільового ресурсу, що задається URL

Протокол HTTP.2

Метод	Опис
CONNECT	Встановлює тунель до сервера, ідентифікованого даним URL.
OPTIONS	Описує параметри зв'язку для цільового ресурсу.
TRACE	Виконує тестування повідомлення методом петлі а також повертає шлях до цільового ресурсу.

Методи запиту GET та POST

Методи запиту (запити) GET та POST дозволяють повернути дані, але задаються та виконуються по-різному.

Параметри запиту GET передаються у рядку URL після символу ‘?’, а параметри запиту POST – окремо від рядка URL.

Параметри GET вказують у вигляді пар: <ім’я>=<значення>, між якими ставлять символ ‘&’, наприклад: espv=2&ie=UTF-8.

У рядку URL допускаються тільки символи ASCII за виключенням спеціальних символів (‘&’, ‘+’ та деяких інших) а також пропусків.

Для того, щоб вказати у параметрах запиту інші символи, треба виконати так зване «кодування» URL. У Python це виконує функція urlencode:

params = urllib.parse.urlencode(query, encoding=enc)

- де query – це словник параметрів або послідовність кортежів з двох елементів, які позначають ім’я та значення параметру відповідно; enc – стандарт кодування для не ASCII-символів.

Кодування та декодування URL

Якщо треба закодувати сам URL, а не тільки параметри, застосовують функцію quote:

string = urllib.parse.quote(string, encoding=enc)

- де string – це рядок; enc – стандарт кодування для не ASCII-символів.

Обернене перетворення здійснюють за допомогою функції unquote:

string = urllib.parse.unquote(string, encoding=enc)

string = urllib.parse.unquote(string, encoding=enc)

- де string – це рядок; enc – стандарт кодування для не ASCII-символів.

Приклад запиту у мережі (версія 1)

Отримати дані з сторінки у мережі за допомогою методу запиту GET.

Для передачі запиту використаємо сторінку енциклопедії університету: http://eu.univ.kiev.ua

Будемо передавати параметри запиту та записувати відповідь сервера у локальний файл.

Стандарт кодування символів для параметрів запиту будемо отримувати зі сторінки.

Відправка даних для запиту POST

Як вже було зазначено, параметри запиту POST передаються окремо від рядка URL.

Для цього треба у функції urlopen вказати другий параметр:

response = urllib.request.urlopen(url, data)

- де data – це рядок байтів, що містить дані запиту POST.
- Цей рядок можна отримати зі словника за допомогою функції urlencode.

Як визначити, які саме параметри треба передати серверу?

На деяких серверах є документація, що описує потрібні параметри.

Якщо ж такої документації немає, варто проаналізувати html-код сторінки, яка формує запит у інтерактивному режимі.

Поля даних, як правило містяться у формах на сторінці.

Форма має тег <form>. Ім’я програми, що обробляє запит на сервері, є значенням атрибуту action цього тегу.

Відправка даних для запиту POST.2

Поля введення у формах мають тег <input>, а ім'я параметру з точки зору програми, що обробляє запит на сервері, є значенням атрибуту name тегу <input>.

Наприклад, фрагменти html-коду сторінки пошуку електронного каталогу бібліотеки Київського університету мають вигляд:

```
<form action="result.php3" method="POST" name="search_f">
…
                <td align="left"><div
class="marg">Назва:</div></td>
                <td colspan="3"><input type="text"
name="title" value="" cols="" size="56" …>
                <td align="left"><div
class="marg">Автор:</div></td>
                <td colspan="2"><input type="text"
name="author" cols="35" value="" size="35" …>
…
                </td>
```

Відправка даних для запиту POST.3

Звідси можна зробити висновок, що

- програма, що обробляє запит, називається result.php3,
- поле «Назва» має ім’я title,
- поле «Автор» - ім’я author.

Приклад запиту у мережі (версія 2)

Отримати відомості про книги заданого автора із заданою назвою з електронного каталогу бібліотеки Київського університету.

