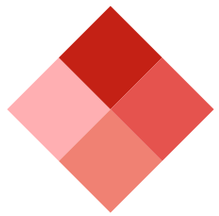


파이썬을 활용한 업무자동화

10회차: 컴퓨터 자동화로 업무자동화 완성하기

목차

10회차: 각자의 사내 프로그램 다루기 : 키보드, 마우스 제어 - 심화

- 복습
- 각자의 사내 프로그램 다루기 : 키보드, 마우스 제어 - 심화
  - 자동화로 엑셀 파일 만들기
  - 카카오톡 메신저로 나에게 해당 엑셀 파일 보내기
- 웹 자동화 심화 (3)
  - 11번가 상품들의 가격 엑셀에 누적하기

복습

지난 강의 때 뭐했지..?

웹 자동화 심화 및 키보드 마우스 제어

- 웹 자동화 심화: xpath 심화
  - 11번가 상품 검색 및 리스트 가져오기
- 키보드, 마우스 제어하기: 기본
  - 기본 라이브러리 소개

```
    # id가 thisClick을 포함하는 모든 태그를 가져옴
    results_list = driver.find_elements_by_xpath('//li[contains(@id,
\'thisClick\')]')

    # 위 가져온 태그들에 대해 반복
    for elem in results_list:
        # 제목, 금액, 판매자 부분을 가져옴
        title_tag = elem.find_element_by_class_name('info_tit')
        price_tag = elem.find_element_by_class_name('sale_price')
        mall_tag = elem.find_element_by_class_name('benefit_tit')
        print("[%s] %s : %s"%(mall_tag.text, title_tag.text, price_tag.text))
        # 태그 안에 고객응대우수 라는 문자가 포함된 span태그가 있는 경우
        # 고객응대우수 판매자를 출력하고, 그렇지 않은 경우 아무 동작하지않음
        try:
            span = elem.find_element_by_xpath('.//span[text()=\'고객응대우수\']')
            print("고객응대우수 판매자입니다.")
        except:
            pass
    # data-log-actionid-label이라는 속성값이 sellername인 a태그를 전부 가져와 출력
    seller_list = driver.find_elements_by_xpath('//a[@data-log-actionid-
label=\'sellername\']')
    for seller in seller_list:
        print(seller.text)

except Exception as e:
    print(e)
finally:
    driver.quit()
```

```python
import pyautogui

# 사이즈를 가져옴
screenWidth, screenHeight = pyautogui.size()
# 사이즈를 확인
print(screenWidth, screenHeight)

# 중앙으로 마우스 이동
pyautogui.moveTo(screenWidth/2, screenHeight/2)

# 오른쪽으로 마우스 이동
pyautogui.moveRel(100, None)

# 터미널 창으로 이동
pyautogui.moveTo(100, screenWidth/2)

# 키 입력
pyautogui.typewrite('keyboard test')

# 특수 키입력
pyautogui.press('enter')
pyautogui.press('enter')
# 이미지로 이동
point = pyautogui.locateCenterOnScreen('a.png')
if point:
    # MacOS의 Retina를 사용하는 경우 가상 픽셀로 해상도를 나타내기 때문에
    # /2가 필요
    pyautogui.click(x=point[0]/2,y=point[1]/2,clicks=2,interval=0.1)
    # 일반적인 경우에는 /2를 할 필요가 없음
    pyautogui.click(x=point[0],y=point[1],clicks=2,interval=0.1)
```

각자의 사내 프로그램 다루기

수업의 제약

- 사내에서 사용하는 프로그램은 너무 다양하고 강의에서 다루기 어려움
- 다음 과정을 키보드, 마우스 제어와 함께 실습
  - 기존 자동화를 통해 엑셀 파일 생성
  - 메신저로 해당 파일 전송
    - 카카오톡 나에게 보내기

