ІНФОРМАТИКА ТА ПРОГРАМУВАННЯ

Тема 23. Робота з даними у офісних документах

Застосування групи Microsoft Office

- Застосування групи Microsoft Office широко використовуються для рішення різноманітних задач.
- Перш за все, це стосується текстового процесору Microsoft Word та електронних таблиць Microsoft Excel.
- Microsoft Word дає можливість створювати та обробляти складені документи, які включають текст, зображення, таблиці, різноманітні об’єкти.
- Microsoft Excel дозволяє зберігати та обробляти табличну інформацію, виконуючи над нею певні перетворення, що задаються формулами.
- Ці застосування використовують у так званій «малій» автоматизації: коли треба автоматизувати якісь повсякденні операції, але для цих операцій не існує спеціалізованих програмних систем (або ці системи є занадто дорогими).

Бібліотеки Python для роботи з офісними документами

- Велика популярність згаданих застосувань призвела до того, що у світі на сьогодні накопичено величезний обсяг даних, що зберігаються у документах Microsoft Office.
- Ці дані також треба обробляти, аналізувати тощо.
- Тому використання файлів офісних документів, доступ до даних та перетворення цих даних становлять самостійний інтерес.
- Microsoft Word та Microsoft Excel мають власні засоби автоматизації, які називають VBA (Visual basic for Applications).
- Але ці засоби працюють тільки в середовищі самих застосувань.
- Інші мови програмування також надають можливості для обробки даних офісних документів.
- При цьому, наявність застосувань Microsoft Word або Microsoft Excel не є обов’язковою.
- У Python є декілька бібліотек (пакетів) для роботи з офісними документами.
- Ми розглянемо бібліотеки python-docx та openpyxl відповідно для Microsoft Word та Microsoft Excel.

Встановлення python-docx та openpyxl

- Оскільки python-docx та openpyxl є зовнішніми бібліотеками та не входять до стандартного пакету поставки Python, їх треба встановити.
- Встановлення не є складним та здійснюється за допомогою спеціальної програми pip3.
- Для встановлення у Unix (Linux) треба виконати команди:

sudo pip3 install python-docx

та

sudo pip3 install openpyxl

Встановлення python-docx та openpyxl.2

- Для встановлення у MS Windows (якщо Python встановлено у каталог C:\Python34) треба виконати команди:

C:\Python34\Scripts\pip3.exe install python-docx

- або C:\Python34\Scripts\easy_install.exe python-docx

та

C:\Python34\Scripts\pip3.exe install openpyxl

- або C:\Python34\Scripts\easy_install.exe openpyxl
  - На окремих комп'ютерах є складності із встановленям допоміжного пакету lxml. У такому випадку слід використати бінарну інсталяцію останньої версії, яку можна знайти за адресою https://pypi.python.org/simple/lxml/ або на сайті http://matfiz.univ.kiev.ua/pages/39
- Після встановлення можемо імпортувати python-docx та openpyxl або їх внутрішні модулі звичайною інструкцією import:

```
import docx
import openpyxl
```

Короткий огляд структури документу MS Word

- Документ MS Word може містити текст, рисунки, таблиці, інші об’єкти.
- Документ складається з параграфів.
- Кожен параграф має власний стиль та форматування: вирівнювання, інтервал між рядками, відступи рядків тощо.
- Кожний параграф, в свою чергу, складається з текстових потоків (runs або character runs).
- **Текстовий потік** – це рядок разом зі стилем та форматуванням: шрифт, написання тощо.
- Тобто, частини параграфу, які мають різне форматування тексту, будуть належати до різних текстових потоків.
- З точки зору зовнішнього представлення, документ розбивається на розділи (sections).
- Кожний розділ містить властивості, що визначають розташування вмісту документу на сторінці: розмір сторінки, орієнтація сторінки, поля тощо.
- Таблиця у документі складається з клітинок, які організовано у рядки та стовпчики.
- Кожна клітинка може містити декілька параграфів.

