# Python 시작하기

## Python

2021/05/15 10:20 <http://blog.naver.com/550sn/222332202508>



드디어 제가 Python을 다루기로 마음먹었습니다.

소소한 업무 자동화를 꿈꾸며, 여러 툴들을 이용해 왔지만, 그 끝에 언젠가는 이 Python을 다뤄봐야지 라고 마음을 먹어 왔지만, 지금까지 엄두가 안났었는데, 이제 시작해 보려고 합니다.

뭐, 개발자 출신도 아니고 실력이 그리 뛰어난것도 아니어서 깊은 내용을 다루는 데에는 한계가 있겠지만, 잔마왕 스타일 로 하나씩 풀어나가 볼까 합니다.

이러면서 공부하는거죠 뭐~~~

1. 설치하기

Python 설치 그리 어렵지 않습니다. 이미 많은 분들이 알고 계신것처럼 무료로 제공되는 언어인지라, Python 공식 사이 트에 접근하면 무료로 다운로드 받을 수 있습니다.



### [Download Python](https://www.python.org/downloads/)

[The official home of the Python Programming Language www.python.org](https://www.python.org/downloads/)

생각하기 귀찮고, 고민하기 귀찮아서 그냥 최신 버전으로 그냥 다운로드 받고 설치 하고 옵션버튼도 그냥 다 선택하고 깔 았습니다.

이런 좋은 소스를 풀어주는 선량한 개발자들이 뭔가 함정을 파놓거나 하지는 않았으리라는 믿음으로.. (너무 순진한 걸까요?)



저는 어짜피 전문 개발자도 아니고, 실력이 출중한것도 아니니, 그냥 많은 사람들이 사용하는 IDE도 그대로 끌어와서 사용 하렵니다. PyCharm을 설치하기로 했습니다 .이 또한 공식 사이트에서 쉽게 다운로드 받을 수 있습니다.

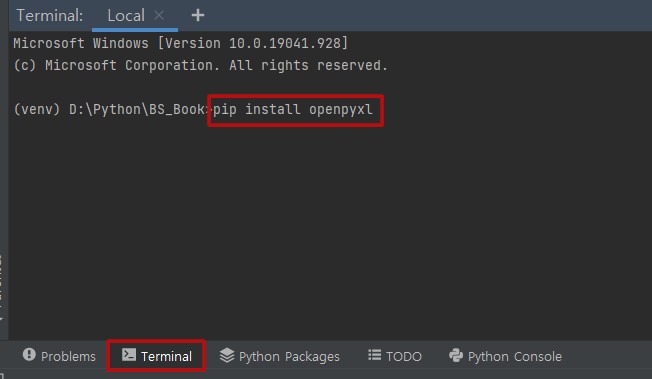
[**다운로드 PyCharm: JetBrains가 만든 전문 개발자용 Python IDE** 최신 버전 다운로드: PyCharm (Windows, macOS, Linux) www.jetbrains.com](https://www.jetbrains.com/ko-kr/pycharm/download/#section%3Dwindows)

마찬가지로 생각하기 귀찮아서... 체크박스 다 체크하고 설치했습니다.

1. Test 코드 작성하기 (Excel 다루기)

Python이 굉장히 좋은점 중의 하나는 필요한 기능이 있을때 Library를 다운로드 받으면 필요한 기능을 굉장히 쉽게 사용 할 수 있다는 점이 되겠습니다.

Library를 설치하는 방법은 Pycharm에서 화면 하단에 있는 "Terminal"을 선택한 다음 Terminal 창에서 pip install 설치할 라이브러리 이름



위와 같은 방식으로 실행해 주면 필요한 라이브러리가 깔립니다.

설치된 라이브러리는 코드를 작성할때 import 구문을 사용하여 불러오면 제공되는 기능을 잘 사용할 수 있습니다. 사용법 을 익히는 것이 쉽지만은 않지만, 파이썬의 최고 장점은 이 언어를 사용하는 사람이 워낙 많고, 그래서 참고할만한 자료가 매우 많으니 Google에서 검색만 잘해도 필요한 왠만한 기능들은 모두 찾아서 구현할 수 있습니다.

저는 예시로 Excel을 다루기 위한 대표적인 library중의 하나인 'openpyxl'을 호출하고, 특정 파일의 특정한 Cell 위치에 특정 값을 입력한 다음 다른이름으로 저장하는 코드를 만들어 보겠습니다.

import openpyxl WB = openpyxl.load\_workbook("D:\Test\Test\Test.xlsx") WS = WB["Test"] WS["B2"] = "Test" WB.save("D:\Test\Test\S\_Test.xlsx")

Test라는 이름의 엑셀파일을 불러와서 "Test"라는 이름의 Sheet의 "B2"위치에 "Test"라는 값을 입력하고 저장하는 매우 단순한 코드입니다.

실제 실행해 보면 몇초 안되어 실행이 완료되는데, 이는 엑셀을 실행하고 값을 입력하고 저장하고 다시 닫는 동작으로 고 려하여 생각해 보면 정말 빠른 동작임을 알 수 있습니다.

이 또한 Python과 같은 언어를 활용한 RPA의 최고 장점이 아닐까 하는 생각을 해 봅니다. 이왕 시작한거 이제는 Python으로도 필요한 기능들을 하나씩 구현해 볼까 합니다.

제 사이틀 방문해 주시는 분들에게 많은 도움이 되면 좋겠고, 무엇보다 제 실력이 이번 기회로 많이 늘 수 있는 기회를 얻

을 수 있으면 좋겠습니다.

# Excel을 삼킨 Python (openpyxl)

## Python

2021/05/18 04:57 <http://blog.naver.com/550sn/222336080574>



Python의 가장 강력한 점 중의 하나는 필요한 기능이 있을때, 필요한 기능을 제공해 주는 Library만 설치해서 활용해도 상 당수의 원하는 기능을 구현할 수 있다는 것이라고 할 수 있습니다.

Python을 이용해 어떤 기능을 구현하고 그 결과를 저장해야 할 일이 있을 때 그 결과 데이터가 대용량의 표와 같은 형태를 가지고 있다면, 엑셀 형식으로 변환하여 저장하는 것이 도움이 될 수 있습니다.

반대로, 내가 다루고자 하는 어떤 조건값을 읽어 들어오는 일이 있을 때 이들 값들을 엑셀 파일의 형태로 보관하고 있는 경 우도 많기 때문에 이들 엑셀 파일을 파이썬에서 다루는 것은 사용처가 많은 일이라고 할 수 있습니다.

Python에서 엑셀을 다루는 대표적인 라이브러리중 하나가 바로 "openpyxl"입니다.

이 "openpyxl" 라이브러리에 대한 상세 정보는 다음의 공식 사이트에서 찾아볼 수 있습니다.

[**openpyxl - A Python library to read/write Excel 2010 xlsx/xlsm files — openpyxl 3.0.7 documentation** openpyxl stable Tutorial Simple usage Performance Optimised Modes Inserting and deleting rows and columns, moving ranges of cells Working with Pandas and NumPy Charts Comments Working with styles Additional Worksheet Properties Conditional Formatting Pivot Tables Print Settings Using filters and sor... openpyxl.readthedocs.io](https://openpyxl.readthedocs.io/en/stable/)

1. 설치 및 시작하기

먼저 openpyxl을 사용하기 위해서는 라이브러리 설치부터 해야 합니다. python 터미널에서

pip install openpyxl

을 입력하면 openpyxl 라이브러리 설치가 가능합니다.

설치되고 난 다음에는 Python 코드를 구성할때 해당 라이브러리를 불러오는 구문만 추가하면 필요한 기능을 불러와서 활 용이 가능합니다.

import openpyxl

이 구문만 추가하면 openpyxl에서 제공하는 기능들을 활용하여 필요한 기능을 구현할 수 있습니다.

1. 파일 생성하기, 불러오기 그리고 저장하기

파이썬에서 엑셀 파일을 새롭게 만드는 경우에는 Workbook 개체만 하나 새로 추가해 주면 됩니다. wb = openpyxl.Workbook()

기존에 있던 파일을 불러올 때는 load\_workbook 이라는 명령어를 사용하면 됩니다.

wb = openpyxl.load\_workbook("불러올 파일 경로")

저장 할때는 save 명령어를 사용하면 됩니다. 단 이때 Workbook으로 선언된 개체에 save 명령어를 추가해야 합니다. wb.save("저장할 파일 경로")

이와같이 Workbook 개체를 하나 선언하고 지정해 줌으로써 Python에서는 해당 파일을 다루게 됩니다.

1. 엑셀에 필요한 값 입력하기

우리가 엑셀에서 어떤 작업을 할 때 실질적인 값은 엑셀 파일의 Sheet 안에 있는 Cell 하나하나에 값을 입력하게 됩니다. 따라서 Sheet를 선택하고, Sheet에서의 Cell 위치를 선택해서 해당 위치에 값을 추가하면 필요한 값을 입력할 수 있습니 다.

엑셀 파일의 "Sheet1" 이라는 이름의 Sheet에 "A2"위치에 "Test"라는 값을 입력하고 싶으면 아래와 같이 차래로 코드를 구성해 주면 됩니다.

ws = wb["Sheet1"] ws["A2"] = "Test"

여기서 어떤 Sheet인지 Sheet 이름은 모르겠고, 엑셀 파일을 실행했을때 가장 먼저 뜨는 Sheet의 가장 아래 데이터가 없 는 첫번째 행에 첫번째 칸부터 차례대로 1, 2, 3의 값을 넣겠다고 하면 다음과 같은 방식으로도 입력이 가능합니다.

ws = wb.active ws.append([1, 2, 3])

Workbook을 선택한 상태에서 active라는 명령어는 wb을 불러온 상태를 기준으로 active상태에 있는 Sheet를 선택하라 는 의미이고, Worksheet 하부의 append라는 명령어는 이후에 배열정보를 추가하라는 의미가 됩니다.

엑셀의 셀 주소 중 Colum 정보는 숫자의 형태가 아니라 A, B, C, ... 와 같이 알파벳이 증가하는 형태의 구조를 가지고 있 어서 Column 값을 증가 시키면서 값을 다뤄야 할 때는 불편함이 있을 수 있습니다.

이를 보조할 수 있는 방법이 두가지가 있는데, cell 이라는 명령어를 활용하여 값까지 바로 입력하는 방법이 있고, "get\_column\_letter" 기능을 활용하여 숫자값을 문자로된 Column 값으로 변환하여 입력해 주는 방법이 있습니다. cell을 활용하는 방법은 아래 코드와 같습니다.

ws.cell(column=4,row=6,value="Cell\_Input")

위와 같은 방법으로 하면 4번째 Colum, 6번째 Row. 즉, "D6" 칸에 "Cell\_Input"이라는 값을 입력하게 됩니다.

"get\_column\_letter"기능을 활용하기 위해서는 해당 기능을 불러오는것부터 실행해야 하며, 아래의 코드와 같이 구성할 수 있습니다.

from openpyxl.utils.cell import get\_column\_letter ws[get\_column\_letter(6) + str(10)] = "Get Test"

첫줄은 "openpyxl" 라이브러리 → "utils" → "cell" 이라는 곳 아래에 있는 "get\_column\_letter"이라는 기능을 끌어와서 사용할수 있게 하라는 의미입니다.

두번째 줄에서 Cell 주소를 입력하는 구문에서 "get\_column\_letter"라는 기능을 끌어왔으며 6이 입력되어 있어서 6번째 엑셀 Colum의 주소값을 반환해 달라는 내용이 됩니다. 따라서 위의 구문을 따를 경우 "F10"에 해당하는 위치에 "Get Test"라는 값을 입력하게 됩니다.

1. 엑셀에서 필요한 값 가져오기

필요한 값을 가져오는 방법은 Cell 주소에 ".value" 를 추가하여 해당 위치의 값을 가져오는 방식으로 필요한 값을 가져올

수 있습니다.

ws["B2"] = "Test" ReadTest = ws["B2"].value print(ReadTest)

위와 같이 입력된 경우 worksheet의 "B2" 위치에 "Test"라는 값을 입력했고, 이 입력된 값은 ReadTest 라는 변수에 그 값으로 저장되었으며,

print 명령에 의해 실행창에서 "Test"라는 값을 인쇄하는 것을 볼 수 있습니다.

1. 그 외

그 외에도 엑셀에서 보다 상세한 입력이 필요한 항목이 있거나 편집이 필요한 내용이 있을 경우에는 많은 부분을 openpyxl 내에서 제공하는 함수로 조정할 수 있습니다.

여기서 추가로 설정할 수 있는것으로는 하이퍼 링크, 폰트, 행/열의 높이/폭 등 대부분의 값들을 바꿀 수 있습니다.

필요한 내용이 있을 경우에는 Google에서 "openpyxl 필요한 내용" 과 같은 방식으로 검색을 하면 필요한 정보를 정말 무 궁무진하게 받아볼 수 있고, 내게 필요한 정보도 바로 바로 복사해 가져오는 방식으로 구현하면 왠만한 내용은 해결할 수 있습니다.

그리고, 실제 파이썬을 이용하여 데이터를 다뤄 보면

1. 엑셀 파일을 전혀 실행하는 동작도 없는데 값이 바뀌어져 있다
2. 정보를 처리하고 종료하는데 걸리는 시간이 굉장히 짧게 걸린다 라는 점에서 놀라게 됩니다.

화면에 사람이 보기 좋은 방식으로 띄우는것을 생략하는 것 만으로도 굉장히 빠른 속도로 정보 처리가 된다는 반증일 수 있을 것 같습니다.

실제 컴퓨터가 정보를 바꾸는 중간 과정을 확인할 수가 없고 결과만 바로 보게 하는 면이 있기에, 동작의 중간 중간을 이해 하기가 까다로운 면이 있지만, 그만큼의 속도와 효율성이 있기에 한번쯤 익혀볼만한 가치는 충분히 있다고 생각됩니다.

※ 유튜브 영상을 다루면서 만든 샘플 파일은 다음과 같습니다. **첨부파일**

Excel\_Study.py [파일 다운로드](http://blogattach.naver.net/be2ba2196c5b3383ae4d2d1b26c5bdcc643f7b/20210510_5_blogfile/550sn_1620654325552_JN6nGg_py/%2FExcel_Study.py)

# [공지사항] 파이썬 시리즈를 준비하며

## Python

2021/05/19 12:57 <http://blog.naver.com/550sn/222357145367>



"??를 삼킨 파이썬"이라는 이름의 시리즈를 준비하고 있습니다.

(이미 첫번째 영상은 올렸습니다.) 엑셀, Web, PPT...

라이브러리만 끌어오고 조금만 찾아보면 뚝딱 뚝딱 기능을 구현하는 파이썬...

그리스 신화에 나오는 큰 뱀인 피톤을 어원으로 하고 있어서 로고가 뱀의 모양인데, 라이브러리를 꿀꺽 꿀꺽 삼키며 기능 을 구현해 내는 모습과도 어울리는 로고가 아닌가 하는 생각도 해 봅니다.

게시 주기를 변경하고자 합니다.

기존의 화, 토요일 게시하던 주기를 **화, 금요일로 변경**하겠습니다. 블로그 주제의 특성상 주말에는 시청자 분이 줄어드는 특성을 보이는데,

토요일 게시는 아무래도 적절하지 않은것 같아서 금주부터 변경하겠습니다.

# Web을 삼킨 Python (selenium)

## Python

2021/05/21 04:18 <http://blog.naver.com/550sn/222336292010>



Python으로 활용이 가능한 업무 자동화 응용 분야는 많은 것이 있겠지만, 빠지지 않고 권해지는 영역중 하나가 웹크롤링 영역, 즉 인터넷을 검색하며 내게 필요한 정보를 자동으로 끌어오는 기능입니다.

이와 같은 웹 크롤링시 가장 많이 사용하는 라이브러리 중의 하나가 selenium + 웹드라이버의 조합입니다.

selenium의 경우에는 앞서 VBA에서 selenium basic을 다루면서 관련 내용을 다룬적이 있어서 보다 편하게 접할 수 있습 니다.

공식적인 selenium webdriver 라이브러리와 관련된 정보는 다음의 링크에서 확인이 가능합니다. [**WebDriver :: Documentation for Selenium**](https://www.selenium.dev/documentation/en/webdriver/)

[WebDriver WebDriver drives a browser natively, as a user would, either locally or on a remote machine using the Selenium server, marks a leap forward in terms of browser automation. Selenium WebDriver refers to both the language bindings and the implementations of the individual browser controlling ... www.selenium.dev](https://www.selenium.dev/documentation/en/webdriver/)

※ 실제 기능을 구현하다 보면 공식적인 문서를 찾기 보다는 구박사 검색을 더 많이 활용하게 되긴 합니다.

1. selenium 라이브러리 설치 및 시작하기

라이브러리를 설치하는 것은 그리 어렵지 않습니다. Python 터미널에서 pip install selenium

상기 명령어만 입력하면 설치가 완료 됩니다.

추가로 설치해 줘야 하는 것이 사용할 웹 드라이버 입니다.

크롬을 사용할 경우에는 아래의 링크에서 내가 사용하는 크롬과 일치하는 버전의 크롬 웹 드라이버를 다운로드 받아 둬야 합니다 .

### [ChromeDriver - WebDriver for Chrome - Downloads](https://chromedriver.chromium.org/downloads)

[Current Releases If you are using Chrome version 91, please download ChromeDriver 91.0.4472.19 If you are using Chrome version 90, please download ChromeDriver 90.0.4430.24 If you are using Chrome version 89, please download ChromeDriver 89.0.4389.23 If you are using Chrome version 88, please chromedriver.chromium.org](https://chromedriver.chromium.org/downloads)

그리고, 이 드라이버는 python에서 실행시 해당 파일을 찾아서 실행 해야지만 하므로 Python 코드가 있는 프로젝트 폴더 에 같이 복사해 두거나, 내가 정확히 알고 있는 폴더에 저장해 두고, 해당 파일 경로를 정확히 입력해 줘야 합니다.

(또는 Path 로 지정해 두어 아무곳에서나 바로 읽을 수 있게 설정해 둬야 합니다.)

사람에 따라서는 자주 사용하는 웹드라이버가 달라서 크롬 이외의 드라이버를 설치하여 사용도 가능하며, 이는 어디까지 나 Python 코드에서 실행할 드라이버를 선택하여 변경이 가능합니다.

selenium + chrome 드라이버의 조합으로 크롤링일 할 경우 다음과 같이 초기화 입력이 필요합니다. from selenium import webdriver WD = webdriver.Chrome("chromedriver.exe")

이는 selenium 라이브러리에 있는 기능중 webdriver를 끌어와 사용하겠다는 의미이며, Chrome 드라이버를 가져와서 활용하겠다는 의미가 되겠습니다.

1. 웹 정보 가져오기

webdriver를 이용하여 필요한 정보를 가져올 때 주로 사용하는 명령어들을 알아 보면 다음과 같이 정리할 수 있습니다. (제가 주로 사용하는 명령어 위주이며, 이보다 훨씬 다양하고 많은 기능을 라이브러리에서 제공하고 있습니다.) WD.get("검색할 사이트 주소")

get 명령어는 해당 사이트로 찾아 들어가는 명령어 입니다. WD.find\_elements\_by\_class\_name("class 이름")

내게 필요한 개체 값을 찾는 명령어 입니다. (위의 명령어는 class 이름을 기준으로 찾으라는 명령어 입니다.) find\_elements... 가 있고, find\_element...가 있습니다. 여러개의 값을 찾느냐, 첫번째 하나의 값만을 찾느냐의 의미 차 이가 있으니 구분해서 사용해야 합니다.

개체를 찾을때 id, class, name, Tag, 등의 여러 Target 값을 기준으로 하여 찾을 수 있으며, 찾은 개체의 하위 개체를 찾 을 때에는 find\_elements... 로 찾은 다음 그 뒤에 다시 "."으로 연결하여 find\_elements... 로 추가 검색을 하는 방법을 취할 수 있습니다.

찾은 개체의 상세 정보 중 가져오고자 하는 값은 다시 뒤에 "."으로 연결하여 가져올 값을 설정해 주면 됩니다. WD.find\_element\_by\_class\_name("class 이름").text # 화면에 표시되는 text WD.find\_element\_by\_class\_name("class 이름").get\_attribute('href') # 하이퍼 링크 주소 WD.find\_element\_by\_class\_name("class 이름").get\_attribute('src') # 소스 주소

등과 같이 필요한 값들을 추가로 발췌하여 필요한 값을 가져올 수 있습니다.

※ find\_elements로 여러 개체를 가져왔을때에는 "[가져올 순번]" 값을 . 앞에 넣어줘야 합니다. WD.find\_elements\_by\_class\_name("class 이름")[0].text # 첫 개체의 text WD.find\_elements\_by\_class\_name("class 이름")[1].get\_attribute('href') # 둘째 개체 링크 주소 WD.find\_elements\_by\_class\_name("class 이름")[2].get\_attribute('src') # 셋째 개체 소스 주소 또는 어떤 동작을 지시할 수도 있습니다. 가장 대표적인 예가 클릭이 될것 같습니다.

WD.find\_element\_by\_class\_name("class 이름").click()

결국 중요한것은 내가 가져오고자 하는 웹 사이트의 구조를 분석하고, 여기 필요한 값을 가져오게 만드는 것이 될 듯 합니 다.

1. 직접 만들어 보기

이번에 만들어볼 소스는 앞서 다루어 보았던 RPA 기능 구현 과제, ridibook에서의 베스트 셀러 정보 가져오기를 해 보겠 습니다.

from selenium import webdriver import openpyxl WD = webdriver.Chrome("chromedriver.exe") WD.get("https://ridibooks.com/bestsellers/general") WB = openpyxl.load\_workbook("B\_Result.xlsx") WS = WB.active while True: B\_Title = WD.find\_elements\_by\_class\_name("title\_text") # 책제목의 Class Name B\_Link

= WD.find\_elements\_by\_class\_name("thumbnail\_btn") # 링크주소 B\_IMG = WD.find\_elements\_by\_class\_name("thumbnail") # 이미지 B\_Author = WD.find\_elements\_by\_class\_name("author") # 저자 B\_Star = WD.find\_elements\_by\_class\_name("RSGBookMetadata\_StarRate") # 별점 B\_Price = WD.find\_elements\_by\_class\_name("meta\_price\_info") # 가격 for i in range (0,len(B\_Title)): try: WS.append ([B\_Title[i].text,B\_Link[i].get\_attribute("href"), B\_IMG[i].get\_attribute("src"), B\_Author[i].text, B\_Star[i\*2].find\_element\_by\_class\_name("StarRate\_Score").get\_attribute("textContent"), B\_Price[i].text]) except: WS.append([B\_Title[i].text, B\_Link[i].get\_attribute("href"), B\_IMG[i].get\_attribute("src"), B\_Author[i].text,"",B\_Price[i].text]) try: WD.find\_element\_by\_class\_name("btn\_next").click() except: break WB.save("Result.xlsx") WD.quit()

위의 소스가 정상적으로 구동하기 위해서는

1. selenium이 설치 되어 있어야 하고
2. 소스 파일이 있는 폴더에 Chrome 드라이버인 "chromedriver.exe" 파일이 설치되어 있어야 합니다.
3. 소스 파일이 있는 폴더에 "B\_Result.xlsx" 라는 파일이 있어야 합니다.

※ 또는 파일 경로를 바꿔서 소스를 수정해 둬도 됩니다.

만들어진 소스는 다음과 같습니다. **첨부파일**

Sel\_Study.py [파일 다운로드](http://blogattach.naver.net/dc49c073613f38e4c82a4d7f43a2d8a4065cad4a8c/20210509_33_blogfile/550sn_1620492140911_kISaqy_py/%2FSel_Study.py)

※ Youtube 영상에서 만들었던 소스는 다음과 같습니다.

영상을 만들면서 일부 값이 다르게 입력되었을 수도 있지만, 기본 방향과 내용은 대동 소이합니다. **첨부파일**

Sel.py

[파일 다운로드](http://blogattach.naver.net/1c8900b3a6f9f82408ea8dbf83621864c69c6d8c0d/20210510_128_blogfile/550sn_1620654700900_EU648V_py/%2FSel.py) **첨부파일** Result\_Format.xlsx [파일 다운로드](http://blogattach.naver.net/e97cf54e506263d6ff197b4b7197e8933b669a2f/20210510_81_blogfile/550sn_1620654765413_810AI2_xlsx/%2FResult_Format.xlsx) **첨부파일** B\_Result.xlsx

[파일 다운로드](http://blogattach.naver.net/53c64fffecbfb76b47a5c2f0cc2d572b89d322c18c/20210510_297_blogfile/550sn_1620654782284_XM2x1V_xlsx/%2FB_Result.xlsx)

※ Youtube 영상을 만드는 과정에서 실수한 부분이 있습니다. (위의 소스에서는 해당 부분이 수정되어 있습니다.) 엑셀에 정보를 Write 하는 구문에서 이미지 소스 뿐만 아니라 하이퍼 링크 주소에서도

"https://ridibooks.com" 부분을 제외하고 저장해야 합니다.

# Web을 삼킨 Python 2 (Beautifulsoup4)

## Python

2021/05/25 06:00 <http://blog.naver.com/550sn/222336745945>



앞서 Python에서 selenium 과 webdriver를 설치하여 웹에서 내게 필요한 정보를 찾아가는 과정을 알아봤습니다.

이번에도 Python을 사용한다는 점은 동일하지만, HTML 문서를 파싱하는 툴을 "Beautifulsoup4"로 바꿔 봤습니다. Beautifulsoup 라이브러리에 대한 공식 문서는 아래의 링크를 참고하시면 좋을 듯 합니다.

### [Beautiful Soup Documentation — Beautiful Soup 4.9.0 documentation](https://www.crummy.com/software/BeautifulSoup/bs4/doc/)

[Quick Start Here’s an HTML document I’ll be using as an example throughout this document. It’s part of a story from Alice in Wonderland : html\_doc = """<html><head><title>The Dormouse's story</title></head>](https://www.crummy.com/software/BeautifulSoup/bs4/doc/)

[<body> <p class="title"><b>The Dormouse's story</b></p> <p class="story">Once upon a time the... www.crummy.com](https://www.crummy.com/software/BeautifulSoup/bs4/doc/)

앞서 살펴본 webdriver를 활용하는 방식은 인터넷 브라우저를 실행 시키고 브라우저를 통하여 정보를 받고, 표현한 다음, 내게 필요한 정보를 찾아가는 방식이었습니다.

반면 Beautifulsoup를 활용하여 파싱을 할 때는 인터넷 브라우저를 실행하지 않고, HTML 문서 자체를 분해/분석하여 필 요한 정보를 가져갑니다.

기본적인 구조나 정보를 가져오는 방식을 보면 셀레리움을 이용하던 방식과 유사한 모습을 보이지만, 문법이 다르기도 하 지만, 약간 더 복잡하고 어려운 느낌을 받을 수 있습니다. 적당한 선에서 알아서 해주면 좋겠는데, 더 번거롭게 많은 정보 를 요구하기도 합니다.

반면, 실제 실행해 보면 엄청난 속도 차이를 보입니다. HTML 문서를 기반으로 이미지를 불러오고, 사용자가 볼 수 있게 재배치하여 보여주는 과정을 생략하는데, 생각보다 이 속도차이가 어마어마하다고 할 수 있습니다.

1. Beautifulsoup4 라이브러리 설치 및 시작하기 Python에서 라이브러리를 추가하는 것은 어렵지 않습니다.

다만, 한가지 Beautifulsoup4는 HTML 문서를 분해 분석하는 툴인데, 필요한 인터넷 문서를 가져오는 라이브러리도 별도 로 가져와야 합니다.

pip install beautifulsoup4 pip install requests

여기서 requests라는 라이브러리가 웹 페이지에 사이트 정보를 요청하고, 그 값을 받을 수 있는 그 역할을 실행합니다.

python을 실행하고 코드 처음에 beautifulsoup4와 requests를 모두 호출하고 시작해야 합니다. from bs4 import BeautifulSoup import requests

1. 웹정보 가져오기

필요한 웹 정보를 가져오고 HTML 문서를 분석하기 위해서는 다음과 같이 명령어를 내려주면 됩니다. WebPage = requests.get('검색할 주소') soup = BeautifulSoup(WebPage.content, "html.parser") requests.get에서는 해당 주소에서의 HTML 문서를 가져옵니다.

가져온 문서를 WebPage라는 변수로 저장해 준 다음, BeautifulSoup 라이브러리에서 파싱 작업을 합니다.

이제 필요한 값들을 찾아와야 하는데요. 찾아오는 명령어는 다음과 같습니다.

soup.findAll("Tag이름", {"Target 명칭" : "Target 내용"}) soup.find("Tag이름", {"Target 명칭" : "Target 내용"}) 여기서 findAll을 하면 여러개의 개체를 모두 찾아오고, find를 하면 첫번째 내용만 가져옵니다.

만약 Class 이름이 "title\_text"이고, Tag이름이 "span"인 개체를 가져와야 할 경우에는 다음과 같이 코드를 작성하면 됩 니다.

soup.findAll("span", {"class" : "title\_text"})

그리고, 찾은 개체 안에서 다시 개체를 이어서 찾아야 할 경우에는 ".find" 또는 ".findAll"로 하위 개체로의 검색을 이어가 면 됩니다.

찾은 개체에서 필요한 값을 가져올 때는 "."이후 가져올 개체 정보를 입력하면 됩니다. 단, "findAll"을 사용했을 경우에는 몇번째 개체인지에 대한 정보가 포함되어야 합니다.

soup.findAll("Tag이름", {"Target 명칭" : "Target 내용"})[0].get\_text() #1번째 개체의 text soup.findAll("Tag이름",

{"Target 명칭" : "Target 내용"})[1].attrs['href'] #2번째 개체의 링크주소 soup.findAll("Tag이름", {"Target 명칭" : "Target 내용"})[2].attrs['src'] #3번째 개체의 소스주소

BeautifulSoup는 문서를 파싱하긴 하지만, 링크를 찾아 클릭하거나 값을 입력하는 등의 동작을 할 수는 없습니다. 그렇기 때문에 클릭을 했을때 일어나는 동작에 대해서는 소스를 분석하고, 다시 requests를 통해 정보를 요청하고 그 결 과를 받아야 합니다.

특정 하이퍼 링크에서 링크 걸린 주소를 찾아가야 하는 상황이라면, 아래와 같이 정리해 줘야 합니다.

Url = soup.find("Tag이름", {"Target 명칭" :"Target 내용"}).attrs['href'] # 원하는 링크 주소 WebPage = requests.get(Url) # 해당 링크 정보 요청

앞서 살펴본 Selenium + WebDriver의 경우와 다르게 Beautifulsoup에서 어떤 동작을 제어할 수는 없기 때문에 예상되 는 또는 필요한 동작에 대해서 하나하나 추적해서 직접 명령을 내려줘야 하기 때문에 아무래도 개발 난이도는 Selenium

+ WebDriver때에 비해서 훨씬 어렵다고 할 수 있습니다.

1. 직접 만들어 보기

이번에 만들어볼 소스도 앞서 지속적으로 솔루션만 바꿔서 다루어 보았던 RPA 기능 구현 과제, ridibook에서의 베스트 셀 러 정보 가져오기를 해 보겠습니다.

from bs4 import BeautifulSoup import requests import openpyxl WP = requests.get("https://ridibooks.com/ bestsellers/general") Soup = BeautifulSoup(WP.content, "html.parser") WB = openpyxl.load\_workbook("Result\_Format.xlsx") WS = WB.active while True: B\_Title = Soup.findAll("span",{"class" : "title\_text"}) #책제목 B\_Link = Soup.findAll("a", {"class" :"title\_link"}) #링크 'href' B\_Img = Soup.findAll("img", {"class" : "thumbnail"}) #이미지 'src' B\_Author = Soup.findAll("p", {"class" : "author"}) #저자 B\_Star = Soup.findAll("span",{"class" : "RSGBookMetadata\_StarRate"}) # 별점 "span" "StarRate\_Score" B\_Price = Soup.findAll("div", {"class" : "meta\_price\_info"}) # 가격 for i in range (0,len(B\_Title)): try: WS.append([B\_Title[i].get\_text().strip(), "https://ridibooks.com" + B\_Link[i].attrs['href'], "https://img.ridicdn.net/cover/" + B\_Link[i].attrs['href'].split("/")[-1] + "/large#1", '

'.join(B\_Author[i].get\_text().split()), B\_Star[i\*2].find("span", {"class" : "StarRate\_Score"}).get\_text(), ' '.join(B\_Price[i].get\_text().split())]) except: WS.append([B\_Title[i].get\_text().strip(), "https://ridibooks.com" + B\_Link[i].attrs['href'], "https://img.ridicdn.net/cover/" + B\_Link[i].attrs['href'].split("/")[-1] + "/large#1", ' '.join(B\_Author[i].get\_text().split()), "", ' '.join(B\_Price[i].get\_text().split())]) try: B\_Next =

Soup.find("li",{"class":"btn\_next"}).find("a").attrs['href'] print(B\_Next) WP = requests.get("https://ridibooks.com" + B\_Next) Soup = BeautifulSoup(WP.content, "html.parser") except: break WB.save("BS4\_Result.xlsx")

위의 소스가 정상적으로 구동하기 위해서는

1. beautifulsoup4 와 requests 가 설치 되어 있어야 하고
2. D:\Test\Test 폴더에 "Test.xlsx" 라는 파일이 있어야 합니다.

※ 또는 파일 경로를 바꿔서 소스를 수정해 둬도 됩니다.

만들어진 소스는 다음과 같습니다. **첨부파일**

BS4.py

[파일 다운로드](http://blogattach.naver.net/a530b90a102e2f9ab35537073ddba4df762ddafd/20210510_87_blogfile/550sn_1620654993858_l9i33k_py/%2FBS4.py) **첨부파일** Result\_Format.xlsx [파일 다운로드](http://blogattach.naver.net/75e069dac195914d6183e4d6ea0b710daff402e163/20210510_29_blogfile/550sn_1620655002560_RGVVO1_xlsx/%2FResult_Format.xlsx) **첨부파일** BS4\_Result.xlsx [파일 다운로드](http://blogattach.naver.net/bf2aa3196d5a3282af4c2c1a27c4bccc64366e/20210510_134_blogfile/550sn_1620655012805_q9ia3a_xlsx/%2FBS4_Result.xlsx)

# 파이썬으로 살펴보는 고급 엑셀

## Python

2021/05/28 06:50 <http://blog.naver.com/550sn/222345719116>



파이썬으로 할 수 있는 일은 정말 무궁 무진 한 듯 합니다. 라이브러리만 추가하면 그 확장성이 끝도 없이 뻗어나가니 말입 니다.

특히나 업무 자동화 영역에서는 그 끝을 감히 짐작하기 어려울듯 합니다.

이번에는 파이썬을 통해 엑셀을 다루는 라이브러리 'openpyxl'의 고급 기능들을 조금 더 살펴 보겠습니다.

앞서 엑셀을 삼킨 파이썬에서 기본적인 기능은 많이 소개했지만, 그 외에도 내게 필요한 기능들을 구현해 보겠습니다.

음... 좀더 정확히는 Web 상에 있는 좋은 Reference 자료들을 가져와서 제게 필요한 기능들을 구현하는 방식으로 만들어 보겠습니다.

그 제작 과정은 아래 영상을 참고하시면 더 좋을듯 합니다.

이번에 구현하고자 하는 기능은 앞서 만들었던 Ridibooks 베스트 셀러 목록 엑셀 파일을 좀 더 보완하는 것입니다.

엑셀 파일에 이미 있는 정보를 활용하여

1. 각각의 책 Link 주소를 하이퍼 링크로 연결하고 (B열에 있는 정보)
2. 이미지 파일을 다운로드 받아와서 엑셀에 삽입하는 과정을 가져보도록 하겠습니다. (C열에 있는 정보)

엑셀 파일은 아래와 같습니다. **첨부파일**

B\_Result.xlsx [파일 다운로드](http://blogattach.naver.net/b227ae1960573f8fa24121172ac9b1c16a3c4b/20210510_263_blogfile/550sn_1620655932973_9htLZ9_xlsx/%2FB_Result.xlsx)

이상의 모든 작업은 Googling을 통해 참고할 소스를 가져와 만드는 방식으로 진행하겠습니다.

1. 시작하기

처음 시작하는 소스는 다음과 같습니다.

import openpyxl WB = openpyxl.load\_workbook("B\_Result.xlsx") WS = WB.active for i in range(2, WS.max\_row +1): WB.save("MOD\_Result.xlsx")

WB라는 이름으로 엑셀 파일을 불러오고,

WS라는 이름으로 작업을 실행할 WorkSheet를 연결해 둡니다. 데이터가 있는 모든줄에 대해서 반복할 반복구문 For 문을 만들어두고 작업이 마쳐지면 저장을 하게 만들어 둡니다.

이제 필요한 내용을 조금씩 채워 나가겠습니다.

1. 하이퍼링크 걸기

하이퍼 링크를 걸기 위해 openpyxl을 이용한 하이퍼 링크를 거는 방법에 대한 내용을 구글링으로 검색해 봅니다. 저는 검색어로 "python openpyxl hyperlink" 로 찾고, 다음의 링크를 찾았습니다.



### [adding hyperlinks in some cells openpyxl](https://stackoverflow.com/questions/39077661/adding-hyperlinks-in-some-cells-openpyxl)

[I have to generate an excel with summary results. The results are included in a list. Some of the elements are values and some links. I managed to generate the excel with the right format but not](https://stackoverflow.com/questions/39077661/adding-hyperlinks-in-some-cells-openpyxl)

[stackoverflow.com](https://stackoverflow.com/questions/39077661/adding-hyperlinks-in-some-cells-openpyxl)

그리고, 그안에서 참고할만한 소스로 다음과 같은 예제 구문을 찾았습니다. wbook.active['A8'].hyperlink = ["http://www.espn.com"](http://www.espn.com/) wbook.active['A8'].value = 'ESPN' wbook.active['A8'].style = "Hyperlink"

해당 구문을 가져와서 제가 만들고자 하는 소스에 붙여 넣고 필요한 부분을 바꿨습니다.

import openpyxl WB = openpyxl.load\_workbook("B\_Result.xlsx") WS = WB.active for i in range(2, WS.max\_row +1): WS["B" + str(i)].hyperlink = WS["B" + str(i)].value WS["B" + str(i)].style = "Hyperlink" WB.save("MOD\_Result.xlsx")

1. 이미지 다운로드 받기

엑셀에 이미지를 넣기 위해서는 먼저 웹상에서 이미지를 다운로드 받아야 합니다.

이번에도 구글링으로 "python image download by url"라는 키워드로 검색, 아래의 링크를 찾았습니다.



### [Downloading a picture via urllib and python](https://stackoverflow.com/questions/3042757/downloading-a-picture-via-urllib-and-python)

[So I'm trying to make a Python script that downloads webcomics and puts them in a folder on my desktop. I've found a few similar programs on here that do something similar, but nothing quite like ... stackoverflow.com](https://stackoverflow.com/questions/3042757/downloading-a-picture-via-urllib-and-python)

마찬가지로 좋은 소스들이 많이 제공되는데, 그 중 아래의 내용을 가지는 소스를 선택하였습니다.

import requests f = open('00000001.jpg','wb') [f.write(requests.get('http://www.gunnerkrigg.com//comics/](http://www.gunnerkrigg.com//comics/)

00000001.jpg').content) f.close()

같은 요령으로 제게 필요한 부분을 추가하여 정리하였습니다.

import openpyxl import requests WB = openpyxl.load\_workbook("B\_Result.xlsx") WS = WB.active for i in range(2, WS.max\_row +1): WS["B" + str(i)].hyperlink = WS["B" + str(i)].value WS["B" + str(i)].style = "Hyperlink" F\_Name = "IMG/" + WS["B" + str(i)].value.split("/")[-1] + ".png" f = open(F\_Name, 'wb') f.write(requests.get(WS["C" +str(i)].value).content) f.close() WB.save("MOD\_Result.xlsx")

1. 엑셀에 이미지 삽입하기

이제 다운로드 받은 이미지를 엑셀에 삽입하는 구문을 만들어 보겠습니다. 이번에도 구글링을 합니다. 키워드는 "python openpyxl add\_image" 입니다.



### [Insert image in openpyxl](https://stackoverflow.com/questions/10888969/insert-image-in-openpyxl)

[Is it possible to insert an image (jpeg, png, etc) using openpyxl? Basically I want to place a generated image with a chart below it. I don't see anything in the documentation, which seems to be a](https://stackoverflow.com/questions/10888969/insert-image-in-openpyxl)

[stackoverflow.com](https://stackoverflow.com/questions/10888969/insert-image-in-openpyxl)

참고할 소스는 다음과 같습니다.

import openpyxl wb = openpyxl.Workbook() ws = wb.worksheets[0] img = openpyxl.drawing.image.Image('test.jpg') img.anchor = 'A1' ws.add\_image(img) wb.save('out.xlsx') 역시 제가 만들고 있는 소스를 보강해 주겠습니다.

import openpyxl import requests WB = openpyxl.load\_workbook("B\_Result.xlsx") WS = WB.active for i in range(2, WS.max\_row +1): WS["B" + str(i)].hyperlink = WS["B" + str(i)].value WS["B" + str(i)].style = "Hyperlink" F\_Name = "IMG/" + WS["B" + str(i)].value.split("/")[-1] + ".png" f = open(F\_Name, 'wb') f.write(requests.get(WS["C" +str(i)].value).content) f.close() img = openpyxl.drawing.image.Image(F\_Name) img.anchor = 'C' + str(i) WS.add\_image(img) WB.save("MOD\_Result.xlsx")

그림은 잘 붙는데, 이미지가 붙는 칸의 크기가 잘 맞지 않습니다. 이에 대한 부분만 width & height 관련 영역을 조정하여 원하는 사이즈로 맞춰 줍니다.

최종적으로 완성된 소스와 결과 파일은 다음과 같습니다.

import openpyxl import requests WB = openpyxl.load\_workbook("B\_Result.xlsx") WS = WB.active for i in range(2, WS.max\_row +1): WS["B" + str(i)].hyperlink = WS["B" + str(i)].value WS["B" + str(i)].style = "Hyperlink" F\_Name = "IMG/" + WS["B" + str(i)].value.split("/")[-1] + ".png" f = open(F\_Name, 'wb') f.write(requests.get(WS["C" +str(i)].value).content) f.close() img = openpyxl.drawing.image.Image(F\_Name)

I\_Rate = img.width / img.height img.width = 146 img.height = 146 / I\_Rate img.anchor = 'C' + str(i) WS.row\_dimensions[i].height = img.height \*150 /200 WS.add\_image(img) WB.save("MOD\_Result.xlsx") **첨부파일**

Excel\_HL\_IMG.py [파일 다운로드](http://blogattach.naver.net/48dd54e4f0a9ac705cbed9ebd7364c3092c93adcea/20210510_181_blogfile/550sn_1620656415903_eN39Gx_py/%2FExcel_HL_IMG.py) **첨부파일** MOD\_Result.xlsx [파일 다운로드](http://blogattach.naver.net/6aff76c6d58f8e527e9cfbc9f5146e12b0eb18ff02/20210511_153_blogfile/550sn_1620664813614_h08w9V_xlsx/%2FMOD_Result.xlsx) 4.정리하며

파이썬을 마음먹고 공부한지는 얼마되지 않았지만, 대단하다는 생각이 듭니다.

1. 라이브러리가 정말 다양하고 강력합니다.
2. 참고할 Reference가 너무나 많고 좋습니다.
3. 기본적인 문법 자체가 간결하고 익히기 좋습니다.