각자의 사내 프로그램 다루기

라이브러리의 제약

- 캡쳐 그림을 통한 위치 찾기 제약
  - 그림의 인식율이 낮아 재시도 필요
  - 그림의 크기와 PC의 해상도에 따라 속도가 느릴 수 있음
  - **따라서, 가능한 최대한 좌표를 활용하기를 권장**

각자의 사내 프로그램 다루기

자동화 프로그램

만들 프로그램 동작

- 11번가 검색 결과 가져와 엑셀에 저장
- 카카오톡 실행 (로그인이 되어있다고 가정)
- 나에게 메시지 보내기 창을 열기
- 탐색기를 열어 엑셀 파일을 드래그해서 메시지 창으로 전달

각자의 사내 프로그램 다루기

자동화 프로그램

필요한 라이브러리 가져오기

- from subprocess import run
  - cmd, 터미널에서 실행할 수 있는 프로그램을 실행하는 함수
- import pyautogui as py
  - pyautogui.xxxx가 아닌 py.xxxx 로 사용할 수 있음

```
# 11번가로 부터 검색 결과를 가져오는 함수
from auto_11st import get_11st
# 엑셀에 저장하기 위한 클래스
from openpyxl import Workbook
# 프로그램을 실행시키기 위한 함수
from subprocess import run
# py 를 가져옴
# as 는 명명으로 py를 다쓰지 않고 py라는 이름으로 사용하기 위함
import pyautogui as py
# 대기를 위해 time 클래스를 가져옴
import time
```

각자의 사내 프로그램 다루기

자동화 프로그램

라이브러리 제약 해소

- 캡쳐 그림으로 위치를 찾을 때 못찾는 경우가 많음
- 자동으로 재시도 하는 함수 생성를 만들어 사용

```
# 이미지에 해당하는 위치를 못찾을 때가 많음
# 자동으로 3회 재시도 해주는 함수로 생성
def get_target(filename):
    # 3번 반복
    ret = 3
    point = None
    for idx in range(ret):
        point = py.locateCenterOnScreen(filename)
        # 정상적으로 위치를 찾으면 반복문을 빠져 나감
        if point:
            break
        time.sleep(0.5)
    return point
```

각자의 사내 프로그램 다루기

자동화 프로그램

11번가 검색 결과 엑셀에 저장

- 11번가 결과를 가져오는 함수를 다른 파이썬 파일에 작성
- auto_11st.py

```
def get_11st(keyword):
    results = []

    from selenium.webdriver.common.keys import Keys
    from selenium import webdriver
    import time

    # 브라우저 로드
    driver = webdriver.Chrome('chromedriver')

    try:
        # 11번가로 접속
        driver.get('http://11st.co.kr')

        # 검색어 입력을 위해 kwd를 가져옴
        elem = driver.find_element_by_name('kwd')
        # 검색어 초기화한 뒤 자전거 입력
        elem.clear()
        elem.send_keys(keyword)
        elem.send_keys(Keys.RETURN)
```

각자의 사내 프로그램 다루기

자동화 프로그램

11번가 검색 결과 엑셀에 저장

- 11번가 결과를 가져오는 함수를 다른 파이썬 파일에 작성

```
        # id가 thisClick을 포함하는 모든 태그를 가져옴
        results_list = driver.find_elements_by_xpath('//li[contains(@id,
\'thisClick\')]')

        # 위 가져온 태그들에 대해 반복
        for elem in results_list:
            # 제목, 금액, 판매자 부분을 가져옴
            title_tag = elem.find_element_by_class_name('info_tit')
            price_tag = elem.find_element_by_class_name('sale_price')
            mall_tag = elem.find_element_by_class_name('benefit_tit')
            # 제목, 금액, 판매자을 튜플 형태로 리스트에 추가
            results.append((mall_tag.text, title_tag.text, price_tag.text))

    except Exception as e:
        print(e)
    finally:
        driver.quit()

    return results
```