Програма

- вводить або отримує з командного рядка назву та автора,
- формує дані запиту POST,
- відправляє запит
- зберігає отриману сторінку у локальному файлі.

Структурний аналіз HTML-файлів

Звичайно, просте збереження програмою сторінки з Інтренет не є найкращим способом отримання інформації, оскільки такі ж дії можна виконати за допомогою будь-якого браузера.

При написанні специфічних клієнтів метою, як правило є частина інформації зі сторінки, яку повертає сервер.

Щоб отримати інформацію, що є важливою, треба провести структурний аналіз (parse) html-документу.

Цей аналіз дозволяє виділити окремі теги разом з їх атрибутами, а також текст з html-сторінки.

У Python є модуль html.parser, який надає клас HTMLParser для структурного аналізу документу.

Щоб зробити власний аналіз, потрібно описати підклас класу HTMLParser, у якому перевизначити методи, що викликаються при знаходженні тегів або даних:

Структурний аналіз HTML-файлів.2

Метод	Опис
handle_starttag(tag, attrs)	Викликається, коли Python зустрічає початковий тег tag. attrs – це список атрибутів тегу. Кожний атрибут – це кортеж з 2 елементів: (ім’я, значення)
handle_endtag(tag)	Викликається, коли Python зустрічає кінцевий тег tag.
handle_data(data)	Викликається, коли Python зустрічає дані. data – рядок даних.
handle_entityref(name)	Викликається, коли Python зустрічає посилання на сутність. Посилання на сутність – це рядок виду &name, де name – ім’я сутності. Посилання на сутності замінюють у html спеціальні символи, які є частиною синтаксису. Наприклад, &gt замінює символ ‘>’

Структурний аналіз HTML-файлів.3

Метод	Опис
handle_charref(name)	Викликається, коли Python зустрічає посилання на символ. Посилання на символ – це рядок виду &#code, де code – код символу. Застосовується для символів, які не є ASCII-символами.
handle_comment(data)	Викликається, коли Python зустрічає коментар
handle_decl(decl)	Викликається, коли Python зустрічає означення, наприклад, <!DOCTYPE html>

Структурний аналіз HTML-файлів.4

Для того, щоб почати процес аналізу html, треба створити об’єкт класу-нащадку HTMLParser (наприклад, з ім’ям p) та викликати метод feed:

```
p.feed(data)
```

- де data – рядок, що містить текст html.

Далі Python буде обробляти текст та викликати ті методи, які визначені у класі-нащадку.

Приклад: перегляд текстових файлів з завантаженням означень термінів з Вікіпедії

При перегляді текстових файлів у графічному режимі необхідно забезпечити виведення означень з Вікіпедії термінів, що зустрічаються у тексті.

Приклад з перегляду текстових файлів ми розглядали у темі «Графічний інтерфейс».

Скористаємось описаними у цьому прикладі класами TextViewer (безпосередньо відображає текстовий файл у графічному вікні) та FontOpts (дозволяє вибрати розмір та написання шрифту).

Утворимо клас TextViewerWiki - нащадок TextViewer.

Також опишемо класи WikiDefParser, WikiDef та LangOpts.

Клас WikiDefParser

Клас WikiDefParser виконує структурний аналіз сторінки з Вікіпедії з означенням відповідного терміну та формує рядок означення.

В якості означення береться перший параграф з тіла документу, тобто, дані між тегами та .

WikiDefParser є нащадком класу HTMLParser.

Клас має поля:

- self.pieces - список частин тексту означення
- self.in_p - чи знаходимось ми усередині тегу . self.in_p дорівнює
  - False до першого тегу ,
  - True всередині першого тегу ,
  - None після першого тегу

Клас WikiDefParser.2

Конструктор __init__ викликає конструктор батьківського класу та встановлює початкові значення полів.

Клас перевизначає методи handle_starttag, handle_endtag та handle_data.