구박사에게 "프로그램명 라이브러리 원하는기능" 순서로 물어보면 왠만한 내가 하고자 하는 기능은 다 찾을 수 있습니다. 쉽게 찾을 수 있을 뿐만 아니라, 예제 파일이 함께 있기 때문에 금방 금방 사용이 가능합니다.

물론 한단계 더 깊게 들어가면 순간 순간 좌절을 느끼고 벽을 만날 수도 있겠지만, 괜히 겁먹지 말고 한번쯤 도전해볼만한 과제라는 생각이 듭니다.

아! 그리고 엑셀을 띄우지 않고도 그 내부에 있는 Data를 자유자재로 다루는것도 생각보다 강력한것 같습니다.

속도가 훨씬 빠릅니다. 엑셀을 띄워놓고 실행하면, 중간중간에 사람에게 보이는 화면 정보를 바꿔준다고 느려지는데 그부 분이 없다 보니 컴퓨터가 할일 데이터 수정만 몰아치다 보니 이와 같은 속도가 나오는 듯 합니다.

저도 겁먹지 말고 좀 더 진작에 파이썬을 시작해 볼 껄 그랬나 봅니다...

# 파이썬으로 만든 나만의 RPA (Tricks\_Office)

## Python

2021/06/01 03:44 <http://blog.naver.com/550sn/222374974703>



지난 2021년 5/15일, Python을 본격적으로 다루기 시작하면서, Python의 매력에 완전히 빠져 버렸습니다.

이왕 시작한 것, 장기 프로젝트화 하여 제대로 무언가를 만들어 보고자 합니다. 프로젝트 명 "Tricks\_Office".

네, 맞습니다. 업무의 잔머리(잔기술)를 제 나름의 방식으로 영문으로 표현한 이름입니다.

우선은 돌아가는 실행파일부터 만들고, 필요한 기능은 하나 하나 추가해 나가되, 그 과정은 점진적으로 영상 및 블로그의 글로 남기는 형태로 진행하려고 합니다.

현재 구현된 기능들은 다음과 같습니다.

1. **여러 PPT 파일의 여러 단어(Dictionary 파일 기준)를 원하는 단어로 치환하기**
2. **양식이 맞춰진 엑셀 파일을 하나의 파일로 취합하기**
3. **하나의 양식은 아니지만, 유사한 양식의 엑셀 파일 데이터 하나의 파일로 취합하기**
4. **여러 PPT 파일의 Font 하나로 통일하기**
5. **한글 ↔ 영문 자동 번역하기**

압축을 풀고 실행파일을 실행하면 사용이 가능하게 만들었는데, 압축 파일의 용량이 약 40MB쯤 되다 보니 네이버 블로그 에는 용량제한으로 올리지 못하고, 따로 만든 블로그에 올렸습니다.

**아래 링크에서 파일을 다운로드 받으실 수 있습니다.**

### [파이썬 나만의 RPA 툴 제작 프로젝트 (Tricks\_Office)](https://tricks-office.com/python-tricks_office/)

[파이썬으로 나만의 RPA 툴을 직접 만드는 장기 프로젝트, 프로젝트명 “Tricks\_Office” 를 진행하고자 합니다. 구현을 계 획하고 있는 기능은 다음과 같습니다. 여러 PPT 파일의 여러 단어(Excel 형태로 보관된 Dictionary 파일 기준)를 원하는 단어로 치환하기 (구현)양식이 맞춰진 엑셀 파일을 하나의 파일로 취…](https://tricks-office.com/python-tricks_office/)

[tricks-office.com](https://tricks-office.com/python-tricks_office/)

(파이썬이 이것저것 라이브러리를 끌어와서 쉽게 원하는 기능을 만들수 있는 장점이 있는 반면, 메모리, 용량등의 PC 리소스 활용 면에서는 비효율적인 면이 있는 느낌이 있습니다.)

앞으로도 사무직 인원들이 자주 접하게 되는 일반적인 문서 반복 노가다 상황에 범용적으로 활용이 가능한 기능 위주로 기 능 추가해 갈 계획입니다.

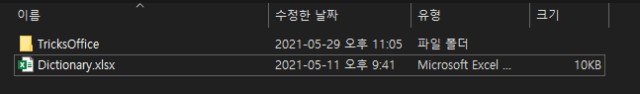
필요한 기능이 있으시면 댓글 달아주시면, 저와 방향성이 맞는 필요 기능을 선별하여 추가해 나가 보겠습니다.(제 실력이 허락하는 범위 내에서요.)

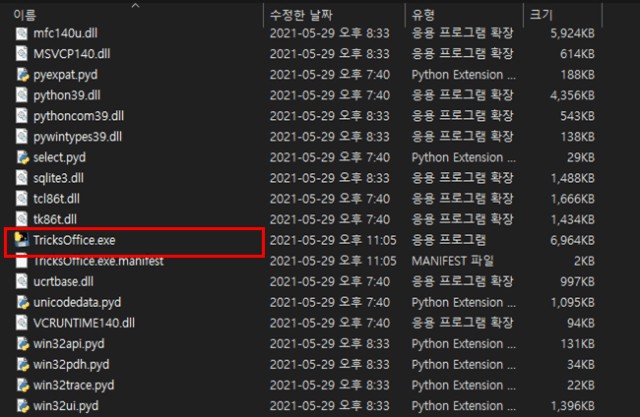
Tricks\_office 사용 방법

**영상을 참고하시면 사용법을 좀 더 쉽게 이해하실 수 있습니다.**

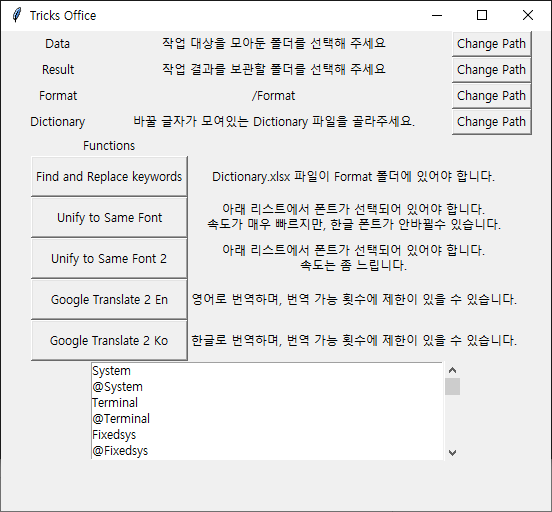
**1) 파일 실행**

위의 링크에서 압축 파일을 다운로드 받아서 압축 해제 하면 다음과 같이 1개의 폴더와 1개의 파일이 생성됩니다. "Dictionary.xlsx" 파일은 여러 파일에서 특정 단어를 일괄 변경할 때 사용하는 바꿀 용어 ↔ 바뀔 용어를 정리한 파일입니 다. 실제 사용시에는 이 파일을 정리한 다음,



이 폴더 안에 들어가면 꽤 많은 파일이 보이는데, 그중 “TricksOffice.exe” 파일을 선택하여 실행해 주시면 됩니다. (한번 사용해 보시고 자주 사용할것 같으면, 해당 실행 파일만 바로가기 버튼을 만들어서 추가해서 사용하셔도 좋을것 같습니 다.)

실행하면 아래와 같은 창이 뜹니다.



제가 봐도 창이 아름답지는 않네요... 아 나의 저주 받은 미적감각... ㅠ.ㅠ

공부하면서 생각한대로 바로바로 만들어지는게 너무 신기한 나머지, 기능구현에만 너무 중점을 줬습니다. 아직 파이썬 GUI 툴의 사용이 익숙하지 않고, 제 미적 감각이 좋지 않은 탓도 있고요.

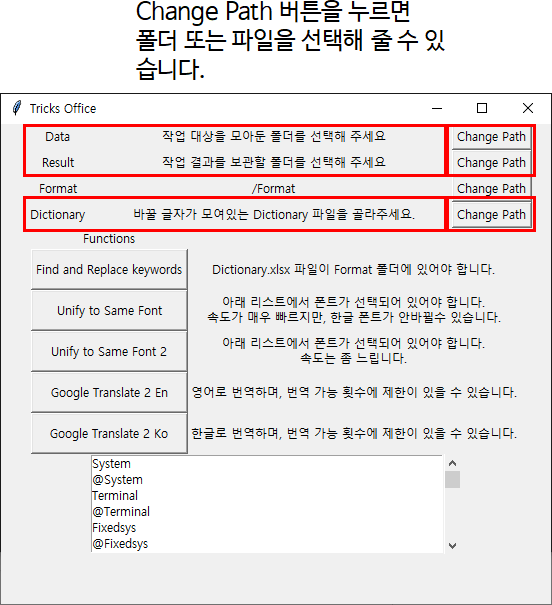
공돌이 감성의 한계가 있는지라 그리 이쁘게 꾸지지는 않을 것 같지만, 시간 나는 대로 조금씩 다듬을 생각이어서, 아마도 조금씩 나아지리라 생각됩니다. 그렇겠죠?

기본적으로 작업하는 방식은 아래와 같습니다.

1. 하나의 폴더에 작업할 대상 파일을 모아 넣는다.
2. 작업하고자 하는 작업을 완료하면 완료 폴더에 작업 완료한 파일을 보관한다.

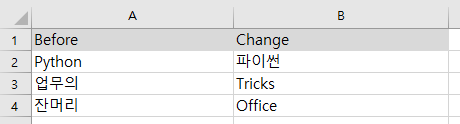
이와 같은 방식으로 작업하다 보니 2가지 정보는 꼭 필요합니다. 작업할 파일이 모여있는 폴더와 작업이 완료된 파일을 보관할 폴더.

아래 이미지에 있는 Data, Result라고 명명되어 있는 곳에 폴더 경로는 필수로 저장되어야 합니다. 경로를 저장하는 방법은 Change Path를 클릭하면 폴더 또는 파일을 선택할 수 있게 되어 있습니다.



1. **여러 파일에서 여러 단어 찾아서 한번에 바꾸기 (PPT전용)**

Data로 설정된 폴더의 파일을 하나하나 읽어와서, Dictionary에 설정된 엑셀 파일 정보를 기반으로하여,



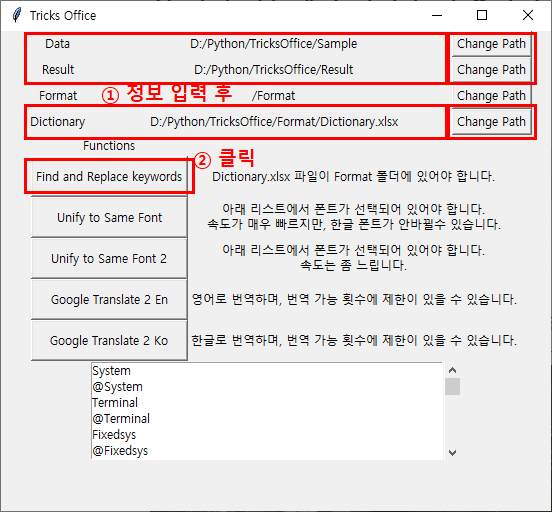
Dictionary 파일

Before에 해당하는 단어를 Change에 해당하는 단어로 일괄 처리하는 기능입니다.

사용방법은, Tricks\_Office 프로그램 화면에서 꼭 필요한 정보인

1. 작업할 파일이 모인 폴더
2. 완료한 파일을 모을 폴더
3. Dictonary 파일 경로

를 선택한 다음 "Find and Replace keywords" 버튼을 클릭해 주면 됩니다.



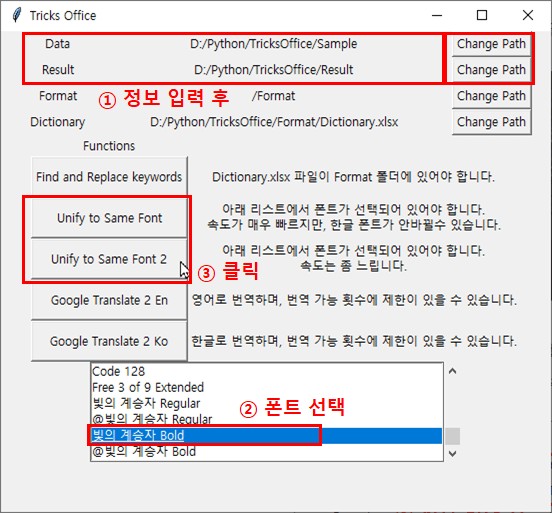
1. **폰트 통일하기 (PPT 전용)**

Data로 설정된 폴더 안의 파일을 하나하나 읽어와서, 맨 아래 칸의 List에서 폰트를 선택하면 선택한 폰트로 모든 파일의 폰트를 바꿔주는 기능입니다.

사용방법은, Tricks\_Office 프로그램 화면에서 꼭 필요한 정보인

1. 작업할 파일이 모인 폴더
2. 완료한 파일을 모을 폴더
3. 바꿀 폰트

를 선택한 다음 "Unify to Same Font" 버튼을 클릭해 주면 됩니다.



현재 Font 변경 기능은 2종류로 배포 되어 있는데요.

1번 : 굉장히 빠른 속도를 자랑하지만, 한글 폰트가 제대로 안바뀌는 경우가 자주 있습니다. python-pptx 라이브러리를 이용해 PPT 파일을 직접 열지 않고 변경하기 때문에 매우 빠릅니다. 다만, 해외 사용자들의 경우 영문폰트 / 한글폰트를 따로 설정하는 경우가 많지 않아서 라이브러리 자체에서 한글폰트를 다루는 것이 매끄럽게 이루어지지 않습니다.

2번 : 1번에 비해서는 굉장히 느리지만, 사람이 일일히 작업하는것 보다는 아주 빠릅니다. Win32 라이브러리를 통해 PPT 파일을 하나하나 열면서 변경하기 때문에 상대적으로 느립니다. 한글 폰트도 문제없이 잘 변경합니다.

1,2번 모두 표 내부, 그룹이 되어 있는 문자까지 바꿀수 있게 구현하였습니다.



1. **번역하기 (PPT전용)**

Data로 설정된 폴더의 파일을 하나 하나 읽어와서, 한글 또는 영어로 번역해 주는 기능입니다.

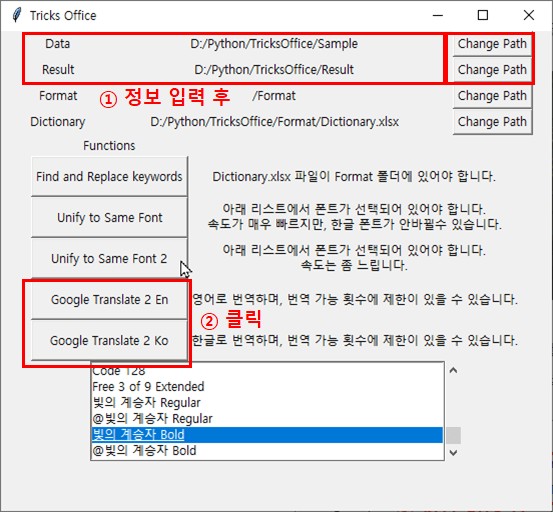
기본적으로 Google Translate를 활용하는 라이브러리를 사용했지만, 정식 구글 API 라이브러리는 아닌 관계로 실질적인 사용 횟수에 제한이 있는 것 같습니다. (관련 내용은 별도 영상으로 다룰 예정입니다.)

사용횟수 제한을 넘겼을 경우에는 번역이 되지 않고 원본 파일 그대로 다시 저장이 되는데, 약 1시간 정도(정확하지는 않 습니다. 1시간 보다는 짧은 느낌이었습니다.) 기다렸다가 다시 실행하면 또 실행이 됩니다.

사용방법은, Tricks\_Office 프로그램 화면에서 꼭 필요한 정보인

1. 작업할 파일이 모인 폴더
2. 완료한 파일을 모을 폴더

를 선택한 다음, "Google Translate 2 En / Ko" 버튼을 클릭해 주면 됩니다.



Google에서 공식적으로 제공하는 Translate API를 활용하고 싶었는데, 파이썬에서 돌아가게 구현은 되는데, exe 파일로 변환만 하면 에러가 나면서 실행이 안되네요.

현재 Google Cloud 라이브러리를 exe파일로 만드는 과정에서 어딘가 문제가 있는것 같은데, 이 부분은 해결되는 대로 추가해서 올리겠습니다.

지금 업데이트 되어 있는 파일은 기능 구현에만 집중되어 있어서 에러 메시지 처리, 예외 케이스에 대해서는 충분히 정리 되고 검증되지 않았습니다. 조금씩 다듬어갈 예정이오니, 불편한 부분은 댓글로 달아주시면 감사하겠습니다.

지난 주말은 이 내용을 구현해 본다고 하얗게 불태워 버렸네요...

사실 업무 자동화라는 큰 방향성을 가지고 채널과 블로그를 운영하면서 궁극적으로는 파이썬으로 가지 않을까 라는 생각 을 막연히 하고는 있었는데, 제게는 왠지 파이썬이 "끝판왕"이라는 이미지가 있어서 늦게 시작한 감도 없지않아 있습니다.

막상 시작하고 보니 파이썬을 시작한 초창기여서 그런지, 이것 저것 해보고 싶은게 너무 많아서 영상을 만드는 속도가 기 능을 구현하는 속도를 못따라오고 있는 현상이 나오고 있습니다. 조바심 내지 말고 하나 하나 풀어나갈까 합니다.

먼저 기능을 사용해 보시고, 그 제작 과정 및 소스가 궁금하신 분들은 앞으로 추가되는 글과 영상들을 참고해 주시면 좋겠 습니다.

# PPT를 삼킨 파이썬(python-pptx)

## Python

2021/06/04 06:50 <http://blog.naver.com/550sn/222347440606>



이번에는 Python에서 PPT를 직접 다룰 수 있는 라이브러리를 다뤄 보겠습니다. 이번에 다루고자 하는 라이브러리는 "python-pptx"입니다.

### [python-pptx — python-pptx 0.6.18 documentation](https://python-pptx.readthedocs.io/en/latest/)

[python-pptx is a Python library for creating and updating PowerPoint (.pptx) files. A typical use would be generating a customized PowerPoint presentation from database content, downloadable by clicking a link in a web application. Several developers have used it to automate production of presentati...](https://python-pptx.readthedocs.io/en/latest/)

[python-pptx.readthedocs.io](https://python-pptx.readthedocs.io/en/latest/)

1. 설치 및 시작하기

먼저 "python-pptx"를 사용하기 위해서는 라이브러리 설치부터 해야 합니다. python 터미널에서

pip install python-pptx

을 입력하면 python-pptx 라이브러리 설치가 가능합니다.

설치되고 난 다음에는 Python 코드를 구성할때 해당 라이브러리를 불러오는 구문만 추가하면 필요한 기능을 불러와서 활 용이 가능합니다.

import pptx

PPT 파일의 경우 사실 어떤 일괄 작업을 할 때를 제외하고는 엑셀에서처럼 데이터 위치를 정확하게 확인하거나 하는 등의 작업이 상대적으로 어렵기 때문에 사실 다루는 것은 PPT 슬라이드, 그리고 Shape 정도가 될것 같습니다.

pptx 라이브러리 전체 보다는 pptx 라이브러리 내의 "Presentation"구문을 주로 활용하는 경우가 많으니, 다음과 같이 호출하는 것이 더 자주 사용될 것도 같습니다.

from pptx import Presentation

1. 파일 생성하기, 불러오기 그리고 저장하기

파이썬에서 PPT 파일을 새롭게 만드는 경우에는 Presentation 개체만 하나 새로 추가해 주면 됩니다. prs = Presentation()

기존에 있던 파일을 불러올 때는 () 안에 파일 이름을 입력해 주면 됩니다. prs = Presentation("불러올 파일 경로")

저장 할때는 save 명령어를 사용하면 됩니다. 단 이때 Workbook으로 선언된 개체에 save 명령어를 추가해야 합니다. prs.save("저장할 파일 경로")

이와같이 Presentation 개체를 하나 선언하고 지정해 줌으로써 Python에서는 해당 파일을 다루게 됩니다.

1. PPT에 글 상자 다루기

파워포인트에서 어떤 작업을 할 때 어떤 동작을 하기 위해서는 정확한 개체를 선택해 줘야 하는데, 그 개체를 선택하기 위 해서는 Presentation → 슬라이드 → 개체 순서로 선택을 해야 합니다.

다른 사람들은 어떨지 모르겠지만, 제가 Python을 활용하여 PPT를 다룰 때 하는 동작은 무언가 단순 반복 작업을 할 때 사용할 것입니다.

즉, PPT 파일 내부에 있는 전체의 개체를 하나하나 찾으면서 어떤 동작을 반복 하는 형식으로 진행할 듯 합니다. 따라서, 어떤 동작을 하더라도 아래의 구문과 같은 방식으로

1. 매 슬라이드를 넘기면서
2. 개체를 하나하나 찾으면서

어떤 필요한 동작을 하는 방식으로 코드를 만들것 같습니다.

prs=Presentation('불러올 파일 경로') slides = prs.slides for slide in slides: shapes = slide.shapes for shape in shapes: 각 shape에서 하고자 하는 동작 prs.save('저장할 파일 경로')

일반적으로 여기서 shape를 다루면 내가 원하는 것을 할 수 있습니다.

그런데, 이 Shape로 올수 있는 개체가 워낙 여러 종류의 개체가 올 수 있는데다가, 각 개체가 필수값으로 가져야 하는 값 이 워낙 다양한지라 다루기가 쉽지만은 않습니다.

제가 생각할때 가장 많이 사용되는 Shape의 형태는 Text\_Frame의 형태일것으로 생각되는데, 이 Text\_Frame을 다루는 게 사실 쉽지만은 않습니다.

따져 봐야 하는것이 많은데요.

1. Text\_Frame이 있을 경우에만 내가 원하는 동작을 하게 해야 합니다.
2. 같은 Text\_Frame 안에 있더라도 문단이 나뉘면 또 따로 살펴 봐야 합니다.
3. 같은 문단 안에 있더라도 폰트가 다르게 적용되거나, 띄어쓰기가 포함 되었을 경우 여기까지 분리해서 봐야 합니다.

결국 여기까지 따지게 되면 위의 구문은 훨씬 복잡해 집니다.

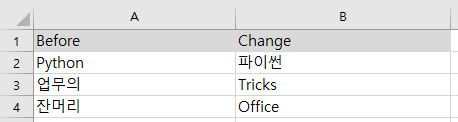
prs=Presentation("Sample.pptx") Slides = prs.slides for slide in Slides: Shapes = slide.shapes for shape in Shapes: if shape.has\_text\_frame: text\_frame = shape.text\_frame for paragraph in text\_frame.paragraphs: for run in paragraph.runs: run 단위로 할 일 prs.save("저장할 파일 경로")

Text를 바꾸는 작업을 할 때 paragraph 또는 run 단위로 내려와서 조정을 하지 않고, text\_frame 단위에서 단순히 Text 만 바꾸면 글자는 원하는 대로 바뀌지만, Font / 글자 크기 / 글자 색깔 등의 모든 설정이 초기값으로 되돌아가게 되기 때 문에 주의해야 합니다.

1. 적용 예시

아래 파일과 같이 여러장의 슬라이드로 이루어진 PPT 파일이 있고, 해당 파일의 몇가지 단어를 일괄 변경하고자 한다고 하겠습니다.

**첨부파일** Sample.pptx [파일 다운로드](http://blogattach.naver.net/b82da41700333287ae482a1a20c6b9c36231cc87/20210512_193_blogfile/550sn_1620760179802_nQk8x0_pptx/%2FSample.pptx)

다음과 같이 엑셀 파일에서 Before에 해당하는 단어를 찾아서 Change에 해당하는 단어로 바꾸고자 한다고 하겠습니다.

**첨부파일** Dictionary.xlsx [파일 다운로드](http://blogattach.naver.net/4edb52e2f7a3aa765ab8dfedd1304a3695c639dee8/20210512_278_blogfile/550sn_1620760170926_2una9D_xlsx/%2FDictionary.xlsx)

이때 이 Dictionary 파일에서 정보를 가져와서 PPT 파일의 단어에서 필요한 단어를 찾아서 바꾸는 작업을 일괄 적용하는 소스를 만들어 보면 아래와 같이 구성할 수 있습니다.

from pptx import Presentation import openpyxl wb = openpyxl.load\_workbook('Dictionary.xlsx') ws = wb.active rows = ws.rows BW = [] AW = [] for cell in rows: BW.append(cell[0].value) AW.append(cell[1].value) prs=Presentation("Sample.pptx") Slides = prs.slides for slide in Slides: Shapes = slide.shapes for shape in Shapes: for i in range(1,len(AW)): if shape.has\_text\_frame: text\_frame = shape.text\_frame for paragraph in text\_frame.paragraphs: for run in paragraph.runs: cur\_text = run.text new\_text = cur\_text.replace(str(BW[i]), str(AW[i])) run.text = new\_text prs.save('Result.pptx')

만들어진 소스와 결과 파일은 다음과 같습니다. **첨부파일**

PPT\_Naver.py [파일 다운로드](http://blogattach.naver.net/b623aa1964533b8ba64525132ecdb4cc6d385c/20210512_284_blogfile/550sn_1620760415618_4j1hs7_py/%2FPPT_Naver.py) **첨부파일** Result.pptx [파일 다운로드](http://blogattach.naver.net/049118abbee9e03c10f295a79b7a007cdf8c73975f/20210512_94_blogfile/550sn_1620760483451_D9qJ3E_pptx/%2FResult.pptx)

1. 그 외

이번에 작성된 소스가 결과적으로 보면 매우 짧게 보이지만, 사실 여기까지 공부하는데 꽤 어려움이 많았습니다.

이전에 VBA 매크로를 공부할 때도 느꼈는데, 파워포인트가 엑셀에 비해서 프로그래밍 관점에서는 훨씬 다루기 어려운것 같습니다.

만드는 과정은 정말 어려웠는데, 한번 만들고 나서 보면 실행되는 속도가 정말 비교가 안되게 빠릅니다.

파일 크기가 얼마나 크든 상관없이 몇초 내에 완료되는것 같습니다. 이와 같은 실행 속도 면을 본다면 이 많은 어려움을 감 내하고도 python은 익힐 가치가 있다는 생각도 듭니다.

이번 소스를 만들어 보면서 참고한 소스의 링크는 다음과 같습니다.

1. PPT text변경하기 참고

검색어 : "python-pptx replace text keep formatting"



### [Python pptx (Power Point) Find and replace text (ctrl + H)](https://stackoverflow.com/questions/37924808/python-pptx-power-point-find-and-replace-text-ctrl-h)

[Question in Short: How can I use the find and replace option (Ctrl+H) using the Python-pptx module? Example Code: from pptx import Presentation nameOfFile = "NewPowerPoint.pptx" #Replace this wi... stackoverflow.com](https://stackoverflow.com/questions/37924808/python-pptx-power-point-find-and-replace-text-ctrl-h)

1. 엑셀 데이터 가져오기 참고

검색어 : "python openpyxl read data as dictionary"



### [Pythonic way to create a dict of Excel column values](https://stackoverflow.com/questions/46717660/pythonic-way-to-create-a-dict-of-excel-column-values)

[I have an Excel worksheet and I want to create a dict which has the cell values as a list with cell column being the key. Let say that spreadsheet has data that looks like, A B C (columns) 1 2 3 ...](https://stackoverflow.com/questions/46717660/pythonic-way-to-create-a-dict-of-excel-column-values)

[stackoverflow.com](https://stackoverflow.com/questions/46717660/pythonic-way-to-create-a-dict-of-excel-column-values)

※ 엑셀 데이터를 가져오는 구문은 데이터 량이 많을 경우 제가 만든 구문 보다는 위의 참고 구문이 좀 더 유용합니다. Dictionary 형태의 변수로 받아 저장하는 방법인데, 활용 가능한 방법이 훨씬 구조적으로 좋습니다.

※ 예제에서 만들어진 프로그램에서는 PPT 안에 Table이 있을 경우에 대해서는 반영되지 않았습니다. 표를 포함하는 구문의 추가 강화가 필요합니다.

1. 최종본 파일

유튜브 영상을 찍으면서 소스를 좀 더 수정하였습니다.

1. Table이 있을 경우 Table 안의 글자를 바꿀수 있게 추가 보완했습니다.
2. Group으로 묶인 개체가 있을 경우 Group 내부에 있는 글들도 바꿀 수 있게 보완했습니다.
3. 반복되는 구문은 별도 Function으로 분리해서 코드를 좀더 보기 좋게 만들었습니다.

만들어진 소스와 파일은 다음과 같습니다.

from pptx import Presentation from pptx.enum.shapes import MSO\_SHAPE\_TYPE import openpyxl def R\_TXT (shp,B\_W,A\_W): for paragraph in shp.text\_frame.paragraphs: for run in paragraph.runs: cur\_text = run.text new\_text = cur\_text.replace(str(B\_W), str(A\_W)) run.text = new\_text wb = openpyxl.load\_workbook("Dictionary.xlsx") ws = wb.active before\_W = [] after\_W = [] for cell in ws.rows: before\_W.append(cell[0].value) after\_W.append(cell[1].value) prs = Presentation("Sample.pptx") for slide in prs.slides: for shape in slide.shapes: for i in range(1, len(after\_W)): if shape.has\_text\_frame: R\_TXT(shape,before\_W[i],after\_W[i]) if shape.has\_table: for row in shape.table.rows: for cell in row.cells: R\_TXT(cell,before\_W[i],after\_W[i]) if shape.shape\_type == MSO\_SHAPE\_TYPE.GROUP: for shp in shape.shapes: if shp.has\_text\_frame: R\_TXT(shp,before\_W[i],after\_W[i]) prs.save("Result.pptx")

**첨부파일** PPT\_Study.py [파일 다운로드](http://blogattach.naver.net/29bc358691c5cd113ddfb88ab6572d51f2a15db865/20210516_243_blogfile/550sn_1621096124620_8j191H_py/%2FPPT_Study.py)

한 폴더에 모인 파일에 대해서 일괄 변경 가능하게 기능을 모아서 구현한 실행파일은 아래의 링크에서 다운로드 받으실 수 있습니다.

### [파이썬 나만의 RPA 툴 제작 프로젝트 (Tricks\_Office)](https://tricks-office.com/python-tricks_office/)

[파이썬으로 나만의 RPA 툴을 직접 만드는 장기 프로젝트, 프로젝트명 “Tricks\_Office” 를 진행하고자 합니다. 구현을 계 획하고 있는 기능은 다음과 같습니다. 여러 PPT 파일의 여러 단어(Excel 형태로 보관된 Dictionary 파일 기준)를 원하는 단어로 치환하기 (구현)양식이 맞춰진 엑셀 파일을 하나의 파일로 취…](https://tricks-office.com/python-tricks_office/)

[tricks-office.com](https://tricks-office.com/python-tricks_office/)

# 번역기를 삼킨 파이썬 (PPT 문서 자동 번역)

## Python

2021/06/08 06:50 <http://blog.naver.com/550sn/222348852644>



해외를 무대로 일을 하다 보면 문서를 번역해야 하는 일이 자주 있습니다.

요즘이야 Google Translate나 Naver 파파고 등과 같이 뛰어난 번역 보조 툴이 있어서 과거에 비해서 훨씬 편하게 번역을 할 수 있기는 하지만, 이 또한 많은 분량의 문서가 있을 경우에는 만만치 않은 작업입니다.

Python의 라이브러리중에 google 번역기를 사용할 수 있게 도와주는 라이브러리가 있어서 이를 활용해 PPT 파일의 문서 를 번역하는 작업을 해 보도록 하겠습니다.

이번에 사용할 라이브러리는 "googletrans" 입니다.

### [googletrans](https://pypi.org/project/googletrans/)

[Free Google Translate API for Python. Translates totally free of charge. pypi.org](https://pypi.org/project/googletrans/)

1. 설치 및 시작하기

"googletrans"를 사용하기 위해서는 라이브러리 설치부터 해야 합니다.

python 터미널에서 아래와 같이 입력하면 설치는 되는데 정상 동작되지는 않습니다. pip install googletrans

실제 예제 구문을 만들어서 실행해 보면 정상 동작하지 않고, 다음의 문구가 포함된 에러 메시지가 뜹니다. "attribute error 'nonetype' object has no attribute 'group'" 어쩌구 저쩌구...

이에 대한 해결 또한 Google에서 에러 구문을 포함하여 검색 하면 답을 찾을 수 있습니다. 검색어 : "python googletrans attribute error 'nonetype' object has no attribute 'group'"



### [googletrans stopped working with error 'NoneType' object has no attribute 'group'](https://stackoverflow.com/questions/52455774/googletrans-stopped-working-with-error-nonetype-object-has-no-attribute-group)

[I was trying googletrans and it was working quite well. Since this morning I started getting below error. I went through multiple posts from stackoverflow and other sites and found probably my ip is stackoverflow.com](https://stackoverflow.com/questions/52455774/googletrans-stopped-working-with-error-nonetype-object-has-no-attribute-group)

여러가지 해결책이 나오는데, 저는 이전 버전을 삭제하고, 그 다음 버전의 알파 버전을 활용하는 방법으로 문제를 해결해 보겠습니다.

pip uninstall googletrans pip install googletrans==3.1.0a0

※ 이글을 읽으실 쯤에는 해당 버전 문제가 해결되어 위의 해결책이 유효하지 않을 수도 있고, 다른 해결 책이 있을 수도 있습니다.

비슷한 난관에 부딪히시더라도, Google에서 영어로 검색하는 방법으로 해결할 수 있으리라 생각됩니다.

소스에는 다음과 같이 선언을 해 주면 활용이 가능합니다. import googletrans

어짜피 저희가 쓸 기능은 Translator 기능이기에, 다음과 같이 선언하여 Translator 기능만 바로 불러서 사용하는 것이 보 다 편리할 수 있습니다.

from googletrans import Translator

1. Translator 불러오기, 번역하기

Translator에서 번역을 지시하는 방법은 다음과 같습니다.

trans = Translator() Trans = trans.translate('안녕하세요.', src='ko', dest='en')

여기서 이미 짐작 하셨겠지만, src는 원본 글의 언어를 뜻하고, dest는 번역한 결과물의 언어를 뜻합니다.

번역한 원문과 결과문을 보기 위해서는 각각 뒤에 ".origin" / ".text"을 입력하면 됩니다. 즉, 아래와 같이 내용을 입력 하면,

from googletrans import Translator trans = Translator() Trans = trans.translate('안녕하세요', src='ko', dest

='en') print(Trans.origin) print(Trans.text) 다음과 같은 결과물을 얻을 수 있습니다. 안녕하세요 Good morning

1. 실제 예제에 응용해 보기

이번에 만들어 볼 내용은 아래 유첨 파일과 같이 한글로 된 파워포인트 자료가 있을때 이를 영문으로 바꿔주는 작업을 해 보도록 하겠습니다.

**첨부파일** Trans\_Sample.pptx [파일 다운로드](http://blogattach.naver.net/5cc940f0e2bdb86448aacdffc322582487d42fcc85/20210512_32_blogfile/550sn_1620791664749_8ZGAjU_pptx/%2FTrans_Sample.pptx)

이를 위해 이전에 다루었던 'python-pptx' 소스도 응용하여 만들어 보도록 하겠습니다. 만들어진 소스 파일은 다음과 같습니다.

from googletrans import Translator from pptx import Presentation trans=Translator() prs=Presentation("Trans\_Sample.pptx") Slides = prs.slides for slide in Slides: Shapes = slide.shapes for shape in Shapes: if shape.has\_text\_frame: text\_frame = shape.text\_frame for paragraph in text\_frame.paragraphs: cur\_text = paragraph.text new\_text = trans.translate(cur\_text, src='ko', dest='en').text paragraph.text = new\_text prs.save('T\_Result.pptx')

실행한 결과로 만들어지는 파일은 다음과 같습니다. **첨부파일**

T\_Result.pptx [파일 다운로드](http://blogattach.naver.net/26b33a899fc7c21e32d0b785b958225efdae55b465/20210513_254_blogfile/550sn_1620842536449_UCS345_pptx/%2FT_Result.pptx)

만들어진 결과를 보면, 번역은 제대로 된것 같은데, 폰트, 글자 크기 등 등이 모두 틀어진 것을 알 수 있습니다.

이는 PowerPoint에서 폰트나 글자 크기 등의 정보를 문단 단위인 paragraph 보다 하부 단위인 "run"단위로 만들어 두는 데, 이 run의 구성이 깨지면서 기본적인글상자가 가지고 있던 설정값으로 되돌아옴으로써 발생하는 현상입니다. 그렇다고 run 단위로 쪼개서 번역기를 돌리면 번역이 단위단위로 이루어지다 보니 매끈하지가 않습니다.

여기서 폰트까지 유지하는 방법을 고민해 보자면 각 paragraph의 첫번째 run의 font를 가져와서 저장해 뒀다가 text를 변경한 다음 다시 덮어쓰는 방법이 있을 수 있습니다.

이를 추가로 정리하면 다음과 같습니다.

from googletrans import Translator from pptx import Presentation trans=Translator() prs=Presentation("Trans\_Sample.pptx") Slides = prs.slides for slide in Slides: Shapes = slide.shapes for shape in Shapes: if shape.has\_text\_frame: text\_frame = shape.text\_frame for paragraph in text\_frame.paragraphs: font=paragraph.runs[0].font Fname = font.name Fsize = font.size cur\_text = paragraph.text new\_text = trans.translate(cur\_text, src='ko', dest='en').text paragraph.text = new\_text for run in paragraph.runs: font = run.font font.name = Fname font.size = Fsize prs.save('T\_Result.pptx')

그 결과는 다음과 같습니다. **첨부파일**

T\_Result2.pptx [파일 다운로드](http://blogattach.naver.net/ed78f14a506667d2fb1d7f4f7593ec9637609228/20210513_173_blogfile/550sn_1620842564517_rxtk0Z_pptx/%2FT_Result2.pptx)

만들어진 소스는 다음과 같습니다. **첨부파일**

Trans\_Study.py [파일 다운로드](http://blogattach.naver.net/7bee67d7c69b9f436f8dead8e4057f03a0f30befa0/20210513_11_blogfile/550sn_1620848255523_916bu1_py/%2FTrans_Study.py)

※ Youtube 영상을 제작하면서 만든 소스와 파일은 아래와 같습니다. 내용은 대동 소이합니다.

from googletrans import Translator from pptx import Presentation translator = Translator() prs = Presentation("Trans\_Sample.pptx") for slide in prs.slides: for shape in slide.shapes: if shape.has\_text\_frame: for paragraph in shape.text\_frame.paragraphs: font = paragraph.runs[0].font FName = font.name FSize = font.size FBold = font.bold cur\_text = paragraph.text new\_text = translator.translate(cur\_text, dest='en').text paragraph.text=new\_text for run in paragraph.runs: font = run.font font.name = FName font.size = FSize font.bold = FBold prs.save('T\_Result2.pptx')

**첨부파일** G\_Trans.py [파일 다운로드](http://blogattach.naver.net/71e46ddec59d95496587e0d2ee0f7509aaf901e784/20210516_151_blogfile/550sn_1621096344305_QG037w_py/%2FG_Trans.py)

1. 그런데...

그런데... 이번 소스는 제가 몇번 적용하고 나니 번역이 되지 않았습니다.

(1시간 정도 지난 다음에 다시 실행하니 정상적으로 동작합니다.)

좀 찾아보니, 이 라이브러리는 한번에 활용 가능한 횟수에 제한이 있는것 같습니다.

(라이브러리 소개 글에는 무료로 무제한 쓸수 있다고 되어 있지만, 결국 번역 엔진은 Google Cloud를 사용하는것 같고, Google Cloud에서 사용을 제한하는것 같습니다.)

만약 번역을 자주 써야 하는 상황이라면, 경우에 따라서는 다른 번역 라이브러리를 찾아야 할 지도 모르겠습니다. 번역하는 구문만 아래와 같은식으로 바꿔주면 될 듯 합니다.

1. google\_trans\_new

google\_trans\_new라는 라이브러리도 있는것 같습니다. 터미널에서 다음과 같이 입력하고

pip install google\_trans\_new

소스 파일에서는 다음과 같이 구성하면 사용이 가능한것 같습니다.

from google\_trans\_new import google\_translator translator = google\_translator() translate\_text = translator.translate('안녕하세요',lang\_tgt='en') print(translate\_text)

결과 값으로 "Good morning"이 나옵니다.

앞서 만들었던 소스는 다음과 같이 바꾸면 될 듯 합니다.

from google\_trans\_new import google\_translator from pptx import Presentation translator = google\_translator() prs=Presentation("Trans\_Sample.pptx") Slides = prs.slides for slide in Slides: Shapes = slide.shapes for shape in Shapes: if shape.has\_text\_frame: text\_frame = shape.text\_frame for paragraph in text\_frame.paragraphs: font=paragraph.runs[0].font Fname = font.name Fsize = font.size cur\_text = paragraph.text new\_text = translator.translate(cur\_text,lang\_tgt='en') paragraph.text = new\_text for run in paragraph.runs: font = run.font font.name = Fname font.size = Fsize prs.save('T\_Result.pptx')

1. google-cloud-translate

이 라이브러리는 Google에서 공식적으로 제공하는 공식 API를 사용할 수 있는 라이브러리 입니다. pip install --upgrade google-cloud-translate

실제로 자주 사용할 일이 있다면, 이 라이브러리를 끌어와서 사용하는 것이 필요할 것 같습니다.

그런데, 이 라이브러리는 Google Cloud Platform에 계정 등록하고, 유료 계정도 만들고, API도 등록하고, Key도 받은 다음 사용이 가능하여, 꽤나 사용전 준비사항도 많이 필요하고, 사용법도 좀 더 복잡합니다.



### [Google Cloud Platform](https://console.cloud.google.com/)

[One account. All of Google. Sign in to continue to Google Cloud Platform Find my account Create account One Google Account for everything Google](https://console.cloud.google.com/)

[console.cloud.google.com](https://console.cloud.google.com/)

※ Google API 등록하고 Key를 만들고 등록하는 과정은 생각보다 까다롭네요... 우선은 제가 참고로 찾아봤던 영상을 아 래 링크로 걸어두고, 저 또한 다음 글과 영상에서 별도로 다뤄 보겠습니다.

사용법도 좀 까다롭고, 한 번에 번역하는 단어 수가 일정 단어수(100만)를 초과하면 유료로 비용이 청구될 수 있어서 사용 시 유의하긴 해야 하지만, 정말 번역의 사용량이 많고, 일정 비용을 지불하더라도 꼭 필요하다면 여기까지 활용하는게 맞 을 것 같습니다.

라이브러리를 깔고 난 다음에도 Google-Cloud를 사용하기 위한 키를 파일로 다운로드 받고, 다운로드 받은 파일 경로를 window 환경에 적용해 줘야 합니다.

from google.cloud import translate\_v2 as translate import os os.environ['GOOGLE\_APPLICATION\_CREDENTIALS'] = r"Google-Cloud Key가 저장된 파일 경로" client = translate.Client() result = client.translate('안녕하세요', target\_language='en') print(result['translatedText']) 결과값으로 "Good morning"이 나옵니다.

앞서 만들었던 소스를 다음과 같이 바꾸면 될것 같습니다.

from google.cloud import translate\_v2 as translate import os from pptx import Presentation os.environ['GOOGLE\_APPLICATION\_CREDENTIALS'] = r"Google-Cloud Key가 저장된 파일 경로" client = translate.Client() prs=Presentation("Trans\_Sample.pptx") Slides = prs.slides for slide in Slides: Shapes = slide.shapes for shape in Shapes: if shape.has\_text\_frame: text\_frame = shape.text\_frame for paragraph in text\_frame.paragraphs: font=paragraph.runs[0].font Fname = font.name Fsize = font.size cur\_text = paragraph.text result = client.translate(cur\_text, target\_language='en') new\_text = result['translatedText'] paragraph.text = new\_text for run in paragraph.runs: font = run.font font.name = Fname font.size = Fsize prs.save('T\_Result.pptx')

번역기 기능을 포함하여, 여러 파일을 한번에 실행할 수 있게 만든 실행 파일을 아래의 링크에서 다운로드 받으실 수 있습 니다.