각자의 사내 프로그램 다루기

자동화 프로그램

11번가 검색 결과 엑셀에 저장

- 다시 auto.py로 돌아와 get_11st()를 호출하여 진행

```
# 자전거 검색 결과 리스트를 가져옴
result_list = get_11st('자전거')

# 뉴스 리스트를 엑셀에 저장
wb = Workbook()
ws = wb.active
ws.append(['판매자명','상품명','가격'])
for product in result_list:
    ws.append(product)

wb.save('/Users/Alghost/Desktop/auto_files/result.xlsx')
```

각자의 사내 프로그램 다루기

자동화 프로그램

카카오톡 실행

- py.PAUSE 설정을 하여 자동화 동작간에 1초로 설정
- py.size()는 화면의 크기를 확인하기 위해 출력
- MacOS는 open명령어 사용, Windows는 프로그램명 사용

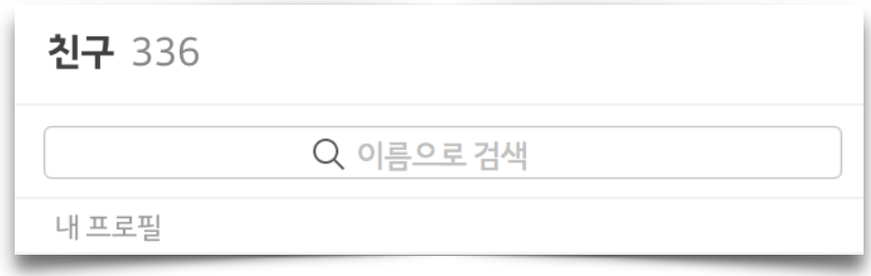
```
# GUI 자동화 동작간에 대기시간을 1초로 설정
py.PAUSE = 1

# 현재 화면 사이즈를 가져옴
screenWidth, screenHeight = py.size()
# 사이즈를 확인
print(screenWidth, screenHeight)

print("카카오톡을 실행합니다.")
# MacOS에서 프로그램을 실행
run(['open', '-a', 'kakaotalk'])
# MacOS에서 프로그램을 실행
# run(['C:\\Program Files (x86)\\Kakao\\KakaoTalk\\KaKaoTalk.exe'])
# 실행이 될때까지 대기
time.sleep(0.5)
```

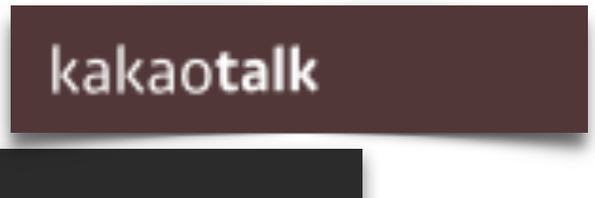
각자의 사내 프로그램 다루기

자동화 프로그램

나에게 메시지 보내기 창 열기

- 오른쪽 그림과 같이 카카오 헤더를 캡쳐
- 해당 이미지를 찾은 후 하단으로 이동
- 즉, 자신에게 메시지 전송






```
print("카카오톡에서 나에게 보내기 위한 설정을 합니다.")
# 친구 목록을 보기 위해 Command+1 을 누름
# Windows는 단축키가 아닌 image로 찾아가야함
py.hotkey('command', '1')

point = get_target('kakao_friends_header.png')
if not point:
    print("카카오 창을 찾을 수 없습니다.")
    quit()

# MacOS의 Retina를 사용하는 경우 가상 픽셀로 해상도를 나타내기 때문에
# /2가 필요
py.click(x=point[0]/2,y=point[1]/2+100)
py.press('enter')
```

각자의 사내 프로그램 다루기

자동화 프로그램

폴더 열기

- MacOS에서는 open 명령어를 통해 폴더를 열 수 있음
- Windows에서는 cmd명령어와 함께 start 명령어로 열어야함

```
# 엑셀 파일이 저장되어있는 폴더 위치를 찾음
# MacOS 에서 폴더 열기
run(['open', '/Users/Alghost/Desktop/auto_files'])
# Windows 에서 폴더 열기
# run(['cmd', '/c', 'start', 'C:\\Users\\Alghost\\Desktop\\auto_files'])
time.sleep(0.5)
```