Основні можливості python-docx

- python-docx включає класи для роботи з документом MS Word.
- Кореневим класом є клас Document – документ.
- Усі можливості бібліотеки python-docx (у тому числі, інші класи) доступні через властивості та методи об’єкту класу Document.
- Для створення нового порожнього документу необхідно створити об’єкт класу Document:

```
doc = docx.Document()
```

- Для відкриття існуючого документу треба також створити об’єкт класу Document, вказавши ім’я файлу існуючого документу filename

```
doc = docx.Document(filename)
```

- Для того, щоб зберегти раніше створений або відкритий документ у файлі newfilename, треба використати метод save класу Document:

```
doc.save(newfilename)
```

Властивості та методи об’єктів класу Document

Властивість або метод	Опис
add_page_break()	Вставляє у документ новий абзац з розривом сторінки
add_paragraph(text='', style=None)	Додати у документ новий абзац з текстом text та стилем style
paragraphs	Список параграфів документу (об’єктів класу Paragraph)
tables	Список таблиць документу (об’єктів класу Table)

Властивості та методи об’єктів класу Paragraph

Властивість або метод	Опис
add_run(text=None, style=None)	Додати у параграф новий текстовий потік з текстом text та стилем style
paragraph_format	Форматування параграфу (об’єкт класу ParagraphFormat)
runs	Список текстових потоків параграфу (об’єктів класу Run)
style	Стиль параграфу
text	Зчеплений текст усіх потоків параграфу

Властивості та методи об’єктів класу Run

Властивість або метод	Опис
add_break(break_type=6)	Додати у потік розрив (за угодою, - розрив рядка)
bold	Чи є шрифт потоку напівгрубим
font	Шрифт потоку (об’єкт класу Font)
italic	Чи є шрифт потоку нахиленим
style	Стиль потоку
text	Текст потоку
underline	Чи є шрифт потоку підкресленим

Приклад: видалення зайвих пропусків у документі MS Word

- Документ MS Word не повинен містити зайвих пропусків. Зайві пропуски утруднюють форматування та зміну документу.
- Тому часто виникає задача видалення зайвих пропусків між словами у тексті документу MS Word.
- Розв’яжемо цю задачу у Python.
- Всю роботу виконує функція delspaces, якій передається в якості параметра ім’я файлу документу.
- Ця функція відкриває документ, проходить усі параграфи та для усіх потоків параграфу видаляє зайві пропуски.
- Після завершення файл зберігається під новим ім’ям.
- Головна частина програми отримує файл в якості параметра при запуску з командного рядка або запрошує введення, якщо такого параметра немає.

Короткий огляд структури робочої книги MS Excel

- Робоча книга MS Excel (workbook) складається з робочих аркушів (worksheet).
- Кожний аркуш представляє собою таблицю з даними. У клітинках таблиці можуть бути розміщені числа, рядки, дати, дані інших типів.
- Також у клітинках можуть міститись формули, що пов’язують дану клітинку з іншими.
- До клітинки таблиці (робочого аркуша) можна отримати доступ за номером рядка та стовпчика таблиці (нумерація починається з 1) або за ім’ям клітинки.

Короткий огляд структури робочої книги MS Excel.2

- Ім’я клітинки – це рядок, що формується з імені стовпчика та номера рядка.
- Стовпчики називають латинськими літерами або послідовностями латинських літер.
- Спочатку у таблиці йдуть стовпчики з іменами з 1 літери від A до Z, потім – з 2 літер - від AA до ZZ, потім з 3 і більше літер.
- Рядки нумеруються натуральними числами.
- Приклад імені клітинки: ”C5”.
- Щоб відрізняти у клітинках формули від даних типу рядок, встановлено, що формула завжди починається зі знаку «=».
- Наприклад, формула у деякій клітинці “=B1+C1” означає, що дані у цій клітинці будуть сумою даних клітинок “B1” та “C1”.
- Робочий аркуш також може містити графіки або діаграми.
- Дані для побудови діаграм та графіків беруться з робочої книги.