### [파이썬 나만의 RPA 툴 제작 프로젝트 (Tricks\_Office)](https://tricks-office.com/python-tricks_office/)

[파이썬으로 나만의 RPA 툴을 직접 만드는 장기 프로젝트, 프로젝트명 “Tricks\_Office” 를 진행하고자 합니다. 구현을 계 획하고 있는 기능은 다음과 같습니다. 여러 PPT 파일의 여러 단어(Excel 형태로 보관된 Dictionary 파일 기준)를 원하는 단어로 치환하기 (구현)양식이 맞춰진 엑셀 파일을 하나의 파일로 취…](https://tricks-office.com/python-tricks_office/)

[tricks-office.com](https://tricks-office.com/python-tricks_office/)

# 구글 Cloud 서비스로 날개를 단 파이썬

## Python

2021/06/10 06:50 <http://blog.naver.com/550sn/222362744092>



구글에서는 우리가 생각하는 것보다 많고도 강력한 서비스를 직/간접적으로 제공하고 있습니다.

직접적으로 구글을 활용하지 않더라도 많은 앱에서 구글에서 제공하는 Cloud 서비스를 기반으로 하는 서비스가 제공됩니 다.

가장 대표적인 Cloud 서비스중의 하나가 Maps 일것 같은데, 그 외에도 번역기나 이미지 인식 등과 같은 처음부터 개발한 다고 하면 막막하기까지한 서비스를 제공해 주고 있습니다.

파이썬에서도 이와 같이 Google Cloud에서 제공하는 API 기능을 활용할 수 있으며, Google에서는 이를 위한 라이브러 리도 별도로 제공하고 있습니다. 다만, 이와 같은 API 기능을 활용하기 위해서는 개발자 정보를 등록하고, 프로젝트를 등 록한 다음 API 키를 받는 일련의 작업이 필요합니다.

이번에는 Google Cloud 서비스 중의 하나인 번역과 이미지 인식에 관련된 API를 등록하고 내 컴퓨터의 Python에서 활 용하는 방법을 알아 보겠습니다.

1. Google cloud 서비스 등록하기

Google Cloud API를 활용하기 위해서 우선 Google API 개발자 등록을 해야 합니다.

### [Google Cloud Platform](https://console.cloud.google.com/)

[One account. All of Google. Sign in to continue to Google Cloud Platform Find my account Create account One Google Account for everything Google](https://console.cloud.google.com/)

[console.cloud.google.com](https://console.cloud.google.com/) 결재 등록하고

새 프로젝트 등록하고 API 추가 하고

사용자 인증정보 등록 사용자 계정 등록

키 등록

1. API 가격 정책

Google Cloud API 서비스는 Google Cloud 서비스에 서비스를 요청한 다음 결과값을 받는 방식으로 진행됩니다.

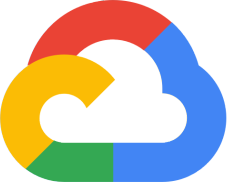
이와 같은 Google Cloud API 서비스는 매일 특정 건수에 대해서는 무료로 제공되지만, 특정 건수를 초과 했을 경우 추가 비용이 발생하는 형태로 비용이 발생합니다.

대개의 경우 해당 서비스는 개인 차원에서 활용할 때에는 무료로 서비스를 받을 수 있는 수준의 서비스가 제공됩니다.

1. Google Translate

Google Translate 서비스의 가격정책을 살펴보면 아래와 같습니다.





### [가격 책정 | Cloud Translation | Google Cloud](https://cloud.google.com/translate/pricing)

[Cloud Translation 문서 리소스 평가 및 리뷰 가격 책정 목차 월별 비용 Cloud Translation - Basic Cloud Translation - Advanced 비용 청구 대상 문자 비용 청구 대상 프로젝트 월별 사용량을 기반으로 Cloud Translation의 비용이 청구됩니](https://cloud.google.com/translate/pricing)

[다. 이 페이지에서는 Cloud Translation - Basic 및 Cloud Translation - Advanced의 사용 수수료를 설명합니다. 텍스트 번역 사용량은 100만 자(영문 기준) 단위로 계산되며 100만은 10 6 ...](https://cloud.google.com/translate/pricing)

[cloud.google.com](https://cloud.google.com/translate/pricing)

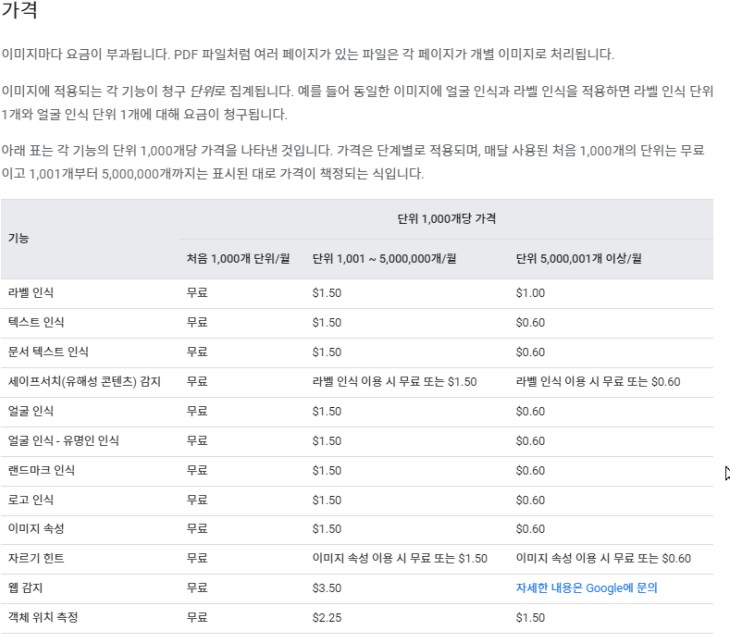
기본 서비스의 가격 정책을 보면 월 50만자가 무료로 제공되는 것을 볼 수 있습니다.

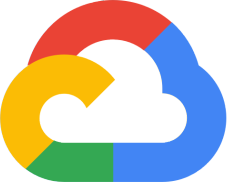
그리고, 50만자 초과시 100만자 마다 $20의 비용이 추가로 청구되는 형식인 것을 알 수 있습니다.

월 50만자 까지는 비용을 들이지 않고 활용이 가능하여 개인적인 목적으로 번역에 활용할 경우라면 충분히 활용할 수 있 으리라 생각됩니다.

1. Google Vision

Google Vision 서비스의 가격정책을 살펴보면 아래와 같습니다





### [가격 책정 | Cloud Vision API | Google Cloud](https://cloud.google.com/vision/pricing)

[Cloud Vision API 문서 리소스 평가 및 리뷰 가격 책정 목차 가격 예시 Google Cloud Platform 비용 Cloud Vision API 는 이미지를 분석하기 위한 여러 기능을 제공합니다. 아래 시나리오에서는 선불 약정 없이 사용한 만큼만 지불하면 됩니 다. 이 API에서 지원하는 기능은 다음과 같습니다. 기능 유형 CROP\_HINTS 이미지에서 자르기 영역으로 제안되는 꼭짓점 을 결정합니다. DOCUMENT\_TEXT\_DETECTION 문서(PDF/TIFF)와 같은 밀집 텍스트 이미지와 필기 입력이 포함된 이 미지에 ...](https://cloud.google.com/vision/pricing)

[cloud.google.com](https://cloud.google.com/vision/pricing)

월 1,000개의 이미지에 대해서는 무료로 서비스가 제공되고, 이후 1,000개 이미지당 $1.5의 비용이 추가로 발생되는 것 을 볼 수 있습니다.

월 1,000개 이미지까지는 무료로 사용이 가능하여 마찬가지로 개인적인 용도 범위에서 활용데에는 큰 무리가 없다고 생 각됩니다.

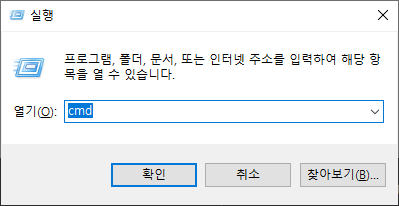
1. 라이브러리 설치 및 활용

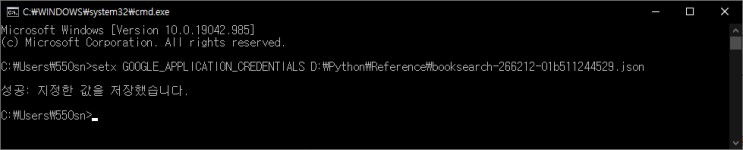
각각의 라이브러리 이름은 "google-cloud-translate" 와 "google-cloud-vision" 입니다. 각각 설치 하려면 터미널에서 아래와 같이 입력하면 됩니다.

pip install --upgrade google-cloud-translate pip install --upgrade google-cloud-vision

소스 코드를 만들때 다른 라이브러리와 마찬가지로 import로 라이브러리를 불러오면 되는데, 한가지 더 신경을 써줘야 하 는것이 Google API 키를 사용할수 있게 등록해 줘야 한다는 것입니다.

앞서 0번 Google Cloud 서비스 등록에서 다운로드 받았던 API 키를 컴퓨터 환경에서 인식하게 만들어 줘야 합니다. Windows + R을 눌러서 시 윈도우 실행창을 띄운 다음, cmd 를 치고 엔터



윈도우 직접 입력창을 연 다음, 아래와 같이 입력해 주면 해당 API키가 윈도우에서 인식이 되게 설정 됩니다. setx GOOGLE\_APPLICATION\_CREDENTIALS API키 json파일 경로

제대로 입력이되면 위와 같이 지정한 값을 저장했다는 메시지가 뜹니다.

이렇게 Setting이 되면, 활용이 가능하다고 되어 있는데, 저의 경우에는 제대로 실행이 되지 않는 것을 볼 수 있었습니다.

저의 경우에는 소스 코드에서 라이브러리를 호출 한 다음, 아래와 같이 코드를 추가하여 환경 설정에 추가해 주면 사용이 가능했습니다.

from google.cloud import translate\_v2 as translate from google.cloud import vision import os os.environ['GOOGLE\_APPLICATION\_CREDENTIALS'] = "API키 json 파일 경로"

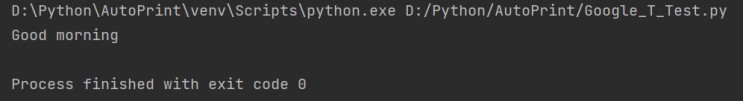
1. Google Cloud API 활용하기

간단한 소스 파일로 확인을 해 보면 다음과 같은 내용이 확인이 가능합니다.

1. 번역기

from google.cloud import translate\_v2 as translate import os os.environ['GOOGLE\_APPLICATION\_CREDENTIALS'] = "API키 json 파일 경로" client = translate.Client() result

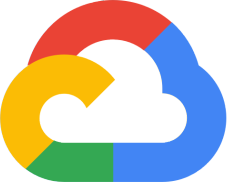
= client.translate('안녕하세요', target\_language='en') print(result['translatedText'])

실행을 하면 아래 이미지와 같이 번역 결과인 "Good morning"값을 회신 주는 것을 볼 수 있습니다.

**첨부파일** Google\_T\_Test.py [파일 다운로드](http://blogattach.naver.net/66f37acadb8b825e7290f7c5f918621ebdef10fbe9/20210523_56_blogfile/550sn_1621760921061_Ev1LI5_py/%2FGoogle_T_Test.py)

1. Vision (그림 인식)

Vision 라이브러리 사용법에 대한 내용은 다음의 링크에서 가이드 받을 수 있습니다.



### [Vision 클라이언트 라이브러리 | Cloud Vision API | Google Cloud](https://cloud.google.com/vision/docs/libraries)

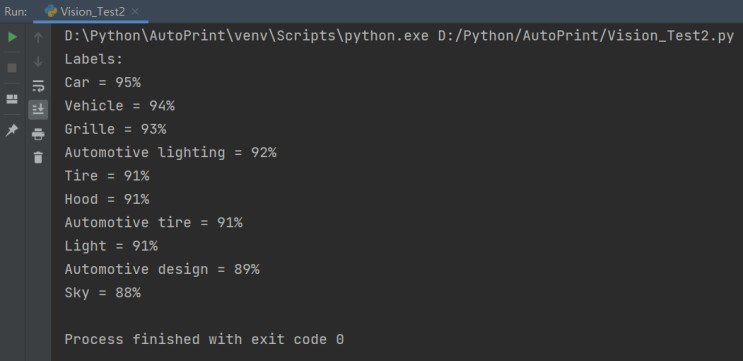
[C#, Go, 자바, Node.js, PHP, Python 또는 Ruby로 Vision 코드 작성을 시작하세요. cloud.google.com](https://cloud.google.com/vision/docs/libraries)

이를 기반으로 아래와 같이 소스를 만들 수 있습니다.

import io import os # Imports the Google Cloud client library from google.cloud import vision os.environ['GOOGLE\_APPLICATION\_CREDENTIALS'] = "API키 json 파일 경로" # Instantiates a client client = vision.ImageAnnotatorClient() # The name of the image file to annotate file\_name = os.path.abspath('이미지 파일 경로') # Loads the image into memory with io.open(file\_name, 'rb') as image\_file: content = image\_file.read() image = vision.Image(content=content) # Performs label detection on the image file response = client.label\_detection(image=image) labels = response.label\_annotations print('Labels:') for label in labels: print(label.description+ " = " + str(int(label.score\*100)) + "%")

다음과 같은 이미지에 대해서 위의 소스를 실행할 경우



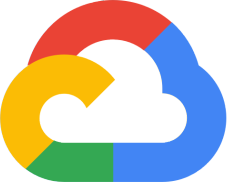
다음과 같은 결과를 얻을 수 있습니다.

이미지를 인식한 결과, 95%의 확률로 "Car"에 대한 이미지로 인식한 것을 볼 수 있습니다. **첨부파일**

Vision\_Test2.py [파일 다운로드](http://blogattach.naver.net/67f27bcbda8b835f7391f6c4f819631fbcee10f7e8/20210523_241_blogfile/550sn_1621760954104_DOHs03_py/%2FVision_Test2.py)

1. Vision (그림 문자로 인식-OCR)

Google-cloud-vision에서는 글자를 문서로 인식하는 서비스도 제공합니다. 아래의 링크에서 참고자료를 찾아 볼 수 있습니다.



### [이미지의 텍스트 감지 | Cloud Vision API | Google Cloud](https://cloud.google.com/vision/docs/ocr)

[Cloud Vision API 문서 가이드 평가 및 리뷰 이미지의 텍스트 감지 목차 광 문자 인식(OCR) 텍스트 감지 요청 GCP 프로 젝트 및 인증 설정 로컬 이미지의 텍스트 인식 원격 이미지의 텍스트 인식 광 문자 인식(OCR) Vision API는 이미지에서 텍스트를 감지하고 추출할 수 있습니다. 다음과 같은 두 가지 주석 특징에서 광 문자 인식(OCR)을 지원합니다.](https://cloud.google.com/vision/docs/ocr)

[TEXT\_DETECTION 은 임의의 이미지에서 텍스트를 감지하고 추출 합니다. 간판이나 표지판이 찍힌 사진을 예시로 들 수 있습니다. JSON은 추출된 전체 ...](https://cloud.google.com/vision/docs/ocr)

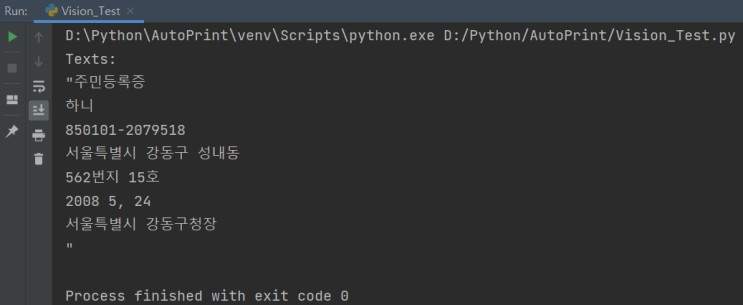
[cloud.google.com](https://cloud.google.com/vision/docs/ocr)

이를 기반으로 아래와 같은 소스를 구성할 수 있습니다.

import os os.environ['GOOGLE\_APPLICATION\_CREDENTIALS'] = "API json 키 파일 경로" from google.cloud import vision import io client = vision.ImageAnnotatorClient() path="이미지 파일 경로" with io.open(path, 'rb') as image\_file: content = image\_file.read() image = vision.Image(content=content) response = client.text\_detection(image=image) texts = response.text\_annotations print('Texts:') print('"{}"'.format(texts[0].description))

다음의 이미지가 있을 경우



이를 실행하면 다음과 같이 결과 값을 얻을 수 있습니다.

**첨부파일** Vision\_Test.py [파일 다운로드](http://blogattach.naver.net/8c199027380706b39a7c1e2e14f28df75705f38f/20210523_255_blogfile/550sn_1621760966020_A1ayZv_py/%2FVision_Test.py)

Youtube 영상을 찍으면서 만든 파일은 다음과 같습니다. 전체 내용은 대동 소이 합니다.

**첨부파일** GoogleCloud.zip [파일 다운로드](http://blogattach.naver.net/11840dbeaaf3f52905e780b28e6f1569ca98658500/20210525_139_blogfile/550sn_1621914463145_zhrC5P_zip/%2FGoogleCloud.zip)

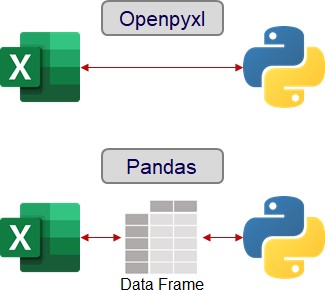
# 판다와 손잡은 파이썬(파이썬 엑셀 다루기)

## Python

2021/06/14 06:50 <http://blog.naver.com/550sn/222362866341>



제가 파이썬을 다루면서 엑셀을 다루는 라이브러리로 openpyxl을 사용 했었는데, Python에 대해 공부를 해 가다 보니 openpyxl보다 pandas 라이브러리의 사용 빈도가 훨씬 높은것 같습니다.

이 두가지 솔루션이 엑셀 파일을 다루는 방식에는 차이가 있는데, openpyxl은 엑셀파일을 있는 그대로 활용하는 방식인 반면, pandas는 엑셀과 같은 스프레드시트 형식의 데이터를 별도의 데이터 관리 형태인 "Data Frame"의 형태로 변환하 여 활용하는 방식입니다.

엑셀 파일의 형태 있는 그대로를 다루는 Openpyxl이 중간단계가 없어서 더 효율적이고 빠를것 같은데, 실제 비교를 해보 면 그렇지는 않습니다. Data Frame의 형태로 데이터를 전환한 다음 다루는 것이 오히려 훨씬 빠릅니다.

이는 이전에도 엑셀 파일 자체가 압축파일이다 보니 직접적으로 엑셀파일을 다루기 위해 압축을 풀었다 저장했다를 반복 하기 때문에 오히려 속도가 느린것이 아닐까 하는 생각이 듭니다.

(※ 엑셀 파일 자체가 압축파일이라는 사실은 이전에 "MS 파일의 비밀"이라는 글로 다룬 적 있습니다.)



### [MS 파일의 비밀(엑셀 보호 강제 해제)](https://blog.naver.com/550sn/222014376518)

[오늘은 MS Office 파일의 충격적인 사실에 대해 알아볼까 합니다. 결론부터 한마디로 말씀 드리면, MS... blog.naver.com](https://blog.naver.com/550sn/222014376518)

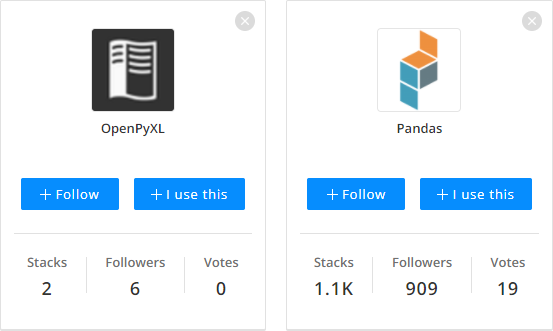
따라서, 시스템이 보다 효율적으로 다룰수 있는 데이터 형태로 전환하여 작업을 완료한 다음에 다시 저장하는 방식을 취하 는 것이 보다 효율적일 수 있기 때문에 이와 같은 속도 차이가 나오는 것으로 보입니다.

아래 링크를 찾아가 보면 pandas와 openpyxl의 인기도(?)를 비교한 내용이 있는데, 압도적으로 pandas가 인기가 있는 것을 알 수 있습니다.

### [OpenPyXL vs Pandas | What are the differences?](https://stackshare.io/stackups/openpyxl-vs-pandas)

[OpenPyXL - We host the world's documentation. Maintained by @ericholscher & @agjhnsn. Not a support channel, please use GitHub.. Pandas - High-performance, easy-to-use data structures and data analysis tools for the Python programming language.](https://stackshare.io/stackups/openpyxl-vs-pandas)

[stackshare.io](https://stackshare.io/stackups/openpyxl-vs-pandas)



openpyxl과 pandas의 속도 비교에 대한 내용도 나오는데, 여기서도 pandas가 훨씬 빠른것을 볼 수 있습니다.

### [[Pandas] Excel Writer 비교 (xlsxwriter VS. openpyxl)](http://blog.naver.com/PostView.nhn?blogId=wideeyed&logNo=221803761031)

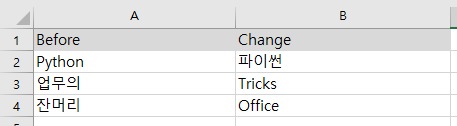
[Pandas(판다스, 팬더스)에서 제공하는 엑셀저장 엔진 xlsxwriter, openpyxl 성능을 비교해보겠습니다. 모... blog.naver.com](http://blog.naver.com/PostView.nhn?blogId=wideeyed&logNo=221803761031)

저 또한 이번에 pandas에 대해 보다 자세히 알아봄에 따라 openpyxl보다 더 애착이 가는것 또한 사실이네요. 오늘은 pandas에 대해서 알아보도록 하겠습니다.

pandas는 openpyxl과 다르게 excel을 다루기 위한 라이브러리라기 보다는 excel과 같은 표 형태로 만들어진 데이터를 가공하기 좋은 Data 형태로 전환하여 다루는 라이브러리라고 봐야 합니다.

대표적으로 많이 사용되는 데이터 형태가 시리즈(series)와 데이터프레임(Data Frame)인데, 시리즈는 열이 하나인 데이 터프레임으로 이해하면 되므로 데이터 프레임 형태에 대해서만 살펴 보도록 하겠습니다.

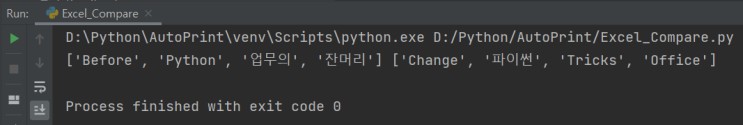
앞선 PPT 바꿀 글자들에 대한 정보를 가져오는 로직을 만들때, 즉, 아래 이미지와 같은 표 정보를 pyton에서 다룰 수 있게 가져올 때



**첨부파일** Dictionary.xlsx [파일 다운로드](http://blogattach.naver.net/f366ef5b467879cce50361516b8df288287f80d9/20210523_291_blogfile/550sn_1621763881351_zoXL9s_xlsx/%2FDictionary.xlsx)

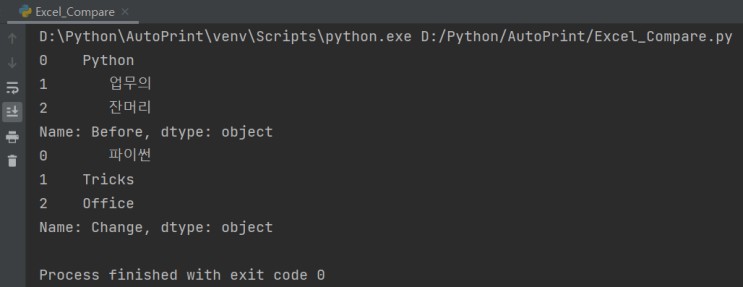
다음과 같은 방식으로 openpyxl을 활용하여 데이터를 가져온 적이 있었습니다.

import openpyxl wb = openpyxl.load\_workbook('Dictionary.xlsx') ws = wb.active rows = ws.rows BW = [] AW = [] for cell in rows: BW.append(cell[0].value) AW.append(cell[1].value) print (BW,AW)

실행해서 결과를 보면 다음과 같이 결과값을 볼 수 있습니다.

그런데, 이와 비슷한 동작을 하기 위해서 pandas에서는 훨씬 간단한 코드로 실행이 가능하고, 만들어진 데이터의 구조도 활용이 훨씬 용이합니다.

import pandas df = pandas.read\_excel('Dictionary.xlsx',sheet\_name='Sheet1') print (df.Before) print (df.Change)

실행한 결과를 보면 다음과 같이 결과값을 볼 수 있습니다.

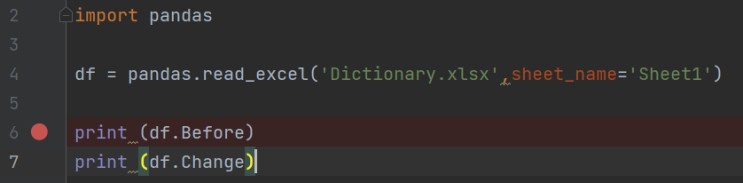
훨씬 코드도 간결하고, 데이터 파일 자체가 정리가 잘되어 있다는 가정하에서는 데이터의 헤더 정보 이름으로 주요 값들을

쉽게 불러올 수 있습니다.

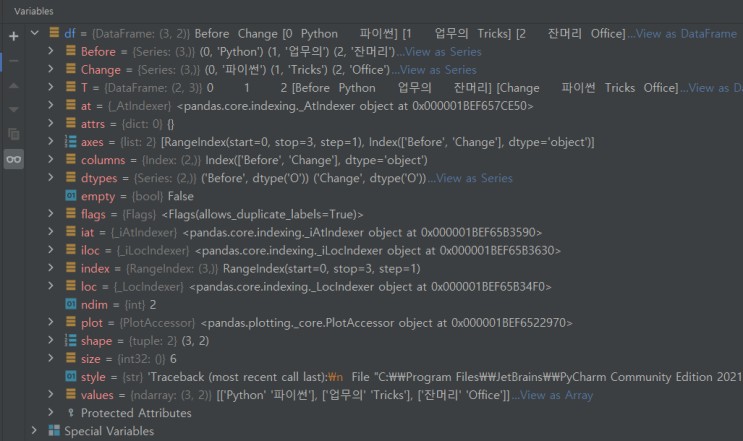
속도도 빠르고, 코드도 간결하지만 Data Frame 형태로 데이터를 변환하여 활용하는 것을 전제로 하고 있기 때문에 엑셀 파일 자체가 다음과 같은 전제 조건을 만족하고 있을 때에는 pandas를 활용하는 것이 더 좋을 수 있습니다.

1. Sheet 안에 하나의 표형태의 정보 Set만 있다.
2. 셀 병합이 없다
3. 1개의 머리글행이 있고, 머릿 글행 정보중에 빈칸이 없다. 여기서 3번조건을 살펴볼까 합니다.

앞서 만들었던 엑셀파일, 그리고 파이썬 코드에서 print 하기전에 중단점을 찍고 Debug 모드로 실행하여 중간에 df라는 Data Frame이 이떻게 데이터를 가지고 있는지를 살펴 보겠습니다.

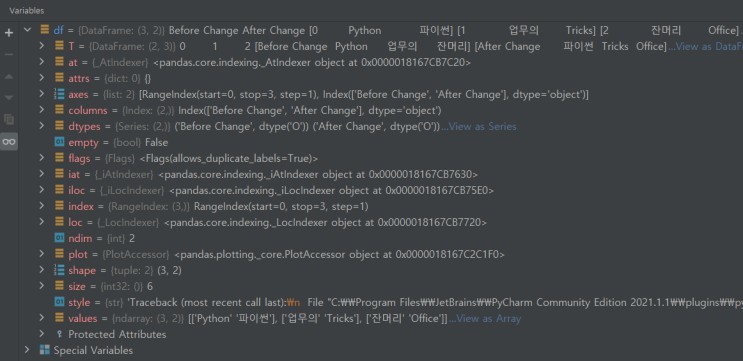


중간에 멈췄을 때 'Variables' 탭에서 정보를 확인하면



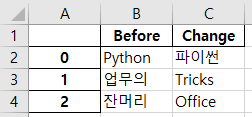
위의 이미지와 같이 헤더이름으로 된 df의 child 개체가 생성 된 것을 볼수 있습니다. 따라서 필요한 값을 쉽게 끌어 올 수 가 있었습니다.

하지만, 아래 이미지와 같이 엑셀 파일의 머릿글행 정보의 중간에 공백이 포함되어 있을 때에는



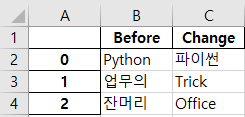
위 이미지와 같이 헤더이름을 사용한 child 개체가 생성이 되지 않는 것을 볼 수 있고, 이 경우 values 에서 행/열의 조합으 로 필요한 값을 가져올 수 밖에 없음을 알 수 있습니다.

pandas를 활용하여 데이터를 저장하는 것은 다음과 같이 정리할 수 있습니다. 데이터프레임.to\_excel('저장할 파일 이름')

앞서 살펴본 소스들을 이어 붙여 보고, 샘플 파일을 정리해보면 다음과 같이 소스를 만들 수 있을 듯 합니다. import pandas df = pandas.read\_excel('Dictionary.xlsx',sheet\_name='Sheet1') df.to\_excel('Dic.xlsx') 결과로 만들어진 Dic.xlsx 파일은 다음과 같이 찾을 수 있습니다.

이번에는 중간에 Change 의 2번째값, Tricks를 Trick으로 바꾼 다음 저장해 보도록 하겠습니다.

import pandas df = pandas.read\_excel('Dictionary.xlsx',sheet\_name='Sheet1') df.Change[1] = "Trick" df.to\_excel('Dic.xlsx')

결과로 만들어진 엑셀파일은 다음과 같습니다.

**첨부파일** Excel\_Compare.py [파일 다운로드](http://blogattach.naver.net/31a42d9e8fd1d50925c7a092ae4f3549eab842a2fd/20210523_141_blogfile/550sn_1621767748920_cxabSh_py/%2FExcel_Compare.py) **첨부파일**

Dic.xlsx

[파일 다운로드](http://blogattach.naver.net/6ffa73c3d38d8b577b99feccf0116b17b4e61cf3eb/20210523_93_blogfile/550sn_1621767757461_7JTcY4_xlsx/%2FDic.xlsx)

원본 엑셀 파일이 전처리가 잘 되어 있어서 대량의 데이터를 다루기 유리하다면 pandas를, 엑셀 자체의 기능을 많이 활용 하여 디자인적인 부분의 편집이 많이 필요하다면 openpyxl을 활용하는 것이 보다 유리할것 같다는 생각이 듭니다.

Youtube 영상을 만들면서 만든 소스 파일은 다음과 같습니다. 내용은 대동 소이합니다. **첨부파일**

ExcelCompare.py [파일 다운로드](http://blogattach.naver.net/52c74efeedbfb66a46a4c3f1cd2c562a89db21cf9c/20210525_39_blogfile/550sn_1621914571088_3RuLH1_py/%2FExcelCompare.py)

# 파이썬으로 엑셀 파일 PPT로 자동 전환하기

## Python

2021/06/17 06:50 <http://blog.naver.com/550sn/222362968315>

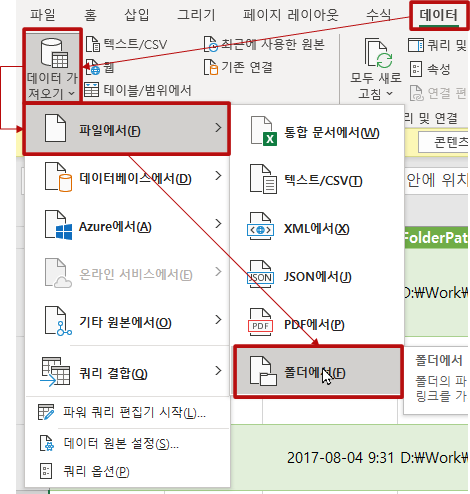


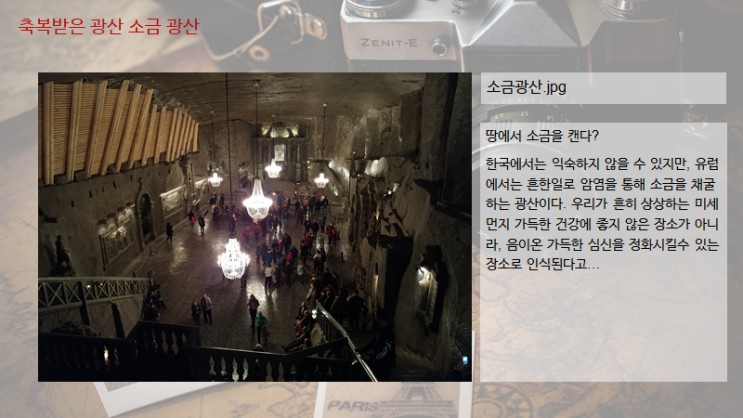
업무를 하다 보면 잘 정리된 많은 양의 엑셀 데이터를 특정한 양식의 PPT 파일로 한번에 전환해야 하는 경우가 있습니다.

다음과 같이 엑셀 파일로 여행을 다녀온 여행지 정보가 잘 정리되어 있다고 했을 때

**첨부파일** Auto\_Excel.xlsx [파일 다운로드](http://blogattach.naver.net/7eeb62d2c39b9a466a88efdde1007a06a5f70fef94/20210523_4_blogfile/550sn_1621771871366_c753Dt_xlsx/%2FAuto_Excel.xlsx)

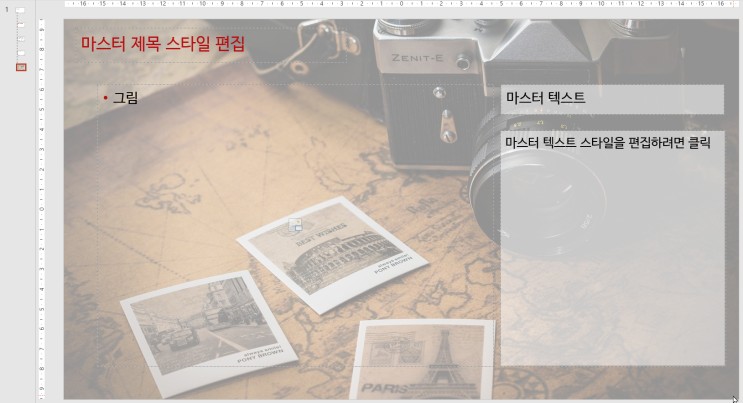
※ 위 엑셀 파일의 경우 이미지 파일을 하나의 폴더에 모아두고 엑셀에서 데이터 → 데이터 가져오기 → 파일에서 → 폴더 에서를 선택하면 이미지 경로 파일과 파일명은 쉽게 정리할 수 있습니다.



이 정보를 아래와 같은 포멧으로 이미지도 붙여 넣고, 제목과 설명, 사진 파일 이름을 붙여 넣는다고 하겠습니다.

python에서 ppt를 다루기 위해서는 python-pptx를 사용할 것입니다.

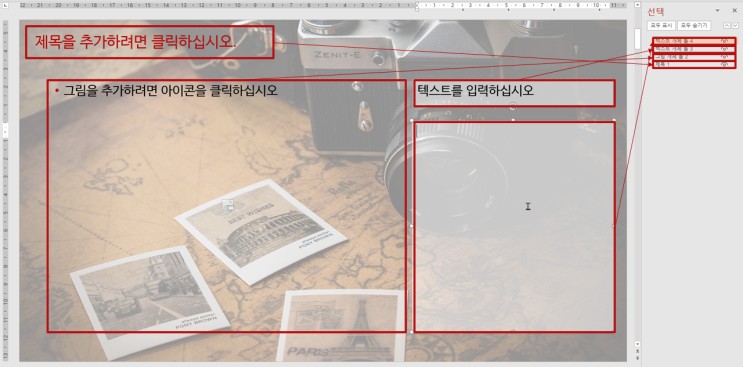
위와 같은 포멧으로 반복해서 이미지를 넣기 위해서는 원하는 양식의 Lay-out을 슬라이드 마스터에 포함시켜 둬야 합니 다.

아래와 같은 형식으로 Lay-out을 먼저 하나 만들어 둡니다.

이때 레이아웃 왼쪽의 마스터 축소판 그림을 살펴보면 첫번째의 마스터 슬라이드를 제외하고, 4번째 lay-out에 지금 구성 한 레이아웃이 포함된 것을 볼 수 있습니다. 이는 0부터 세었을때 "3"에 해당하는 고유번호를 가지는 레이아웃임을 알 수 있습니다.

슬라이드 마스터를 빠져 나와서 새 슬라이드를 하나 추가하고, 추가된 슬라이드는 우리가 새로이 설정한 레이아웃을 적용 합니다.

메뉴에서 홈 → 선택 → 선택창을 선택하면 우측에 각 개체의 이름들이 표시됩니다.



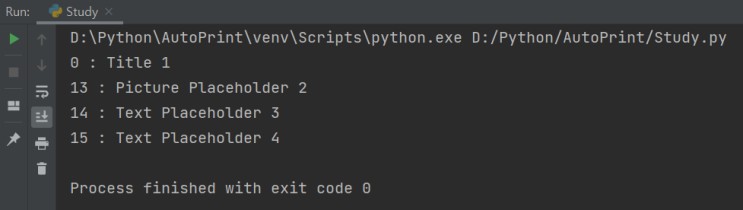
**첨부파일** Auto\_PPT\_Format.pptx [파일 다운로드](http://blogattach.naver.net/4fda53e3f6a3ab775bb9deecd0314b3794c631dbe1/20210525_245_blogfile/550sn_1621914667164_0yX6Lk_pptx/%2FAuto_PPT_Format.pptx)

각각의 이름들이 고유 이름과 번호가 만들어진 것을 확인이 가능한데, 이 번호를 참고로 하여 python-pptx에서는 슬라이 드를 추가했을때 각각의 개체를 어떻게 인식하는지를 살펴 보겠습니다.

먼저 터미널에 ppt를 다루기 위한 라이브러리인 "python-pptx"를 설치해 둡니다. pip install python-pptx

python에서 아래와 같이 코드를 만든다음 먼저 실행을 해 봅니다.

from pptx import Presentation prs = Presentation('Auto\_PPT\_Format.pptx') add\_slide\_layout = prs.slide\_layouts[3] slide = prs.slides.add\_slide(add\_slide\_layout) shapes = slide.shapes for shape in shapes: print (str(shape.placeholder\_format.idx) + " : " + shape.name)

실행 결과 창에서 다음의 정보를 얻을 수 있습니다.

이 정보를 통해 '제목 1'이 0번, '그림 개체 틀 2'이 13번, '텍스트 개체 틀 3'이 14번, '텍스트 개체 틀 4'가 15번 이라는 placeholder 고유 번호를 가지는 것을 확인할 수 있었습니다.

이제 Pandas로 엑셀에서 필요한 정보를 가져오는 로직을 추가한 다음, 우선 첫번째 값들을 방금 확인한 placeholder위치 에 넣겠습니다.

엑셀 파일에서 확인을 했을 때 제목에 들어갈 값은 Head 정보가 "Title", 이미지에 들어갈 정보는 그림을 삽입하되, 그림 경로는 "FolderPath" + "Name", 이미지 이름 정보는 "Name", 이미지에 대한 설명 정보는 "Description"으로 정의 되어 있기 때문에, pandas로 읽어들인 값은 각각의 Head명으로 정리되어 저장이 될 것입니다.

따라서 아래와 같이 입력해 주면 1장의 슬라이드 정보가 추가 되고, 첫번째 이미지에 대한 정보와 이미지가 삽입 될 것입 니다.

import pandas from pptx import Presentation df = pandas.read\_excel('Auto\_Excel.xlsx',sheet\_name='IMG') prs = Presentation('Auto\_PPT\_Format.pptx') add\_slide\_layout = prs.slide\_layouts[3] slide = prs.slides.add\_slide(add\_slide\_layout) shapes = slide.shapes slide.placeholders[0].text= df.Title[0] img = slide.placeholders[13] img.insert\_picture(df.FolderPath[0] + df.Name[0]) slide.placeholders[14].text = df.Name[0] slide.placeholders[15].text = df.Description[0] prs.save('Auto\_PPT\_Result.pptx')

아래 이미지와 같이 한장의 슬라이드가 추가되고 필요한 정보도 정상적으로 잘 입력 된 것을 볼 수 있습니다.

이제 데이터를 모두 읽을 때까지 반복하게 반복문만 한줄 추가하고, 첫번째 개체를 가져오던것을 행을 바꿔가면서 실행하 게 i값으로 대체해 주겠습니다.

import pandas from pptx import Presentation df = pandas.read\_excel('Auto\_Excel.xlsx',sheet\_name='IMG') prs = Presentation('Auto\_PPT\_Format.pptx') for i in df.index: add\_slide\_layout = prs.slide\_layouts[3] slide = prs.slides.add\_slide(add\_slide\_layout) shapes = slide.shapes slide.placeholders[0].text= df.Title[i] img = slide.placeholders[13] img.insert\_picture(df.FolderPath[i] + df.Name[i]) slide.placeholders[14].text = df.Name[i] slide.placeholders[15].text = df.Description[i] prs.save('Auto\_PPT\_Result.pptx')

실행한 결과는 다음과 같습니다.

**첨부파일** Study.py [파일 다운로드](http://blogattach.naver.net/ce5bd261712b2af6da385f6d51b0cab61547b05de0/20210523_21_blogfile/550sn_1621771915865_teaq0r_py/%2FStudy.py) **첨부파일**

Auto\_PPT\_Result.pptx [파일 다운로드](http://blogattach.naver.net/f065ec594c7b7acfe6006252688ef18b2b708edf/20210523_191_blogfile/550sn_1621772388267_SZ7eyu_pptx/%2FAuto_PPT_Result.pptx)

※ 결과 pptx 파일은 용량이 부족하여 이미지를 많이 삭제한 상태로 올렸습니다.

Youtube 영상을 만들면서 다시 만든 코드는 아래와 같습니다. 내용은 대동 소이합니다. **첨부파일**

AutoPPT.py [파일 다운로드](http://blogattach.naver.net/7bee67d7c69b9f436f8dead8e4057f03a0f204ef98/20210525_22_blogfile/550sn_1621914693163_47ErS2_py/%2FAutoPPT.py)

# 쓰기 편하게 다시 태어나는 파이썬 (GUI, Tkinter)

## Python

2021/06/21 06:50 <http://blog.naver.com/550sn/222375152366>



지금까지 파이썬을 활용하여 많은 유용한 기능을 구현했습니다.

기능 기능 자체는 더할나위 없는데, 사실 만들어진 기능을 사용하는 것이 사용자 친화적이라고 보기에는 어려움이 있습니 다.

오늘부터 정리하는 작업은 만드는 우리는 꽤 많이 괴롭더라도, 일반 사용자도 사용할 수 있는 GUI 환경을 만들고, 파이썬 이 깔려있지 않은 곳에서도 실행이 가능하게 프로그램을 재구성하는 것을 알아보도록 하겠습니다.

금번 글에서는 먼저 GUI 환경을 꾸미는 것 까지 알아보도록 하겠습니다.