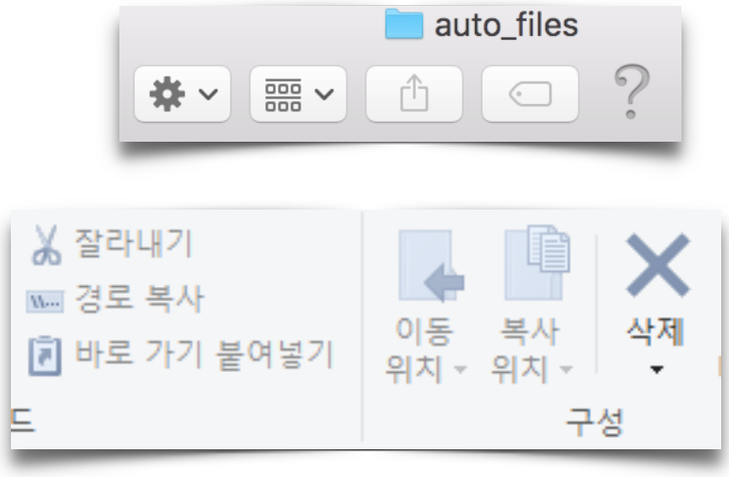
각자의 사내 프로그램 다루기

자동화 프로그램

폴더 열기

- 파인더의 상단을 캡쳐하여 header.png로 저장
- 윈도우는 탐색기의 상단을 캡쳐하여 header.png로 저장
- 헤더로부터 일정 좌표만큼 이동하여 파일의 위치를 찾음
- 파일의 위치를 file_point라는 변수에 저장

```
# 파인더의 헤더를 찾음
point = get_target('header.png')
if not point:
    print("파인더를 찾을 수 없습니다.")
    quit()

file_point = list(point)
file_point[0] -= 300
file_point[1] += 110
```



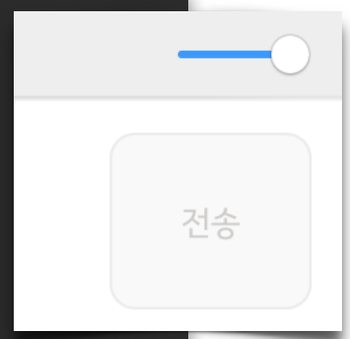
각자의 사내 프로그램 다루기

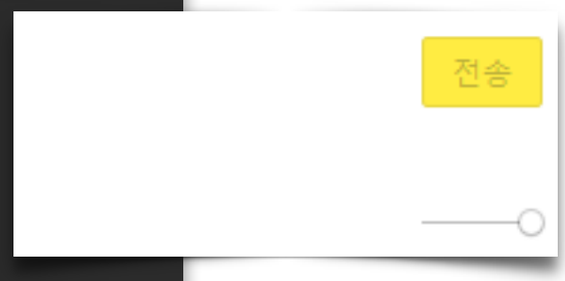
자동화 프로그램

폴더 열기

- 카카오톡의 전송 버튼을 캡쳐하여 send_btn.png로 저장
- 전송 버튼에서 왼쪽으로 좌표를 움직여 드래그할 위치를 target_point에 저장
- moveTo, dragTo를 활용하여 파일을 드래그하여 전송

```
# 카카오톡의 전송 버튼을 찾음
point = get_target('send_btn.png')
if not point:
    print("전송 버튼을 찾을 수 없습니다.")
    quit()

target_point = list(point)
target_point[0] -= 300

# 헤더로부터 55픽셀 아래로 내려와 파일을 클릭
py.moveTo(file_point[0]/2, file_point[1]/2)
py.dragTo(target_point[0]/2, target_point[1]/2, 2, button='left')
```