WikiDefParser також має властивість getdef, що повертає текст означення.

Клас WikiDef

Клас WikiDef призначений для читання статті Вікіпедії за запитом та повернення означення.

Він використовує клас WikiDefParser.

Клас має поле:

- self._def - означення терміну

Конструктор __init__ за заданим терміном (p_str) та мовою (lang) відкриває та аналізує статтю.

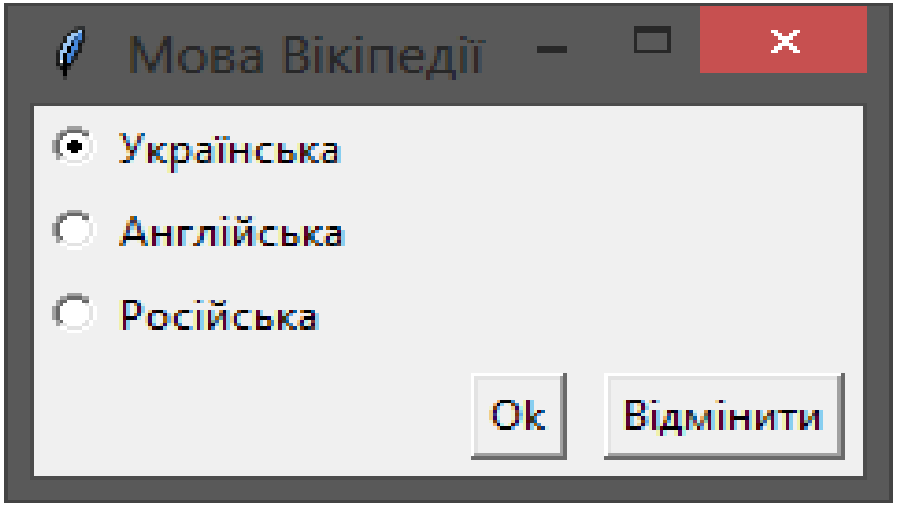
Мову треба визначати, оскільки URL Вікіпедії починається з двох літер – мови: ‘uk’, ‘en’ або ‘ru’.

WikiDef також має властивість definition, що повертає текст означення.

Клас LangOpts

Клас LangOpts призначено для вибору мови Вікіпедії.

Графічний інтерфейс містить радіокнопки та кнопки команд.



Клас LangOpts.2

Клас має поля:

- self.top - вікно верхнього рівня у якому розміщено елементи
- self.cancel - чи було натиснуто кнопку "Відмінити"
- self.langvar - змінна, пов'язана з радіокнопками
- self.language - мова Вікіпедії (спочатку - початкова мова init_lang)

Конструктор __init__ викликає внутрішній метод _make_widgets для створення елементів інтерфейсу.

Методи ok_handler та cancel_handler обробляють натиснення кнопок «Ok» та «Відмінити» відповідно.

Метод get повертає результат: вибрану мову.

Якщо натиснуто кнопку «Відмінити», то повертає None.

Цей метод, як правило, викликається після завершення вибору.