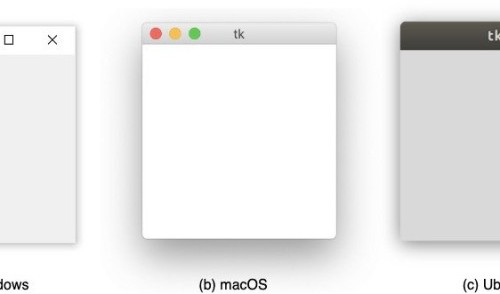
지금까지 파이썬에서 만들어진 프로그램은 파이참에서 실행 명령을 내리면서 실행되는 형태로 진행되어 왔습니다.

그에 비해 우리가 일반적으로 사용하는 프로그램을 생각해 보면, 새로운 창을 열고 버튼을 누르는 등의 방식으로 프로그램 을 사용하는 것을 알 수 있습니다.

이들 창, 버튼과 같이 그래픽을 기반으로 한 사용자 환경을 GUI(Graphic User Interface) 라고 부르며, Python에서도 이 와 같은 그래픽 기반의 명령창을 만들수 있는 기능을 제공합니다.

GUI를 구성하기 위한 많은 라이브러리가 있지만, Python에 이미 기본적으로 내장된 라이브러리로 "tkinter"라는 라이브 러리가 있습니다. 선호도도 높고 하니, 저는 이 "tkinter" 라이브러리를 활용하여 화면을 구성해 보겠습니다.

Google에서 "tkinter tutorial"이라고 검색을 하면 역시 많은 설명 자료가 나옵니다. 처음에 보기 좋은 자료는 아래의 링 크로 찾아진 자료였습니다.



### [Python GUI Programming With Tkinter – Real Python](https://realpython.com/python-gui-tkinter/)

[In this article, you'll learn the basics of GUI programming with Tkinter, the de-facto Python GUI framework. Master GUI programming concepts such as widgets, geometry managers, and event handlers. Then, put it all together by building two applications: a temperature converter and a text editor.](https://realpython.com/python-gui-tkinter/)

[realpython.com](https://realpython.com/python-gui-tkinter/)

기본적인 사용법에 대한 설명으로는 기본 개념을 잡기는 좋은데, 저는 역시 실전 위주로 찾는것이 좋은지라, 여러가지 경 우를 따지며 검색을 해 봤습니다.

저의 경우 가장 이해하기 좋고, 빠르게 필요한 내용을 이해할 수 있는 참고 자료는 Google에서 "tkinter multiple buttons"로 검색했을 때 찾아진 아래 링크의 글이었습니다.

[Playing with buttons The first thing to do is to import the Tkinter module. Generally this is done by importing the module into our local namespace so that we can use the classes and constants by their names (like Label, Button, TOP) instead of having to constantly qualify everything (like Tkinter.L...](https://www.openbookproject.net/py4fun/gui/tkPhone.html)

[www.openbookproject.net](https://www.openbookproject.net/py4fun/gui/tkPhone.html)

특히, 제일 마지막 예시로 나온 소스 코드문은 제가 구현하고자 하는 방식, 기능이 거의 다 들어있어서 참고하기 매우 좋았 습니다.

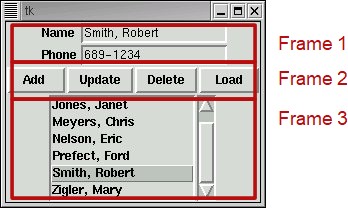
[# t k P h o n e . p y # from Tkinter import \* from phones import \* def whichSelected () : print "At %s of %d"](https://www.openbookproject.net/py4fun/gui/tkPhone.py)

[% (select.curselection(), len(phonelist)) return int(select.curselection()[0]) def addEntry () : phonelist.append ([nameVar.get(), phoneVar.get()]) setSelect () def u...](https://www.openbookproject.net/py4fun/gui/tkPhone.py)

[www.openbookproject.net](https://www.openbookproject.net/py4fun/gui/tkPhone.py)

Python에서 GUI를 구성할 때는 모든 내용을 마우스 조작 없이 코딩으로 명령을 해야 하기 때문에 실제 작업을 하기 전에 는 종이에라도 간단하게 그려보고 실질적인 코딩을 시작하는 것을 추천합니다.

위 예시 사이트에서 구현한 GUI 또한 결과물을 잘 보면



위의 이미지에 표시한것처럼 3개의 영역 Frame1, Frame2, Frame3의 영역으로 나누어 진 것을 볼 수 있습니다. Frame 1의 경우에는 Lable - Text Box 형태로 구성된 것을 볼 수 있고,

Frame 2의 경우에는 4개의 버튼이 일렬로 정렬된 것을 볼 수 있습니다. Frame 3은 1개의 List Box가 1개의 Scrollbar와 연결된 것을 볼 수 있습니다.

소스코드 중 제게 필요한 부분만 발췌를 하면 아래와 같습니다. 데이터를 가져오고 연결하는 부분은 지금 당장 필요하지는 않으므로, GUI 화면을 구성하는, 제게 의미 있는 부분을 하나하나 살펴 보겠습니다.

# t k P h o n e . p y # from Tkinter import \* from phones import \* def makeWindow () : win = Tk() frame1 = Frame(win) frame1.pack() Label(frame1, text="Name").grid(row=0, column=0, sticky=W) nameVar = StringVar() name = Entry(frame1, textvariable=nameVar) name.grid(row=0, column=1, sticky=W) Label(frame1, text="Phone").grid(row=1, column=0, sticky=W) phoneVar= StringVar() phone= Entry(frame1, textvariable=phoneVar) phone.grid(row=1, column=1, sticky=W) frame2 = Frame(win) # Row of buttons frame2.pack() b1 = Button(frame2,text=" Add ",command=addEntry) b2 = Button(frame2,text="Update",command=updateEntry) b3 = Button(frame2,text="Delete",command=deleteEntry) b4 = Button(frame2,text=" Load ",command=loadEntry) b1.pack(side=LEFT); b2.pack(side=LEFT) b3.pack(side=LEFT); b4.pack(side=LEFT) frame3 = Frame(win) # select of names frame3.pack() scroll = Scrollbar(frame3, orient=VERTICAL) select = Listbox(frame3, yscrollcommand=scroll.set, height=6) scroll.config (command=select.yview) scroll.pack(side=RIGHT, fill=Y) select.pack(side=LEFT, fill=BOTH, expand=1) return win win = makeWindow() win.mainloop()

한부분 한부분씩 살펴 보겠습니다.

# t k P h o n e . p y # from Tkinter import \* def makeWindow () : win = Tk() win = makeWindow() win.mainloop()

실행이 되면, "Tkinter" 라이브러리를 먼저 불러오고, makeWindow라는 함수를 정의해 줍니다.

"win"이라는 이름으로 Tkinter의 Tk 개체로 선언을 해주고, 이를 mainloop라는 명령어로 계속 떠있게 만들어 주는것이 기본 틀입니다. 사실 여기까지 해서 실행하면 제대로 돌아가지는 않습니다. "win"안에 들어갈 내용물을 아무것도 넣지 않 았기 때문입니다.

한가지 라이브러리를 불러오는 구문에서 살펴볼것이 import \* 인데요. 이와 같은 방식으로 선언이 되어 있으면, "Tkinter"로 시작하는 라이브러리 내부 함수를 앞에 "Tkinter"를 사용하는 개체라는 인식을 시켜줄 필요 없이 바로 그냥 인식한다는 것을 의미합니다.

위 예문에서 Tk() 함수가 대표적인데, 만약 다른 라이브러리를 불러오는것 처럼 사용하게 된다면, 아래와 같이 정리하게 되는 개념이라고 보시면 좋겠습니다.

import Tkinter win = Tkinter.Tk()

이제 그 안에 들어갈 내용물을 다룬 코드를 살펴 보겠습니다. 먼저 Frame 1에 해당하는 영역을 보겠습니다.

frame1 = Frame(win) frame1.pack() Label(frame1, text="Name").grid(row=0, column=0, sticky=W) nameVar

= StringVar() name = Entry(frame1, textvariable=nameVar) name.grid(row=0, column=1, sticky=W) Label(frame1, text="Phone").grid(row=1, column=0, sticky=W) phoneVar= StringVar() phone= Entry(frame1, textvariable=phoneVar) phone.grid(row=1, column=1, sticky=W)

win 안에 있는 Frame 개체를 선언해주고, 이를 pack() 해줬습니다.

이 pack() 이 들어가야 비로소 앞서 선언한 win에서의 내용물이 들어가고 창이 뜨게 됩니다.

위 소스에서 보면 pack()을 먼저 해 준 다음에, 안에 들어갈 내용물인 Label과 Entry를 입력한 것을 볼 수 있습니다. 입력된 개체들의 경우

1. 어느 개체 안에 삽입이 될 것인지?

* 이 소스에서는 frame1에 삽입할 것임

1. 어떤 text값을 가질 것인지?

* 이 소스에서는 text 또는 textvariable 형태로 입력을 밥을 것임

에 대한 내용이 잘 명기되어 있습니다.

그리고, 개체를 만든 다음 .grid로 표 형태로 나뉘어진 구역 안에, 원하는 위치에 들어가게 row와 column 값을 설정하는 구조로 되어 있습니다.

소스에서 보면 2개의 label은 각각 row 0, 1 / column 0 으로 설정하여 첫번째 열의 1,2행에 들어가게 했고, Entry는 row 0,1 / column 1으로 설정하여 두번째 열의 1,2행에 들어가게 했습니다.

이와 같은 구조를 이해하고, Frame 2를 살펴 보겠습니다.

frame2 = Frame(win) # Row of buttons frame2.pack() b1 = Button(frame2,text=" Add ",command=addEntry) b2 = Button(frame2,text="Update",command=updateEntry) b3 =

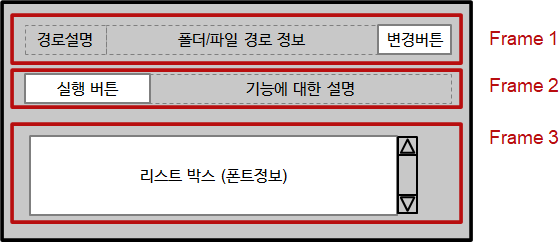
Button(frame2,text="Delete",command=deleteEntry) b4 = Button(frame2,text=" Load ",command=loadEntry) b1.pack(side=LEFT); b2.pack(side=LEFT) b3.pack(side=LEFT); b4.pack(side=LEFT) 기본적인 구조는 다르지 않은데, 각각의 버튼에, 명령어를 붙였다는점이 조금 다릅니다. command라는 이름으로 들어가 있으며, 그 뒤에 들어간 값은 소스 내에서 정의된 다른 함수들을 실행하게 되어 있습니다.

그리고, 이번에는 grid를 사용하지 않고, pack이라는 명령어로 단추를 배치한 것을 볼 수 있습니다. b1부터 순차적으로 배치해서 맨 왼쪽에서 부터 차래로 Add, Update, Delete, Load 의 단추가 들어가게 구성한 것을 볼 수 있습니다.

Frame 3을 살펴 보겠습니다.

frame3 = Frame(win) # select of names frame3.pack() scroll = Scrollbar(frame3, orient=VERTICAL) select = Listbox(frame3, yscrollcommand=scroll.set, height=6) scroll.config (command=select.yview) scroll.pack(side=RIGHT, fill=Y) select.pack(side=LEFT, fill=BOTH, expand=1)

Frame 3에서는 Scrollbar와 Listbox를 만들어 두고, 두 개체를 서로 y축 방향이 동기화 되게 연결지어준것을 볼 수 있습 니다. 그리고, 이번에도 Pack의 형태로 각각 오른쪽과 왼쪽에 배치한 것을 볼 수 있습니다.

이를 참고 삼아 우리가 만들고자 하는 화면을 구상해 보겠습니다.

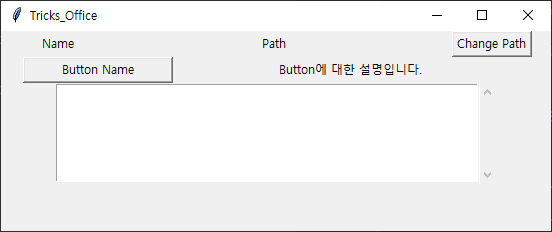
구성하고자 하는 화면은 위와 같습니다. 저 역시 3개의 Frame으로 구성되어 있고, 각각의 영역에는 라벨, 버튼, 리스트 박 스가 들어가 있습니다.

이를 소스로 정리하면 다음과 같이 정리할 수 있습니다.

from tkinter import \* win = Tk() win.geometry("550x200") win.title('Tricks\_Office') frame1 = Frame(win) frame1.pack() lbName = Label(frame1, text="Name",width = 10).grid(row=0, column=0, sticky=W) lbPath = Label(frame1, text="Path",width = 50).grid(row=0, column=1, sticky=W) btnPath

=Button(frame1,text="Change Path",width = 10).grid(row=0, column=2, sticky=W) frame2 = Frame(win) # Row of buttons frame2.pack() btnAction = Button(frame2,text="Button Name",width = 20).grid(row=0, column=0, sticky=W) lbAction = Label(frame2, text="Button에 대한 설명입니다.",width = 50).grid(row=0, column=1, sticky=W) frame3 = Frame(win) # select of names frame3.pack() scroll = Scrollbar(frame3,

orient=VERTICAL) select = Listbox(frame3, yscrollcommand=scroll.set, height=6,width = 60) scroll.config (command=select.yview) scroll.pack(side=RIGHT, fill=Y) select.pack(side=LEFT, fill=BOTH, expand=1) win.mainloop()

이를 실행하면 다음과 같은 GUI가 만들어 집니다.

만들어진 파일은 다음과 같습니다. **첨부파일**

Tricks\_Office.py [파일 다운로드](http://blogattach.naver.net/16830ab9adf3f22e02e087b58968126ecd9c6283f9/20210531_127_blogfile/550sn_1622396789463_m59102_py/%2FTricks_Office.py)

이와 같은 작업을 반복하여 최종적으로 만들어진 실행 파일은 다음의 링크에서 다운로드 받으실 수 있습니다.

### [파이썬 나만의 RPA 툴 제작 프로젝트 (Tricks\_Office)](https://tricks-office.com/python-tricks_office/)

[파이썬으로 나만의 RPA 툴을 직접 만드는 장기 프로젝트, 프로젝트명 “Tricks\_Office” 를 진행하고자 합니다. 구현을 계 획하고 있는 기능은 다음과 같습니다. 여러 PPT 파일의 여러 단어(Excel 형태로 보관된 Dictionary 파일 기준)를 원하는 단어로 치환하기 (구현)양식이 맞춰진 엑셀 파일을 하나의 파일로 취…](https://tricks-office.com/python-tricks_office/)

[tricks-office.com](https://tricks-office.com/python-tricks_office/)

다음 글과 영상에서는 이렇게 만들어진 GUI를 지금까지 만들어 두었던 기능과 연결하는 작업을 해 보겠습니다.

# 쓰기 편하게 다시 태어나는 파이썬2(GUI에 필요한 정보 입력하기)

## Python

2021/06/24 06:50 <http://blog.naver.com/550sn/222382647160>



지난번 글에서 tkinter의 툴들을 이용하여 GUI를 꾸미는 개략적인 방법을 알아봤습니다.

### [쓰기 편하게 다시 태어나는 파이썬 (GUI, Tkinter)](https://blog.naver.com/550sn/222375152366)

[지금까지 파이썬을 활용하여 많은 유용한 기능을 구현했습니다. 기능 기능 자체는 더할나위 없는데, 사실 ... blog.naver.com](https://blog.naver.com/550sn/222375152366)

오늘은 이를 기본으로 하여

1. 실제 사용할 버튼을 만들어 추가하고
2. 버튼에 기능을 추가하고 (폴더/파일 경로 가져오기)
3. Master Data를 만들어 연결하고
4. 폰트 리스트를 가져오는 작업을 해 보도록 하겠습니다.

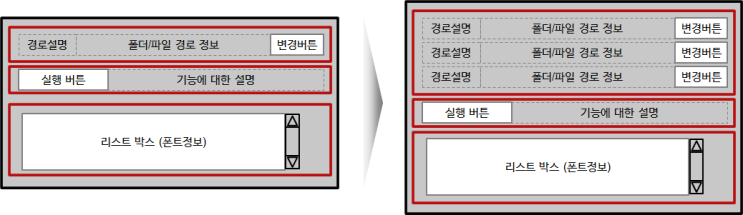
이전 글에서 만들었던 소스코드에서 부터 이어서 작업을 시작하겠습니다.

from tkinter import \* win = Tk() win.geometry("550x200") win.title('Tricks\_Office') frame1 = Frame(win) frame1.pack() lbName = Label(frame1, text="Name",width = 10).grid(row=0, column=0, sticky=W) lbPath = Label(frame1, text="Path",width = 50).grid(row=0, column=1, sticky=W) btnPath

=Button(frame1,text="Change Path",width = 10).grid(row=0, column=2, sticky=W) frame2 = Frame(win) # Row of buttons frame2.pack() btnAction = Button(frame2,text="Button Name",width = 20).grid(row=0, column=0, sticky=W) lbAction = Label(frame2, text="Button에 대한 설명입니다.",width = 50).grid(row=0, column=1, sticky=W) frame3 = Frame(win) # select of names frame3.pack() scroll = Scrollbar(frame3, orient=VERTICAL) select = Listbox(frame3, yscrollcommand=scroll.set, height=6,width = 60) scroll.config (command=select.yview) scroll.pack(side=RIGHT, fill=Y) select.pack(side=LEFT, fill=BOTH, expand=1) win.mainloop()

1. 실제 사용할 버튼 추가

지난번에는 기본적인 틀만 잡았고, 해당하는 틀의 형식에 맞게 추가하는 작업을 해 보도록 하겠습니다.



기존에 잡았던 틀에서 기능이 추가되더라도 반복적으로 필요한 정보에 대한 내용을 첫번째 프레임에 폴더 경로의 형태로 가져오는 부분을 만들었습니다. 그런데, 이부분은 구조가 동일한 형태에서 내용이 반복적으로 만들어 집니다. 컴퓨터에게 일을 시킬때 가장 큰 이익을 얻을 수 있는 부분이 이처럼 어떤 규칙아래에서 반복되는 일이 있을때 이를 반복 문으로 구현하는 것일텐데, 이와 같이 필요한 정보를 얻는것 또한 반복문을 사용하지 않을 이유가 없습니다.

Google에서 "python tkinter multiple buttons"라는 키워드로 검색하여 다음의 링크를 찾아가면 이번에도 도움을 받을 수 있는 참고자료가 있습니다.



### [Tkinter. Create multiple buttons with "different" command function](https://stackoverflow.com/questions/27198287/tkinter-create-multiple-buttons-with-different-command-function)

[first of all, sorry for the title, I couldn't find a better one. The following code is a minimalized version of a problem I have in my Python program (I am a newbie btw.). def onClick(i): pri...](https://stackoverflow.com/questions/27198287/tkinter-create-multiple-buttons-with-different-command-function)

[stackoverflow.com](https://stackoverflow.com/questions/27198287/tkinter-create-multiple-buttons-with-different-command-function)

여기에 있는 참고할만한 소스를 가져오면 다음과 같습니다.

※ 이 글에서 위쪽에 있는 코드는 제대로 실행되지 않는 질문에 대한 코드이고,

아래에 있는 Answer에 대한 코드가 질문에 대한 해답으로, 보완할 부분을 보완한 코드입니다.

def onClick(i): print "This is Button: " + str(i) return def start(): buttons = [] win = Tkinter.Tk() for i in range(5): b = Tkinter.Button(win, height=10, width=100, command=lambda i=i: onClick(i)) b.pack() buttons.append(b)

여기에 있는 반복문에 해당하는 문구를 끌어오면서 내게 필요한 정보를 연결시키는 구조를 만들면 되겠습니다. 내게 필요한 구조는 다음의 3가지로 정리할 수 있습니다.

1. 작업할 파일이 모여있는 폴더 주소에 대한 정보
2. 작업이 완료된 파일을 저장할 폴더 주소에 대한 정보
3. Dictionary 파일 경로에 대한 정보

우리가 만들었던 frame 1에 대한 내용을 다음과 같이 바꿀 수 있습니다.

def onClick(i): print "This is Button: " + str(i) return frame1 = Frame(win) frame1.pack() lbName = [] lbPath = [] btnPath =[] for i in df.index: lbName.append(Label(frame1, text=df.Item[i], width=10)) lbName[i].grid(row=i, column=0, sticky=W) lbPath.append(Label(frame1, text=df.Detail[i], width=50)) lbPath[i].grid(row=i, column=1, sticky=W) btnPath.append(Button(frame1, text="Change Path", width=10,command=lambda i=i: onClick(i))) btnPath[i].grid(row=i, column=2, sticky=W)

1. 버튼에 기능을 추가 (폴더/파일 경로 가져오기) 이제, 버튼에 대한 동작을 바꿔 보겠습니다.

버튼을 누르면 폴더 또는 파일을 선택하고 그에 대한 경로를 끌어오는 부분을 구현했으면 좋겠습니다. 이 또한 구글에서 "python tkinter select folder"로 검색을 합니다.



### [How to select a directory and store the location using tkinter in Python](https://stackoverflow.com/questions/11295917/how-to-select-a-directory-and-store-the-location-using-tkinter-in-python)

[I am creating a GUI with a browse button which I only want to return the path. I've been looking at solutions using code like below. Tkinter.Button(subframe, text = "Browse", command = self.](https://stackoverflow.com/questions/11295917/how-to-select-a-directory-and-store-the-location-using-tkinter-in-python)

[stackoverflow.com](https://stackoverflow.com/questions/11295917/how-to-select-a-directory-and-store-the-location-using-tkinter-in-python)

마찬가지로 내게 필요한 소스를 가져올 수 있습니다.

from tkinter import filedialog from tkinter import \* root = Tk() root.withdraw() folder\_selected = filedialog.askdirectory()

우리가 필요한 기능에 대한 소스를 정리하다보니, 폴더/파일을 가져오는 구역인 Frame1이 사실은 2개의 구역으로 나눠 야 함을 알 수 있습니다. 폴더를 다루는 영역과 파일을 다루는 영역으로요.

frame 1 다음에 frame1h 라는 이름의 프레임을 추가하고, 정리하겠습니다. 이를 소스에 입력하면 다음과 같이 정리할 수 있습니다.

from tkinter import filedialog def onClick(i): folder\_selected = filedialog.askdirectory() lbPath[i].config(text=folder\_selected) return def onClickf(i): folder\_selected = filedialog.askopenfile() lbPathf[i].config(text=folder\_selected.name) return frame1 = Frame(win) frame1.pack() lbName = [] lbPath = [] btnPath =[] for i in df.index: lbName.append(Label(frame1, text=df.Item[i], width=10)) lbName[i].grid(row=i, column=0, sticky=W) lbPath.append(Label(frame1, text=df.Detail[i], width=50)) lbPath[i].grid(row=i, column=1, sticky=W) btnPath.append(Button(frame1, text="Change Path", width=10,command=lambda i=i: onClick(i))) btnPath[i].grid(row=i, column=2, sticky=W) frame1h = Frame(win) frame1h.pack() lbNamef = [] lbPathf = [] btnPathf =[] for i in dff.index:

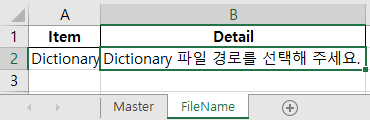
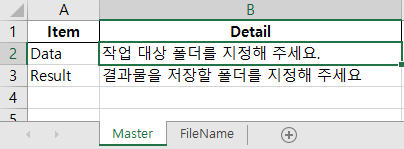
lbNamef.append(Label(frame1h, text=dff.Item[i], width=10)) lbNamef[i].grid(row=i, column=0, sticky=W) lbPathf.append(Label(frame1h, text=dff.Detail[i], width=50)) lbPathf[i].grid(row=i, column=1, sticky=W) btnPathf.append(Button(frame1h, text="Change Path", width=10,command=lambda i=i: onClickf(i))) btnPathf[i].grid(row=i, column=2, sticky=W)

1. Master Data 만들기

위와같이 정리하면 내게 필요한 Folder 정보를 탐색기 형태의 틀에서 가져올 수 있습니다.

그런데, 이와 같이 프로그램을 실행하면 매번 프로그램을 실행할 때 마다 작업 폴더 경로를 새롭게 입력해 줘야하는 번거 로움이 있습니다. 보통 한번 정해진 폴더는 잘 안바꾸는 경향이 있을텐데 말이죠...

그래서 이부분은 파이썬 소스 내부가 아니라, 외부에 별도의 파일 형태로 보관하고 있다가 필요할때 가져오는 형식으로 정 리해 두면 조금더 편할듯 합니다.

엑셀 파일로 다음과 같이 2개의 Sheet로 만들어진 파일을 만들어 둡니다.

이제 앞서 만들었던 버튼 로직에서 데이터 부분을 이 data frame 영역으로 바꿔 줍니다. 그리고, 값을 바꿨을때는 엑셀에 저장하는 로직을 추가해 둡니다.

그런데, 엑셀에 추가할때 한번에 2개의 Sheet를 저장해야 합니다.

이 또한 구글에서 "pandas write excel multiple sheets"과 같이 검색하여 필요한 소스를 찾아봅니다.

[**Example: Pandas Excel with multiple dataframes — XlsxWriter Documentation** ############################################################################## # #](https://xlsxwriter.readthedocs.io/example_pandas_multiple.html)

[An example of writing multiple dataframes to worksheets using Pandas and # XlsxWriter. # # Copyright 2013-2021, John McNamara, jmcnamara@cpan.org # import pandas as pd # Create some Pandas dataframes from some data. d...](https://xlsxwriter.readthedocs.io/example_pandas_multiple.html)

[xlsxwriter.readthedocs.io](https://xlsxwriter.readthedocs.io/example_pandas_multiple.html)

이번에도 힌트를 얻을 소스를 찾았습니다.

# Create a Pandas Excel writer using XlsxWriter as the engine. writer = pd.ExcelWriter('pandas\_multiple.xlsx', engine='xlsxwriter') # Write each dataframe to a different worksheet. df1.to\_excel(writer, sheet\_name='Sheet1') df2.to\_excel(writer, sheet\_name='Sheet2') df3.to\_excel(writer,

sheet\_name='Sheet3') # Close the Pandas Excel writer and output the Excel file. writer.save()

이를 토대로 writer라는 ExcelWriter 개체를 만들어두고, 여기서 Sheet별로 저장을 한다음에 저장을 해 주면 된다는 것을 알 수 있습니다.

여기까지 반영하면 소스가 다음과 같이 추가 보완됩니다.

import pandas as pd def onClick(i): folder\_selected = filedialog.askdirectory() lbPath[i].config(text=folder\_selected) df.Detail[i] = folder\_selected writer = pd.ExcelWriter('Master/ Master.xlsx', engine='xlsxwriter') df.to\_excel(writer, sheet\_name='Master', index=False) dff.to\_excel(writer, sheet\_name='FileName',index=False) writer.save() return def onClickf(i): folder\_selected = filedialog.askopenfile() lbPathf[i].config(text=folder\_selected.name) dff.Detail[i] = folder\_selected.name writer = pd.ExcelWriter('Master/Master.xlsx', engine='xlsxwriter') df.to\_excel(writer, sheet\_name='Master', index=False) dff.to\_excel(writer, sheet\_name='FileName',index=False) writer.save() return df= pd.read\_excel("Master/Master.xlsx",sheet\_name="Master") dff= pd.read\_excel("Master/ Master.xlsx",sheet\_name="FileName")

1. 폰트 리스트를 가져오기

이번에는 세번째 프레임에 있는 폰트 리스트 상자에 현재 내가 사용하는 폰트 리스트를 가져오는 작업을 해 보도록 하겠습 니다.

이번에도 구글 검색을 하겠습니다. "python font list"로 검색을 하면 아래의 자료를 찾아 볼 수 있습니다.



### [List available font families in `tkinter`](https://stackoverflow.com/questions/39614027/list-available-font-families-in-tkinter)

[In many tkinter examples available out there, you may see things like: canvas.create\_text(x, y, font=('Helvetica', 12), text='foo') However, this may not work when run in your computer (the result stackoverflow.com](https://stackoverflow.com/questions/39614027/list-available-font-families-in-tkinter)

이번에도 훌륭한 힌트를 얻을 수 있습니다.

from tkinter import \* from tkinter import font root = Tk() root.title('Font Families') fonts=list(font.families()) fonts.sort()

이렇게 가져온 font list를 Frame3에 있는 리스트에 입력해 줘야 하는데, 이 부분은 이미 지난번 글에서 키워드 "tkinter multiple buttons"로 찾았던 참고 문서에서 힌트를 얻을 수 있습니다.

[# t k P h o n e . p y # from Tkinter import \* from phones import \* def whichSelected () : print "At %s of %d"](https://www.openbookproject.net/py4fun/gui/tkPhone.py)

[% (select.curselection(), len(phonelist)) return int(select.curselection()[0]) def addEntry () : phonelist.append ([nameVar.get(), phoneVar.get()]) setSelect () def u...](https://www.openbookproject.net/py4fun/gui/tkPhone.py)

[www.openbookproject.net](https://www.openbookproject.net/py4fun/gui/tkPhone.py)

for name,phone in phonelist : select.insert (END, name) 이를 소스에 추가하면 다음과 같이 정리 됩니다.

fonts=list(font.families()) select.delete(0,END) for item in fonts : select.insert (END, item) 이제 부분 부분 수정한 내용을 모두 모아서 하나의 파일로 만들면 다음과 같이 구성할 수 있습니다.

from tkinter import \* from tkinter import filedialog from tkinter import font import pandas as pd def onClick(i): folder\_selected = filedialog.askdirectory() lbPath[i].config(text=folder\_selected) df.Detail[i] = folder\_selected writer = pd.ExcelWriter('Master/Master.xlsx', engine='xlsxwriter') df.to\_excel(writer, sheet\_name='Master', index=False) dff.to\_excel(writer, sheet\_name='FileName',index=False) writer.save() return def onClickf(i): folder\_selected = filedialog.askopenfile() lbPathf[i].config(text=folder\_selected.name) dff.Detail[i] = folder\_selected.name writer = pd.ExcelWriter('Master/Master.xlsx', engine='xlsxwriter') df.to\_excel(writer, sheet\_name='Master', index=False) dff.to\_excel(writer, sheet\_name='FileName',index=False) writer.save() return win = Tk() win.geometry("550x250") win.title('Tricks\_Office') df= pd.read\_excel("Master/Master.xlsx",sheet\_name="Master") dff= pd.read\_excel("Master/Master.xlsx",sheet\_name="FileName") fonts=list(font.families()) frame1 = Frame(win) frame1.pack() lbName = [] lbPath = [] btnPath =[] for i in df.index: lbName.append(Label(frame1, text=df.Item[i], width=10)) lbName[i].grid(row=i, column=0, sticky=W) lbPath.append(Label(frame1, text=df.Detail[i], width=50)) lbPath[i].grid(row=i, column=1, sticky=W) btnPath.append(Button(frame1, text="Change Path", width=10,command=lambda i=i: onClick(i))) btnPath[i].grid(row=i, column=2, sticky=W) frame1h = Frame(win) frame1h.pack() lbNamef = [] lbPathf = [] btnPathf =[] for i in dff.index: lbNamef.append(Label(frame1h, text=dff.Item[i], width=10)) lbNamef[i].grid(row=i, column=0, sticky=W) lbPathf.append(Label(frame1h, text=dff.Detail[i], width=50)) lbPathf[i].grid(row=i, column=1, sticky=W) btnPathf.append(Button(frame1h, text="Change Path", width=10,command=lambda i=i: onClickf(i))) btnPathf[i].grid(row=i, column=2, sticky=W) frame2 = Frame(win) # Row of buttons frame2.pack() # Keywork Change 버튼 btnPPT\_KC = Button(frame2,text="Button",width = 20) btnPPT\_KC.grid(row=0, column=0, sticky=W) lbPPT\_KC = Label(frame2, text="Button 설명",width = 50) lbPPT\_KC.grid(row=0, column=1, sticky=W) frame3 = Frame(win) # select of names frame3.pack() scroll = Scrollbar(frame3, orient=VERTICAL) select = Listbox(frame3, yscrollcommand=scroll.set, height=6,width = 60) scroll.config (command=select.yview) scroll.pack(side=RIGHT, fill=Y) select.pack(side=LEFT, fill=BOTH, expand=1) select.delete(0,END) for item in fonts : select.insert (END, item) win.mainloop()

여기까지 구성된 Python 파일은 다음과 같습니다. **첨부파일**

Tricks\_Office\_Pre.py [파일 다운로드](http://blogattach.naver.net/47d25bebfea3a37f53b1d6e4d839433f9ccd37d2e5/20210604_244_blogfile/550sn_1622784616325_Q9j2s1_py/%2FTricks_Office_Pre.py)

다음 글에서는 이렇게 만들어진 Python GUI에 실제 이전에 만들었던 Python 소스들을 연결하고, EXE 파일을 다루는 과 정을 알아보도록 하겠습니다.

최종 결과물 파일은 아래의 링크에서 다운로드 받으실 수 있습니다.

### [파이썬 나만의 RPA 툴 제작 프로젝트 (Tricks\_Office)](https://tricks-office.com/python-tricks_office/)

[파이썬으로 나만의 RPA 툴을 직접 만드는 장기 프로젝트, 프로젝트명 “Tricks\_Office” 를 진행하고자 합니다. 구현을 계 획하고 있는 기능은 다음과 같습니다. 여러 PPT 파일의 여러 단어(Excel 형태로 보관된 Dictionary 파일 기준)를 원하는 단어로 치환하기 (구현)양식이 맞춰진 엑셀 파일을 하나의 파일로 취…](https://tricks-office.com/python-tricks_office/)

[tricks-office.com](https://tricks-office.com/python-tricks_office/)

# 쓰기 편하게 다시 태어나는 파이썬3 (프로그램 연결, EXE 파일 만들기)

## Python

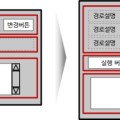
2021/06/28 06:50 <http://blog.naver.com/550sn/222382776900>



앞서 만든 프로그램 GUI를 기반으로,

1. 실질적인 동작이 가능하게 실행 버튼을 만들고,
2. EXE 파일 형태의 실행 파일을 만들어 보도록 하겠습니다.

※ iOS 사용자 분들에게는 죄송합니다. 제 컴퓨터 환경이 Win 밖에 없어서, iOS에 대한 배려가 없네요... ㅠ.ㅠ 기본적으로 제가 솔루션을 찾아가는 방법이 구글 검색에 의존하는 방식이니,

제 글과 영상에서 어떤 식으로 검색하면 좋을지에 대한 힌트는 얻으실수 있을듯 합니다.

### [쓰기 편하게 다시 태어나는 파이썬2(GUI에 필요한 정보 입력하기)](https://blog.naver.com/550sn/222382647160)

[지난번 글에서 tkinter의 툴들을 이용하여 GUI를 꾸미는 개략적인 방법을 알아봤습니다. 오늘은 이를 기...](https://blog.naver.com/550sn/222382647160)

[blog.naver.com](https://blog.naver.com/550sn/222382647160)

지금까지 파이썬 코드로 구현한 파일을 가져오고, 이를 연결하여 실행이 가능하게 연결짓겠습니다.

from tkinter import \* from tkinter import filedialog from tkinter import font import pandas as pd import os from tkinter import messagebox def onClick(i): folder\_selected = filedialog.askdirectory() lbPath[i].config(text=folder\_selected) df.Detail[i] = folder\_selected writer = pd.ExcelWriter('Master/ Master.xlsx', engine='xlsxwriter') df.to\_excel(writer, sheet\_name='Master', index=False) dff.to\_excel(writer, sheet\_name='FileName',index=False) writer.save() return def onClickf(i): folder\_selected = filedialog.askopenfile() lbPathf[i].config(text=folder\_selected.name) dff.Detail[i] = folder\_selected.name writer = pd.ExcelWriter('Master/Master.xlsx', engine='xlsxwriter') df.to\_excel(writer, sheet\_name='Master', index=False) dff.to\_excel(writer, sheet\_name='FileName',index=False) writer.save() return win = Tk() win.geometry("550x250") win.title('Tricks\_Office') df= pd.read\_excel("Master/ Master.xlsx",sheet\_name="Master") dff= pd.read\_excel("Master/Master.xlsx",sheet\_name="FileName") fonts=list(font.families()) frame1 = Frame(win) frame1.pack() lbName = [] lbPath = [] btnPath =[] for i in df.index: lbName.append(Label(frame1, text=df.Item[i], width=10)) lbName[i].grid(row=i, column=0, sticky=W) lbPath.append(Label(frame1, text=df.Detail[i], width=50)) lbPath[i].grid(row=i, column=1, sticky=W) btnPath.append(Button(frame1, text="Change Path", width=10,command=lambda i=i: onClick(i))) btnPath[i].grid(row=i, column=2, sticky=W) frame1h = Frame(win) frame1h.pack() lbNamef = [] lbPathf = [] btnPathf =[] for i in dff.index: lbNamef.append(Label(frame1h, text=dff.Item[i], width=10)) lbNamef[i].grid(row=i, column=0, sticky=W) lbPathf.append(Label(frame1h, text=dff.Detail[i], width=50)) lbPathf[i].grid(row=i, column=1, sticky=W) btnPathf.append(Button(frame1h, text="Change Path", width=10,command=lambda i=i: onClickf(i))) btnPathf[i].grid(row=i, column=2, sticky=W) frame2 =

Frame(win) # Row of buttons frame2.pack() # Keywork Change 버튼 btnPPT\_KC = Button(frame2,text="Button",width = 20) btnPPT\_KC.grid(row=0, column=0, sticky=W) lbPPT\_KC = Label(frame2, text="Button 설명",width = 50) lbPPT\_KC.grid(row=0, column=1, sticky=W) frame3 = Frame(win) # select of names frame3.pack() scroll = Scrollbar(frame3, orient=VERTICAL) select = Listbox(frame3, yscrollcommand=scroll.set, height=6,width = 60) scroll.config (command=select.yview) scroll.pack(side=RIGHT, fill=Y) select.pack(side=LEFT, fill=BOTH, expand=1) select.delete(0,END) for item in fonts : select.insert (END, item) win.mainloop()

1. 실행버튼 만들고 연결하기

실행 버튼에서 해줄 일은 두가지 입니다.

1. 폴더에서 파일 이름별로 쪼개서
2. 이전에 만들었던 파이썬 파일로 명령 보내기

이를 위해서 먼저 폴더에서 파일이름 리스트를 가져오는 내용을 살펴 보겠습니다. 구글에서 "python filename from directory"로 검색을 합니다.

이번에도 사용 가능한 힌트를 얻을 수 있습니다. import os arr = os.listdir() print(arr)

이를 소스에 추가하고, 반복문을 만들겠습니다.

import os files = os.listdir(불러올 파일경로) for file in files: 반복할 동작

버튼을 눌렀을때 동작을 하게 만들어야 하기 때문에 소스 앞부분에 def를 통해 함수를 만들어 둡니다. 그리고, 이렇게 만들어둔 함수는 버튼에 연결시켜 둡니다.

버튼을 눌렀을때 하는 동작은 폴더에서의 파일 리스트를 가져오고, 파일 리스트에 대하여 반복문을 실행하면서 예전에 만 들었던 소스코드에서 기능을 사용하는 것입니다.

그와 같이 동작을하게 하기 위해서 기존에 만들었던 소스 코드의 내부 기능들도 함수화 해 둬야 합니다.

이전에 만들었던 Dictionary 파일을 읽어 들인 다음, PPT에서 keyword를 바꾸는 파이썬 파일을 우선 복사해와서 지금 작 업하고 있는 프로젝트 폴더에 저장해 둡니다. (파일 이름은 PPT\_Study.py였습니다.)

기존 소스는 아래와 같습니다.

from pptx import Presentation from pptx.enum.shapes import MSO\_SHAPE\_TYPE import openpyxl def R\_TXT (shp,B\_W,A\_W): for paragraph in shp.text\_frame.paragraphs: for run in paragraph.runs: cur\_text = run.text new\_text = cur\_text.replace(str(B\_W), str(A\_W)) run.text = new\_text wb = openpyxl.load\_workbook("Dictionary.xlsx") ws = wb.active before\_W = [] after\_W = [] for cell in ws.rows: before\_W.append(cell[0].value) after\_W.append(cell[1].value) prs = Presentation("Sample.pptx") for slide in prs.slides: for shape in slide.shapes: for i in range(1, len(after\_W)): if shape.has\_text\_frame: R\_TXT(shape,before\_W[i],after\_W[i]) if shape.has\_table: for row in shape.table.rows: for cell in row.cells: R\_TXT(cell,before\_W[i],after\_W[i]) if shape.shape\_type == MSO\_SHAPE\_TYPE.GROUP: for shp in shape.shapes: if shp.has\_text\_frame: R\_TXT(shp,before\_W[i],after\_W[i]) prs.save("Result.pptx")

그리고, 해당 파일의 기능을 def로 함수화 하여 만들고, 필요한 변수를 함수에서 받아올 수 있게 ()안에 정리해 둡니다. 변경한 소스는 아래와 같습니다.

from pptx import Presentation from pptx.enum.shapes import MSO\_SHAPE\_TYPE import openpyxl def R\_TXT (shp,B\_W,A\_W): for paragraph in shp.text\_frame.paragraphs: for run in paragraph.runs: cur\_text = run.text new\_text = cur\_text.replace(str(B\_W), str(A\_W)) run.text = new\_text def PPT\_KC\_Action(file,dpath,rpath,DFName): wb = openpyxl.load\_workbook(DFName) ws = wb.active before\_W = [] after\_W = [] for cell in ws.rows: before\_W.append(cell[0].value) after\_W.append(cell[1].value) prs = Presentation(dpath + "/" + file) for slide in prs.slides: for shape in slide.shapes: for i in range(1, len(after\_W)): if shape.has\_text\_frame: R\_TXT(shape,before\_W[i],after\_W[i]) if shape.has\_table: for row in shape.table.rows: for cell in row.cells: R\_TXT(cell,before\_W[i],after\_W[i]) if shape.shape\_type == MSO\_SHAPE\_TYPE.GROUP: for shp in shape.shapes: if shp.has\_text\_frame: R\_TXT(shp,before\_W[i],after\_W[i]) prs.save(rpath + "/" + file)

변경한 소스에서는 file / dpath / rpath / DFName라는 이름으로 변수를 받아오게 되어 있습니다.

1. file : 작업할 파일이름
2. dpath : 작업할 파일이 있는 폴더 경로
3. rpath : 작업 결과를 저장할 폴더 경로
4. DFName : Dictionary 파일이름(경로 포함)

이 부분을 버튼을 눌렀을때 일어나는 동작에 불러와서 사용하게 하면 됩니다.

같은 폴더에 실행하고 싶은 다른 파일의 함수가 있을 경우 사용법은 다음과 같습니다. import 파일이름 파일이름.함수이름(필요한변수들)

어딘가 친숙한 형태죠? 네, 맞습니다.

우리가 유용하게 사용하고 있는 수많은 라이브러리들 또한 이와 같은 형태로 누군가가 앞서서 끌어쓸 수 있게 만들어 놓았 기 때문에 우리는 import라는 명령어 하나 만으로 편리하게 필요한 기능을 사용할 수 있습니다. 이점은 늘 감사하게 생각 합니다.

이 부분을 추가로 정리하면 다음과 같습니다.

import PPT\_Study def PPT\_KC\_Action(): files = os.listdir(df.Detail[0]) for file in files: PPT\_Study.R\_KWord(file,df.Detail[0],df.Detail[1],dff.Detail[0])

그 외에 추가로 필요한 기능들도 같은 방식으로 정리하겠습니다. 정리된 Python 파일은 다음과 같습니다.