Основні можливості openpyxl

- openpyxl включає класи для роботи з робочою книгою MS Excel.
- Кореневим класом є клас Workbook – робоча книга.
- Усі можливості бібліотеки openpyxl (у тому числі, інші класи) доступні через властивості та методи об’єкту класу Workbook.
- Для створення нової робочої книги з одним робочим аркушем необхідно створити об’єкт класу Workbook:

```
wb = openpyxl.Workbook()
```

- Для відкриття існуючої робочої книги треба викликати функцію load_workbook, вказавши ім’я файлу існуючого документу filename

```
wb = openpyxl.load_workbook(filename)
```

- Для того, щоб зберегти раніше створений або відкритий документ у файлі newfilename, треба використати метод save класу Workbook:

```
wb.save(newfilename)
```

- Для збереження завантаженої робочої книги у файлі з тим же ім’ям, треба викликати

```
wb.save()
```

Доступ до робочого аркуша

- Робочий аркуш – це об’єкт класу Worksheet.
- Отримати доступ до відповідного робочого аркуша робочої книги wb можна декількома способами.
- Можна вибрати так званий активний робочий аркуш, над яким виконувались останні дії

```
ws = wb.active
```

- Можна вказати ім’я робочого аркуша

```
ws = wb[name]
```

- Отримати список імен робочих аркушів можна так:

```
lst = wb.get_sheet_names()
```

Доступ до робочого аркуша.2

- Робоча книга є також ітератором, тобто ми можемо використати цикл по всіх робочих аркушах

```
for sheet in wb:
    print(sheet.title)
```

- Нарешті, можна вказати номер робочого аркуша (починаючи з 0)

```
ws = wb.worksheets[index]
```

- Щоб створити робочий аркуш з ім’ям name, треба викликати метод create_sheet об’єкту класу Workbook

```
ws = wb.create_sheet(title=name)
```

Доступ до окремих клітинок у робочому аркуші

- Нехай, ws – це робочий аркуш. Окремі клітинки аркуша є об’єктами класу Cell.
- Тоді отримати доступ до клітинки можна, вказавши її ім’я

```
c = ws[name]
```

- наприклад,

```
c = ws[name]
```

- або використавши метод cell та вказавши номер рядка (row) та стовпчика (col), наприклад,

```
c = ws.cell(row = 2, column = 4)
```

Доступ до окремих клітинок у робочому аркуші.2

- Властивість Cell.value дає змогу прочитати або змінити значення клітинки, наприклад,

```
c.value = 25
```

- Є можливість одразу додати рядок у кінець таблиці:

```
ws.append(lst)
```

  - де lst – список значень, які будуть записані у клітинки нового рядка.
- Щоб перебрати усі рядки з даними робочого аркуша ws, застосовують властивість ws.rows, а усі стовпчики, - ws.columns
- Окрім цього, ws.min_row та ws.max_row повертають мінімальний та максимальний номери рядків з даними, а ws.min_column та ws.max_ column повертають мінімальний та максимальний номери стовпчиків з даними у робочому аркуші.

Діаграми та графіки

- MS Excel добре вміє зображувати графіки та діаграми.
- Його можливості ширші, ніж у пакета matplotlib, який ми розглядали у темі «Наукові обчислення».
- Кожна діаграма базується на даних, які складаються з так званих «рядів даних».
- Ряд даних – це або частина стовпчика, або частина рядка робочого аркуша.
- Також діаграма може використовувати діапазон даних для відображення на осі OX.
- Для побудови діаграми у openpyxl треба виконати такі кроки:
  - Створити діаграму відповідного типу
  - Додати ряди даних (та значення осі OX)
  - Додати діаграму до робочого аркуша

Діаграми та графіки.2

- Для створення діаграми треба створити об’єкт класу, який описує потрібний тип діаграми.
- Деякі з цих класів:
  - AreaChart – діаграма областей
  - BarChart – діаграма зі стовпчиків
  - LineChart (або ScatterChart) – графік
  - PieChart – кругова діаграма
- Наприклад,
- `chart1 = ScatterChart()`

Діаграми та графіки.3

- Щоб додати ряд даних до діаграми, використовуємо об’єкт Reference, який повертає посилання на діапазон клітинок робочого аркуша, та об’єкт Series, який створює ряд даних:
- `ydata = Reference(ws, min_col=k, min_row=2, max_row=n+1)`
- `s = Series(ydata, xvalues=xdata)`
- `chart1.append(s)`
- Додати діаграму до робочого аркуша можна так:
- `ws.add_chart(chart1, "E1")`
- Після цього діаграма буде розташована, починаючи з клітинки "E1".

Приклад: побудова графіків функцій у MS Excel

- Побудувати графіки функцій у MS Excel
- Дані для графіків отримуються з масивів numpy, які створюються у розглянутому у темі «наукові обчислення» прикладі табулювання функції.
- Будує графіки функція plotfunc1 (plotfunc2 у версії 2).
- Функція виконує дії згідно з розглянутими вище кроками побудови діаграми.