Клас TextViewerWiki

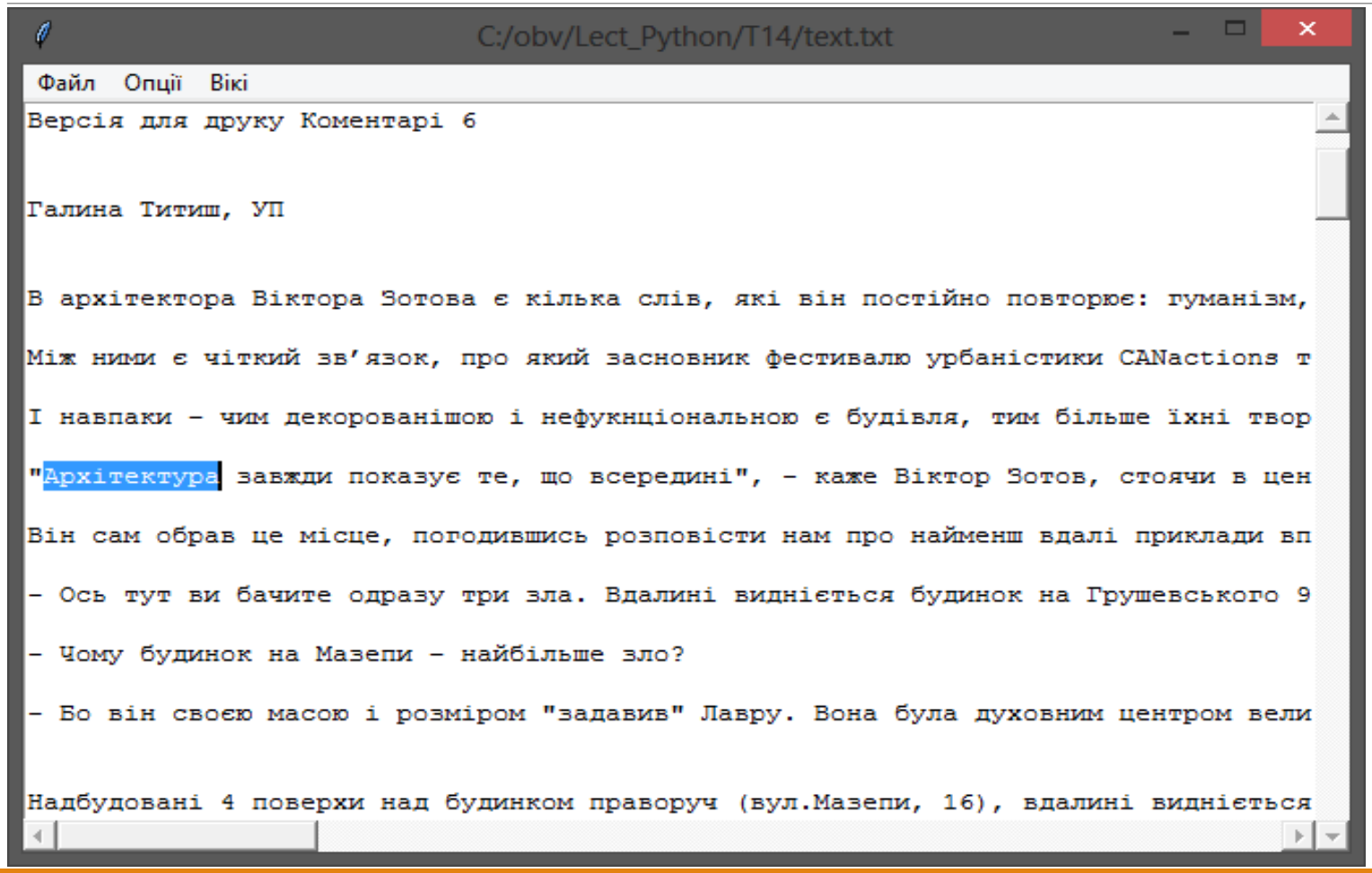
Клас TextViewerWiki створює графічний інтерфейс для перегляду текстових файлів та виведення означень з Вікіпедії.