1. 실행 파일 만들기

완성된 파이썬 파일은 나 홀로 사용하기에는 아무 문제가 없지만, 파이썬을 잘 모르는 우리의 직장 동료들과 나눠쓰고 싶 을 경우 파이썬이 깔려있지 않더라도 실행 될 수 있게 실행 파일로 만들어 주는 과정이 필요합니다.

이 과정을 알아보도록 하겠습니다.

이번에도 구글 검색을 합니다. "python to exe file"



### [Create Executable from Python Script using Pyinstaller - Data to Fish](https://datatofish.com/executable-pyinstaller/)

[Create Executable from Python Script using Pyinstaller Python / April 9, 2020 Looking to create an executable from Python script using pyinstaller? If so, I’ll show you the full steps to accomplish this goal in Windows .](https://datatofish.com/executable-pyinstaller/)

[Steps to Create an Executable from Python Script using Pyinstaller Step 1: Add... datatofish.com](https://datatofish.com/executable-pyinstaller/)

이번에 찾은 실행 방법은 코드에서 실행하기 보다는 Terminal에서 실행해야 하는 것 같습니다. 먼저 필요한 라이브러리인 pyinstaller를 깔고,

pip install pyinstaller

Terminal에서 다음과 같이 입력하면 실행파일 형태로 만들어 줍니다. pyinstaller 파이썬파일이름.py

이렇게 하면 프로젝트가 있는 폴더의 dist 폴더에 파이썬 파일이름으로 된 폴더가 하나 만들어지고, 그 안에 실행파일을 포 함한 많은 파일이 구성되어 있으며, 그중 exe 파일도 포함되어 구성되는 것을 볼 수 있습니다.

이렇게 보니 무언가 너무 복잡해 보입니다. exe 파일 하나로만 구성하고 싶은데... 그러면 아래와 같이 옵션을 추가해 주면 하나의 파일로 만들어 줍니다. pytinstaller --onefile 파이썬파일이름.py

마찬가지로 dist 폴더 아래 해당 파이썬 파일 이름으로 된 실행파일이 하나 생성됩니다. 이제 실행을 해 보면... 안타깝게도 실행이 되지 않습니다.

이는 파이썬 코드에서 라이브러리나 파이썬 파일 방식이 아닌 파일을 끌어와서 사용하는 경우, 그 파일을 찾지 못해 발생 하는 현상으로 보입니다.

따라서 해당하는 Master file을 만들어진 실행 파일에서 인식할 수 있게 복사를 해와 주는 작업이 필요합니다.

현재 소스에서 Master파일은 파이썬 실행파일이 있는 폴더에 "Master"라는 폴더아래 "Master.xlsx"라는 파일이름으로

연계되어 있습니다.

그렇기 때문에 이 폴더 자체를 복사해 와서 실행파일이 만들어진 폴더에 붙여 넣으면 이후 정상적으로 동작하는 것을 볼 수 있습니다.

단, Master.xlsx라는 파일이 반드시 실행파일과 같이 다녀야 한다는 것이 항상 신경 써야 하는 점이겠네요...

3. 그외...

Python이 많은 라이브러리를 끌어와서 비교적 쉽고 빠르게 강력한 기능을 구현할 수 있는 강점이 있는 반면, 메모리 관리

/ 용량 관리에 최적화 되어 있는 프로그램은 아니라는 생각이 듭니다. (물론 제 공부가 부족해서 그럴 수도 있습니다. ) 굉장히 간단하고, 단순한 기능만 구현해도 기본적으로 파일 크기가 30메가를 넘어가는데, 이는 실행 파일 자체의 용량 보 다는 같이 다니는 라이브러리 파일들의 용량이 큰 것이 가장 큰 원인인것 같습니다.

이 크기를 줄이는 방법이 어렵네요..

만약, 이와 같은 방식으로 파이썬으로 만들어야 하는 실행 파일이 여러개 있다면, onefile 옵션으로 만들기 보다는 directory 형태로 만드는 것을 추천합니다. directory 형태로 만들경우 라이브러리 파일들을 모아 놓은 폴더에 작은 크기 의 exe 파일만 추가로 넣어도 사용하는 라이브러리가 동일하다면 실행이 가능합니다.

이렇게 만들어진 Python 파일은 아래와 같습니다.

from tkinter import \* from tkinter import filedialog from tkinter import font import pandas as pd import PPT\_Study import os from tkinter import messagebox def onClick(i): folder\_selected = filedialog.askdirectory() lbPath[i].config(text=folder\_selected) df.Detail[i] = folder\_selected writer = pd.ExcelWriter('Master/Master.xlsx', engine='xlsxwriter') df.to\_excel(writer, sheet\_name='Master', index=False) dff.to\_excel(writer, sheet\_name='FileName',index=False) writer.save() return def onClickf(i): folder\_selected = filedialog.askopenfile() lbPathf[i].config(text=folder\_selected.name) dff.Detail[i] = folder\_selected.name writer = pd.ExcelWriter('Master/Master.xlsx', engine='xlsxwriter') df.to\_excel(writer, sheet\_name='Master', index=False) dff.to\_excel(writer, sheet\_name='FileName',index=False) writer.save() return def PPT\_KC\_Action(): files = os.listdir(df.Detail[0]) for file in files: PPT\_Study.PPT\_KC(file,df.Detail[0],df.Detail[1],dff.Detail[0]) messagebox.showinfo(title="Finish", message="Job Finished!") return win = Tk() win.geometry("550x250") win.title('Tricks\_Office') df= pd.read\_excel("Master/Master.xlsx",sheet\_name="Master") dff= pd.read\_excel("Master/ Master.xlsx",sheet\_name="FileName") fonts=list(font.families()) frame1 = Frame(win) frame1.pack() lbName

= [] lbPath = [] btnPath =[] for i in df.index: lbName.append(Label(frame1, text=df.Item[i], width=10)) lbName[i].grid(row=i, column=0, sticky=W) lbPath.append(Label(frame1, text=df.Detail[i], width=50)) lbPath[i].grid(row=i, column=1, sticky=W) btnPath.append(Button(frame1, text="Change Path", width=10,command=lambda i=i: onClick(i))) btnPath[i].grid(row=i, column=2, sticky=W) frame1h = Frame(win) frame1h.pack() lbNamef = [] lbPathf = [] btnPathf =[] for i in dff.index:

lbNamef.append(Label(frame1h, text=dff.Item[i], width=10)) lbNamef[i].grid(row=i, column=0, sticky=W) lbPathf.append(Label(frame1h, text=dff.Detail[i], width=50)) lbPathf[i].grid(row=i, column=1, sticky=W) btnPathf.append(Button(frame1h, text="Change Path", width=10,command=lambda i=i: onClickf(i))) btnPathf[i].grid(row=i, column=2, sticky=W) frame2 = Frame(win) # Row of buttons frame2.pack() # Keywork Change 버튼 btnPPT\_KC = Button(frame2,text="Find and Replace keywords (PPT)",width = 20, command=lambda:PPT\_KC\_Action()) btnPPT\_KC.grid(row=0, column=0, sticky=W) lbPPT\_KC = Label(frame2, text="Dictionary 정보를 기반으로 하여, Keywords를 바꿔주는 기능입니다. \n" "Dictionary 파일 경 로가 정확하게 지정되어 있어야 합니다.",width = 50) lbPPT\_KC.grid(row=0, column=1, sticky=W) frame3 = Frame(win) # select of names frame3.pack() scroll = Scrollbar(frame3, orient=VERTICAL) select = Listbox(frame3, yscrollcommand=scroll.set, height=6,width = 60) scroll.config (command=select.yview) scroll.pack(side=RIGHT, fill=Y) select.pack(side=LEFT, fill=BOTH, expand=1) select.delete(0,END) for item in fonts : select.insert (END, item) win.mainloop()

**첨부파일** Tricks\_Office.py [파일 다운로드](http://blogattach.naver.net/e97cf54e506263d6ff197b4b7197e89231689723/20210604_227_blogfile/550sn_1622783191657_BgYawY_py/%2FTricks_Office.py)

이와 같은 요령으로 다른 파이썬 파일들도 연계하여 만든 파일은 아래와 같습니다. **첨부파일**

Tricks\_Office.zip [파일 다운로드](http://blogattach.naver.net/60f57cccdc8184587496f1c3ff1e6418bbea1ffdee/20210604_82_blogfile/550sn_1622783393618_6Gvuip_zip/%2FTricks_Office.zip)

상기 내용으로 연계하여 Zip 파일로 만들때는 창 Layout에 대한 부분을 조금 더 보기 좋게 다듬었습니다. 필요한 작업은 Google 검색으로 찾은 문서를 참고하여 했으며, 이때 검색어 및 참고 문헌은 아래와 같습니다.

"tkinter label border"



### [Border for tkinter Label](https://stackoverflow.com/questions/39416021/border-for-tkinter-label)

[Not really relevant but i'm building a calendar and I have a lot of Label widgets, and therefore it will look alot nicer if I had some borders for them! I have seen you can do this for other widgets](https://stackoverflow.com/questions/39416021/border-for-tkinter-label)

[stackoverflow.com](https://stackoverflow.com/questions/39416021/border-for-tkinter-label) "tkinter margin"



### [How to add a margin to a tkinter window?](https://stackoverflow.com/questions/3643235/how-to-add-a-margin-to-a-tkinter-window)

[So I have so far a simple python tkinter window and i'm adding text, buttons, etc. snippet: class Cfrm(Frame): def createWidgets(self): self.text = Text(self, width=50, height=10) ...](https://stackoverflow.com/questions/3643235/how-to-add-a-margin-to-a-tkinter-window)

[stackoverflow.com](https://stackoverflow.com/questions/3643235/how-to-add-a-margin-to-a-tkinter-window)

실행파일은 40MB 수준이어서 네이버 블로그에는 링크가 불가능한 관계로 개인적으로 운영하는 다른 블로그에 올렸습니 다.

아래 링크를 참고하세요.

### [파이썬 나만의 RPA 툴 제작 프로젝트 (Tricks\_Office)](https://tricks-office.com/python-tricks_office/)

[파이썬으로 나만의 RPA 툴을 직접 만드는 장기 프로젝트, 프로젝트명 “Tricks\_Office” 를 진행하고자 합니다. 구현을 계 획하고 있는 기능은 다음과 같습니다. 여러 PPT 파일의 여러 단어(Excel 형태로 보관된 Dictionary 파일 기준)를 원하는 단어로 치환하기 (구현)양식이 맞춰진 엑셀 파일을 하나의 파일로 취…](https://tricks-office.com/python-tricks_office/)

[tricks-office.com](https://tricks-office.com/python-tricks_office/)

# 파이썬으로 PPT Font 통일하기

## Python

2021/07/01 06:50 <http://blog.naver.com/550sn/222384801239>



제가 Youtube 채널을 시작했던 당시 처음으로 만들었던 PPT 매크로가 PPT 폰트 통일이었습니다.

PPT로 많은 자료를 만드는 분들이 굉장히 필요로 하는 기능중의 하나가 폰트 통일이었던것 같습니다. 이번에는 파이썬에서 이와 같은 기능을 구현해 보겠습니다.

PPT 매크로에서 이미 만든 기능이지만, 파이썬에서 다시 만들어 보는데에는 이번에 진행한 장기 Project와 상관이 있습니 다.

금번 Project에서는 특정 폴더에 반복 작업을 할 파일을 모아두고 실행시 일괄 변경 작업을 하는 방식으로 기능을 구현했 습니다.

하나의 파일에 대해서 이와 같은 작업을 할 수 있게 기능을 구현해 두면, GUI와 연계 해서 폴더내에서의 파일들을 읽어와 반복 할 수 있게 기능을 구현할 수 있겠습니다.

1. python-pptx 라이브러리로 구현하기

이 구문 자체는 예전에 번역기를 삼킨 파이썬에서 다루었던 소스를 그대로 응용하면 크게 어렵지 않게 구현이 가능합니다.

※ 번역기를 삼킨 파이썬 연결

from google.cloud import translate\_v2 as translate import os from pptx import Presentation os.environ['GOOGLE\_APPLICATION\_CREDENTIALS'] = r"Google-Cloud Key가 저장된 파일 경로" client = translate.Client() prs=Presentation("Trans\_Sample.pptx") Slides = prs.slides for slide in Slides: Shapes = slide.shapes for shape in Shapes: if shape.has\_text\_frame: text\_frame = shape.text\_frame for paragraph in text\_frame.paragraphs: font=paragraph.runs[0].font Fname = font.name Fsize = font.size cur\_text = paragraph.text result = client.translate(cur\_text, target\_language='en') new\_text = result['translatedText'] paragraph.text = new\_text for run in paragraph.runs: font = run.font font.name = Fname font.size = Fsize prs.save('T\_Result.pptx')

오히려 있던 기능에서 번역기에 대한 기능만 빼고, Font name만 변경하면 됩니다.

from pptx import Presentation prs=Presentation("Sample.pptx") Fname = "font Name" Slides = prs.slides for slide in Slides: Shapes = slide.shapes for shape in Shapes: if shape.has\_text\_frame: text\_frame = shape.text\_frame for paragraph in text\_frame.paragraphs: for run in paragraph.runs: runfont.name = Fname prs.save('T\_Result.pptx')

Table안에 글이 있는 경우, Group 안에 글이 있는 경우를 포함하기 위해 소스를 다음과 같이 추가/정리해 둡니다. 그리고, 아예 함수화 하여 GUI에서 연결해 가져갈 수 있게 구성해 둡니다.

from pptx import Presentation from pptx.enum.shapes import MSO\_SHAPE\_TYPE def C\_Font (shp,Fname): for paragraph in shp.text\_frame.paragraphs: for run in paragraph.runs: run.font.name = Fname def AutoFont(file,Tpath,Rpath,Fname): prs = Presentation(Tpath + '/' + file) for slide in prs.slides: for shape in slide.shapes: if shape.has\_text\_frame: C\_Font(shape,Fname) if shape.has\_table: for row in shape.table.rows: for cell in row.cells: C\_Font(cell,Fname) if shape.shape\_type == MSO\_SHAPE\_TYPE.GROUP: for shp in shape.shapes: if shp.has\_text\_frame: C\_Font(shp,Fname) prs.save(Rpath + '/' + file) Tpath = 'Sample' Rpath

= 'Result' Fname = '맑은 고딕' file = 'Sample.pptx' AutoFont(file,Tpath,Rpath,Fname) 일단, 실행은 되는데, 완벽하지는 않습니다.



영문 영역은 Font가 잘 바뀌었는데, 한글 영역은 바뀌지 않았습니다. 이런 부분이 보통 개선이 가장 어렵습니다.

한국 특화된 영역, 특화된 로직이 필요한 부분에는 전체 Library에서 그 기능을 잘 제공하지 않는 경우가 많거든요.

제가 검색 능력이 모자라서 그럴수도 있고, python-pptx 라이브러리가 그 기능을 제공하지 않아서 그럴수도 있는데, 저는 python-pptx 라이브러리에서 한글 폰트(MS에서는 FarEast라고 별도 폰트를 관리할 수 있게 해 주고 있습니다. 아마도 해당 폰트 분리 기능은 아시아권에만 제공되는 기능일 수도 있을듯 합니다.)를 다루는 기능을 지원하지 않았습니다.

1. win32com 라이브러리로 구현하기

결국 다른 방법을 찾아야 하는데, 이와 같은 방법으로는 아예 PowerPoint Application을 불러와서 직접 다루는 방법이 있습니다.

이는 windows에서 실행 가능한 프로그램을 실행하게 하면서 그 내부를 제어하는 win32com이라는 라이브러리를 활용 하여 구현이 가능합니다.

구글에서 "python win32com open pptx"로 검색을 하면 다음과 같은 예제 파일을 받을 수 있습니다.



### [Opening a Powerpoint presentation, Saving as PDF and closing the application using WIN32APIs](https://stackoverflow.com/questions/55227428/opening-a-powerpoint-presentation-saving-as-pdf-and-closing-the-application-usi)

[I have written a small script to open a powerpoint file, save it as PDF and close powerpoint. It looks like the command ppSaveAsPDF is not getting recognized. It says NameError: name 'ppSaveAsPDF... stackoverflow.com](https://stackoverflow.com/questions/55227428/opening-a-powerpoint-presentation-saving-as-pdf-and-closing-the-application-usi)

import win32com.client, sys FILENAME = "C:\\Users\\Swaroop\\Desktop\\Scripts\\Test.pptx" APPLICATION = win32com.client.Dispatch("PowerPoint.Application") PRESENTATION = APPLICATION.Presentations.Open(FILENAME, ReadOnly= False)

PRESENTATION.SaveAs("C:\\Users\\Swaroop\\Desktop\\Output.pdf", 32) APPLICATION.Quit()

PRESENTATION = None APPLICATION = None

이를 토대로 하여 내게 필요한 소스를 만들어 보겠습니다.

아, 먼저 pywin32 라이브러리부터 설치 해 줘야 합니다. 터미널에서 pip install pywin32

그리고 소스 파일을 작성해야 하는데, Powerpoint Application을 끌어오는 부분까지는 앞선 자료를 참고하면 되겠는데, 이후의 소스 구성에 대한부분은 검색이 좀 어렵습니다.

예전에 만들었던 폰트 변경에 대한 VBA 소스를 참조하겠습니다. [**PPT Font 통일 Part 2. (그룹 포함, PPT 매크로)**](https://blog.naver.com/550sn/221784564251)

[오늘은, 예전에 올렸던 자동 폰트 통일하기 기능에 다음의 두가지를 추가 보완하고자 합니다. 1. 소스를 조... blog.naver.com](https://blog.naver.com/550sn/221784564251)

Private Sub cmd\_AFont\_Click() Dim SL\_Count, SH\_Count, SL\_Num, SH\_Num As Integer Dim EFont, HFont As String Dim tbl As Table Dim t\_R, t\_C As Integer '폰트명 읽어오기 HFont = txt\_HFont.Text EFont = txt\_EFont.Text SL\_Count = ActivePresentation.Slides.Count ' 모든 슬라이드에 대하여 폰트를 변경합니다. For SL\_Num = 1 To SL\_Count SH\_Count = ActivePresentation.Slides(SL\_Num).Shapes.Count For SH\_Num = 1 To SH\_Count ' 선택된 개체가 표일때 표의 각 Cell마다 폰트를 변경합니다. If ActivePresentation.Slides(SL\_Num).Shapes(SH\_Num).HasTable = msoTrue Then ' 표일 경우 폰트 변경 Set tbl = ActivePresentation.Slides(SL\_Num).Shapes(SH\_Num).Table For t\_R = 1 To tbl.Rows.Count For t\_C = 1 To tbl.Columns.Count With tbl.Cell(t\_R, t\_C).Shape.TextFrame.TextRange .Font.NameFarEast = HFont '한글폰트

.Font.Name = EFont '기본폰트 End With Next t\_C Next t\_R ' 선택된 개체가 Text Frame일때 폰트를 변경합니다. ElseIf ActivePresentation.Slides(SL\_Num).Shapes(SH\_Num).HasTextFrame = msoTrue Then ' Text Frame 일 경 우 폰트 변경 ActivePresentation.Slides(SL\_Num).Shapes(SH\_Num).TextFrame.TextRange.Font.NameFarEast = HFont ActivePresentation.Slides(SL\_Num).Shapes(SH\_Num).TextFrame.TextRange.Font.NameAscii = EFont End If Next SH\_Num Next SL\_Num frm\_AFont.Hide End Sub

이를 참고하여 다음과 같이 작성합니다.

import win32com.client def AutoFont2(file,Tpath,Rpath,FName): powerpoint = win32com.client.Dispatch('PowerPoint.Application') Sepa = '\\' Tpath = Tpath.replace("/","\\") Rpath = Rpath.replace("/","\\") ppt = powerpoint.Presentations.Open (Tpath + Sepa + file, WithWindow=False) for slide in ppt.Slides: for shape in slide.shapes: if shape.HasTextFrame == -1: shape.TextFrame.TextRange.Font.NameFarEast = FName shape.TextFrame.TextRange.Font.Name = FName if shape.HasTable == -1: for row in shape.Table.Rows: for cell in row.cells: cell.Shape.TextFrame.TextRange.Font.NameFarEast = FName cell.Shape.TextFrame.TextRange.Font.Name = FName try: for GI in shape.GroupItems: GI.TextFrame.TextRange.Font.NameFarEast = FName GI.TextFrame.TextRange.Font.Name = FName except: pass ppt.SaveAs (Rpath + Sepa + file) ppt.Close ()

Tpath = 'D:/Python/TricksOffice/Sample' Rpath = 'D:/Python/TricksOffice/Result' Fname = '맑은 고딕' file = 'Sample.pptx' AutoFont2(file,Tpath,Rpath,Fname)

※ 어떻게 보면 이번 코드 작성이 지금까지 해왔던 다른 코드 수정에 비해서 가장 까다로울 수 있습니다.

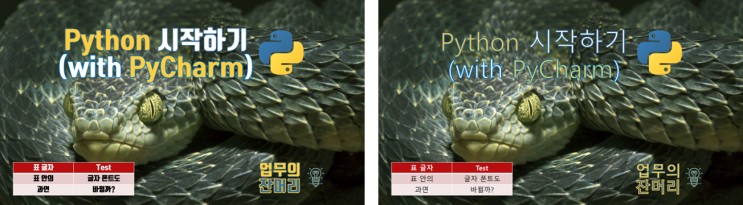
1. 이와 같은 방법을 취하는 사람이 그리 많지 않았던듯 합니다. 참고할 문서를 찾기가 어려웠습니다. (제 검색 능력이 부족해서 그럴 수도 있습니다. )
2. 파이썬 라이브러리가 아니라 Application을 직접 끌어오는 방식으로 구현할 경우, 파이참에서 사용 가능한 메서드의 키워드를 찾아주는 등의 도움을 받기 어렵습니다.
3. 오히려 예전에 VBA로 작성했던 VBA 소스를 가져와서 재활용하는 것이 더 도움이 될 수 있습니다. 대소문자도 구분하여 정확히 입력해야 하기 때문에, VBA에서 작성한 다음 가져오는 것도

한 방법이 될 수 있을것 같습니다.

1. 파이썬에서는 '\'가 있을때 이를 특수 문자를 사용하기 위한 기호로 인식하기 때문에

보통 /로 값을 치환하여 사용합니다. 이 부분을 '\'로 인식할 수 있게 바꾸는 과정이 필요합니다.

1. 특이하게 MSO True는 -1 값을 가지고 있으므로, 조건문에서는 -1을 참일때로 사용해야 합니다.

이렇게 만든 소스로 실행을 하면, 이번에는 잘 실행됩니다.

한글 폰트도, 표안의 글자도, 그룹안의 글자도 모두 잘 변경된 것을 볼 수 있습니다. 다만, 실행 속도가 정말 느립니다.

지금까지 다뤘던 파이썬의 다른 라이브러리들은 각각의 전용 Application을 사용하지 않고, 데이터에서 필요한 값만을 끌 어와서 사용했기 때문에 전용 소프트웨어로 하나하나 열면서 작업하는것과는 비교가 안되는 빠른 속도를 자랑했습니다. 반면, 이 win32com을 사용하는 경우에는 PowerPoint를 직접 띄워서 작업하는 방식이기 때문에 PPT를 통해서 직접 파 일을 열고, 작업하고, 닫는 형식이기 때문에 시간이 오래 걸립니다.

그런데, 이와 같은 win32com 기능을 활용해야만 하는 경우도 있습니다.

1. Excel, PPT 등과 같은 전용 Application 안에서만 실행되는 VBA를 실행해야 할 때
2. 회사 보안 등의 이유로 DRM 이 걸려 있어서 정식 Application을 실행하지 않고는 수정이 불가능할 때

처리 속도가 많이 느리다고 하기는 했지만, 사람이 하나하나 수작업 하는것 보다는 빠릅니다.

옵션에서 "WithWindow = False"로 해주면 사람이 보기좋게 화면에 띄우는 작업이 생략되기 때문에 이것 만으로도 제법 속도 개선이 이루어 집니다.

※ 좀 느리더라도 제대로 진행되고 있는것인지를 확인하고 싶으면 이 옵션을 True로 해둬도 됩니다.

만들어진 소스 파일은 다음과 같습니다. **첨부파일**

PPT\_Win.py [파일 다운로드](http://blogattach.naver.net/a732bb081c2c2d98b15735053fd9a6dc7e2cd605/20210604_122_blogfile/550sn_1622794786498_7Tx33m_py/%2FPPT_Win.py)

폰트 통일 기능 포함하여, 여러 파일을 일괄 처리 할수 있는 실행파일은 아래 링크에서 다운로드 받을 수 있습니다.

### [파이썬 나만의 RPA 툴 Tricks\_Office](http://tricks-office.com/python-tricks_office/)

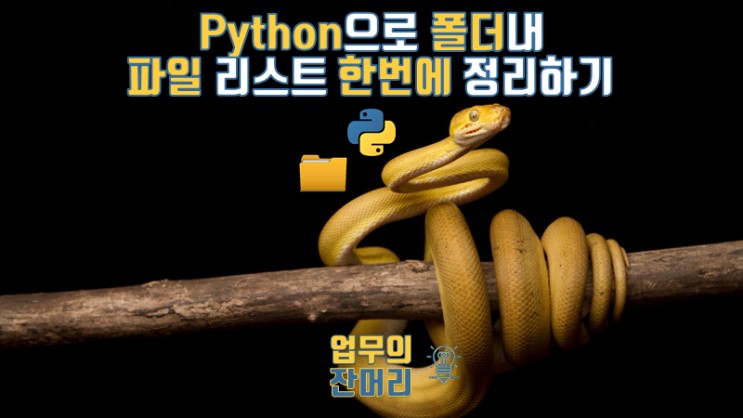
[Python으로 만든 개인용 업무 자동화 RPA툴 Tricks-Office를 공개합니다. 대표적인 PPT와 Excel 문서 노가다 작업을 버 튼 한번으로 실행할 수 있게 구현했습니다.…](http://tricks-office.com/python-tricks_office/)

[tricks-office.com](http://tricks-office.com/python-tricks_office/)

# 파이썬으로 폴더내의 파일리스트 한번에 정리하기

## Python

2021/07/05 06:50 <http://blog.naver.com/550sn/222385113610>



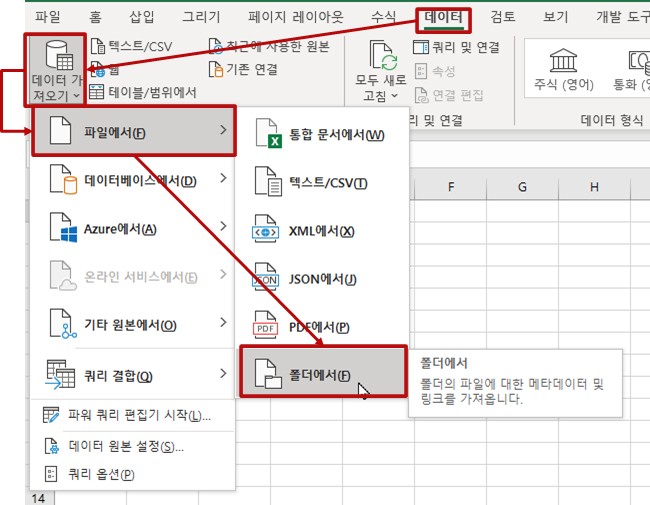
업무를 하다 보면 특정 폴더 안에 파일 리스트를 조사해야 할 때가 있습니다. 오늘은 이 파일 리스트를 한번에 조사할 수 있는 방법에 대하여 알아 보겠습니다.

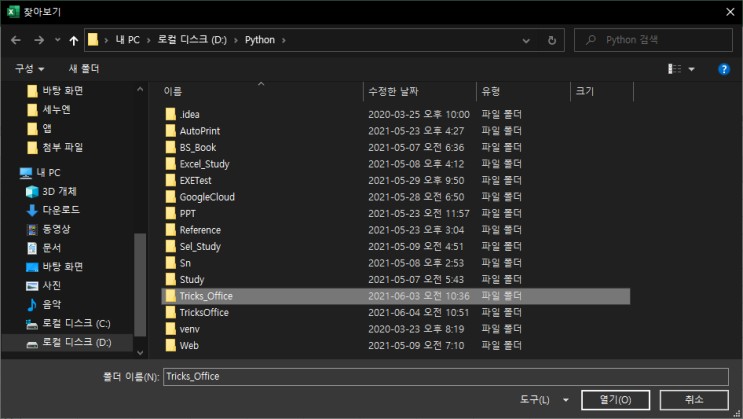
1. 엑셀의 기본 기능을 이용하는 방법

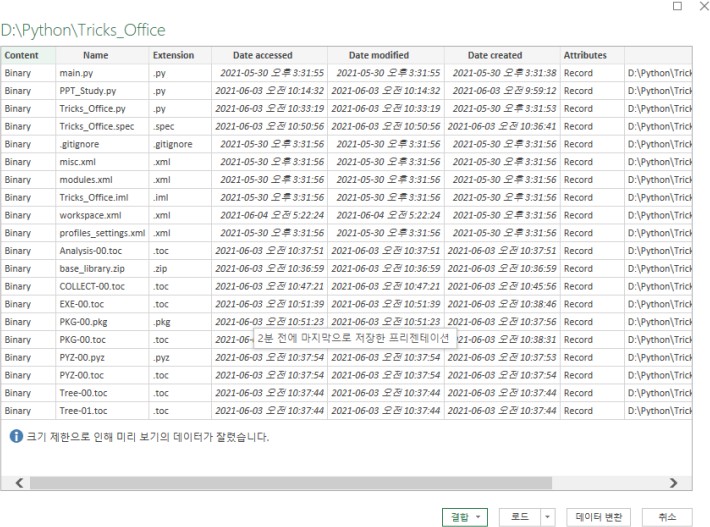
이때 가장 손쉽게 사용할 수 있는 방법은 엑셀에서 기본적으로 제공하는 데이터 관련 기능입니다.

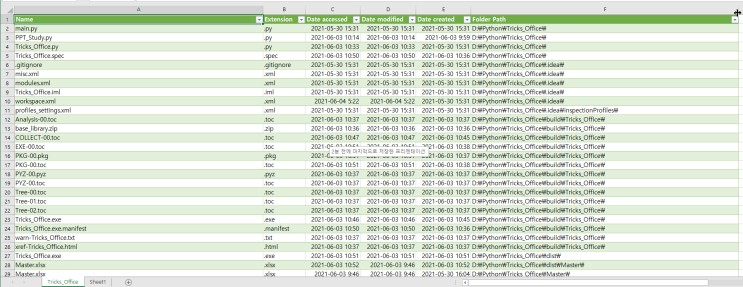
(파워쿼리에 해당하는 기능으로 버전에 따라서는 파워쿼리를 추가기능으로 추가해야 할 수도 있으며, 메뉴의 구조는 조금 씩 다를 수 있습니다.)

메뉴에서 데이터 → 데이터 가져오기 → 파일에서 → 폴더에서를 선택하면



그러면 폴더를 선택하는 창이 뜨고, 여기서 폴더를 선택하면 해당 폴더 안에 있는 파일 리스트를 자겨와 정리를 해 줍니다.





파일 이름, 폴더, 확장자명, 데이터 생성일, 경로까지 완벽하게 정보가 생성되는 것을 볼 수 있습니다.

그런데 말입니다...

안타깝게도 파일 사이즈에 대한 정보는 여기 포함되어 있지 않습니다.

만약 각 파일 리스트의 파일 크기에 대한 값도 정리가 필요하다면, 다른 방법을 찾아야 할 듯 합니다.

1. 파이썬으로 해결하기

이번에도 내가 참고해 볼만한 정보를 찾아 보겠습니다. 구글에서 "python file list in folder"로 검색을 해 봅니다.



### [How do I list all files of a directory?](https://stackoverflow.com/questions/3207219/how-do-i-list-all-files-of-a-directory)

[How can I list all files of a directory in Python and add them to a list? stackoverflow.com](https://stackoverflow.com/questions/3207219/how-do-i-list-all-files-of-a-directory)

이번에도 참고할만한 소스가 깔끔하게 나옵니다.

from os import walk f = [] for (dirpath, dirnames, filenames) in walk(mypath): f.extend(filenames) break 여기서 내가 가져오고 싶은 값은 파일이름, 경로, 날짜 그리고 파일 크기 입니다.

이번에도 구글에서 "python file size"로 검색합니다.



### [How can I check file size in Python?](https://stackoverflow.com/questions/2104080/how-can-i-check-file-size-in-python)

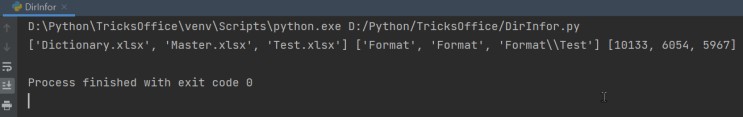
[I am writing a Python script in Windows. I want to do something based on the file size. For example, if the size is greater than 0, I will send an email to somebody, otherwise continue to other thi... stackoverflow.com](https://stackoverflow.com/questions/2104080/how-can-i-check-file-size-in-python)

이번에도 참고할만한 소스를 찾을 수 있습니다. os.stat('somefile.txt').st\_size

그러면, 이 두 내용을 조합하고, 내게 필요한 정보를 정리하면 다음과 같이 코드를 만들 수 있습니다.

import os def DFInfo(TPath): FList = [] FPath = [] Fsize = [] FDate = [] for (dirpath, dirnames, filenames) in os.walk(TPath): for file in filenames: FList.append(file) FPath.append(dirpath) Fsize.append(os.stat(dirpath + "/" + file).st\_size) FDate.append(os.stat(dirpath + "/" + file).st\_ctime) print (FList, FPath, Fsize, FDate) return Path = "Format" DFInfo(Path)

화면에서 필요한 정보가 다음과 같이 표시 되는 것을 볼 수 있습니다.



이제, 이 데이터를 엑셀로 저장했으면 좋겠습니다.

Pandas에서 Dataframe을 만들고 저장하는 법을 찾아보겠습니다. 구글에서 "pandas make dataframe from list"로 검색합니다.

### [Different ways to create Pandas Dataframe - GeeksforGeeks](https://www.geeksforgeeks.org/different-ways-to-create-pandas-dataframe/)

[A Computer Science portal for geeks. It contains well written, well thought and well explained computer science and programming articles, quizzes and practice/competitive programming/company interview Questions.](https://www.geeksforgeeks.org/different-ways-to-create-pandas-dataframe/)

[www.geeksforgeeks.org](https://www.geeksforgeeks.org/different-ways-to-create-pandas-dataframe/)

참고할만한 소스로 찾은 내용은 다음과 같습니다.

import pandas as pd # intialise data of lists. data = {'Name':['Tom', 'nick', 'krish', 'jack'], 'Age':[20, 21, 19, 18]} # Create DataFrame df = pd.DataFrame(data) # Print the output. df

상기 내용을 참고로 하여 아래와 같이 소스를 수정합니다.

import os import pandas as pd def DFInfo(TPath): FList = [] FPath = [] Fsize = [] FDate = [] for (dirpath, dirnames, filenames) in os.walk(TPath): for file in filenames: FList.append(file) FPath.append(dirpath) Fsize.append(os.stat(dirpath + "/" + file).st\_size) FDate.append(os.stat(dirpath + "/" + file).st\_atime) data =

{'FileName': FList, 'Path': FPath, 'FileSize':Fsize, 'FileDate':FDate} df = pd.DataFrame(data) df.to\_excel('FileList.xlsx') return Path = "Format" DFInfo(Path)

이제 실행하면 원하는 값은 저장이 되는데, 시간 정보가 이상합니다. 구글에서 "python convert os time"로 검색하여 자료를 찾아 봅니다.



### [How can I convert os.path.getctime()?](https://stackoverflow.com/questions/19501711/how-can-i-convert-os-path-getctime)

[How can I convert os.path.getctime() to the right time? My source code is: import os print("My Path: "+os.getcwd()) print(os.listdir(".")) print("Root/: ",os.listdir...](https://stackoverflow.com/questions/19501711/how-can-i-convert-os-path-getctime)

[stackoverflow.com](https://stackoverflow.com/questions/19501711/how-can-i-convert-os-path-getctime)

참고할 코드를 찾을 수 있습니다.

from datetime import datetime datetime.fromtimestamp(1382189138.4196026).strftime('%Y-%m-%d

%H:%M:%S')

이를 토대로 다음과 같이 소스를 수정해 보겠습니다.

import os import pandas as pd from datetime import datetime def DFInfo(TPath): FList = [] FPath = [] Fsize = [] FDate = [] for (dirpath, dirnames, filenames) in os.walk(TPath): for file in filenames: FList.append(file) FPath.append(dirpath) Fsize.append(os.stat(dirpath + "/" + file).st\_size) ModTime = datetime.fromtimestamp(os.stat(dirpath + "/" + file).st\_mtime).strftime('%Y-%m-%d %H:%M:%S') FDate.append(ModTime) data = {'FileName': FList, 'Path': FPath, 'FileSize':Fsize, 'FileDate':FDate} df = pd.DataFrame(data) df.to\_excel('FileList.xlsx') return Path = "Format" DFInfo(Path)

이상의 결과가 가능하면 결과값을 모아놓는 폴더로 모이게 설정을 해 두면 좋겠습니다. 이를 조금만 더 수정해 보겠습니다.

import os import pandas as pd from datetime import datetime def DFInfo(TPath,RPath): FList = [] FPath = [] Fsize = [] FDate = [] for (dirpath, dirnames, filenames) in os.walk(TPath): for file in filenames: FList.append(file) FPath.append(dirpath) Fsize.append(os.stat(dirpath + "/" + file).st\_size) ModTime = datetime.fromtimestamp(os.stat(dirpath + "/" + file).st\_mtime).strftime('%Y-%m-%d %H:%M:%S') FDate.append(ModTime) data = {'FileName': FList, 'Path': FPath, 'FileSize':Fsize, 'FileDate':FDate} df = pd.DataFrame(data) df.to\_excel(RPath + '/' + 'FileList.xlsx',index=False) return Path = "Format" RPath = "Result" DFInfo(Path,RPath)

완성된 파일은 다음과 같습니다. **첨부파일**

DirInfor.py [파일 다운로드](http://blogattach.naver.net/c055dc6f7c2d24f8d43651635fbec4b81b4bb154e7/20210604_30_blogfile/550sn_1622809856701_lgze7f_py/%2FDirInfor.py)

# 정규식을 삼킨 파이썬 (파일 필터)

## Python

2021/07/08 06:50 <http://blog.naver.com/550sn/222385189403>



이번 장기 프로젝트를 통해서 자료를 만들다 보니 우연히도 지금까지 만든 기능 모두가 모두 PPT 파일만을 다루게 되어 있 었습니다.

취약점이 있는데, 해당 폴더에 PPT 가 아닌 파일이 있을 때 에러가 발생하면서 진행이 되지 않는 이슈가 있었습니다.

그래서, 구글에서 "python file name filter"로 검색을 하면 참고할만한 자료를 찾을 수 있습니다.



### [Get a filtered list of files in a directory](https://stackoverflow.com/questions/2225564/get-a-filtered-list-of-files-in-a-directory)

[I am trying to get a list of files in a directory using Python, but I do not want a list of ALL the files. What I essentially want is the ability to do something like the following but using Pytho...](https://stackoverflow.com/questions/2225564/get-a-filtered-list-of-files-in-a-directory)

[stackoverflow.com](https://stackoverflow.com/questions/2225564/get-a-filtered-list-of-files-in-a-directory)

files = [f for f in os.listdir('.') if re.match(r'[0-9]+.\*\.jpg', f)]

대략적으로 봤을때, 파일 정보를 읽어온 다음, ".jpg"가 포함된 파일을 찾아오는 로직 같은데, 알수없는 외계어가 섞여 있 습니다. 그리고, re라는 약간은 생소한 라이브러리가 사용된것 같고요.

'[0-9]+.\*\.jpg'

이 위에 표시된 외게어 느낌의 글이 정규식이고, 이 정규식을 다루는 라이브러리가 "re" 입니다.

정규식이 처음에 접할때 생소하고 익히기 어려운데, 한번 익혀두면 두고 두고 써먹을데가 많다고 하니 한번쯤 살펴 볼만 합니다.

관련 내용을 모두 머리속에 넣어 두는것도 좋겠지만, 어디서 이와 관련된 내용을 찾을 수 있는지를 알아두고, 필요할때 마 다 찾아 보는것도 좋은 방법중 하나가 될듯 합니다.

구글에서 "정규식"으로 찾으면 참고할만한 자료가 매우 많이 나옵니다.

### [위키독스](https://wikidocs.net/4308)

[온라인 책을 제작 공유하는 플랫폼 서비스 wikidocs.net](https://wikidocs.net/4308)

여기 내용을 잘 살펴보면, 우리가 어떤 파일을 찾을때 비교에 사용되는 구분자로 사용할만한 내용을 꽤 많이 정리할 수 있

습니다.

많은 내용이 있지만, 제가 꼭 사용할 몇가지만 찾아 보겠습니다.

1. . : 줄바꿈 문자를 제외한 모든 문자열을 의미
2. [ ] : 대괄호 안의 문자들의 클레스를 의미 (Ex. [0-9] : 숫자)
3. \* : \*앞의 문자가 없거나 여러번 반복되는 경우를 의미

4) + : + 앞의 문자가 1번 이상 반복되는 경우를 의미

5) ? : ? 앞의 문자가 없거나 1번 나오는 경우를 의미 이상의 내용으로 앞서 씌여진 정규식 부분을 해석하면 '[0-9]+.\*\.jpg'

제일처음에 [0-9]+ 라고 표시가 되어 있습니다.

첫글자는 숫자이어야 하는데, 그 숫자는 1번 이상 나와야 된다라는 의미입니다. 숫지로 시작하는 파일이름만이 처음의 [0-9]+ 라는 조건을 만족합니다.

두번째 구문이 .\*라고 되어 있는데, 그 뒤로는 어떤 값이 와도 두번째 조건은 만족이라는 뜻입니다.

그 다음영역이 \.jpg인데, 여기서 "\."은 "."을 의미합니다. 정규식 내에서는 "."이 모든 글자를 의미하기 때문에 "."을 표시하기 위해서는 "."으로 표시를 하는 것입니다. 그런데, 이는 [.] 으로 문자 클레스로 묶어도 동일한 효과를 얻을수 있습 니다.

이 영역이 사실상 우리가 파일인지 아닌지를 구분하기 위한 영역입니다.

우리가 만들고자 하는 정규식은 파일 확장자 명이 ".ppt" 또는 ".pptx" 인 파일입니다. 즉 ".ppt"가 반드시 포함되어 있어 야 하고, 그 앞에는 어떤 글자가 오든 상관없다는 표시를 해야 합니다.

이를 정규식으로 표현하면 다음과 같이 표현할 수 있습니다. '.\*[.]ppt'

이제 이 정규식을 사용하는 함수인 re를 알아볼 차례인데, 여기서는 정규식에 들어있는 내용이 문자열에 포함되는 것인지 를 알아보는 부분만 살펴 보면되니, re.match(정규식,비교할문서) 를 통해 True/False를 되돌려 준다는 점만 이해하면 될 것 같습니다.

마찬가지로 re라이브러리도 불러와 줘야 하니 터미널에서 라이브러리 설치와 처음에 import로 불러오는 동작도 해 줘야 합니다.

pip install re import re ...

이제 이를 토대로 앞서 만들었던 Tricks\_Office 파일을 수정하면 아래와 같이 소스를 정리할 수 있겠습니다.