- Версія 2 відрізняється від версії 1 тим, що на одному рисунку зображено декілька графіків.

Додаткові можливості python-docx

- За роботу з таблицями MS Word відповідає клас Table.
- Цей клас, зокрема, дозволяє отримати клітинку таблиці t – об'єкт класу _Cell - за заданим рядком та стовпчиком:

```
c = t.cell(row, col)
```

- Проаналізувати або змінити вміст клітинки можна, застосувавши властивість класу _Cell – paragraphs

```
c.paragraphs
```

- Метод класу Table row_cells(n) повертає послідовність клітинок таблиці з рядка з номером n (нумерація починається з 0).

```
t.row_cells(n)
```

Додаткові можливості python-docx.2

- Властивості

```
t.rows
```

```
t.columns
```

повертають послідовність рядків (стовпчиків) таблиці t.

- Розглянемо ще 2 класи, які знадобляться для нашого наступного прикладу.
- Клас ParagraphFormat дає можливість проаналізувати або змінити параметри форматування абзацу, а клас Font – повертає або змінює шрифт та написання символів текстового потоку.

Властивості об’єктів класу ParagraphFormat

Властивість	Опис
alignment	Вирівнювання параграфу (по лівому краю, по центру тощо)
line_spacing	Інтервал між рядками у параграфі
first_line_indent	Відступ першого рядка від краю параграфу
keep_together	Чи обов’язково тримати весь параграф на одній сторінці
keep_with_next	Чи обов’язково тримати поточний параграф разом з наступним на одній сторінці
left_indent	Поле зліва від краю сторінки до лівого краю параграфу
line_spacing_rule	Визначає один з стандартних інтервалів між рядками: одинарний, подвійний або півтора інтервали
page_break_before	Чи треба вставляти розрив сторінки перед параграфом
right_indent	Поле справа від краю сторінки до правого краю параграфу
space_after	Відстань між даним та наступним параграфом
space_before	Відстань між даним та попереднім параграфом
widow_control	Чи обов’язково залишати на сторінці більше одного рядка параграфу

Властивості об’єктів класу Font

Властивість	Опис
all_caps	Чи потрібно всі літери переводити у верхній регістр
color.rgb	Колір шрифту
name	Ім’я шрифту
size	Розмір шрифту
small_caps	Чи потрібно всі літери переводити у верхній регістр та зображувати меншим розміром
strike	Чи зображувати текст закресленим
subscript	Чи є текст нижнім індексом
superscript	Чи є текст верхнім індексом

Приклад: Злиття даних з файлів MS Word, MS Excel за шаблоном MS Word

- Розглянемо більш змістовний та складний приклад використання бібліотек python-docx та openpyxl.
- Це злиття даних з файлів MS Word, MS Excel за шаблоном MS Word.
- Під злиттям мають на увазі побудову документів, що містять у вказаних місцях дані з інших документів або електронних таблиць.
- При злитті використовують шаблон, у якому вказують місця для вставки даних.
- Потім вибирають дані по одному запису та вставляють у шаблон.
- Останню операцію повторюють для всіх записів з даними.

Приклад: Злиття даних з файлів MS Word, MS Excel за шаблоном MS Word.2

- MS Word має функціональність злиття – це побудова списків розсилки.
- Але, по-перше, дозволяється використання даних тільки з одного джерела, по-друге, не переноситься шрифт при вставці даних з іншого документа MS Word, зокрема, нижні та верхні індекси.
- Складемо програму, яка здійснює злиття даних.
- Нехай місця у шаблоні, у які треба вставити дані, позначаються рядками у фігурних дужках, наприклад ‘{Num}’.
- Джерел даних може бути декілька, вони повинні бути таблицями у документах MS Word або робочими аркушами у книгах MS Excel.
- Одне з полів вважається провідним: кількість повторень злиття відповідає кількості даних провідного параметру.