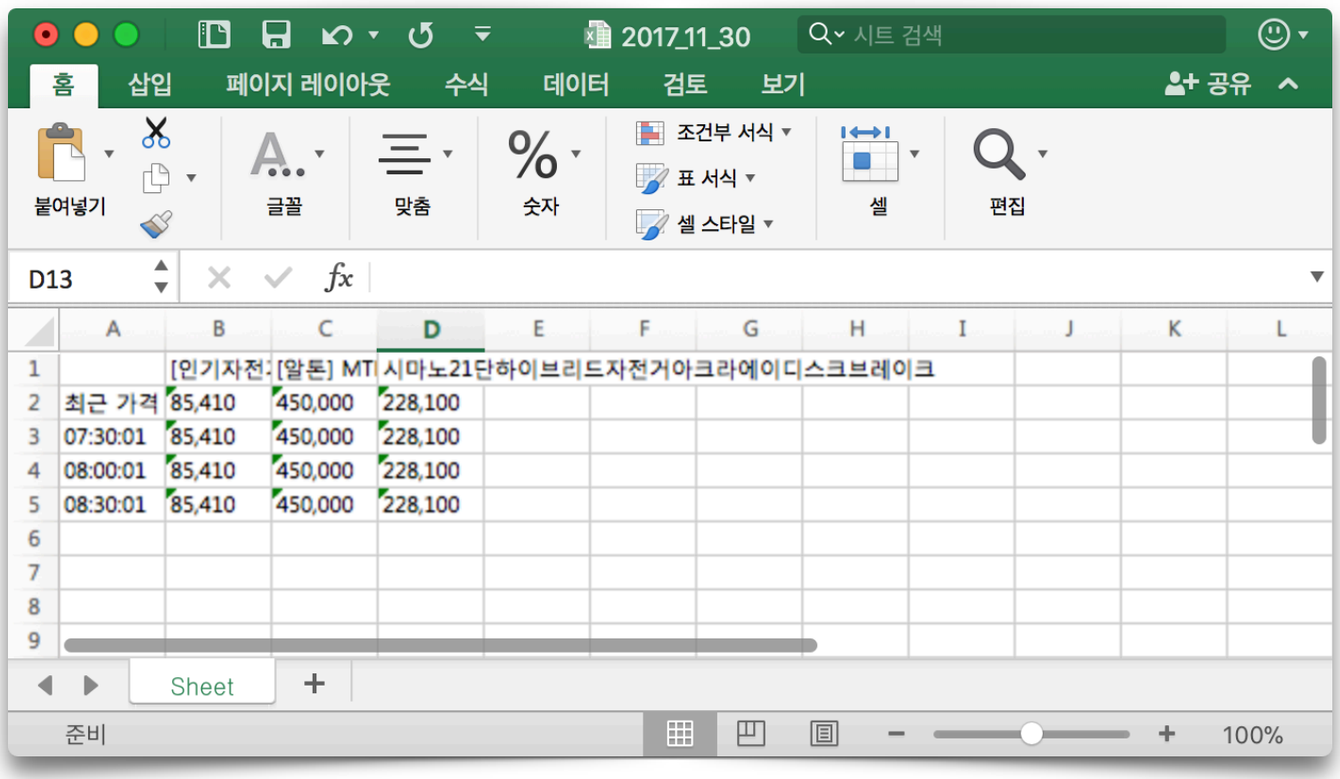


웹 자동화 심화 (3)

11번가 상품들의 가격 엑셀에 누적하기

자동화 과정

- 텍스트파일에 누적하고자 하는 상품들의 링크를 저장
- 텍스트파일로부터 링크들을 읽어옴
- 각 링크에서 정보 분석
- 날짜별 엑셀파일에 추가
  - 엑셀파일이 없는 경우 생성



웹 자동화 심화 (3)