여전히 소스에서 비효율적으로 정리된 부분도 많은데, 아직 제가 실력이 모자라서 실력을 키워나가면서 조금씩 해야 할듯 합니다.

from tkinter import \* from tkinter import filedialog from tkinter import font import pandas as pd import os from tkinter import messagebox import PPT\_Study import PPT\_Font import PPT\_Win import PPT\_Trans import re #Master Data 저장하기 def saveMaster(): writer = pd.ExcelWriter('Master.xlsx', engine='xlsxwriter')

df.to\_excel(writer, sheet\_name='Master', index=False) dff.to\_excel(writer, sheet\_name='FileName', index=False) writer.save() return #폴더 경로 변경 def onClick(i): folder\_selected = filedialog.askdirectory() lbPath[i].config(text=folder\_selected) df.Detail[i] = folder\_selected saveMaster() return #파일경로 변경 def onClickf(i): folder\_selected = filedialog.askopenfile() lbPathf[i].config(text=folder\_selected.name) dff.Detail[i]

= folder\_selected.name saveMaster() return #Dictionary 기준, Keyword 변경 def PPT\_KC\_Action(): files = [f for f in os.listdir(df.Detail[0]) if re.match('.\*[.]ppt', f)] for file in files: PPT\_Study.R\_KWord(file,df.Detail[0],df.Detail[1],dff.Detail[0]) messagebox.showinfo(title="Finish", message="Job Finished!") return #PPT Format 통일 (python-pptx) def UFontAc(): files = [f for f in os.listdir(df.Detail[0]) if re.match('.\*[.]ppt', f)] for file in files: PPT\_Font.AutoFont(file, df.Detail[0], df.Detail[1], select.get(select.curselection())) messagebox.showinfo(title="Finish", message="Job Finished!") return #PPT Format 통일 (win 32) def U2FontAc(): files = [f for f in os.listdir(df.Detail[0]) if re.match('.\*[.]ppt', f)] for file in files: PPT\_Win.AutoFont2(file, df.Detail[0], df.Detail[1], select.get(select.curselection())) messagebox.showinfo(title="Finish", message="Job Finished!") return #영어 로 구글 번역 (google-trans) def GTrans2EnAc(): files = [f for f in os.listdir(df.Detail[0]) if re.match('.\*[.]ppt', f)] for file in files: PPT\_Trans.GTrans(file, df.Detail[0], df.Detail[1], 'en') messagebox.showinfo(title="Finish", message="Job Finished!") return #한글로 구글 번역 (google-trans) def GTrans2KoAc(): files = [f for f in os.listdir(df.Detail[0]) if re.match('.\*[.]ppt', f)] for file in files: PPT\_Trans.GTrans(file, df.Detail[0], df.Detail[1], 'ko') messagebox.showinfo(title="Finish", message="Job Finished!") return #창 호출 & 초기값 설정 win = Tk() win.geometry("600x650") win.title('Tricks\_Office') df= pd.read\_excel("Master.xlsx",sheet\_name="Master") dff= pd.read\_excel("Master.xlsx",sheet\_name="FileName") fonts=list(font.families()) #fram1(폴더경로) 설정 frame1 = Frame(win,pady=10) frame1.pack() lbName = [] lbPath = [] btnPath =[] for i in df.index: lbName.append(Label(frame1, text=df.Item[i], width=10)) lbName[i].grid(row=i, column=0, sticky=W,padx=5, pady=5) lbPath.append(Label(frame1, text=df.Detail[i], width=50,height=2,relief="groove")) lbPath[i].grid(row=i, column=1, sticky=W) btnPath.append(Button(frame1, text="Change Path", width=10, command=lambda i=i: onClick(i))) btnPath[i].grid(row=i, column=2, sticky=W) #frame1h(파일 경로) 설정 frame1h = Frame(win,pady=10) frame1h.pack() lbNamef = [] lbPathf = [] btnPathf =[] for i in dff.index: lbNamef.append(Label(frame1h, text=dff.Item[i], width=10)) lbNamef[i].grid(row=i, column=0, sticky=W,padx=5, pady=5) lbPathf.append(Label(frame1h, text=dff.Detail[i], width=50, height=2,relief="groove")) lbPathf[i].grid(row=i, column=1, sticky=W) btnPathf.append(Button(frame1h, text="Change Path", width=10, command=lambda i=i: onClickf(i))) btnPathf[i].grid(row=i, column=2, sticky=W) #framePPT 실행 버튼 설정 (PPT전용) framePPT = Frame(win, padx=10, pady=10,bd=1,relief=SOLID) framePPT.pack() #PPT전용 표기 Label(framePPT,text = "PPT 전용").grid(row=0,column=1) # Keywork Change 버튼 btnPPT\_KC = Button(framePPT,text="Find and Replace keywords",width = 21, height=2, command=lambda:PPT\_KC\_Action()) btnPPT\_KC.grid(row=1, column=1, sticky=W,padx=5, pady=5) lbPPT\_KC = Label(framePPT, text="Dictionary 정보를 기반으로,

Keywords를 바꿔주는 기능입니다. \n" "Dictionary 파일 경로가 정확하게 지정되어 있어야 합니다.", width=50,height=2,relief="groove") lbPPT\_KC.grid(row=1, column=2, sticky=W) #Font 통일1 UFontbtn = Button(framePPT, text="Unify to Same Font",width = 21,height=2, command=lambda:UFontAc()) UFontbtn.grid(row=3, column=1,padx=5, pady=5) UFontlbl = Label(framePPT, text="아래 리스트에서 폰트가 선택되어 있어야 합니다.\n" + "속도가 매우 빠르지만, 한글 폰트가 안바뀔수 있습니다.", width=50,height=2,relief="groove") UFontlbl.grid(row=3, column=2) #Font 통일2 U2Fontbtn = Button(framePPT, text="Unify to Same Font 2",width = 21,height=2, command=lambda:U2FontAc()) U2Fontbtn.grid(row=4, column=1,padx=5, pady=5) U2Fontlbl = Label(framePPT, text="아래 리스트에서 폰트 가 선택되어 있어야 합니다.\n" +

**첨부파일** Tricks\_Office.py [파일 다운로드](http://blogattach.naver.net/940188392c1f1eab826406360cea95ef4d15e74f/20210604_27_blogfile/550sn_1622812495354_vrc7K1_py/%2FTricks_Office.py)

참고로 이와 같은 정규식을 조금 응용한다면, 이미지 파일에 대해서 구분하기 위해서는 아래와 같이 정규식으로 표현을 할 수 있을것 같습니다.

'.\*([.]jpg|[.]png|[.]gif)'

# 파이썬 엑셀 데이터 자동 합치기

## Python

2021/07/12 06:50 <http://blog.naver.com/550sn/222395157243>



"Tricks-Office"에서 구현한 기능 중 하나인, 같은 양식의 엑셀 데이터 합치기 기능의 구현 내용에 대하여 살펴 보도록 하 겠습니다.

파이썬에서 엑셀과 같은 스프레드 시트 형태 (표형태)의 데이터를 다루는데 있어서 pandas라는 라이브러리는 정말 강력 한 힘을 발휘 합니다. 이 강력한 기능을 활용하면 같은 형식의 엑셀 데이터를 병합하는 것은 굉장히 쉽게 구현이 가능합니 다.

이번 Case는 다음과 같이 여러개의 엑셀 시트에 같은 양식의 데이터가 모여 있을때, **첨부파일**

네이버.xlsx [파일 다운로드](http://blogattach.naver.net/64f178c8d881805c7092f5c7fb1a601cbfe812f8ca/20210612_147_blogfile/550sn_1623470861290_M8A4Zw_xlsx/%2F%B3%D7%C0%CC%B9%F6.xlsx) **첨부파일** 네이버2.xlsx [파일 다운로드](http://blogattach.naver.net/bc29a01300373683aa4c2e1e24c2bdc76234c28c/20210612_12_blogfile/550sn_1623470726810_lFkaW0_xlsx/%2F%B3%D7%C0%CC%B9%F62.xlsx) **첨부파일** 네이버3.xlsx [파일 다운로드](http://blogattach.naver.net/90058c3d241b1aaf8660023208ee91eb4e18ef47/20210612_88_blogfile/550sn_1623470733674_2GjPzn_xlsx/%2F%B3%D7%C0%CC%B9%F63.xlsx)

이 데이터 들을 한곳에 모으는, 이와 같은 작업을 해 보도록 하겠습니다.

pandas에서 각각의 정보를 Data Frame 형태로 읽어 왔을때를 가정하고, 이를 합치는 방법을 찾아 보겠습니다. 구글에서 "pandas merge dataframe"로 검색을 합니다.

검색되는 여러 글중 아래의 글을 보면 정리가 정말 잘되어 있습니다.



### [Merge, join, concatenate and compare — pandas 1.2.4 documentation](https://pandas.pydata.org/pandas-docs/stable/user_guide/merging.html)

[Merge, join, concatenate and compare pandas provides various facilities for easily combining together Series or DataFrame with various kinds of set logic for the indexes and relational algebra functionality in the case of join / merge-type operations. In addition, pandas also provides utilities to c...](https://pandas.pydata.org/pandas-docs/stable/user_guide/merging.html)

[pandas.pydata.org](https://pandas.pydata.org/pandas-docs/stable/user_guide/merging.html)

예제 소스를 찾아보면 정말 간결하게 답이 나와 있습니다. frames = [df1, df2, df3] result = pd.concat(frames)

엑셀에서 Data Frame 형태로 값을 가져오고, 저장하는 부분이 남아 있지만, 2개 이상의 엑셀 파일을 병합하는 것은 정말 저 두줄로 끝입니다.

물론, 병합이 제대로 잘 되기 위해서는 전제 조건이 있습니다. 머리글행이 잘 읽어 들어와 져 있어야 하고, 모든 데이터가 머리글행으로 잘 정리가 되어 있어야 합니다.

이 전제 조건만 만족되면 같은 형식의 데이터를 병합하는 것은 이것이 끝입니다.

※ 만약 시간이 허락된다면 위의 필요한 구문 이외에도 여러가지 2개 이상의 DataFrame을 병합하는 방법을 같이 봐 두시 면 큰 도움이 됩니다. 특히 대용량 데이터 간의 Vlookup 과 같은 조합을 걸어야 할 경우 여기 소개된 join 관련 내용을 익 혀두시면 정말 유용합니다.

그러면, 이와 같은 작업을 잘 하기 위해서 데이터를 읽어올 때 값을 잘 읽어 와야 합니다. 데이터를 읽어오는 부분을 보강해 주겠습니다.

빈 줄이 있을 때 이 줄을 건너뛰고 실제 데이터가 있는 영역에서 부터 값을 가져오게 해 보겠습니다.

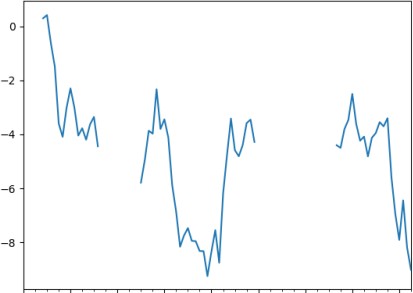
구글에서 "pandas skip rows"로 검색을 하여 필요한 정보를 찾아 봅니다. [**Skip rows during csv import pandas**](https://stackoverflow.com/questions/20637439/skip-rows-during-csv-import-pandas)

[I'm trying to import a .csv file using pandas.read\_csv(), however I don't want to import the 2nd row of the data file (the row with index = 1 for 0-indexing). I can't see how not to import it beca... stackoverflow.com](https://stackoverflow.com/questions/20637439/skip-rows-during-csv-import-pandas)

pd.read\_csv(StringIO(s), skiprows=[1], header=None)

처음 값을 읽어들일때 skiprows 라는 옵션 값으로 몇줄을 건너뛸 것인지만 결정해 주면 되는 것을 알 수 있습니다.

그렇다면, 어느칸부터 값이 들어있는지를 확인할 수 있는 로직을 고민해 보겠습니다. 구글에서 "pandas select no data"로 검색합니다.



### [Working with missing data — pandas 1.2.4 documentation](https://pandas.pydata.org/pandas-docs/stable/user_guide/missing_data.html)

[Working with missing data In this section, we will discuss missing (also referred to as NA) values in pandas. Note The choice of using NaN internally to denote missing data was largely for simplicity and performance reasons. Starting from pandas 1.0, some optional data types start experimenting with... pandas.pydata.org](https://pandas.pydata.org/pandas-docs/stable/user_guide/missing_data.html)

pd.isna(df2["one"])

pandas에 속한 메소드 중에 isna라는 속성을 활용하면 값이 비었는지 아닌지를 알 수 있습니다.

이제 제가 구현하고자 하는 기능에서 필요한 부분들은 찾아졌습니다. 다음과 같은 방식으로 파일을 읽어 들어오는 부분을 구현하면 되겠습니다.

1. 파일 전체를 DataFrame의 형태로 가져온다. 이때 머리글행이 없는것으로 인지하고 가져온다.
2. 한줄씩 읽어들어가면서 DataFrame 전체 기준으로 마지막 열에 값이 있는지를 판단한다.
3. 가장 마지막 열에 처음으로 값이 들어온 위치를 머리글행이라고 판단한다.
4. 머리글행 앞까지의 행을 무시하고 다시 DataFrame 형태로 값을 읽어 온다.

위의 내용을 구현하기 위해 pandas에서 한행씩 읽어가면서 반복문을 수행하는 부분을 찾아보겠습니다. 구글에서 "python pandas loop through rows and columns"로 검색을 합니다.



### [How to iterate over rows in a DataFrame in Pandas](https://stackoverflow.com/questions/16476924/how-to-iterate-over-rows-in-a-dataframe-in-pandas)

[I have a DataFrame from Pandas: import pandas as pd inp = [{'c1':10, 'c2':100}, {'c1':11,'c2':110},](https://stackoverflow.com/questions/16476924/how-to-iterate-over-rows-in-a-dataframe-in-pandas)

[{'c1':12,'c2':120}] df = pd.DataFrame(inp) print df Output: c1 c2 0 10 100 1 11 110 2... stackoverflow.com](https://stackoverflow.com/questions/16476924/how-to-iterate-over-rows-in-a-dataframe-in-pandas)

for ind in df.index: print df['c1'][ind], df['c2'][ind] 위와 같은 참고 자료를 찾을 수 있습니다.

여기까지 찾은 내용들을 조합하여 소스로 정리하면 다음과 같이 정리할 수 있습니다. l\_row=0 df1=pd.read\_excel("읽을 파일이름", header=None) for x in df1.index: if not

pd.isna(df1.values[x,df1.columns.size-1]) : l\_row = x break df1=pd.read\_excel("읽을 파일 이름", skiprows = l\_row)

이제 읽어온 값에서 중복값이 있을수 있으니, 중복값을 빼는 방법을 찾아 보겠습니다. 구글에서 "pandas drop duplicates"로 검색을 합니다. [**pandas.DataFrame.drop\_duplicates — pandas 1.2.4 documentation**](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop_duplicates.html)

[pandas.DataFrame.drop\_duplicates DataFrame. drop\_duplicates ( subset = None , keep = 'first' , inplace = False , ignore\_index = False ) [source] Return DataFrame with duplicate rows removed. Considering certain columns is optional. Indexes, including time indexes are ignored. Parameters subset colum... pandas.pydata.org](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop_duplicates.html)

df.drop\_duplicates()

drop\_duplicates라는 함수만 붙여주면 쉽게 해결 되는 것을 볼 수 있습니다.

여기까지를 외부에서 기존의 Data\_Frame을 가지고 와서 새로운 파일 경로의 값을 이어붙이는 하나의 독립된 함수로 만 들면 아래와 같이 만들 수 있습니다.

def Excel\_M(file,SPath,df): l\_row=0 df1=pd.read\_excel(SPath + "/" + file, header=None) for x in df1.index: if not pd.isna(df1.values[x,df1.columns.size-1]) : l\_row = x break df1=pd.read\_excel(SPath + "/" + file, skiprows

= l\_row) frames = pd.concat([df1, df]).drop\_duplicates() return frames

여기서 file은 파일 이름, SPath는 작업할 파일이 있는 폴더 경고, df는 이전에 만들어 두었던 DataFrame을 의미합니다.

Youtube 영상을 제작하면서 만든 파일은 아래와 같습니다. (위의 예제와 일부 변수명이 다를 수 있습니다.

**첨부파일** Excel\_M.py [파일 다운로드](http://blogattach.naver.net/5cc940f0e2bdb86448aacdffc322582487d028cc7a/20210613_211_blogfile/550sn_1623560435499_E2xA7Z_py/%2FExcel_M.py)

이제 GUI 영역에 이와 관련된 코드를 추가하면 다음과 같이 2개 파일에 추가할 내용을 정리할 수 있습니다.

※ 아래 소스는 Tricks-Office 전체 소스중 금번에 추가된 부분 일부만 가져온 내용입니다.

1. GUI 영역

from tkinter import \* from tkinter import filedialog from tkinter import messagebox import pandas as pd

import Excel\_F import re import os # 엑셀 데이터 합치기 함수 def Excel\_Merge(): files = [f for f in os.listdir(작업 대상폴더경로) if re.match('.\*[.]xls', f)] elist=[] dfs=pd.DataFrame(elist) for file in files: dfs = Excel\_F.Excel\_M(file, 작업대상폴더경로, dfs) dfs.to\_excel(결과저장경로 + "/" + "Merg\_Result.xlsx",index=False) messagebox.showinfo(title="Finish", message="Job Finished!") return # 창 호출 & 초기값 설정 win = Tk() win.geometry("600x800") win.title('Tricks\_Office') # frameExcel 실행 버튼 설정 (엑셀전용) frameXl = Frame(win, padx=10, pady=10,bd=1,relief=SOLID) frameXl.pack() # 엑셀 표기 Label(frameXl,text = "Excel 전 용").grid(row=0,column=0) # 엑셀 파일 병합 MXlbtn = Button(frameXl, text="Excel Merge",width = 21, height=2,command=Excel\_Merge) MXlbtn.grid(row=1, column=0,padx=5) MXllbl = Label(frameXl, text="대 상 폴더에 있는 엑셀 파일을 Merge", width=50,height=2,relief="groove") MXllbl.grid(row=1, column=1)

1. 함수 영역

import pandas as pd # 원본 Dataframe에 추가로 읽어온 엑셀 파일의 Data Frame을 병합 def Excel\_M(file,SPath,df): l\_row=0 df1=pd.read\_excel(SPath + "/" + file, header=None) for x in df1.index: if not pd.isna(df1.values[x,df1.columns.size-1]) : l\_row = x break df1=pd.read\_excel(SPath + "/" + file, skiprows = l\_row) frames = pd.concat([df1, df]).drop\_duplicates() return frames

이상의 기능을 포함하여 Tricks-Office 전체 소스는 다음과 같습니다. **첨부파일**

Tricks\_Office\_210610.zip [파일 다운로드](http://blogattach.naver.net/891c9521360203b69f791b2b11f788f25703f77d/20210612_276_blogfile/550sn_1623473232597_86r70f_zip/%2FTricks_Office_210610.zip)

전체 기능을 포함하여 exe 파일로 만들어진 자료는 아래의 링크에서 다운로드 받으실 수 있습니다.

### [파이썬 나만의 RPA 툴 Tricks\_Office](https://tricks-office.com/python-tricks_office/)

[파이썬으로 만든 나만의 RPA 툴 “Tricks\_Office” 를 공유합니다. 구현한 기능은 다음과 같습니다. 여러 PPT 파일의 여러 단어(Excel 형태로 보관된 Dictionary 파일 기준)를 원하는 단어로 치환하기양식이 맞춰진 Excel 파일을 하나의 파일로 취합하기하나의 양식은 아니지만, 유사한 양식의 엑셀 파일 데이터…](https://tricks-office.com/python-tricks_office/)

[tricks-office.com](https://tricks-office.com/python-tricks_office/)

# 파이썬 엑셀 데이터 자동 정리 (스마트 스토어 양식 기준)

## Python

2021/07/15 06:50 <http://blog.naver.com/550sn/222395307498>



엑셀 데이터를 정리하다 보면 같은 형식의 파일을 취합할 때도 있지만, 유사한 형식의 파일을 병합하여 자료를 만들어야 할 때가 있습니다.

예를 들어 여러군데의 스마트 스토어가 있고, 각 스마트 스토어에서 데이터를 주고 받기 위한 양식이 조금씩 차이가 있는 경우를 생각해 보겠습니다.

**첨부파일** 네이버.xlsx [파일 다운로드](http://blogattach.naver.net/42d75eeefba3a67a56b4d3e1dd3c463a99ce30d4e5/20210612_252_blogfile/550sn_1623473671351_u5EVXP_xlsx/%2F%B3%D7%C0%CC%B9%F6.xlsx) **첨부파일** 사이소.xls [파일 다운로드](http://blogattach.naver.net/35a0299a8bd1d10d21c3a496aa4b314deeb947a2f3/20210612_115_blogfile/550sn_1623473679136_he8fF8_xls/%2F%BB%E7%C0%CC%BC%D2.xls) **첨부파일** 옥션.xls

[파일 다운로드](http://blogattach.naver.net/2fba338097c9cb173bd9be8cb0512b57f4a35dbf72/20210612_9_blogfile/550sn_1623473688863_mbI0Ao_xls/%2F%BF%C1%BC%C7.xls)

각각의 양식이 요구하는 값들은 비슷하지만, 머리글 용어가 다른 경우가 있고, 전체 데이터 중 내게 필요한 데이터는 한정 적일 경우가 있습니다.

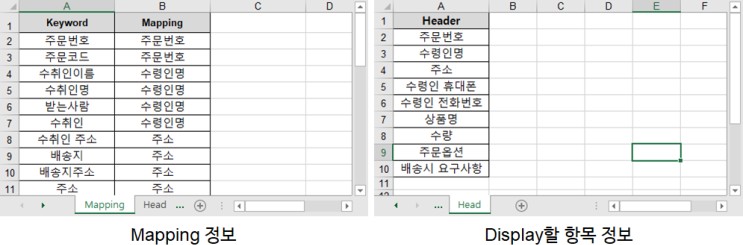
이때 내게 필요한 데이터만, 머리글 용어를 같게 통일해 주고 취합해 오는 작업을 해 보겠습니다.

앞서 구현한 엑셀 머지 기능을 거의 그대로 이용할건데, 2가지 기능을 추가해야 합니다.

1. 머리글행의 이름을 mapping 정보에 따라 재정리 해 줘야 합니다.
2. 내가 보고 싶은 항목만 뽑아서 볼 수 있게 정리해 줘야 합니다.

상기 2가지 내용을 구현하기 위해서 mapping 정보와 보고자 하는 항목 정보를 가지고 있는 Format 파일을 하나 만들어 두겠습니다.

**첨부파일** Total\_Order\_Format.xlsx [파일 다운로드](http://blogattach.naver.net/1b8e07b4a1fdff230fed8ab884651f63c097698815/20210612_9_blogfile/550sn_1623474090749_O0Q74b_xlsx/%2FTotal_Order_Format.xlsx)



Mapping 정보에서는 왼쪽 "Keyword"칸에 있는 머리말이 있을 경우 오른쪽 "Mapping"칸에 있는 머리말로 바꿔주는 작업을 하게 할 것입니다.

Display할 항목 정보에 표기된 순서대로 각각의 머리말에 해당하는 값을 모아서 하나의 파일로 정리하여 보여주게 될 것 입니다.

먼저 머리글 이름을 바꾸기 위해 이와 관련된 내용을 찾아 보겠습니다. 구글에서 "pandas column rename"으로 검색을 합니다. [**pandas.DataFrame.rename — pandas 1.2.4 documentation**](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.rename.html)

[pandas.DataFrame.rename DataFrame. rename ( mapper = None , index = None , columns = None , axis = None , copy = True , inplace = False , level = None , errors = 'ignore' ) [source] Alter axes labels. Function / dict values must be unique (1-to-1). Labels not contained in a dict / Series will be lef...](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.rename.html)

[pandas.pydata.org](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.rename.html) df.rename(columns={"A": "a", "B": "c"})

위와 같은 한줄 문법으로 Rename이 가능한 것을 확인했습니다. 해당 부분만 for 문으로 감싸주면 되겠습니다.

그 다음 알아볼 내용은 내게 필요한 값만 가져오는 부분입니다. 구글에서 "pandas select columns"로 검색을 합니다.



### [Selecting multiple columns in a Pandas dataframe](https://stackoverflow.com/questions/11285613/selecting-multiple-columns-in-a-pandas-dataframe)

[I have data in different columns, but I don't know how to extract it to save it in another variable. index a b c 1 2 3 4 2 3 4 5 How do I select 'a', 'b' and save it in to d...](https://stackoverflow.com/questions/11285613/selecting-multiple-columns-in-a-pandas-dataframe)

[stackoverflow.com](https://stackoverflow.com/questions/11285613/selecting-multiple-columns-in-a-pandas-dataframe)

columns = ['b', 'c'] df1 = pd.DataFrame(df, columns=columns)

그리고, 반복문으로 해당 내용을 감싸주기 위하여 DataFrame에서 줄을 증가시켜가면서 값을 가져오는 로직을 찾아봅니 다.

구글에서 "python pandas loop through rows and columns"로 검색합니다. [**How to iterate over rows in a DataFrame in Pandas**](https://stackoverflow.com/questions/16476924/how-to-iterate-over-rows-in-a-dataframe-in-pandas)

[I have a DataFrame from Pandas: import pandas as pd inp = [{'c1':10, 'c2':100}, {'c1':11,'c2':110},](https://stackoverflow.com/questions/16476924/how-to-iterate-over-rows-in-a-dataframe-in-pandas)

[{'c1':12,'c2':120}] df = pd.DataFrame(inp) print df Output: c1 c2 0 10 100 1 11 110 2... stackoverflow.com](https://stackoverflow.com/questions/16476924/how-to-iterate-over-rows-in-a-dataframe-in-pandas)

for index, row in df.iterrows(): print(row['c1'], row['c2'])

이전에 만들었던 엑셀에서 필요한 값을 가져오는 로직과 합쳐서 아래와 같이 정리할 수 있습니다.

def Excel\_ReA(file,SPath,dfMap,dfH,dfs): l\_row=0 df1=pd.read\_excel(SPath + "/" + file, header=None) for x in df1.index: if not pd.isna(df1.values[x,df1.columns.size-1]) : l\_row = x break df1=pd.read\_excel(SPath + "/"

+ file, skiprows = l\_row) for index, row in dfMap.iterrows(): df1.rename(columns = {row[0]:row[1]}, inplace = True) frames = pd.DataFrame(df1,columns=dfH.Header) frames = pd.concat([dfs, frames]).drop\_duplicates() return frames

여기서 dfMap은 앞서 본 mapping table의 값을 Data frame의 형태로 가져온 값이고, dfH는 보고자 하는 항목의 머리 글행 정보를 가져온 값입니다.

만들어진 함수를 불러오는 GUI 관련 화면은 아래의 로직으로 구현할 수 있습니다.

from tkinter import \* from tkinter import filedialog import pandas as pd import os from tkinter import messagebox import Excel\_F import re #Excel 파일 양식에 따라 병합하기 def Excel\_ReArg(): files = [f for f in os.listdir("작업할 파일 폴더경로") if re.match('.\*[.]xls', f)] dfMap = pd.read\_excel("양식 파일 경로", sheet\_name='Mapping') dfH = pd.read\_excel("양식 파일 경로", sheet\_name='Head') elist = [] dfs = pd.DataFrame(elist) for file in files: dfs = Excel\_F.Excel\_ReA(file,"작업할 파일 폴더경로",dfMap,dfH,dfs) dfs.to\_excel("저장할 파일 폴더경로" + "/" + "ReArrange\_Result.xlsx",index=False) messagebox.showinfo(title="Finish", message="Job Finished!") return # 창 호출 & 초기값 설정 win = Tk() win.geometry("600x800") win.title('Tricks\_Office') # 엑셀 표기 Label(frameXl,text = "Excel 전 용").grid(row=0,column=0) # 엑셀 파일 양식에 맞춰서 재배열 병합 RAXlbtn = Button(frameXl, text="Excel ReArrange",width = 21, height=2,command=Excel\_ReArg) RAXlbtn.grid(row=2, column=0,padx=5) RAXllbl

= Label(frameXl, text="양식에 맞춰서 엑셀 파일을 재정렬 후 병합 \n" "Format 파일이 지정되어 있어야 합니다.", width=50,height=2,relief="groove") RAXllbl.grid(row=2, column=1)

상기 내용 포함하여 완료된 소스는 다음과 같습니다. **첨부파일**

Tricks\_Office\_210610.zip

[파일 다운로드](http://blogattach.naver.net/cf5ad360702b2bf7db395e6c50b1cbb71443bc5dea/20210612_93_blogfile/550sn_1623483321232_5SBOth_zip/%2FTricks_Office_210610.zip)

※ Youtube 에서 다룬 소스는 다음과 같습니다. **첨부파일**

Excel\_M.py [파일 다운로드](http://blogattach.naver.net/e772fb415a6c6dd8f11775457f99e69c396a9635/20210614_233_blogfile/550sn_1623608103854_HYx3zj_py/%2FExcel_M.py)

Tricks-Office 전체 실행 파일 및 관련 내용은 아래 링크에서 확인이 가능합니다.

### [파이썬 나만의 RPA 툴 Tricks\_Office](https://tricks-office.com/python-tricks_office/)

[파이썬으로 만든 나만의 RPA 툴 “Tricks\_Office” 를 공유합니다. 구현한 기능은 다음과 같습니다. 여러 PPT 파일의 여러 단어(Excel 형태로 보관된 Dictionary 파일 기준)를 원하는 단어로 치환하기양식이 맞춰진 Excel 파일을 하나의 파일로 취합하기하나의 양식은 아니지만, 유사한 양식의 엑셀 파일 데이터…](https://tricks-office.com/python-tricks_office/)

[tricks-office.com](https://tricks-office.com/python-tricks_office/)

# 프로젝트를 다룬다면, Git!

## Python

2021/07/22 06:50 <http://blog.naver.com/550sn/222411855782>



금번에 개인 RPA 과제 Tricks-Office를 만들면서 가장 어려웠던 점 중 하나는 소스를 관리하는 부분이었습니다. 직장인의 신분으로 취미 생활로 별도 프로젝트 과제를 만들다 보니 짬짬이 시간을 내서 코딩을 할 수 밖에 없었고, 집에서 는 아이들의 과제로 인해 한번씩 PC사용권을 빼앗기면 다른곳에 아이디어를 기록해 두거나 따로 소스를 만들었다가도 다 시 합치는데 고생을 많이 했습니다.

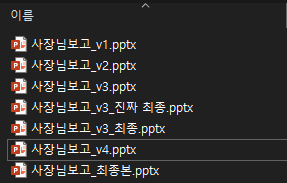
그 와중에 중간에는 배포 사고(제대로 실행되지 않는 실행파일을 업데이트하는)도 한번 발생 했었고요... ㅠ.ㅠ

이와 같은 현상은 IT 시스템을 개발하는 곳에서는 수시로 발생할 수 있는 일이고, 일반적으로 프로그래밍을 하는 사람들이 조심하는 부분 중 하나입니다.

오늘은 이를 위해 활용되는 대표적인 툴인 Git에 대하여 한번 알아볼까 합니다.

1. Git의 탄생 배경

우리가 일반적인 업무를 할때를 생각해 보겠습니다.

보고서를 만들때 임원진에게 보고해야 하는 중요 문서의 경우에는 대개 여러번의 수정과정을 거치고, 그 과정을 여러 버전 의 파일로 보관해 두곤 합니다.

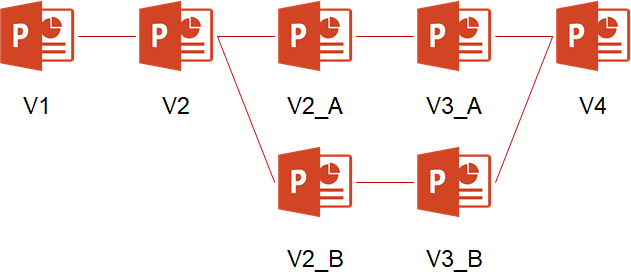
그 과정에서 여러번의 변경 과정을 만들면서 비슷한 파일이 여러개가 파상되어 나오게 되고, 나중의, 나중의 나중에는 어 떤 파일이 진짜 최종본인지 중간에 어떤 내용들이 어떻게 바뀌었는지가 헷깔리게 되곤 합니다.

그러면 최종 버전 단 하나의 파일만 남기면 되지 않느냐? 라고 할 수도 있지만, 그러지 못하고 여러개의 버전 파일로 만들 어서 관리하는 이유는 수정하는 과정에서 오히려 잘못수정하여 이전의 자료로 되돌아 가야 할 때도 있고, 경우에 따라서는 1안, 2안과 같이 복수의 선택안에 대하여 정리하여 준비하다가 최종안을 결정하는 경우도 있기 때문입니다.

이와 같은 현상은 문서작업 뿐만 아니라 프로그램 개발 과정에서도 수시로 일어나게 됩니다.

상황이 이러하다 보니 대개의 경우 우리는 특정한 파일 이름에 대한 네이밍 룰을 정해두고, 반복해서 파일을 저장하면서

이를 관리하는 것이 일반적인 업무용 문서 관리 방법이었습니다.



이와 같은 방법은 그래도 약속된 룰만 잘 지킨다면 이력은 관리가 되겠지만, 몇가지 문제점이 있을 수 밖에 없습니다.

1. 같은 파일을 변경된 내용 포함하여 반복 저장하는 형태이기 때문에 저장 공간을 많이 차지한다.
2. 버전 이름만으로는 중간에 어떤 변화가 어떻게 일어났는지를 알기가 어렵고 이력 관리가 어렵다.
3. A안, B안과 같이 분리하여 작업한 내용을 합칠때 상호 변경된 내용을 추적하여 합치기 어렵다.

IT 프로젝트 즉 프로그램을 만드는 작업을 할 때는 위의 내용과 유사한 문제가 늘, 자주 발생하게 되다 보니 이를 관리하기 위한 별도의 툴을 만들어 낸 것이 Version Control System(VCS)이며, 가장 인기 있는 VCS 툴 중 하나가 바로 Git 입니 다.

Git에서는

1. 프로젝트 내에서 변경된 내용만 저장하여 관리하기 때문에 저장 공간을 많이 차지 하지 않고
2. 중간 중간에 Git에 저장하는 과정에서는 상세 설명 입력이 의무화 되어서 이력 관리가 용이하고
3. Branch 개념이 있어서 분리하여 작업하고 각자의 작업을 합칠때 상호 비교하는 기능이 제공되어 편리합니다.

앞서 살펴본 3가지 어려운 점을 해결할 수 있는 도구로써 기능을 제공하고 있지만, 역시 기능에 대한 사용법을 알고 써야 그 효익을 충분히 누릴 수 있을듯 합니다.

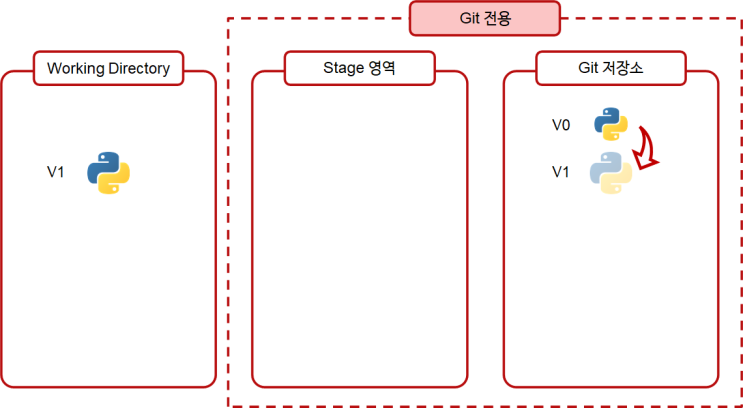
1. Git의 기본 구동 방식

Git을 사용할 때 한가지는 꼭 개념을 잡고 넘어가야 합니다. Git에서 파일을 인식하고 저장하고 관리하는 방식입니다. 우 리가 그냥 파일로 인식하는 문서들을 3단계로 나누어서 Git은 추적 관리하는 것을 원칙으로 하고 있습니다.

우리가 일반적인 파일을 저장하고, 다른 프로그램에서도 같이 활용하는 Working Directory라는 영역이 있고, Gi 에서만

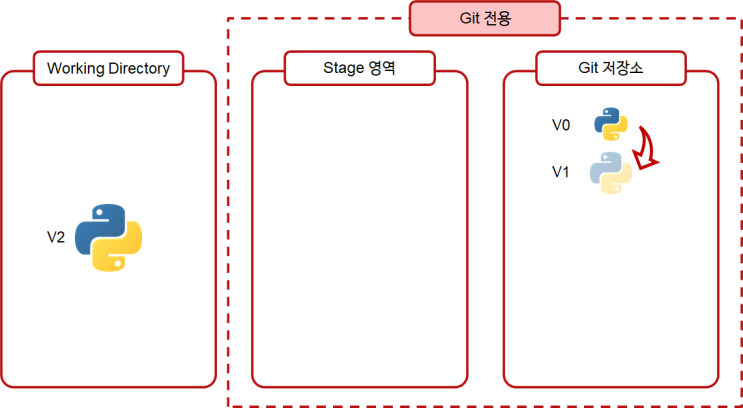
관리하는 Stage 라는 영역과 Git 저장소 영역이 따로 존재합니다.

Git 저장소에서는 우리가 작업한 결과파일을 보관하고 있는데, 저장하는 형태가 조금 다릅니다.

현재 파일의 형태를 그대로 저장하고 있는것이 아니고, 우리가 Git에 저장을 해 줄 때 마다 그때 그때 변화한 내용에 대한 내용만을 추가로 저장하는 형태로 저장됩니다.

위의 이미지 상으로 볼 때, Working Directory에서는 V1 이라는 현재 버전의 파일의 형태로 저장되어 있고, 인식하는 반 면 Git 저장소에서는 최초 파일 생성 되었을때의 모습인 V0이라는 모습+ 변경된 내용 에 대한 정보 형태로 저장된다고 볼 수 있습니다.

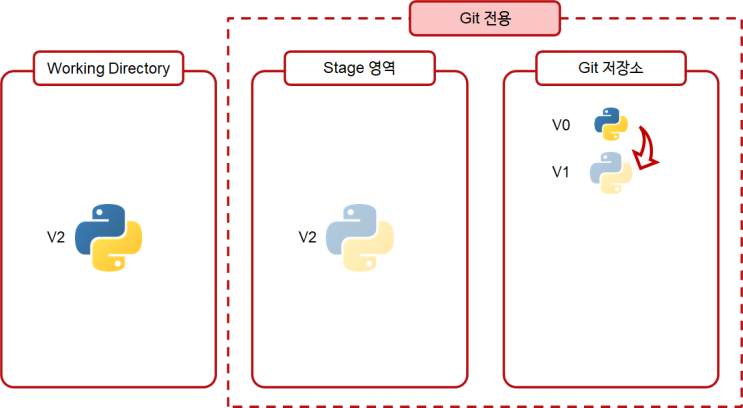
어떤 작업을 통해 실제 파일의 내용이 변경되었다고 하면,

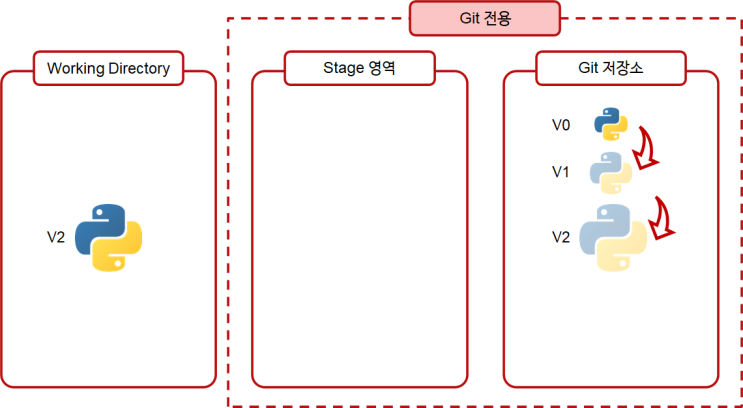


이때 변경내용을 Git에서도 인식시키고 저장하기 위해서는 Working Directory와 Git 저장소의 중간에 변경된 내용을 업 데이트하기 위한 별도의 공간인 Stage영역이 있어서, Git에 저장을해 줄 때는 이 Stage 영역에서 비교, 확인 후에 필요한 정보를 업데이트 하게 됩니다.

그렇다 보니 Git에 저장을 해 두는 방법은 조금 번거로울 수 있습니다.

먼저 Stage 영역에 변경되고 수정된 파일들의 내용을 추적 관리하게 올려주고,



Git 저장소에 저장되게 다시 한번 저장을 해 주는 방식으로 저장되고 관리된다고 보면 되겠습니다.

여기까지 기본 개념, 활용되는 로직만 보면 보기에 따라서는 오히려 일반적인 파일로 작업 할 때 보다 번거롭게 느꼊질 수 도 있습니다.

아니, 실제로 처음 사용해 보면 상당히 번거로운 요소가 분명히 있습니다.

그럼에도 이와 같은 툴을 사용하는 이유는 복잡하고 여러번의 변경을 거치게 되는 프로젝트를 다루게 될 경우 어느정도의 진도가 나가고 난 다음에 이력을 추적하거나, 과거 버전으로 오히려 되돌아 가거나, Case별로 케이스를 만들어 추적 하는 등의 특이 작업이 생겼을때 이를 보다 편하고 정확하게 작업을 할 수 있기 때문입니다.

1. Git 설치 하기

Git을 설치하기 위해서는 아래 링크의 Git 사이트로 접속하여 해당 S/W를 다운로드 받아서 설치하면 됩니다. [**Git**](https://git-scm.com/)

[About The advantages of Git compared to other source control systems. Documentation Command reference pages, Pro Git book content, videos and other material. Downloads GUI clients and binary releases for all major platforms. Community Get involved! Bug reporting, mailing list, chat, development and ...](https://git-scm.com/)

[git-scm.com](https://git-scm.com/)

그리고, Git의 기본 사용 방법 자체는 터미널에 직접 입력하는 방식인데, 이 부분은 아무래도 번거로움이 있으니 GUI 화면 으로 볼 수 있는 솔루션을 설치하는 것을 추천합니다.

대표적인 Git GUI 툴 중 하나인 Sourcetree를 아래 링크를 통해 설치하시면 사용이 가능합니다.



### [Sourcetree | Free Git GUI for Mac and Windows](https://www.sourcetreeapp.com/)

[A Git GUI that offers a visual representation of your repositories. Sourcetree is a free Git client for Windows and Mac.](https://www.sourcetreeapp.com/)

[www.sourcetreeapp.com](https://www.sourcetreeapp.com/)

이들 S/W들은 참으로 고맙게도 무료로 사용이 가능하게 배포되어 있습니다.

1. Git 사용해 보기

사용 방법이... 글로 설명하려니 어렵네요...

아래의 유튜브 영상을 참고해 주시기 바랍니다.

### [YouTube](https://youtu.be/i2MACxpLYhQ)

[YouTube에서 마음에 드는 동영상과 음악을 감상하고, 직접 만든 콘텐츠를 업로드하여 친구, 가족뿐 아니라 전 세계 사람 들과 콘텐츠를 공유할 수 있습니다.](https://youtu.be/i2MACxpLYhQ)

[youtu.be](https://youtu.be/i2MACxpLYhQ)

1. 참고 영상 및 자료

Git은 이번에 제가 개인 Project를 진행하면서 필요에 의해 접하고 공부하기 시작한 부분으로 현재 저는 아주 기초적인 내 용만 이해하고 있습니다. 제가 Git을 공부하기 위해 구매해서 읽은 책과 찾아본 영상도 추천 드립니다.