Цей інтерфейс включає

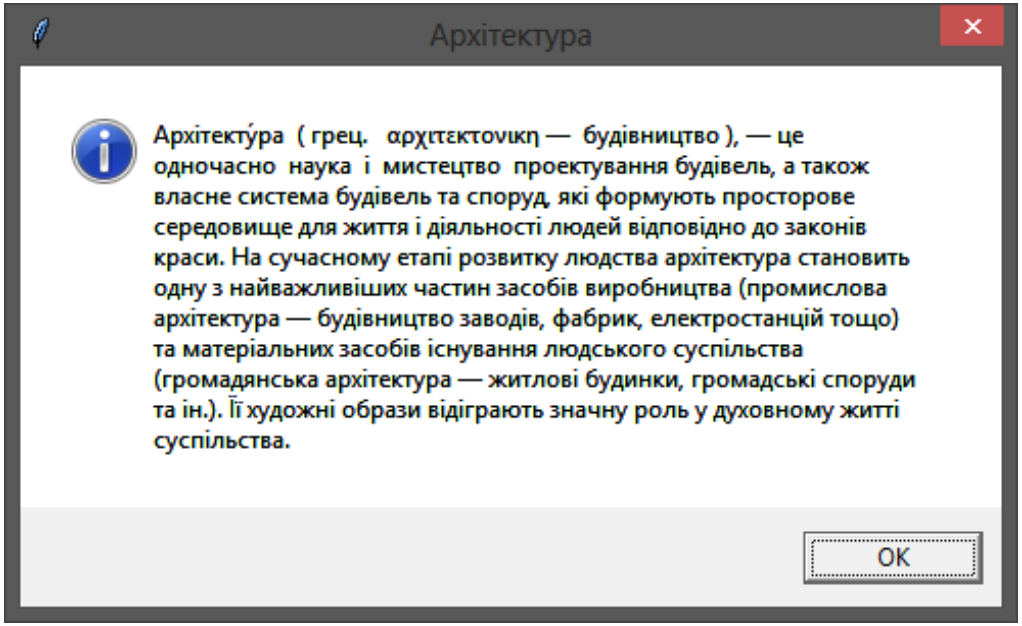
- меню з
  - введенням файлу (меню Файл),
  - вибором розмірів та написання шрифту а також кольорів тексту та фону (меню Опції)
  - зображенням означення з вікіпедії та вибором мови Вікіпедії (Вікі),
- вікно тексту, у яке виводиться текст файлу.

TextViewerWiki є нащадком класу TextViewer.

Клас TextViewerWiki.2



Клас TextViewerWiki.3



Клас має одне власне поле (інші поля успадковуються від TextViewer):

- self. language – мова Вікіпедії

Клас TextViewerWiki.4

Конструктор __init__ викликає конструктор батьківського класу, який викликає внутрішній метод _make_widgets для створення елементів інтерфейсу.

Метод _make_widgets викликає однойменний метод батьківського класу а також будує меню Вікі та приєднує його до меню батьківського класу

Метод displaywiki отримує означення терміну (терміном вважається виділений текст), використовуючи клас WikiDef, а також показує це означення у вікні стандартного повідомлення.

- Якщо термін не знайдено у Вікіпедії, відображається відповідне попередження.

Метод setlanguage обробляє вибір пункту меню встановлення мови.

- Для цього він запускає відповідний діалог (створює об’єкт класу LangOpts)

Резюме

Ми розглянули:

1. Мова HTML. Структура документу HTML
2. Основні теги HTML
3. URL.
4. Веб-клієнти у Python. Пакет urllib
5. Протокол HTTP
6. Методи запиту GET та POST
7. Відправка даних для запиту POST
8. Структурний аналіз HTML-файлів

Де прочитати

1. Обвінцев О.В. Об’єктно-орієнтоване програмування. Курс на основі Python. Матеріали лекцій. – К., Основа, 2017
2. Peter Norton, Alex Samuel, David Aitel та інші - Beginning Python
3. Wesley J. Chun - Core Python Programming - 2001
4. Magnus Lie Hetland - Beginning Python from Novice to Professional, 2nd ed – 2008
5. Mark Lutz - Programming Python. 4th Edition - 2011
6. Прохоренок Н.А. - Python 3 и PyQt. Разработка приложений – 2012
7. Mark Pilgrim - Dive into Python, Version 5.4 - 2004
8. Jim Knowlton - Python Create Modify Reuse – 2008
9. Noah Gift, Jeremy M. Jones - Python for Unix and Linux System Administration
10. John Goerzen -Foundations of Python Network Programming. - 2004
11. 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
13. http://html5book.ru/html-tags/
14. https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html