11번가 상품들의 가격 엑셀에 누적하기

필요한 라이브러리 가져오기

- 새롭게 추가된 라이브러리
  - from os.path import exists
    - 파일이 존재하는지 유무를 확인하는 함수
  - from datetime import datetime
    - 날짜, 시간을 다루는 클래스
  - import string
    - 문자열 포맷을 다룸

```
# 필요한 라이브러리
from selenium import webdriver
from openpyxl import Workbook, load_workbook
# 파일이 존재하는지 확인하는 라이브러리
from os.path import exists
# 현재 시간을 가져오기 위한 라이브러리
from datetime import datetime
# 문자열 포맷을 다루기 위한 라이브러리
import string
```

웹 자동화 심화 (3)

11번가 상품들의 가격 엑셀에 누적하기

시간 가져오기 및 파일명 설정

- datetime.now()는 현재 시간을 가져오는 함수
- '%d_%d_%d.xlsx'%(a,b,c)
  - 문자열을 포맷팅하는 방법
  - %d: 숫자형 변수가 대체됨
  - %s: 문자열 변수가 대체됨
  - %.2f: 숫자형 변수중 소수점 표현시 소수점 갯수 지정

```
>>> a = 1.1123
>>> print('%.2f'%a)
1.11
>>> print('%.1f'%a)
1.1
>>> print('%.3f'%a)
1.112
```

```
# 현재 시간을 가져옴
today = datetime.now()
# 저장할 파일명을 날짜로 생성
today_file_name = '%d_%d_%d.xlsx'%(today.year,today.month,today.day)
```

웹 자동화 심화 (3)

11번가 상품들의 가격 엑셀에 누적하기

저장할 엑셀 가져오기

- 오늘 날짜로 된 엑셀 파일이 있는 경우 load_workbook
- 없을 경우 새로 생성
- 최근 가격을 2행에 유지하기 위해 A2에 최근 가격이라는 문자열 추가

```
# 파일이 존재하는 경우에는 load_workbook을 가져오고
# 없는 경우 새로운 파일 생성
if exists(today_file_name):
    result_xlsx = load_workbook(today_file_name)
else:
    result_xlsx = Workbook()

# 시트를 가져오고 'A2'셀에 최근 가격을 넣음
worksheet = result_xlsx.active
worksheet['A2'] = '최근 가격'
```

웹 자동화 심화 (3)

11번가 상품들의 가격 엑셀에 누적하기

크롬 띄우기

- 크롬이 작을 때 발생할 수 있는 문제를 없애기 위해 크기를 지정
- ChromeOptions클래스를 사용하여 지정할 수 있음

```
try:
    # 크롬에 옵션을 지정하기 위한 클래스 변수
    opts = webdriver.ChromeOptions()
    # 크롬의 크기 지정
    opts.add_argument('window-size=1920,1080')
    # 크롬드라이버의 경로와 옵션을 함께 지정하여 띄움
    driver = webdriver.Chrome('./chromedriver',chrome_options=opts)
```

웹 자동화 심화 (3)

11번가 상품들의 가격 엑셀에 누적하기

텍스트 파일로부터 링크 가져오기

- products.txt파일로부터 링크를 가져옴
  - readlines함수는 모든 내용을 가져오는데 행단위로 잘라서 리스트로 반환함
- 엑셀에 한 행씩 추가할 때 첫번째 컬럼에 시간을 넣음 => row 변수
  - strftime은 인자값으로 입력한 형태로 시간을 표기해줌
  - %H': 시, '%M': 분, '%S': 초

```
# 파일에 작성된 상품 링크들을 전부 가져옴
# readlines()는 모든 텍스트 내용을 행별로 리스트를 만들어 반환하는 함수
product_urls = open('products.txt', 'r').readlines()

# 엑셀에 추가될 행 정보를 담을 리스트 변수를 생성
# 이 변수 첫번째 값에 시간을 추가합니다.
row =[today.strftime("%H:%M:%S")]
```

웹 자동화 심화 (3)