1. 읽은 책

아래 링크의 Git 교과서 라는 책을 보면서 공부했습니다.

솔직히... 읽다가 많이 졸기는 했습니다. 실제로 실습을 해 보면서 따라해 보시길 추천합니다.

### [Git 교과서](https://ridibooks.com/books/754028387?_s=search&_q=git%20%EA%B5%90%EA%B3%BC%EC%84%9C&_rdt_sid=search&_rdt_idx=0)

[기본기를 탄탄하게 다지는 깃(git) 입문서도해와 실습으로 깃, 소스트리, 깃허브를 배우자!버전 관리 시스템(VCS)은 파일 의 변화를 시간에 따라 기록했다가 나중에 특정 시점의 버전을 다시 꺼내올 수 있는 시스템이다. VCS를 사용하면 파일이 나 프로젝트를 이전 상태로 되돌릴 수 있고, 시간에 따라 수정 내용을 비교해볼 수도 있다. 또한, 문제가 있을 때 누... ridibooks.com](https://ridibooks.com/books/754028387?_s=search&_q=git%20%EA%B5%90%EA%B3%BC%EC%84%9C&_rdt_sid=search&_rdt_idx=0)

처음 Git에 대한 개념에서 부터 기초, 고급 응용 기능까지 두루두루 자세하게 설명이 되어 있어서 참고하기 좋습니다.

1. 참고 영상

우선 내게 필요한 기능들만 익혀서 사용한다면, 아래의 영상을 추천합니다.

타임캡슐과 평행세계로의 이동으로 표현한 부분이 인상적인데, 기본 사용법을 정말 잘 설명해 주시고 있습니다.

그리고, 최근 즐겨 보기 시작한 유튜버 드림코딩 by 엘리님의 영상도 기본 개념을 잡는데 정말 큰 도움이 됩니다.

프로그래머가 아닌 저 또한 약간 규모가 큰 프로젝트를 다루기 위해서는 Git을 익히고 사용하는데, 프로그래머를 꿈꾸시 는 분이라면 Git은 선택이 아니라 필수가 아닐까 하는 생각을 해 봅니다.

# IT 협업을 한다면, GitHub

## Python

2021/07/29 06:50 <http://blog.naver.com/550sn/222412031843>



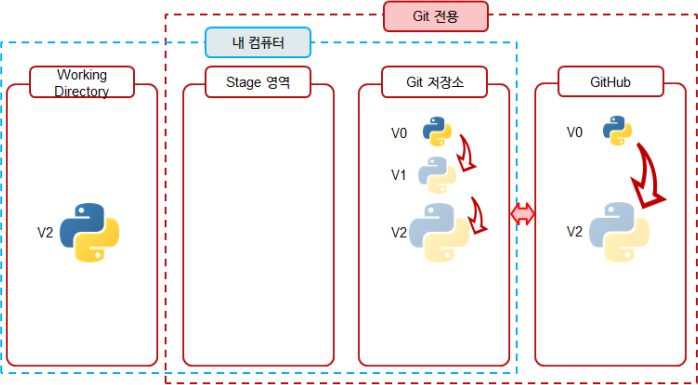
우리는 앞선 글에서 버전 관리 툴인 Git에 대하여 알아 봤습니다. 오늘은 여러사람이 동시 작업시 소스 관리를 도와줄 수 있는 툴인 GitHub에 대해 알아보도록 하겠습니다.

1. 기본 개념

Git과 GitHub는 약간은 다른 개념입니다.

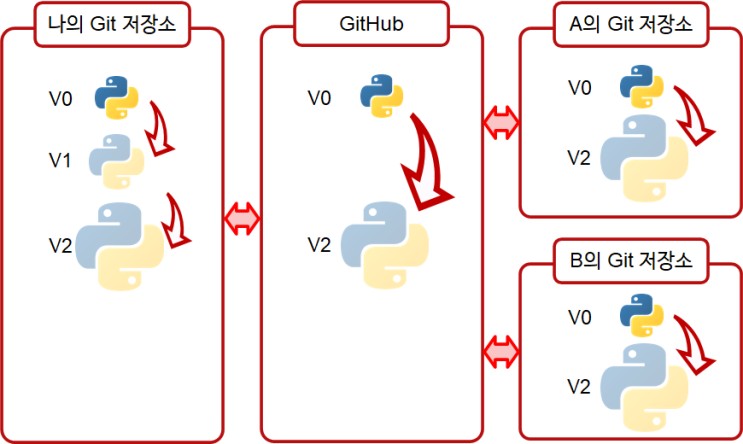
물롯 GitHub가 Git을 Base로 두고 운영되는 부분이 있기 때문에 두가지 개념을 떼어놓고 볼 수는 없지만, 같지는 않습니 다.

Git이 내 컴퓨터 내에서 내 프로젝트를 관리하는 방식이라면, GitHub는 서버 공간에서 내 프로젝트를 관리하는 방식이라 고 할 수 있습니다.



이때 Git 저장소와 GitHub가 나의 변경 History에 대하여 인식하고 저장하는 기본 방법은 동일하며, 내 컴퓨터 안의 Git 저장소와 GitHub의 내용을 비교하면서 변동된 내용을 추적 저장하는 방식으로 소스가 관리됩니다.

이렇게 GitHub에 저장된 소스파일은 GitHub에 접근이 허용된 모든 인원이 해당 소스파일과 소스 이력에 접근을 할 수 있 고, 그 결과에 대해서도 공유가 가능합니다.



물론, 나의 Git 저장소에 있는 내용이 아니라 GitHub에 동기화된 내용 기준으로요.

따라서 내가 만든 소스 파일을 함께 나눠 보고 싶은 상황이 있다면, 이 GitHub를 활용하면, 훨씬편하게 공유가 가능할 뿐 만 아니라, 변동상황이 발생하더라도, 서로 쉽게 인식하고 반영할 수 있습니다.

협업이 필요할 경우에는 이 GitHub에 접근 권한을 열어서 수정이 가능하게 해 두면 바뀐 내용에 대한 추적 관리가 가능합 니다.

1. 사용 방법

아래 링크의 GitHub 사이트에서 가입하면 GitHub의 사용이 가능합니다.



### [GitHub: Where the world builds software](https://github.com/)

[GitHub is where over 65 million developers shape the future of software, together. Contribute to the open source community, manage your Git repositories, review code like a pro, track bugs and feat...](https://github.com/)

[github.com](https://github.com/)

그리고, 이곳에 Repository라는 공간을 만들어 두면 Hub 공간이 추가되고 사용이 가능합니다.

1. 실제 사용해 보기

실제 사용 방법에 대해서 글로 정리하려 하니 꽤 까다롭네요. 아래 영상을 참고하시기 바랍니다.

### [YouTube](https://youtu.be/Vt5LoxTPmCw)

[YouTube에서 마음에 드는 동영상과 음악을 감상하고, 직접 만든 콘텐츠를 업로드하여 친구, 가족뿐 아니라 전 세계 사람 들과 콘텐츠를 공유할 수 있습니다.](https://youtu.be/Vt5LoxTPmCw)

[youtu.be](https://youtu.be/Vt5LoxTPmCw)

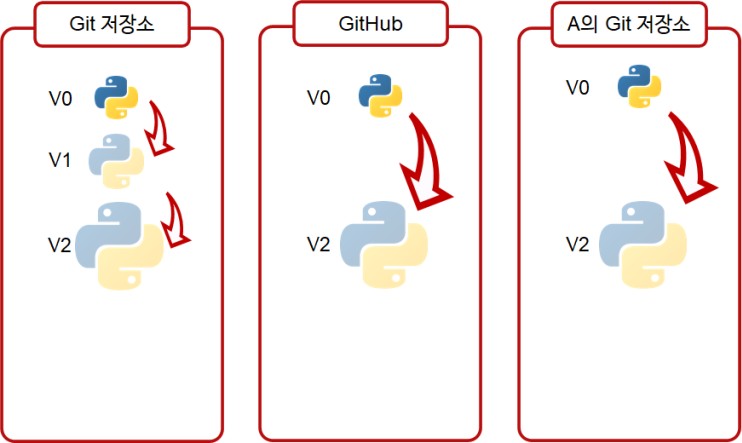
1. 충돌 발생과 해결하기

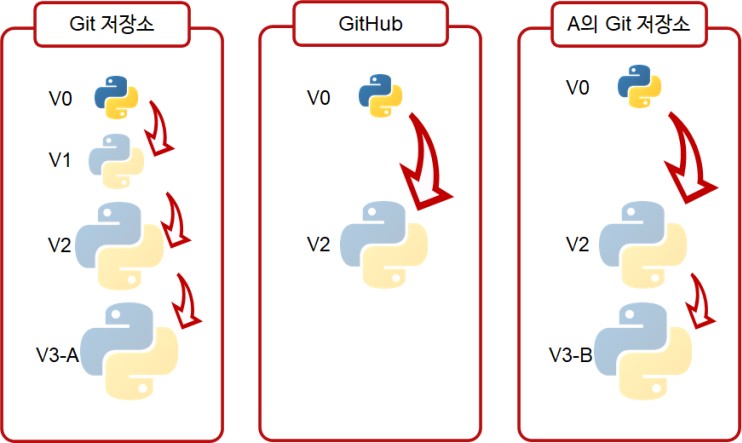
영상에서도 다루었지만, 공동 작업을 할때 충돌 발생과 이에 대한 해결은 가장 중요한 이슈가 아닐까 하는 생각을 해 봅니 다.

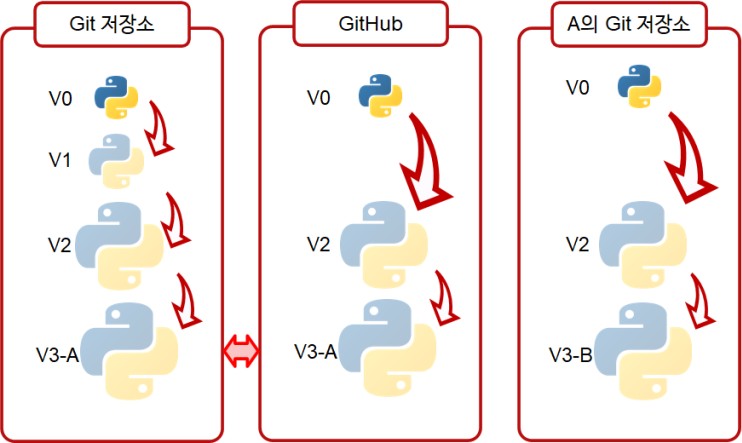
하나의 프로젝트를 여러사람이 함께 작업하다 보면 같은 파일의 같은 부분을 수정하는 일이 발생할 수 있습니다.

이때 Git에서는 충돌 이벤트를 발생시켜서 해당 부분을 정확히 확인하고 수정한 다음에야 다음 작업을 진행할 수 있게 기 능을 지원해 줍니다.

대표적으로 충돌이 발생하는 케이스를 살펴보면 다음과 같은 경우를 생각해 볼 수 있겠습니다. 금번 프로젝트를 A라는 동료와 함께 작업을 하는 경우를 생각해 보겠습니다.

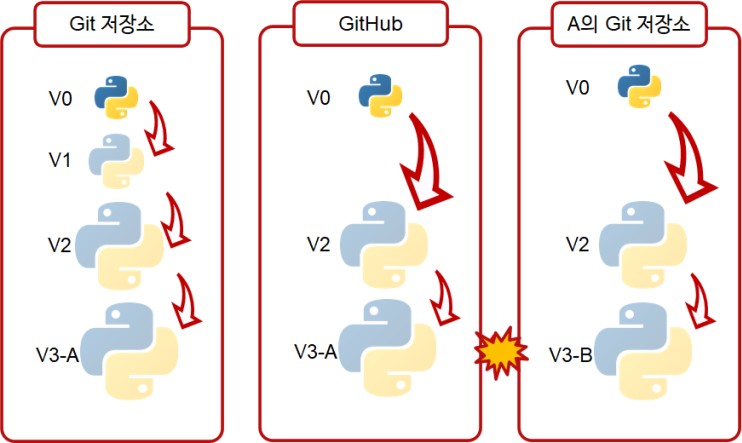


V2 파일을 수정할 일이 있어서 수정을 했는데, 공교롭게도 제가 수정할때 동일한 시점에 A라는 동료도 같은 파일을 다른 방식으로 수정했다고 가정하겠습니다.

그리고, 이어서 제가 만든 Git 저장소의 내용을 GitHub로 Push를 하면 정상적으로 Push가 잘 이루어 집니다. 이를 통해 내가 작업한 내용이 GitHub에 올라가게 됩니다.

문제는 그 다음에 발생합니다.

제가 해당하는 내용을 GitHub에 Push하고 난 다음, 동료 A가 자신이 수정한 내용을 GitHub에 Push하면 이 내용은 충돌 이 나면서 제대로 작동하지 않습니다.



지금 GitHub에 올라와 있는 내용은 V2에서 V3-A로 변경되는 변경 내용을 가지고 있기 때문에 그 다음에 업데이트되어 야 하는 내용은 V3-A에서 다음 단계로 변경되는 내용이 업데이트 되어야 합니다. 이에 비해 동료 A의 Git 저장소에 저장 되어 있는 내용은 V2에서 V3-B로 변경되는 내용을 가지고 있기 때문에, Git Hub 입장에서는 어느 변화를 받아들여야 할 지가 충돌되는 상태에 있는 것입니다.

그래서 이와 같은 일이 발생할 때는 다시 한번 더 Pull으로 최신 정보를 가져온 다음 방금 커밋한 내용과의 차이점을 확인 하고 수정한 다음, 다시 커밋하고 Push 해야 합니다.

1. 팀 업무시 권장하는 작업 방법 & 순서추천

팀 업무를 GitHub를 통하여 작업을 할 경우, 충돌과 같은 최악의 상황이 발생했을때 원만한 해결책을 제공하고는 있지만, 사실 충돌이 발생했을 때 이를 해결하는 것은 항상 꽤 많은 시간을 소요하고 이슈가 발생하기 가장 쉬운 Weakpoint임에 는 틀림 없습니다.

따라서 가능하면 이와 같은 충돌이 발생하는 것을 최소화 하게 일을 나눠서 진행하는 것이 보다 효율적인 방법일 것입니 다.

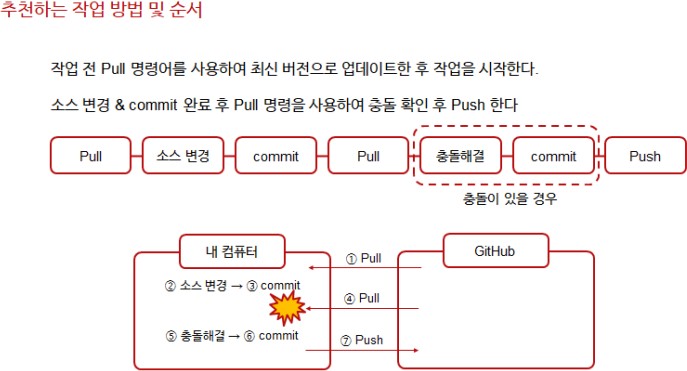
작업 방법 면에서 봤을 때

* 가능한한 프로젝트 멤버들 간 작업해야 하는 파일은 서로 겹치지 않게 나눠서 작업한다.

작업 순서로 봤을때

* 개선 작업전 GitHub에서 Pull로 최신 정보를 받아오고,
* Push로 내 저장소 정보를 업데이트 하기전에 한번 더 GitHub에서 Pull로 정보를 가져온 다음
* 충돌이 있으면 충돌을 해결한 당ㅁ에
* Push로 내 저장소 정보를 업데이트 한다

의 순서로 진행하는 것이 그래도 충돌을 최소화 하고, 작업을 효율적으로 진행할 수 있는 팁이 되지 않을까 생각해 봅니다.



GitHub는 저처럼 혼자서 필요한 것을 만드는 사람들에게는 크게 중요하지 않을 수도 있지만,(솔직히 제게는 소스 공유용 의 용도가 더 큽니다.) 협업이 필요한 사람들에게는 없어서는 안될 필수 툴이라고 생각됩니다.

오늘 제가 다룬 내용은 GitHub에 대한 기초중의 기초, 가장 기본 개념만 다룬것으로, 협업을 준비하시는 분이라면 별도로 Git을 추가공부해 보시길 추천합니다.

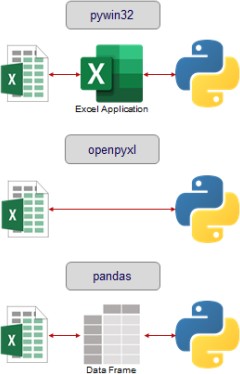
# 파이썬에서 엑셀을 사용하는 3가지 방법

## Python

2021/08/02 06:50 <http://blog.naver.com/550sn/222426744373>



파이썬에서 엑셀 파일을 다룰 수 있는 방법으로는 여러가지가 있습니다.



이전 영상에서도 이미 'openpyxl'을 활용하는 방법과 'pandas'를 활용하는 방법을 소개한 바 있습니다.



### [Excel을 삼킨 Python (openpyxl)](https://blog.naver.com/550sn/222336080574)

[Python의 가장 강력한 점 중의 하나는 필요한 기능이 있을때, 필요한 기능을 제공해 주는 Library만 설치... blog.naver.com](https://blog.naver.com/550sn/222336080574)

### [판다와 손잡은 파이썬(파이썬 엑셀 다루기)](https://blog.naver.com/550sn/222362866341)

[제가 파이썬을 다루면서 엑셀을 다루는 라이브러리로 openpyxl을 사용 했었는데, Python에 대해 공부를 해... blog.naver.com](https://blog.naver.com/550sn/222362866341)

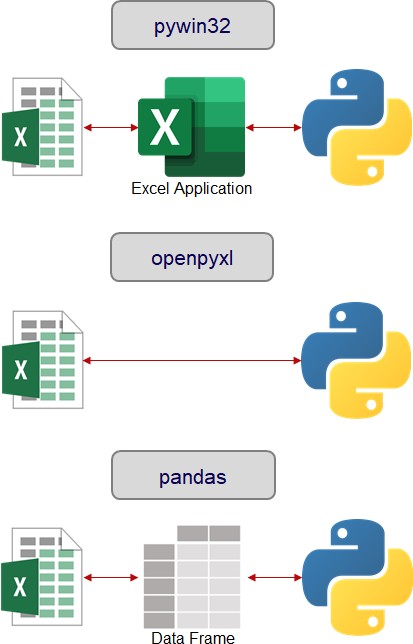
그리고, 엑셀로 다루지는 않았지만, 파워포인트에서 pywin32를 활용하는 방법에 대해서도 소개한 바 있습니다.



### [파이썬으로 PPT Font 통일하기](https://blog.naver.com/550sn/222384801239)

[제가 Youtube 채널을 시작했던 당시 처음으로 만들었던 PPT 매크로가 PPT 폰트 통일이었습니다. P... blog.naver.com](https://blog.naver.com/550sn/222384801239)

크게 보면 엑셀 파일을 다룰 수 있는 방법으로는 위의 3가지 형태가 있을듯 한데, 각각의 방식에 대해서 알아보도록 하겠 습니다.

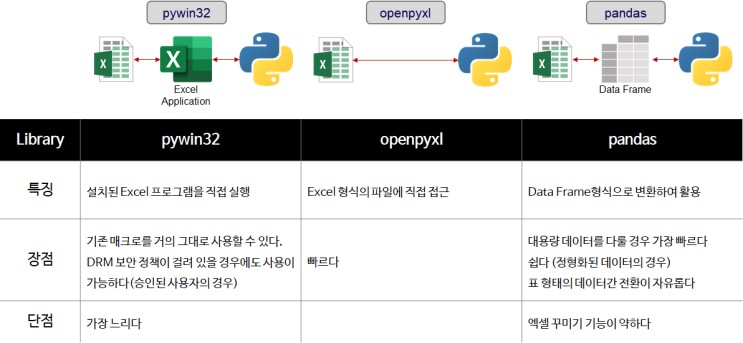


기본적으로 엑셀은 표 형태에 파일이고, 여기 접근하는 3가지 라이브러리는 각각의 다른 방법을 취하고 있습니다.

아예 엑셀 파일의 전용 프로그램인 Excel을 실행하고, 그 프로그램 내에서 명령을 하는 방식 (pywin32), Excel 파일에 직 접 접근하여 원하는 형태로 수정하는 방식(openpyxl), 컴퓨터가 쉽게 인식하는 변수 형태로 변환한 다음에 수정하는 방식 (pandas).

어떤 방식이 무조건 좋다고 만은 할 수 없고, 각각의 특징이 있기 때문에 내가 사용하고자 하는 환경에 맞춰서 사용하는 것 이 중요하겠습니다.

각각의 방식을 간단히 표로 비교하면 아래와 같이 표현이 가능합니다.



※ 파이썬에서 사용이 가능한 라이브러리는 위의 3가지 이외에도 더 있을 수 있지만, 엑셀 파일을 다루는 방식은 상기 3가 지 방법중 한가지 형태를 가지리라 생각됩니다.

1. 속도

pandas > openpyxl >> pywin32 순서로 생각하시면 됩니다.

pandas와 openpyxl는 둘다 엑셀 파일에 직접적으로 접근하기 때문에 불필요한 동작을 모두 배제하여 매우 빠릅니다. 대 용량 데이터로 가지 않는 경우(일반적인 사용자의 경우) pandas와 openpyxl의 속도 차이는 느끼기 어려울 정도로 둘다 매우 빠릅니다. 대용량 데이터를 다루면서 분석을 위한 복잡한 산식이 사용되게 된다면 속도 차이가 조금씩 발생하며, 이 때는 pandas가 가장 유리합니다.

pywin32의 경우에는 직접 Excel Application을 띄우고, 모든 엑셀 기능을 실행한 다음에 필요한 동작을 하므로 매우 느 립니다. 대용량 데이터 & 복잡한 계산식이 사용될 경우 pywin32를 사용하는 방식은 엑셀을 직접 활용하는 방법과 비교해 도 속도 면에서는 장점을 찾기 어렵습니다.

1. 꾸미기

pywin32 > openpyxl >> pandas 순서로 생각하시면 됩니다.

pandas는 보다 정확히 이야기 한다면 엑셀을 위한 라이브러리라기 보다는 표 형태의 데이터를 다루는데 특화된 라이브러 리라고 할 수 있습니다. 그렇다보니 데이터를 꾸미는데에는 전혀 관심이 없습니다. 셀을 병합하거나, 테두리선을 편집하거 나, 행과 열의 높이와 폭을 조정하는 등의 동작에는 별 기능을 제공하지 않습니다.

이에 비해 pywin32와 openpyxl의 경우에는 엑셀 파일을 직접 제어하는 형식이다 보니 이와 같은 꾸미기 기능들은 왠만 한 내용은 다 제공된다고 볼 수 있습니다.

1. 난이도

pandas >> openpyxl > pywin32 순으로 두겠습니다.

사실 난이도라는것이 결국 익숙해지기 나름이어서, 어느쪽이 더 쉽다고 콕찍어서 말하기는 어려울듯 합니다.

사용 목적에 따라서 사용하는 방법에 따라서 체감하는 난이도는 모두 다를것 같습니다.

하지만, 순수하게 데이터를 다루는데만 목적을 둔다면, 아무래도 컴퓨터에서 잘 인식하고 이해하는 변수형태로 변환한 다 음 데이터를 다루는 pandas가 아무래도 이후 데이터를 어떤 방식으로 인식 시키고 필요한 동작을 하게 할 것인가를 고민 할 수 밖에 없는 다른 두 방식에 비해서 월등히 데이터를 다루기 쉽다고 생각됩니다.

물론, 데이터가 표 형태로 정형화가 잘 되어 있다는(병합되어 있거나, 빈칸이 있는 등의 예외사항이 없다는) 전재 하에서 요.

다만, VBA 매크로 사용법이 익숙하신 분들에게는 pywin32의 사용법이나 문법 자체가 기존의 매크로와 유사한점이 많기 때문에 보다 쉽게 느껴질 수도 있습니다.

1. 범용성

pywin32 >> openpyxl > pandas 순으로 점수를 두겠습니다.

범용성이라는 항목 이름은 제가 임의로 정한 이름으로, 조금 특이한 엑셀 작업들을 할 때 대응이 가능하냐 여부의 점수라 고 생각해 주면 좋겠습니다.

먼저 엑셀파일 자체가 매크로 파일을 포함하는 파일의 경우를 생각해 보겠습니다.

pandas는 앞에서도 이야기 했다 시피 오로지 표 형태의 데이터에만 관심이 있기 때문에 VBA 매크로는 전혀 취급하지 않 습니다.

openpyxl의 경우 excel 파일 자체에 접근하기 때문에 이미 만들어져 있는 매크로 영역을 건드리지 않고 그대로 저장하는 것은 가능하지만, 해당 매크로를 실행하는 등의 동작은 어렵습니다. (따로 방법이 있을지도 모르겠지만, 저는 못찾았습니 다.)

pywin32의 경우 Excel 프로그램을 직접 실행시켜서 필요한 동작을 하기 때문에 매크로를 실행시킬 수도 있고, 이를 포함 하여 저장할 수도 있는등 자유로운 활용도를 보여줍니다. (속도를 버린 대신 엑셀의 모든것을 취한다고 보시면 됩니다.) DRM과 같은 보안 관점을 살펴 보겠습니다.

회사에 따라서는 사내 보안을 강화하기 위하여 DRM 솔루션을 도입하여 문서 보안을 설정하는 경우가 있습니다. 이때 이 와 같은 DRM 솔루션이 적용된 문서는 어떤 암호화 과정을 거쳐서 파일에의 접근과 수정이 가능하게 됩니다. 그리고, 이 암호화 과정은 대개 설치된 전용 프로그램, 즉, 엑셀 Application을 통해서 이루어지게 됩니다.

이 말은 pywin32의 방식을 제외한 나머지 2가지 방식으로는 파일의 접근 자체가 불가능하다는 말로 이해하시면 됩니다.

1. 활용 방법

3가지 라이브러리 중 어느 한가지만을 선택하여 활용할 필요는 없다고 생각합니다.

각각의 라이브러리가 가지는 특징과 장/단점을 이해하고, 필요에 따라서 조합하여 사용하는 것이 가장 좋은 방법이리라 생각됩니다.

엑셀에 내장된 매크로를 활용해야 한다거나 DRM이 걸려있는 경우라면,

pywin32 에서 데이터를 불러오고, 불러온 데이터를 pandas로 넣은 다음 시간이 많이 걸리는 대용량 데이터 가공 작업을 완료한 다음에 다시 pywin32로 마무리 하는 방식을 취할 수 있습니다.

특수 조건이 걸려있지는 않지만 보기좋게 편집하는 과정이 필요한 경우라면,

openpyxl만 활용해서 데이터를 수정할 수도 있고, pandas에서 데이터 가공 작업을 완료하고, openpyxl에서 마지막 가 공/편집 하는 과정을 추가하는 방식을 취할 수도 있습니다.

내게 주어진 해결해야하는 문제점이 무엇인가를 잘 파악하고, 그에 맞는 해결책을 찾아가시는 것을 추천합니다.

다만, 한가지 추천하는 것은 너무 어떤 툴에 연연하지는 않았으면 합니다.

세상에는 수많은 선량한 잔머리 대마왕들이 있고, 그들이 공유하는 잔머리 툴의 세계는 무궁무진합니다.

특정한 한 두가지 툴에 연연하다가 또 다른 유용한 툴을 활용해볼 기회를 놓치는 것은 꽤나 아까운 일이 아닐까 하는 생각 을 해 봅니다.

영상에서 사용된 엑셀 파일 및 소스는 다음과 같습니다. **첨부파일**

Excel\_Test.xlsx [파일 다운로드](http://blogattach.naver.net/51c44dfdefb5b56945a7c0f2ce2f55298adc23c083/20210711_27_blogfile/550sn_1625979258963_ipD36A_xlsx/%2FExcel_Test.xlsx)

#Pandas import pandas as pd df = pd.read\_excel(FName) print(df)

#openpyxl import openpyxl wb = openpyxl.load\_workbook(FName) ws = wb.active Place = [] Country = [] for cell in ws.rows: Place.append(cell[0].value) Country.append(cell[1].value) print (Place,Country)

# pywin32 import win32com.client xl = win32com.client.Dispatch('Excel.Application') wb = xl.Workbooks.Open(FName) ws = wb.Worksheets(1) Place = [] Country = [] for i in range(2,7): Place.append(ws.Range("A" + str(i)).Value) Country.append(ws.Range("B" + str(i)).Value) print (Place,Country) wb.Close(False) xl.Quit

# Word를 삼킨 파이썬

## Python

2021/08/23 06:50 <http://blog.naver.com/550sn/222455259588>



오늘은 파이썬에서 Word를 다룰수 있는 라이브러리를 다뤄 보겠습니다.

이번에 다룰 라이브러리는 "python-docx"입니다.

### [python-docx — python-docx 0.8.11 documentation](https://python-docx.readthedocs.io/en/latest/)

[What it can do Here’s an example of what python-docx can do: from docx import Document from docx.shared import Inches document = Document () document . add\_heading ( 'Document Title' , 0 ) p = document . add\_paragraph ( 'A plain paragraph having some ' ) p . add\_run ( 'bold' ) . bold = True p . add\_... python-docx.readthedocs.io](https://python-docx.readthedocs.io/en/latest/)

기본적인 문법이 Word에 맞게 새로 구성되기는 하지만, 의외로 많은 부분이 python-pptx에서 제공하는 기능을 유사하 게 사용할 수 있어서 이전에 다루었던 PPT를 삼킨 파이썬의 내용을 참고하면 많은 도움을 받을 수 있습니다.



### [PPT를 삼킨 파이썬(python-pptx)](https://blog.naver.com/550sn/222347440606)

[이번에는 Python에서 PPT를 직접 다룰 수 있는 라이브러리를 다뤄 보겠습니다. 이번에 다루고자 하는 ... blog.naver.com](https://blog.naver.com/550sn/222347440606)

1. 설치 및 시작하기

먼저 "python-docx"를 사용하기 위해서는 라이브러리 설치부터 해야 합니다. python 터미널에서

pip install python-docx

를 입력하면 python-docx가 설치 됩니다.

설치되고 난 다음에는 Python 코드를 구성할때 해당 라이브러리를 불러오는 구문만 추가하면 필요한 기능을 불러와서 활 용이 가능합니다.

import docx

Word 파일의 경우 문서를 다루는 것을 기반으로 하기 때문에, docx 라이브러리 내의 "Document"구문을 주로 활용하는 경우가 많으니, 다음과 같이 호출하면 됩니다.

from docx import Document

1. 파일 생성하기, 불러오기 그리고 저장하기

Word에서 파일을 생성하고, 불러오고, 저장하는 방법은 python-pptx에서 사용하는 방법과 상당히 유사합니다. 새파일을 생성하는 방법은 다음과 같습니다.

doc = Document()

기존에 있던 파일을 불러올 때는 () 안에 파일 이름을 입력해 주면 됩니다. doc = Document("불러올 파일 경로와 파일명")

저장 할때는 save 명령어를 사용하면 됩니다. doc.save("저장할 파일경로와 파일명")

이와같이 Word의 Document 개체를 하나 선언하고 지정해 줌으로써 Python에서는 해당 파일을 다루게 됩니다.

1. 적용 예시

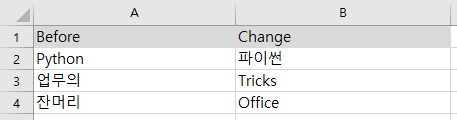
Word는 어떻게 보면 PPT에 비해서는 다루기 용이한 면이 있습니다.

PPT에서는 슬라이드마다 사용하는 개체를 찾은 다음 개체 정보 안에서의 특성을 찾아가야 하는데, Word의 경우 기본적 으로 PPT에서의 글상자 개체가 지정되어 있는 상태라고 보는것이 좋을듯 합니다.

예시로 만들어 보고자 하는 사례는 다음과 같습니다. **첨부파일**

Sample.docx [파일 다운로드](http://blogattach.naver.net/059019aabfe9e13d11f394a69a7b017dde88759963/20210803_231_blogfile/550sn_1627951042614_A0aVSQ_docx/%2FSample.docx)

위와 같이 파이썬 리스트로 만든 docx 파일이 있다고 하겠습니다.

그리고, 아래와 같이 바꾸기 전 용어와 바꾼 후 용어를 엑셀로 지정했다고 하겠습니다.

**첨부파일** Dictionary.xlsx [파일 다운로드](http://blogattach.naver.net/2abf368592c9ce123edcbb89b5542e52f1a75bbe77/20210803_212_blogfile/550sn_1627951200151_QZ51HE_xlsx/%2FDictionary.xlsx)

이때 이 Dictionary 파일에서 정보를 가져와서 PPT 파일의 단어에서 필요한 단어를 찾아서 바꾸는 작업을 일괄 적용하는 소스를 만들어 보면 아래와 같이 구성할 수 있습니다.

from docx import Document import pandas as pd doc = Document("Master/Sample.docx") df = pd.read\_excel("Master/Dictionary.xlsx") for x in df.index: for paragraph in doc.paragraphs: for run in paragraph.runs: cur\_text = run.text new\_text = cur\_text.replace(df.Before[x], df.Change[x]) run.text = new\_text doc.save("Master/Result.docx")

만들어진 소스와 결과 파일은 다음과 같습니다. **첨부파일**

Word\_F.py [파일 다운로드](http://blogattach.naver.net/6df871c1d1818955799bfccef2136915b6e01cfbd5/20210803_87_blogfile/550sn_1627951285400_557OZm_py/%2FWord_F.py) **첨부파일** Result.docx [파일 다운로드](http://blogattach.naver.net/ef7af3495a6465d0f91f7d4d7791ee9430609a2e/20210803_4_blogfile/550sn_1627951319423_3558OI_docx/%2FResult.docx)

이전에 Python-pptx로 유사한 기능을 가진 소스를 만들어본 경험이 있고, 엑셀 파일은 pandas로 다뤄본 경험이 있다보 니, 비교적 쉽게 원하는 기능을 구현할 수 있었습니다.

Python이 처음 진입 장볍이 높고 어려움이 있음에도 한번 익히고 나면 이와 같이 약간의 수정으로도 여러방향의 응용이 가능하여 어느 정도 익숙해지고 Reference 자료가 쌓이고 나면 더 없이 유용한 툴이 될 수 있으리라 생각됩니다.

# 한/글을 삼킨 파이썬 1

## Python

2021/08/26 06:50 <http://blog.naver.com/550sn/222460454384>



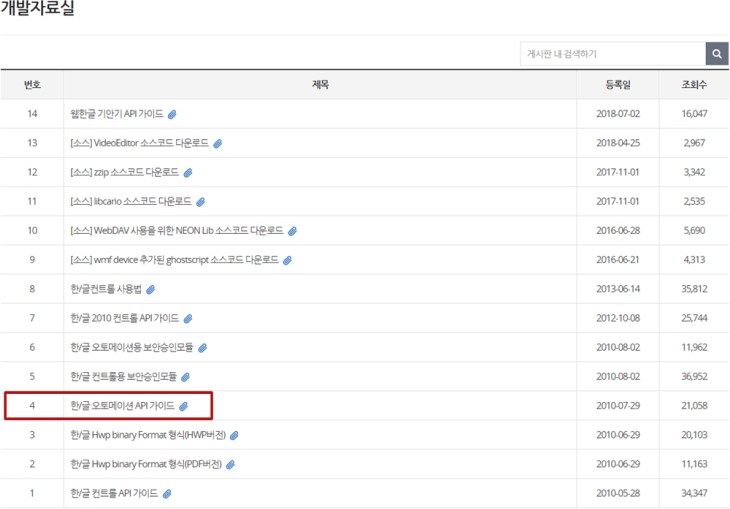
오늘은 우리 나라 관공서에서 많이 사용하는 워드 프로세스인 한컴 오피스의 한/글을 Python에서 활용하는 방법을 알아 보도록 하겠습니다.

안타깝게도 한/글의 경우에는 우리나라를 제외한 다른 나라에서는 거의 사용이 되지 않기 때문에 python-docx, python- pptx, openpyxl 등과 같이 Python 전용으로 제공되는 라이브러리는 찾아보기 어렵습니다.

하지만, 다행스럽게도 win32 application을 활용하면 필요한 기능을 구현할 수 있습니다. 한컴 오피스 개발 자료실에 접속하면 한/글 API를 사용할 수 있는 자료를 찾을 수 있습니다. [**개발자료실 : 한글과컴퓨터**](https://www.hancom.com/board/devdataList.do?gnb0=25&gnb1=81)

[개발자료실 검색어 입력 검색 번호 제목 등록일 조회수 14 웹한글 기안기 API 가이드 2018-07-02 16,047 13 [소스] VideoEditor 소스코드 다운로드 2018-04-25 2,967 12 [소스] zzip 소스코드 다운로드 2017-11-01 3,342 11 [소스] libcario 소스코드 다운로드 2017-11-01 2,535 10 [소스] WebDAV 사용을 위한 NEON Lib 소스코드 다운로드](https://www.hancom.com/board/devdataList.do?gnb0=25&gnb1=81)

[2016-06-28 5,690 9 [소스] wmf device 추가된 ghostscript 소스코드 다운로드 ... www.hancom.com](https://www.hancom.com/board/devdataList.do?gnb0=25&gnb1=81)



개발 자료실에 있는 내용중 한/글 오토메이션 API 가이드에 대한 내용에 한글 Control과 관련되어 있는 내용이 포함되어 있습니다.

한/글을 파이썬으로 자동화 하기 위해서는 이 게시글을 다운로드 받아서 필요한 내용이 있을 때 찾아보시면 도움이 됩니 다.

한/글의 경우 외국에서 그리 많은 사용 사례가 없는지라, 기능 구현과 관련하여서 참고자료를 찾기가 그리 용이하지는 않 습니다.

유튜브 채널중에 회사원 코딩이라는 채널이 있습니다. 여기서 한/글 파이썬을 많이 다루고 있어서 참고하고 공부하기 좋 은 내용이 많습니다.

### [회사원코딩](https://www.youtube.com/channel/UCDB-3Raasov1tC5_qd7jwLg)

[보통 회사원을 위한 파이썬 코딩, 업무효율 향상을 위한 각종 도구, 핵심만 골랐습니다. 코딩으로 아낀 시간에 회사에서 자 기계발합시다. 그리고 야근하지 맙시다. 가정과 직장에서의 삶에 모두 최선을 다하는 여러분, 존경합니다. www.youtube.com](https://www.youtube.com/channel/UCDB-3Raasov1tC5_qd7jwLg)

금번에 다룰 한/글 -파이썬은 몇개의 글로 나누어서 다뤄 보도록 하겠습니다.

한/글을 다루는 것이 꽤 난이도가 있어서 제작이 오래걸리는 것이 하나의 이유이고, 긴 휴가 기간 이후 제 페이스를 아직 찾지 못한 제가 또 하나의 이유입니다.

오늘은 간단히 사용법을 익히는 수준으로만 다뤄 보도록 하겠습니다.

1. 설치 및 시작하기

별도의 파이썬용 라이브러리가 없기 때문에 먼저 한컴 오피스에서 제공하는 Application을 실행하여 사용할 수 있는 Win 32 application 라이브러리 설치부터 해야 합니다. 이전에 ppt Pont 변경에서 사용했던것 처럼 win32com 라이브러리 를 설치하고, 사용하도록 하겠습니다.

### [파이썬으로 PPT Font 통일하기](https://blog.naver.com/550sn/222384801239)

[제가 Youtube 채널을 시작했던 당시 처음으로 만들었던 PPT 매크로가 PPT 폰트 통일이었습니다. P... blog.naver.com](https://blog.naver.com/550sn/222384801239)

python 터미널에서 pip install pywin32

를 입력하면 pywin32가 설치 됩니다.

설치되고 난 다음에는 Python 코드를 구성할때 해당 라이브러리를 불러오는 구문도 추가해야 하지만, 한/글 Application 도 실행해 줘야 하기 때문에 다음과 같은 소스코드가 처음에 꼭 추가되어야 합니다.

해당 코드는 한/글이 설치된 PC에서만 사용이 가능합니다.

import win32com.client hwp = win32com.client.gencache.EnsureDispatch('HWPFrame.HwpObject')

그 뒤로는 hwp라는 개체 형태로 한/글 프로그램을 실행한 다음에 내가 활용하고자 하는 기능들을 하나하나 실행해 가능 방식으로 한/글 자동화를 만들 수 있습니다.

그리고, 작업이 종료되면 한/글을 종료처리 해 주어야 합니다. hwp.Quit()

1. 파일 생성하기, 불러오기 그리고 저장하기

한/글에서 파일을 생성하고, 불러오고, 저장하는 방법은 다음과 같습니다. 새파일을 생성하는 방법은 다음과 같습니다.

hwp.Run("FileNew")

기존에 있던 파일을 불러올 때는 Open이라는 명령어를 사용합니다. hwp.Open("불러올 파일명(경로포함)")

저장 할때는 SaveAs 명령어를 사용하면 됩니다. hwp.SaveAs("저장할 파일명(경로포함)")

이와같이 hwp 개체를 하나 선언하고 그 하위의 명령어들을 실행 함으로써 한/글을 다루게 됩니다.

이들 명령어에 대한 설명은 앞서 사이트에서 다운로드 받았던 "HwpAutomation.hwp" 파일과 "ActionObject.hwp"파 일에 나와 있습니다.

**첨부파일** HwpAutomation.hwp [파일 다운로드](http://blogattach.naver.net/9c098036221716a38a6c0e3e04e29de7431de33c/20210824_25_blogfile/550sn_1629745302439_A8XFQ5_hwp/%2FHwpAutomation.hwp)

**첨부파일** ActionObject.hwp [파일 다운로드](http://blogattach.naver.net/4dd851e1f5a9a97559bbdceed233493596c33bd9e1/20210824_245_blogfile/550sn_1629745223082_E71V58_hwp/%2FActionObject.hwp)

설명이 상세하다고 하기는 어렵겠지만, 워낙 많은 기능들을 한번에 압축해서 담았다 보니 한눈에 들어오지는 않습니다. 인 내심을 가지고 하나하나 읽어보고 시도해 보는것이 방법이 될 것 같습니다.

"HwpAutomation.hwp" 파일에는 한글에서 사용 가능한 자동화 명령어들에 대한 기능 설명이 되어 있고, "ActionObject.hwp"파일에는 한글에서 사용 가능한 명령어중 가장 많은 기능을 포함하고 있는 "Run"에서 실행 가능한 기능들에 대한 설명이 포함되어 있습니다.

1. 한/글에서 파일 열고 위치 이동해 보기

오늘은 우선 간단하게 몇가지 자주 사용할만한 기능들을 소스 코드로 구현해 보겠습니다.

먼저 Run이라는 명령어를 사용하여 문서의 가장 처음과 가장 끝으로 이동하는 구문을 작성해 보면 다음과 같습니다. #방법 1 : Run을 사용 hwp.Run("MoveDocBegin") hwp.Run("MoveDocEnd")

같은 기능이지만, MovePos와 위치에 대한 고유번호를 사용해서 작성하면 다음과 같습니다. #방법 2 : MovePos와 고유 번호를 사용 hwp.MovePos(2) hwp.MovePos(3)

이번에는 MovePos와 위치에 대한 ID를 사용해서 작성하면 다음과 같습니다.

#방법 3 : MovePos와 고유 ID를 사용 hwp.MovePos("moveTopOfFile") hwp.MovePos("moveBottomOfFile") 이번에는 약간의 응용으로 글의 맨 처음으로 온다음 범위 설정을 하되 해당 줄의 끝까지를 선택했다가 선택 해지 해 보도 록 하는 명령을 순차적으로 작성하면 다음과 같습니다.