Злиття даних з файлів MS Word, MS Excel за шаблоном MS Word. Реалізація

- У нашій програмі використаємо три класи: MergeSource, SourceItem та Merger.
- Клас MergeSource призначено для під'єднання до джерел даних та повернення даних по кроках.
- Під’єднання до джерел даних здійснюється у конструкторі класу, а повернення, - як у класі-ітераторі.
  - Тобто, клас підтримує ітераційний протокол.
- Клас MergeSource містить поля
  - self.lead - ім'я поля, яке є провідним параметром
  - self.fields - словник, що має ключами імена полів, а значеннями - об'єкти класу SourceItem
- та методи __init__, __iter__, __next__.

Злиття даних з файлів MS Word, MS Excel за шаблоном MS Word. Реалізація.2

- Клас SourceItem призначено для під'єднання до одного джерела даних – стовпчика таблиці MS Word або стовпчика робочого аркуша книги MS Excel - та повернення елементів даних по кроках.
- Під’єднання до джерел даних здійснюється у конструкторі класу, а повернення, - у методі next.
- Клас SourceItem містить поля
  - self.type - тип джерела даних ('word' чи 'excel')
  - self.rootobj - кореневий об'єкт документ (Document) або робоча книга (Workbook)
  - self.islead - чи є поле провідним
  - self.obj - об'єкт з даними: клітинка аркуша Excel (Cell) або клітинка таблиці Word (_Cell)
  - self.parent - об'єкт, що містить self.obj: аркуш Excel (Worksheet) або таблиця Word (Table)
  - self.row – номер поточного рядка, з якого вибираються дані
  - self.col – номер стовпчика, з якого вибираються дані

Злиття даних з файлів MS Word, MS Excel за шаблоном MS Word. Реалізація.3

- Клас SourceItem містить методи __init__, _findexcel, _findword, next.
- Методи _findexcel та _findword викликаються з конструктора та шукають дані відповідного поля у аркуші MS Excel або таблиці MS Word.
- Метод next повертає наступний елемент даних для заданого поля.
- Якщо поле є провідним та дані закінчились, - цей метод ініціює виключення StopIteration.

Злиття даних з файлів MS Word, MS Excel за шаблоном MS Word. Реалізація.4

- Клас Merger призначено для злиття даних згідно шаблону, який є документом MS Word.
- При створенні об’єкту класу Merger йому повинні бути передані: ім’я файлу-шаблону, словник імен полів злиття та файлів, у яких треба шукати джерело даних відповідного поля, ім’я провідного параметру.
- Клас Merger використовує клас MergeSource для під'єднання до джерел даних та повернення даних по кроках.
- Клас Merger містить поля
  - self.indoc - об'єкт Document з файлу-шаблону
  - self.mergesrc - об'єкт класу MergeSource - містить джерела даних та повертає записи з даними злиття
  - self.outfile - ім'я файлу-результату
  - self.outdoc - об'єкт Document файлу-результату

Злиття даних з файлів MS Word, MS Excel за шаблоном MS Word. Реалізація.5

- Клас Merger містить методи __init__, merge, _process_template, _process_paragraph, _process_run, _get_field_pos.
- Метод merge здійснює злиття, використовуючи шаблон та об’єкт self.mergesrc.
- Внутрішні методи _process_template, _process_paragraph та _process_run здійснюють один крок злиття відповідно для всього шаблону, одного параграфу, одного текстового потоку.
- Метод _process_run використовує внутрішній метод _get_field_pos для визначення першої позиції одного з полів злиття у текстовому потоці (якщо таке поле є).
- Внутрішні методи також використовують функції para_format_copy та run_format_copy для копіювання форматування абзацу та текстового потоку у новий документ, оскільки ця функціональність відсутня у python-docx.
- Основна частина програми вводить параметри з конфігураційного файлу, створює об’єкт класу Merger та здійснює злиття.

Резюме

- Ми розглянули:
  1. Встановлення python-docx та openpyxl
  2. Основні можливості python-docx
  3. Властивості та методи об’єктів класів Document, Paragraph, Run
  4. Основні можливості openpyxl
  5. Доступ до робочого аркуша
  6. Доступ до окремих клітинок у робочому аркуші
  7. Діаграми та графіки
  8. Додаткові можливості python-docx

Де прочитати

1. https://python-docx.readthedocs.org/en/latest/
2. http://openpyxl.readthedocs.io/en/default/index.html
3. http://habrahabr.ru/post/232291/
4. http://habrahabr.ru/post/99923/
5. Леонтьев Виталий. Microsoft Office. – 2007. http://www.e-reading.club/book.php?book=84357