11번가 상품들의 가격 엑셀에 누적하기

상품들에 대해 반복문을 수행

- 텍스트 파일로부터 가져온 링크들을 반복문으로 순회
- 각 상품 페이지로 이동
- 각 상품이름을 B2, C2, D2와 같은 셀에 추가하기 위해 인덱스 필요
- 추후 이 인덱스를 알파뱃으로 변경
  - 예: 1 => B, 2 => C, 3 => D

```
    # 상품 별로 컬럼을 변경하며 값을 넣기 위해 인덱스 값 생성
    column_idx = 1
    # 상품들에 대해 반복문을 수행
    for product in product_urls:
        # 상품 주소로 이동
        driver.get(product)
```

웹 자동화 심화 (3)

11번가 상품들의 가격 엑셀에 누적하기

상품 가격 가져오기

- 상품을 xpath로 가져옴
  - //div[contains(@class, 'prdc_default_info')]//strong[@class='sale_price']
  - class에 prdc_default_info가 포함된 div하위태그중 class가 sale_price인 strong 태그


```
        price_xpath = "//div[contains(@class,'prdc_default_info')]//
strong[@class='sale_price']"
        # 위 xpath로 가격정보를 가져옴
        price_tag = driver.find_element_by_xpath(price_xpath)
        # 가격을 엑셀에 추가하기 위해 행정보를 담은 변수에 추가
        row.append(price_tag.text)
```

가격정보를 가져오기 위한 xpath

웹 자동화 심화 (3)

11번가 상품들의 가격 엑셀에 누적하기

상품 가격 가져오기

- 상품을 xpath로 가져옴
  - //div[@class='heading']/h2
  - class가 heading인 div태그 하위의 h2태그


```
# 상품명 태그를 가져옴
title_tag = driver.find_element_by_xpath("//div[@class='heading']/h2")
```

웹 자동화 심화 (3)

11번가 상품들의 가격 엑셀에 누적하기

상품명 추가를 위해 컬럼 계산

- string.ascii_uppercase: 'ABCDE…Z'
  - 문자열이기 때문에 인덱싱으로 각 알파뱃 접근 가능
  - 즉, 0 => A, 1 => B로 변경가능
  - 이 값을 활용하여 시트의 첫번째 행에 상품명 추가
  - 이 값을 활용하여 시트의 두번째 행에 최근 가격값 추가

```
# 상품명을 B컬럼부터 1씩 증가하면서 넣기 위해
# 인덱스값을 알파뱃으로 변경해야함
# string.ascii_uppercase는 'ABCD...Z'문자열
# 1번째 문자가 B이고, C, D, E...로 인덱스를 사용하여 가져올 수 있음
column = string.ascii_uppercase[column_idx]
# 상품명을 'B1', 'C1', 'D1', ... 에 갱신
worksheet[column+'1'] = title_tag.text
worksheet[column+'2'] = price_tag.text
# 다음 컬럼에 내용을 쓰기 위해 컬럼 인덱스를 증가
column_idx += 1
```

웹 자동화 심화 (3)