#한줄을 선택했다가 해제해 보기 hwp.MovePos(2) hwp.Run("Select") hwp.Run("MoveLineEnd") hwp.Run("Cancel")

지금까지 테스트해본 소스 내용들을 하나로 모으면 다음과 같습니다.

import win32com.client hwp = win32com.client.gencache.EnsureDispatch('HWPFrame.HwpObject') hwp.Run("FileNew") hwp.Open("D:/Python/TOffice/Master/Sample.hwp") #방법 1 : Run을 사용 hwp.Run("MoveDocBegin") hwp.Run("MoveDocEnd") #방법 2 : MovePos와 고유 번호를 사용 hwp.MovePos(2) hwp.MovePos(3) #방법 3 : MovePos와 고유 ID를 사용 hwp.MovePos("moveTopOfFile") hwp.MovePos("moveBottomOfFile") #한줄을 선택했다가 해제해 보기 hwp.MovePos(2) hwp.Run("Select") hwp.Run("MoveLineEnd") hwp.Run("Cancel") #결과 파일 저장해 보기 hwp.SaveAs("D:/Python/TOffice/Master/ Result.hwp") #종료하기 hwp.Quit()

위의 소스 파일을 만들어서 저장한 다음 Debug 모드로 한줄씩 실행해 보면서 한/글 안에서 내가 원하는 대로 동작하는지 하나하나 살펴보시면 도움이 될듯 합니다.

유튜브 영상을 제작하면서 만들었던 예제 파일은 다음과 같습니다. **첨부파일**

HWP\_py.py [파일 다운로드](http://blogattach.naver.net/b124ad1963543c8ca142221429cab3cd6b3f50/20210825_80_blogfile/550sn_1629858277394_V7fe5A_py/%2FHWP_py.py)

이상으로 파이썬에서 한/글을 다루는 기초적인 방법에 대하여 알아 봤습니다.

다음 글에서는 앞서 Word를 다룰때와 같은 기능을 구현하는 방법에 대하여 고민하여 구현해 보도록 하겠습니다.

# 한/글을 삼킨 파이썬 2

## Python

2021/08/30 06:50 <http://blog.naver.com/550sn/222485531060>



지난번에 알아봤던 파이썬에서 한/글을 활용하는 방법을 응용하여 여러 keyword를 미리 만들어둔 Dictionary 파일의 형 식에 맞게 한번에 변경하는 기능을 구현해 보도록 하겠습니다.

1. 작업 개요

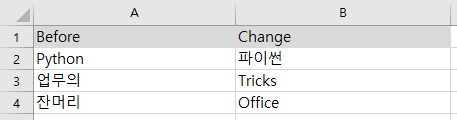
이번에 구현하고자 하는 기능은 사실 앞서 Word-Python에서 구현했던 것과 동일한 기능이라고 할 수 있습니다.

### [Word를 삼킨 파이썬](https://blog.naver.com/550sn/222455259588)

[오늘은 파이썬에서 Word를 다룰수 있는 라이브러리를 다뤄 보겠습니다. 이번에 다룰 라이브러리는 "p... blog.naver.com](https://blog.naver.com/550sn/222455259588)

**첨부파일** Sample.hwp [파일 다운로드](http://blogattach.naver.net/39ac259680dfdd012dcfa89aa6473d41e2b74da814/20210807_135_blogfile/550sn_1628296853685_mC1QkU_hwp/%2FSample.hwp)

위의 첨부 파일과 같이 파이썬 리스트로 만든 hwp 파일이 있다고 하겠습니다. 그리고, 아래와 같이 바꾸기 전 용어와 바꾼 후 용어를 엑셀로 지정했다고 하겠습니다.



**첨부파일** Dictionary.xlsx [파일 다운로드](http://blogattach.naver.net/f164ed584c7a7bcee7016353698ff08a2d7b81e2/20210807_123_blogfile/550sn_1628296951267_AldBew_xlsx/%2FDictionary.xlsx)

이때 이 Dictionary 파일에서 정보를 가져와서 한/글 파일의 단어에서 필요한 단어를 찾아서 바꾸는 작업을 일괄 적용하는 소스를 만드는 것을 최종 목표로 해 보겠습니다.

실질적으로 한/글에서 하는 일은 다음의 동작이 될 것입니다.

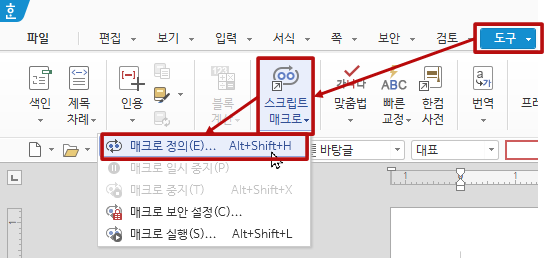
1. 찾아 바꾸기를 실행
2. 찾을 단어에 "Before" 값 입력
3. 바꿀 단어에 "Change" 값 입력
4. Dictionary파일의 리스트가 종료 될 때까지 1~3번 반복

기능 자체로는 한/글에서 있는 찾아 바꾸기만을 활용하는 기능으로 이 기능에 대한 내용을 가져오면 좋을듯 합니다.

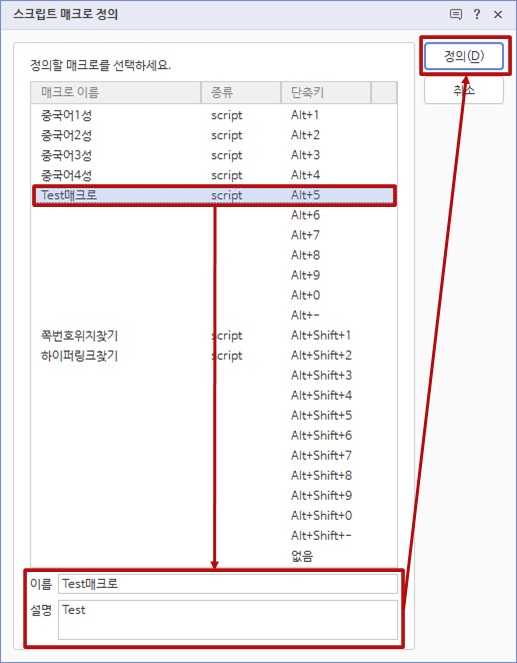
1. 매크로 기록 & 소스 분석하기

다행스럽게도 한/글에서는 매크로 기록 기능을 제공하고 있고, 기록된 매크로 기능의 소스를 읽어올 수 있습니다.

이때 만들어지는 소스는 자바스크립트 문법에 따라서 자동으로 만들어지는데, 이 코드 부분에서 필요한 영역만 가져와서 파이썬 문법에 맞게 바꿔주면 활용이 가능합니다.

한/글에서 도구 → 스크립트 매크로 → 매크로 정의 를 선택하면 (또는 단축키 Alt + Shift + H) 스크립트 매크로 정의라는 별도의 창이 뜹니다.

여기서 빈칸을 하나 누르거나 기존의 값이 있는 칸을 선택한 다음에 이름과 설명에 간략한 정보를 넣고 "정의" 버튼을 누 르면 그 다음부터 내가 하는 동작이 녹화가 됩니다.

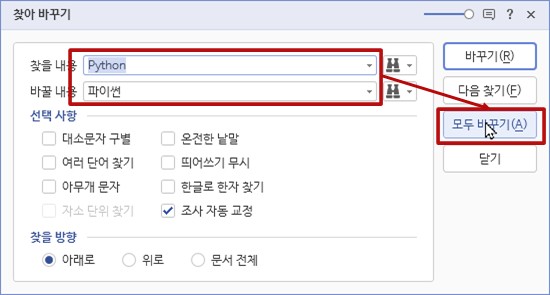


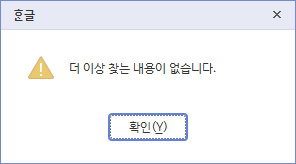
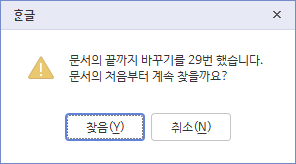
예제 문서에서 "Python"을 "파이썬"으로 전체 바꾸는 작업을 기억시켜 보도록 하겠습니다.

정의 버튼을 누르고 나면 마우스 아래 조그만한 아이콘이 표시되며 기록이 되고 있음이 표시 됩니다.

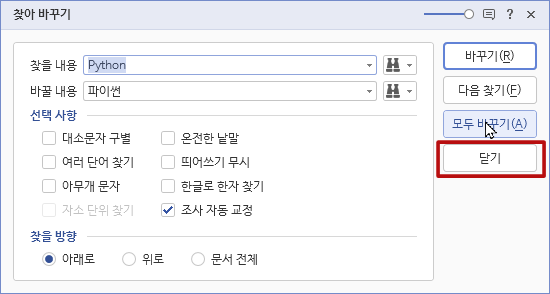


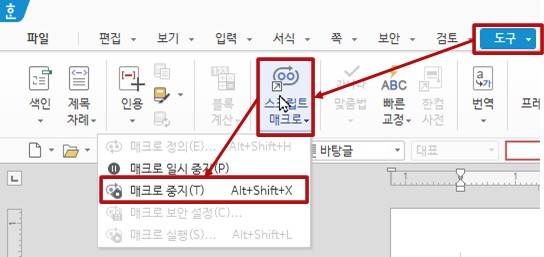
Ctrl + H 버튼을 눌러서 찾아 바꾸기 창을 띄운 다음 우리가 필요한 정보를 입력하고 전체 바꾸기를 실행하겠습니다.



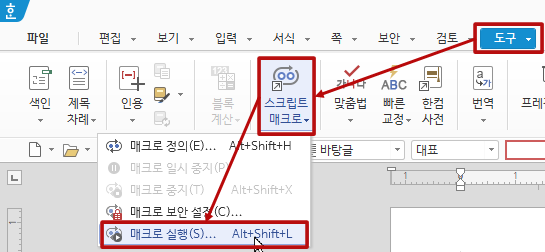
실행하고 나면 찾아바꾸기 작업이 완료되었다는 확인 창이 두번 뜹니다.

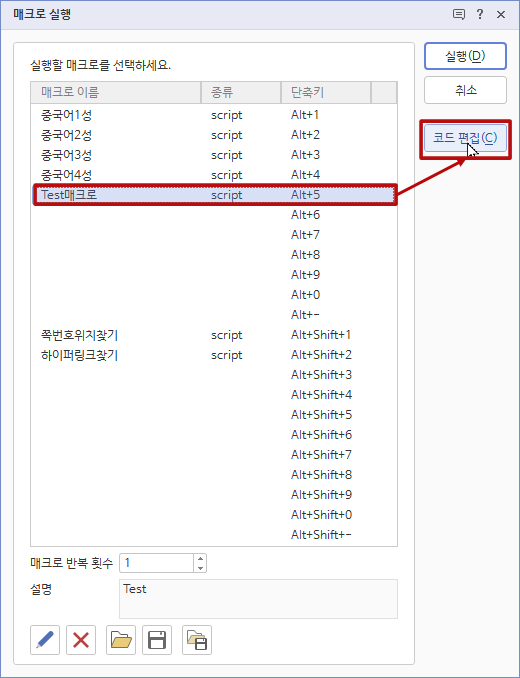
확인을 선택하고 찾아 바꾸기의 닫기 버튼을 눌러 작업을 완료합니다.



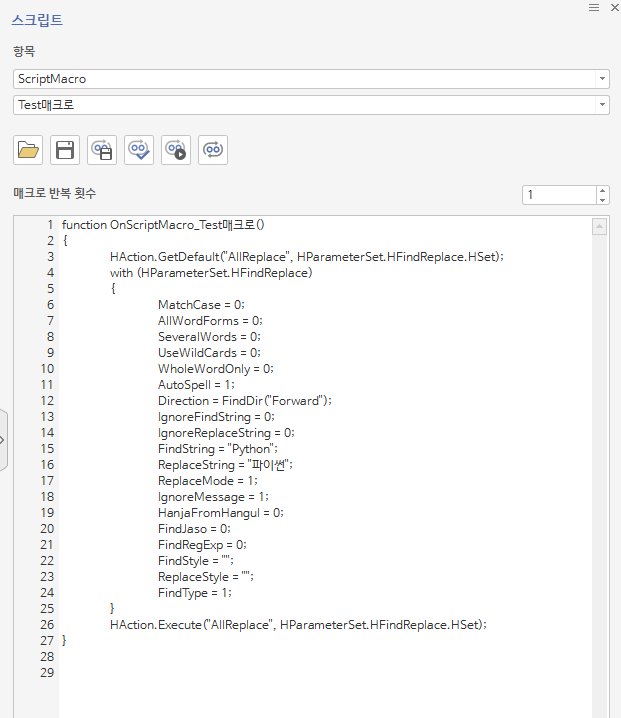
스크립트 매크로에서 매크로 중지 버튼을 누릅니다.

이제 자동으로 만들어진 소스를 찾아 보겠습니다. 도구 → 스크립트 매크로 → 매크로 실행을 누릅니다.

방금 만든 소스를 선택하고 코드 편집을 클릭합니다.



그러면, 프로그램 우측에 스크립트라는 창이 하나 더 생기면서 방금 작업한 내용에 대한 소스가 쭉 표시가 됩니다.



소스내용을 복사해 오면 다음과 같습니다.

function OnScriptMacro\_Test매크로() { HAction.GetDefault("AllReplace", HParameterSet.HFindReplace.HSet); with (HParameterSet.HFindReplace) { MatchCase = 0; AllWordForms = 0; SeveralWords = 0; UseWildCards = 0; WholeWordOnly = 0; AutoSpell = 1; Direction = FindDir("Forward"); IgnoreFindString = 0; IgnoreReplaceString = 0; FindString = "Python"; ReplaceString = "파이썬"; ReplaceMode = 1; IgnoreMessage

= 1; HanjaFromHangul = 0; FindJaso = 0; FindRegExp = 0; FindStyle = ""; ReplaceStyle = ""; FindType = 1; } HAction.Execute("AllReplace", HParameterSet.HFindReplace.HSet); }

한글 매크로에서 어떤 기능을 기록하면 기본적으로 다음과 같은 순서로 작업이 되는것 같습니다. 작업할 동작 이름의 함수를 불러오고 → 중요 파라미터값들 입력하고 → 불러온 동작 함수 실행하고.

위의 동작도 따지고 보면 아래와 같이 정리할 수 있을듯 합니다.

HAction.GetDefault("동작 이름", 중요 Parameter) 중요 Parameter 입력 HAction.Execute("동작이름", 중요 Parameter);

1. 파이썬에 적용해 보기

한/글에서 자동으로 생성해 주는 문법 자체가 자바 스크립트의 문법을 따르다 보니 이를 파이썬으로 조금씩만 바꿔주면 되 는데, 이전에 한/글을 다루면서 만들었던 소스에 이 내용을 덧붙여 보도록 하겠습니다.

중요 Parameter중에서는 제게 의미가 있는 Parameter는 FindString 값과 ReplaceString 값이기 때문에 이에 대한 Parameter만 가져오도록 하겠습니다.

그리고, 실제 실행했을때 떴었던 메시지를 매번 클릭하는것은 번거롭기 때문에 메세지를 생략하는 내용에 대한 변수인 "IgnoreMessage" 값 까지만 추가로 가져오도록 하겠습니다.

만들어진 소스는 다음과 같습니다.

import win32com.client hwp = win32com.client.gencache.EnsureDispatch('HWPFrame.HwpObject') hwp.Run("FileNew") hwp.Open("D:/Python/TOffice/Master/Sample.hwp") hwp.HAction.GetDefault("AllReplace", hwp.HParameterSet.HFindReplace.HSet) hwp.HParameterSet.HFindReplace.FindString = "Python" hwp.HParameterSet.HFindReplace.ReplaceString = "파이썬" hwp.HParameterSet.HFindReplace.IgnoreMessage = 1 hwp.HAction.Execute("AllReplace", hwp.HParameterSet.HFindReplace.HSet) hwp.SaveAs("D:/Python/TOffice/Master/Result.hwp") hwp.Quit() 기존의 파일을 열고, 저장하는 부분 이외에 추가된 내용에서는 한/글 매크로로 만들어진 소스에 "hwp."라고 파이썬에서 불러온 개체 안의 속성들을 활용한다고 정의한 부분이 변경되었고, with로 묶여있던 부분을 with 밖으로 빼냈습니다. 이제 실행하면 파일을 불러오고 찾아 바꾸기 작업을 자동으로 하는 것을 볼 수 있습니다.

1. Dictionary 반복문 추가하기

금번에 만들 소스가 Excel 파일에서 찾아 바꿀 단어를 찾아서 반복하는 내용이었기 때문에 이에 대한 소스를 추가하여 앞 서 만들어진 소스를 감싸도록 하겠습니다.

이 반복문은 앞서 Word를 다룰때 사용했던 로직과 동일하기 때문에 그대로 가져오도록 하겠습니다. import win32com.client import pandas as pd hwp =

win32com.client.gencache.EnsureDispatch('HWPFrame.HwpObject') df = pd.read\_excel("Master/ Dictionary.xlsx") hwp.Run("FileNew") hwp.Open("D:/Python/TOffice/Master/Sample.hwp") for x in df.index: hwp.HAction.GetDefault("AllReplace", hwp.HParameterSet.HFindReplace.HSet) hwp.HParameterSet.HFindReplace.FindString = df.Before[x] hwp.HParameterSet.HFindReplace.ReplaceString

= df.Change[x] hwp.HParameterSet.HFindReplace.IgnoreMessage = 1 hwp.HAction.Execute("AllReplace", hwp.HParameterSet.HFindReplace.HSet) hwp.SaveAs("D:/Python/TOffice/Master/Result.hwp") hwp.Quit()

1. 메세지 박스 생략하기

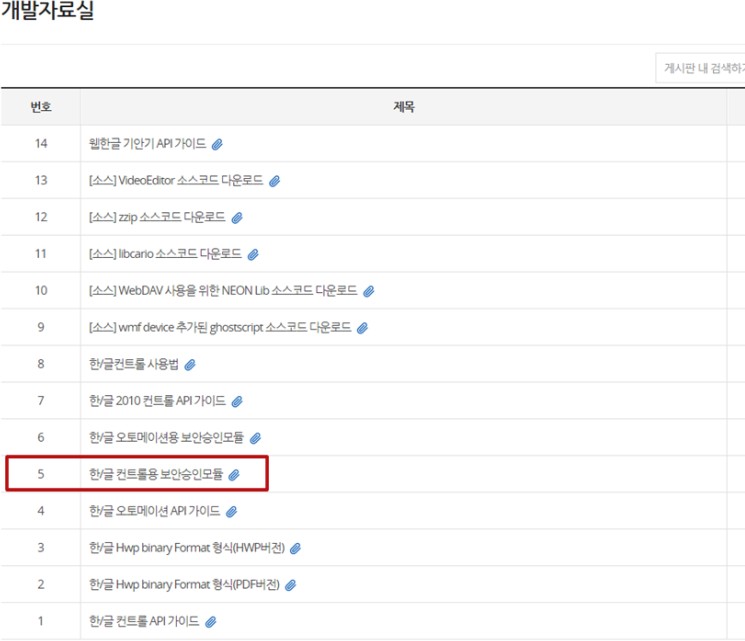
자동으로 동작하긴 하는데 매번 파일을 불러올때 마다 보안 에러 메세지가 뜹니다.



이 에러 메세지가 뜨지 않게 하는 방법이 있는데요. 생각보다는 까다롭습니다. 보안 승인 모듈을 다운로드 받아서 레지스터리 등록을 해줘야 하기 때문인데요.

이 방법은 정말 자세하게 잘 다루어준 유튜브 영상이 있어서 해당 영상으로 소개하는 것으로 대신할까 합니다. (이 영상보다 자세하게 잘 다룰 자신이 없는것도 한가지 이유입니다.)

보안 승인 모듈에 필요한 파일은 한컴 오피스 개발 자료실에서 다운로드 받을 수 있습니다.

앞서 한/글 - 파이썬에 대해 공부할때 많이 참고했다고 소개드렸던 "회사원 코딩" 사이트에 소개된 내용입니다.

보안 모듈이 등록되면 소스 코드에서 파일을 불러오기 전 아래의 구문을 추가해 주면 파일을 불러올때 보안 팝업이 뜨지 않습니다.

hwp.RegisterModule('FilePathCheckDLL','SecurityModule') 여기까지 해서 완성된 소스는 다음과 같습니다.

import win32com.client import pandas as pd hwp = win32com.client.gencache.EnsureDispatch('HWPFrame.HwpObject') df = pd.read\_excel("Master/ Dictionary.xlsx") hwp.RegisterModule('FilePathCheckDLL','SecurityModule') hwp.Run("FileNew") hwp.Open("D:/Python/TOffice/Master/Sample.hwp") for x in df.index: hwp.HAction.GetDefault("AllReplace", hwp.HParameterSet.HFindReplace.HSet) hwp.HParameterSet.HFindReplace.FindString = df.Before[x] hwp.HParameterSet.HFindReplace.ReplaceString = df.Change[x] hwp.HParameterSet.HFindReplace.IgnoreMessage = 1 hwp.HAction.Execute("AllReplace", hwp.HParameterSet.HFindReplace.HSet) hwp.SaveAs("D:/Python/TOffice/Master/Result.hwp") hwp.Quit() 소스 파일과 결과로 만들어진 hwp 파일은 다음과 같습니다.

**첨부파일** HWP\_py.py [파일 다운로드](http://blogattach.naver.net/83169f283c0809bc957311211bfd82f85f0ef496/20210827_113_blogfile/550sn_1630037773693_6mvR1p_py/%2FHWP_py.py) **첨부파일** Result.hwp [파일 다운로드](http://blogattach.naver.net/c653da697a2122fed230576559b8c2be1d48b550d0/20210827_129_blogfile/550sn_1630037804617_9OXkk6_hwp/%2FResult.hwp)

Youtube 영상을 만들면서 만든 소스코드는 다음과 같습니다. 내용은 대동 소이합니다. **첨부파일**

HWP\_py\_YT.py [파일 다운로드](http://blogattach.naver.net/86139a2c320d0cb9907614241ef887fd5a0bf393/20210830_190_blogfile/550sn_1630291811940_SY6jeb_py/%2FHWP_py_YT.py)

Python이 처음 진입 장벽이 높고 어려운데, 한/글은 더 어려운것 같습니다.

참고할만한 Reference가 충분히 많지 않다 보니 한컴에서 제공하는 개발용 문서를 꼼꼼히 읽어보는 수 밖에 없는데다가 예제 문서도 MS Office나 구글 오피스 대비 충분하지 않아서 여러번 시도해 보면서 하나 둘 구현해 볼 수 밖에 없다는 점 이 아쉽습니다.

그래도, 한가지 다행스러운 점은 한/글 매크로에서 기록한 내용들이 자바스크립트 형태의 소스를 남기며 자동으로 기록되 는 기능이 있다는 점이리라 생각됩니다.

한/글을 정말 많이 사용할 수 밖에 없는 상황이 아니라면 다른 사무용 S/W에 비해서 공부량 대비 효율은 상대적으로 낮을 수도 있겠습니다. 그럼에도 한국에서는 한/글 자동화의 Needs가 있는 곳은 계속 존재하기 때문에 한/글을 많이 사용하시 는 분들이라면 한번쯤 익혀 보시는것도 추천합니다.

한/글 자동화에 대한 부분을 공부하고 살펴보다 보니 한/글은 참 비운의 소프트웨어라는 생각이 많이 듭니다. 지극히 개인적인 의견이긴 하지만, 워드 프로세스 자체로만 보면 한/글은 최고의 프로그램이라고 생각됩니다.

이와 같은 자동화에 대한 가이드도 이미 2010년경에 굉장히 높은 수준으로 제공되었었고 그뒤로 지금까지 큰 수정이 없 을만큼 완성도 또한 수준 높았다고 생각됩니다.

워드 프로세스 하나로만 보면 강력하고 우수하지만, MS Office를 경쟁 상대로 본다면 사무용 S/W로써 경쟁해야 하는 프 로그램들이 엑셀, 파워포인트였기에 이 경쟁(특히 엑셀)에서 밀려서 입지가 좁아지지 않았나 하는 생각을 해 봅니다.

# 수백개의 문서를 한번에 PDF로 바꾸기(Python, PPT/Docx/hwp/ images)

## Python

2021/09/02 06:50 <http://blog.naver.com/550sn/222486790485>



오늘은 다양한 형태의 사무용 SW 파일을 PDF 파일로 저장해 보도록 하겠습니다.

일단 대부분의 사무용 SW 파일은 PDF 파일로 저장하는 기능을 제공하는 기능이 있으므로 굳이 파이썬으로 기능을 만들 필요가 없겠지만, 여러개의 문서 파일를 한번에 PDF 파일로 변경해야 하는 경우라면 충분히 도움을 받을 수 있으리라 생 각됩니다.

1. 공통 로직

ToDo 폴더와 Result 폴더를 만들어 두고, ToDo 폴더의 파일을 하나하나 읽어와서 Result 폴더로 PDF 파일로 전환하여 저장하는 로직을 만들도록 하겠습니다.

import os import re TDPath = "D:\\Data\\ToDo" RPath = "D:\\Data\\Result" #PPT to PDF files = [f for f in os.listdir(TDPath) if re.match('.\*[.]ppt', f)] for file in files: # PPT 파일을 PDF로 바꾸는 로직 #Word to PDF files = [f for f in os.listdir(TDPath) if re.match('.\*[.]doc', f)] for file in files: # Word 파일을 PDF로 바꾸는 로직 #HWP to PDF files = [f for f in os.listdir(TDPath) if re.match('.\*[.]hwp', f)] for file in files: # HWP 파일을 PDF로 바꾸는 로직 #Image to PDF files = [f for f in os.listdir(TDPath) if re.match('.\*([.]jpg|[.]png|[.]gif)', f)] for file in files: # HWP 파일을 PDF로 바꾸는 로직

여기서는 정규식을 사용하여 파일 필터가 걸린 파일 리스트를 가져오는 기능으로 files 리스트를 만들고, 반복문을 추가했 습니다.

정규식을 사용하는 방법과 관련하여서는 이전에 다루었던 아래 글을 참고하시면 도움이 될듯 합니다.



### [정규식을 삼킨 파이썬 (파일 필터)](https://blog.naver.com/550sn/222385189403)

[이번 장기 프로젝트를 통해서 자료를 만들다 보니 우연히도 지금까지 만든 기능 모두가 모두 PPT 파일만... blog.naver.com](https://blog.naver.com/550sn/222385189403)

1. PPT to PDF

PPT 파일을 PDF로 변환하기 위해 "python pptx to pdf"로 검색을 하면 여러 글이 나오는데, PPT를 다루는 라이브러리 중 하나의 python-pptx에서는 해당 기능을 제공하지 않고, Power Point Application을 직접 띄워서 PDF 파일 형태로 다시 저장해야 한다는 것을 찾아볼 수 있습니다.



### [How to convert a .pptx to .pdf using Python](https://stackoverflow.com/questions/31487478/how-to-convert-a-pptx-to-pdf-using-python)

[I have been looking to convert a .pptx file to a .pdf file through a Python script for several hours but nothing seems to be working. What I have tried: I have tried 1) this script which calls](https://stackoverflow.com/questions/31487478/how-to-convert-a-pptx-to-pdf-using-python)

[stackoverflow.com](https://stackoverflow.com/questions/31487478/how-to-convert-a-pptx-to-pdf-using-python)

검색한 Source를 참고하여 PPT to PDF 기능을 구현해 보면 아래와 같이 소스를 구성할 수 있습니다. import os import re import win32com.client powerpoint = win32com.client.Dispatch("Powerpoint.Application") TDPath = "D:\\Data\\ToDo" RPath =

"D:\\Data\\Result" #PPT to PDF files = [f for f in os.listdir(TDPath) if re.match('.\*[.]ppt', f)] for file in files: # PPT 파일을 PDF로 바꾸는 로직 deck = powerpoint.Presentations.Open(os.path.join(TDPath, file)) deck.SaveAs(os.path.join(RPath, file + ".pdf"), 32) # formatType = 32 for ppt to pdf deck.Close() powerpoint.Quit()

여기서 SaveAS 안에 들어있는 32라는 숫자가 들어가는 위치가, Power Application의 SaveAs 기능에서 파일의

formatType를 표시하는 자리로 32가 PDF 파일을 의미합니다.

소스 로직은 하나하나의 파일을 불러온 다음, PDF 파일 형태로 저장하는 로직이 만들어 집니다.

그런데, 한가지 아쉬운것이 새로 저장되는 파일은 "~.pptx.pdf" 라는 형태로 파일이름이 저장됩니다. 파일이름에 이미 확장자명이 들어가 있는 상태에서 pdf라는 확장자를 추가하다 보니 이와 같은 현상이 생깁니다.

이 부분을 개선하기 위해 "python rename file extension"라는 키워드로 검색을 해 보면, 역시 참고할만한 자료를 많이 찾을 수 있습니다.



### [Changing file extension in Python](https://stackoverflow.com/questions/2900035/changing-file-extension-in-python)

[Suppose from index.py with CGI, I have post file foo.fasta to display file. I want to change foo.fasta's file extension to be foo.aln in display file. How can I do it?](https://stackoverflow.com/questions/2900035/changing-file-extension-in-python)

[stackoverflow.com](https://stackoverflow.com/questions/2900035/changing-file-extension-in-python)

이 내용을 참고로 하여 위의 소스를 수정하면 다음과 같이 만들수 있습니다. import os import re import win32com.client powerpoint =

win32com.client.Dispatch("Powerpoint.Application") TDPath = "D:\\Data\\ToDo" RPath = "D:\\Data\\Result" #PPT to PDF files = [f for f in os.listdir(TDPath) if re.match('.\*[.]ppt', f)] for file in files: # PPT 파일을 PDF로 바꾸는 로직 deck = powerpoint.Presentations.Open(os.path.join(TDPath, file)) pre, ext = os.path.splitext(file) deck.SaveAs(os.path.join(RPath, pre + ".pdf"), 32) # formatType = 32 for ppt to pdf deck.Close() powerpoint.Quit()

1. Word to PDF

Word 파일을 PDF로 만드는 기능도 PPT 에서 처럼 Word Application을 사용해도 되겠지만, 구글에서 "python docx to pdf"로 검색을 해 보면 이 경우에는 별도의 라이브러리가 있는것을 찾아볼 수 있습니다.



### [docx2pdf](https://pypi.org/project/docx2pdf/)

[Convert docx to pdf on Windows or macOS directly using Microsoft Word (must be installed). pypi.org](https://pypi.org/project/docx2pdf/)

이를 활용하면 보다 쉽게 기능을 구현할 수 있습니다.

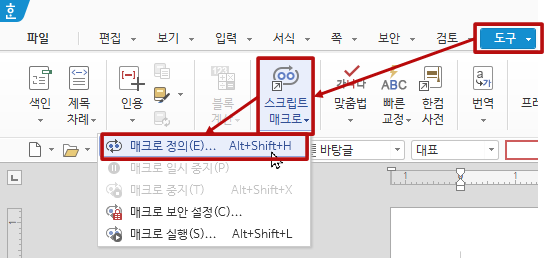
Word를 PDF로 만드는 소스는 다음과 같이 정리할 수 있습니다.

import os import re from docx2pdf import convert TDPath = "D:\\Data\\ToDo" RPath = "D:\\Data\\Result" #Word to PDF files = [f for f in os.listdir(TDPath) if re.match('.\*[.]doc', f)] for file in files: # Word 파일을 PDF로 바꾸는 로직 pre, ext = os.path.splitext(file) convert(os.path.join(TDPath, file), os.path.join(RPath, pre + ".pdf"))

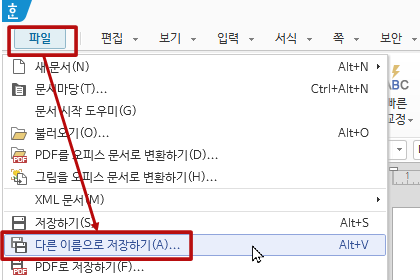
소스 자체는 가장 단순하지 않나 생각됩니다.

1. HWP to PDF

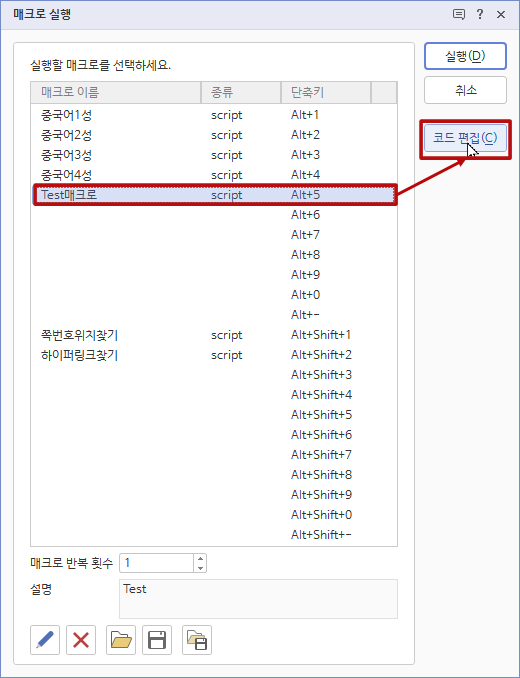
한컴 오피스 한글의 경우에는 역시 소스를 찾아보기 쉽지 않습니다.

예제 파일을 하나 열어서 도구 → 스크립트 매크로 → 매크로 정의를 한 상태에서

파일 → 다른이름으로 저장하기를 사용하여 PDF파일로 저장해 봅니다.



매크로 중지를 선택한 다음 매크로 실행 → 코드 편집을 선택합니다.



그러면 상당히 긴 코드가 만들어져 있는 것을 볼 수 있습니다.

function OnScriptMacro\_Test매크로() { HAction.GetDefault("PrintToPDFEx", HParameterSet.HPrint.HSet); with (HParameterSet.HPrint) { PrinterPaperSize = PrintPaper(""); PrinterPaperWidth = 59530; PrinterPaperLength = 84190; Collate = 1; UserOrder = 0; PrintToFile = 0; UsingPagenum = 1; ReverseOrder = 0; Pause = 0; PrintImage = 1; PrintDrawObj = 1; PrintClickHere = 0; PrintAutoFootnoteLtext = "^f"; PrintAutoFootnoteCtext = "^t"; PrintAutoFootnoteRtext = "^P쪽 중 ^p쪽"; PrintAutoHeadnoteLtext = "^c"; PrintAutoHeadnoteCtext = "^n"; PrintAutoHeadnoteRtext = "^p"; PrintFormObj = 1; PrintMarkPen = 0; PrintBarcode = 1; PrintPronounce = 0; } HAction.Execute("PrintToPDFEx", HParameterSet.HPrint.HSet); HAction.GetDefault("FileSaveAs\_S", HParameterSet.HFileOpenSave.HSet); with (HParameterSet.HFileOpenSave) { FileName = "D:\\Data\\ToDo\\Sample.pdf"; Format = "PDF";

Attributes = 0; } HAction.Execute("FileSaveAs\_S", HParameterSet.HFileOpenSave.HSet); }

코드 내용을 잘 살펴보면, 사실상 PDF 파일 형태로 저장하는 부분은 아래의 맨 마지막에 있는 내용들임을 알 수 있습니다. HAction.GetDefault("FileSaveAs\_S", HParameterSet.HFileOpenSave.HSet); with (HParameterSet.HFileOpenSave) { FileName = "D:\\Data\\ToDo\\Sample.pdf"; Format = "PDF"; Attributes = 0; } HAction.Execute("FileSaveAs\_S", HParameterSet.HFileOpenSave.HSet);

이를 토대로 소스를 작성하면 다음과 같이 작성 됩니다. import os import re import win32com.client hwp =

win32com.client.gencache.EnsureDispatch('HWPFrame.HwpObject') hwp.RegisterModule('FilePathCheckDLL', 'SecurityModule') TDPath = "D:\\Data\\ToDo" RPath = "D:\\Data\\Result" #HWP to PDF files = [f for f in os.listdir(TDPath) if re.match('.\*[.]hwp', f)] for file in files: # HWP 파일을 PDF로 바꾸는 로직 hwp.Open(os.path.join(TDPath, file)) pre, ext = os.path.splitext(file) hwp.HAction.GetDefault("FileSaveAs\_S", hwp.HParameterSet.HFileOpenSave.HSet) hwp.HParameterSet.HFileOpenSave.FileName = os.path.join(RPath, pre + ".pdf") hwp.HParameterSet.HFileOpenSave.Format = "PDF" hwp.HAction.Execute("FileSaveAs\_S", hwp.HParameterSet.HFileOpenSave.HSet); hwp.Quit()

그런데, 이렇게 실행하면 중간에 에러 메시지가 발생하면서 동작하지 않습니다. "FileName"을 찾을수가 없다고 하면서 요.

이 부분때문에 좀 고생을 했는데, 아무래도 HWPFrame에 정의되어 있는 "FileName"이 실제 동작시에는 대소문자가 다 르게 적용되어 있는것 같습니다. 이 FileName을 모두 소문자로 바꿔주면 정상 동작합니다.

import os import re import win32com.client hwp = win32com.client.gencache.EnsureDispatch('HWPFrame.HwpObject') hwp.RegisterModule('FilePathCheckDLL', 'SecurityModule') TDPath = "D:\\Data\\ToDo" RPath = "D:\\Data\\Result" #HWP to PDF files = [f for f in os.listdir(TDPath) if re.match('.\*[.]hwp', f)] for file in files: # HWP 파일을 PDF로 바꾸는 로직 hwp.Open(os.path.join(TDPath, file)) pre, ext = os.path.splitext(file) hwp.HAction.GetDefault("FileSaveAs\_S", hwp.HParameterSet.HFileOpenSave.HSet) hwp.HParameterSet.HFileOpenSave.FileName = os.path.join(RPath, pre + ".pdf") hwp.HParameterSet.HFileOpenSave.Format = "PDF" hwp.HAction.Execute("FileSaveAs\_S", hwp.HParameterSet.HFileOpenSave.HSet); hwp.Quit()

※ 이부분은 저혼자 한참 끙끙 대다가 이번에도 회사원 코딩 채널의 도움을 받았습니다. 한컴 오피스 한/글 자동화는 이 채널이 아니었음 공부하기 정말 힘들뻔 했습니다.

1. Image to PDF

마지막으로 다뤄볼 내용은 이미지 파일을 모아서 PDF로 만드는 작업입니다.

"python image to pdf"으로 검색해 보면 마찬가지로 좋은 자료를 많이 찾을 수 있습니다. [**Create PDF from a list of images**](https://stackoverflow.com/questions/27327513/create-pdf-from-a-list-of-images)

[Is there any practical way to create a PDF from a list of images files, using Python? In Perl I know that module. With it I can create a PDF in just 3 lines: use PDF::FromImage; ... my $pdf = PDF::](https://stackoverflow.com/questions/27327513/create-pdf-from-a-list-of-images)

[stackoverflow.com](https://stackoverflow.com/questions/27327513/create-pdf-from-a-list-of-images)

여러 형태의 소스가 존재하지만, fpdf라는 라이브러리를 사용하는 소스가 가장 사용하기 편할것 같아서 이를 기반으로 하 여 아래와 같이 소스를 작성할 수 있습니다.

import os import re from fpdf import FPDF pdf = FPDF('L') TDPath = "D:\\Data\\ToDo" RPath = "D:\\Data\\Result" #Image to PDF files = [f for f in os.listdir(TDPath) if re.match('.\*([.]jpg|[.]png|[.]gif)', f)] for file in files: # img를 PDF로 바꾸는 로직 pdf.add\_page() pdf.image(os.path.join(TDPath,file), 0, 0, 330) pdf.output(os.path.join(RPath,"IMG2PDF.pdf"), "F")

여기서 처음에 선언한 FPDF('L') 에서 "L"은 가로방향을 의미합니다.

이미지를 삽입하는 영역에서 숫자는 차례대로 x축 위치, y축 위치, 폭, 높이를 의미하는데, 이 값들을 입력하지 않으면 왼 쪽 맨위에 그림 이미지 크기 기준으로 삽입하게 되비니다.

기본적으로 만들어지는 PDF 파일은 A4 사이즈이고, 처음에 가로방향으로 설정을 했기 때문에 폭을 가득 채우게 330mm 사이즈로 이미지를 삽입하게 소스를 만들면 위와 같이 구성할 수 있습니다.

1. Summary

여기까지 만들어진 소스를 모드 모으면 아래와 같이 정리할 수 있습니다.

import os import re import win32com.client from docx2pdf import convert from fpdf import FPDF powerpoint = win32com.client.Dispatch("Powerpoint.Application") hwp = win32com.client.gencache.EnsureDispatch('HWPFrame.HwpObject') hwp.RegisterModule('FilePathCheckDLL', 'SecurityModule') pdf = FPDF('L') TDPath = "D:\\Data\\ToDo" RPath = "D:\\Data\\Result" #PPT to PDF files = [f for f in os.listdir(TDPath) if re.match('.\*[.]ppt', f)] for file in files: # PPT 파일을 PDF로 바꾸는 로직 deck = powerpoint.Presentations.Open(os.path.join(TDPath, file)) pre, ext = os.path.splitext(file) deck.SaveAs(os.path.join(RPath, pre + ".pdf"), 32) # formatType = 32 for ppt to pdf deck.Close() powerpoint.Quit() #Word to PDF files = [f for f in os.listdir(TDPath) if re.match('.\*[.]doc', f)] for file in files: # Word 파일을 PDF로 바꾸는 로직 pre, ext = os.path.splitext(file) convert(os.path.join(TDPath, file), os.path.join(RPath, pre + ".pdf")) #HWP to PDF files = [f for f in os.listdir(TDPath) if re.match('.\*[.]hwp', f)] for file in files: # HWP 파일을 PDF로 바꾸는 로직 hwp.Open(os.path.join(TDPath, file)) pre, ext = os.path.splitext(file) hwp.HAction.GetDefault("FileSaveAs\_S", hwp.HParameterSet.HFileOpenSave.HSet) hwp.HParameterSet.HFileOpenSave.filename = os.path.join(RPath, pre + ".pdf") hwp.HParameterSet.HFileOpenSave.Format = "PDF" hwp.HAction.Execute("FileSaveAs\_S", hwp.HParameterSet.HFileOpenSave.HSet); hwp.Quit() #Image to PDF files = [f for f in os.listdir(TDPath) if re.match('.\*([.]jpg|[.]png|[.]gif)', f)] for file in files: # img를 PDF로 바꾸는 로직 pdf.add\_page() pdf.image(os.path.join(TDPath,file), 0, 0, 330) pdf.output(os.path.join(RPath,"IMG2PDF.pdf"), "F")

**첨부파일** PDF\_F.py [파일 다운로드](http://blogattach.naver.net/b623aa1964533b8ba64525132ecdb4ca653751/20210828_8_blogfile/550sn_1630133232547_qcl879_py/%2FPDF_F.py)

하나하나로 보면 많이 복잡하거나 하지는 않은데, 사무용 문서 형태가 워낙 다양하다 보니 전체 소스는 상당히 길어졌습니 다.

그래도 한번에 여러 가지 형태의 사무용 문서 파일을 다루는 방법을 접해 볼 수 있었다는 점에서는 의미가 있지 않나 생각 해 봅니다.

※ 유튜브 영상을 만들면서 정리한 소스는 아래와 같습니다. 블로그글과 대동소이 합니다. **첨부파일**

PDF\_py\_YT.py [파일 다운로드](http://blogattach.naver.net/23b63f8c9acfc71b37d5b280bc5d275bf8ac55b677/20210902_31_blogfile/550sn_1630550015918_O180BM_py/%2FPDF_py_YT.py)