11번가 상품들의 가격 엑셀에 누적하기

엑셀에 데이터 저장 후 프로그램 종료

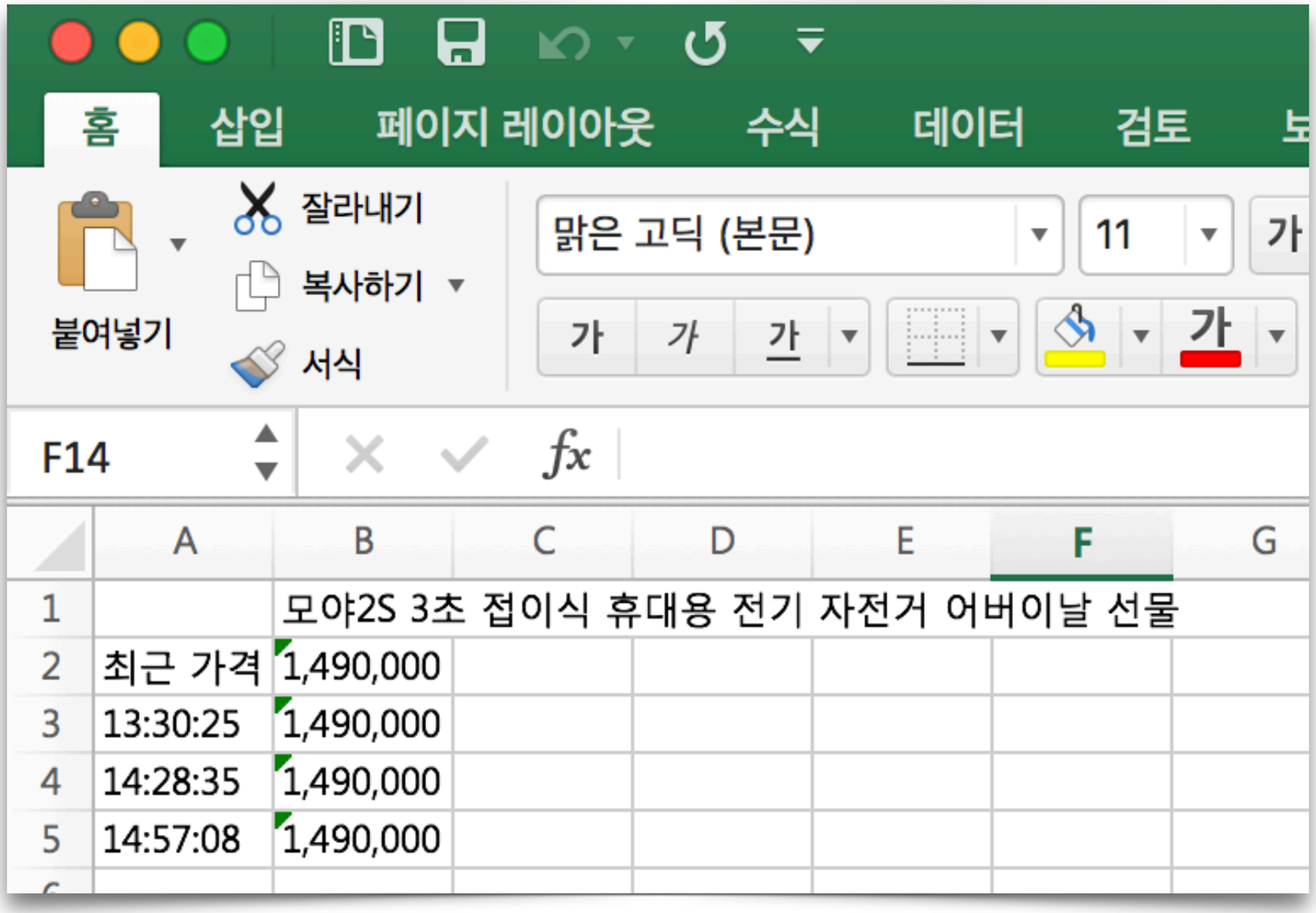
- 각 상품별로 가격이 추가된 row변수를 append
- 엑셀파일 저장

```
        # 엑셀에 한 행 추가 한 후 저장
        worksheet.append(row)
        result_xlsx.save(today_file_name)
    except Exception as e:
        print(e)
    finally:
        driver.quit()
```

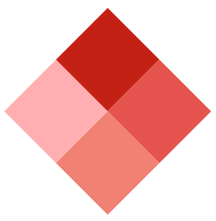
웹 자동화 심화 (3)

11번가 상품들의 가격 엑셀에 누적하기

프로그램 실행 결과



	A	B	C	D	E	F	G
1		모야2S 3초 접이식 휴대용 전기 자전거 어버이날 선물					
2	최근 가격	1,490,000					
3	13:30:25	1,490,000					
4	14:28:35	1,490,000					
5	14:57:08	1,490,000					



Good Bye

See you next time