

주요 학습 목표

- pyInstaller 이용하기
- 콘솔창 없이 실행 하기
- 1개의 exe 파일로 만들기

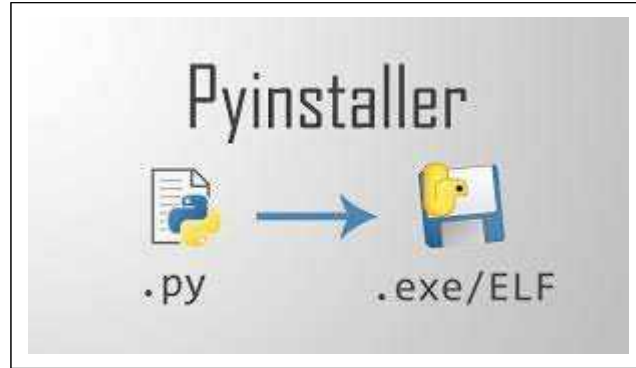
Chapter 05. pyInstaller를 이용한 실행파일 만들기

5.1 pyInstaller는 무엇인가?

파이썬은 매우 간결하다는 것을 알 수 있었다. 아주 중요한 규칙이라는 것이 4칸씩 들여 쓰는 정도만으로 가독성과 직관적인 문법을 지원 한다. 그러나 파이썬은 실행 파일이 아닌 소스코드 형태로 제공 되기 때문에 사용자는 앞서 설명한 과정들을 거쳐야 하고 최소한 Python.org를 방문 해서 python 인터프리터를 다운로드 받아 설치 해야한다. 그러나 이것도 만만치 않은 작업이다. 왜냐하면 첫째 파이썬 자체의 버전이 다른 경우가 발생 하고 설치되지 않은 패키지로 인해서 실행이 안되는 경우가 많다.

이러한 문제를 해결 하기 위해 pyInstaller가 개발 되었다. pyInstaller는 파이썬 코드를 효율적으로 1개의 실행 파일로 만들어 주는 역할을 함으로서 사용자가 별도의 과정 없이 바로 실행이 가능하도록 해줌으로서 파이썬 프로그램의 배포와 실행이 가능하도록 해줘 개발의 생산성을 매우 획기적으로 높일수 있다.

다음 [그림 5-1]은 소스코드를 실행 파일로 만드는 Pyinstaller에 대한 삽화이다.



[그림 5-1] 소스코드와 실행파일[출처 : <https://medium.com/>]

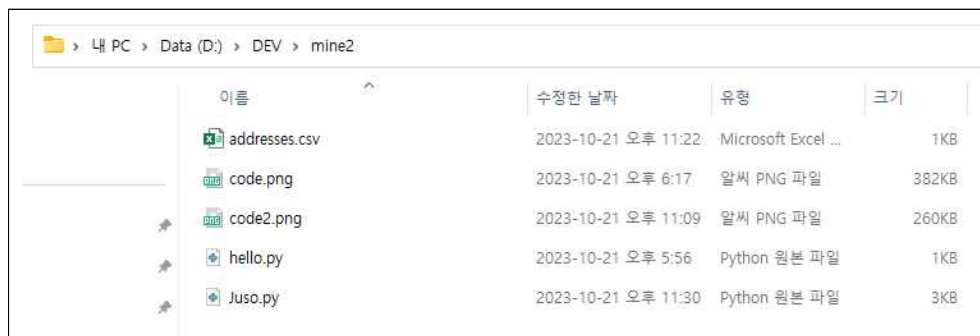
또한 Pyinstaller는 윈도는 물론 리눅스 등에서도 동작이 가능한 실행 파일을 생성할 수 있고, 데이터 파일과 같은 종속성을 포함 시켜 실행 파일 1개로 만드는 작업까지 가능하다.

5.2 탐색기를 이용하여 작업 환경 복귀 하기

여기서 잠깐 살펴보고 넘어갈 것은 컴퓨터를 리부팅 하였거나 하루가 지났거나 여러 가지 사정으로 프로그램을 다음 날 하게 되었을 때를 가정하여 VSCODE와 가상환경으로 복귀하는 방법에 대하여 설명한다.

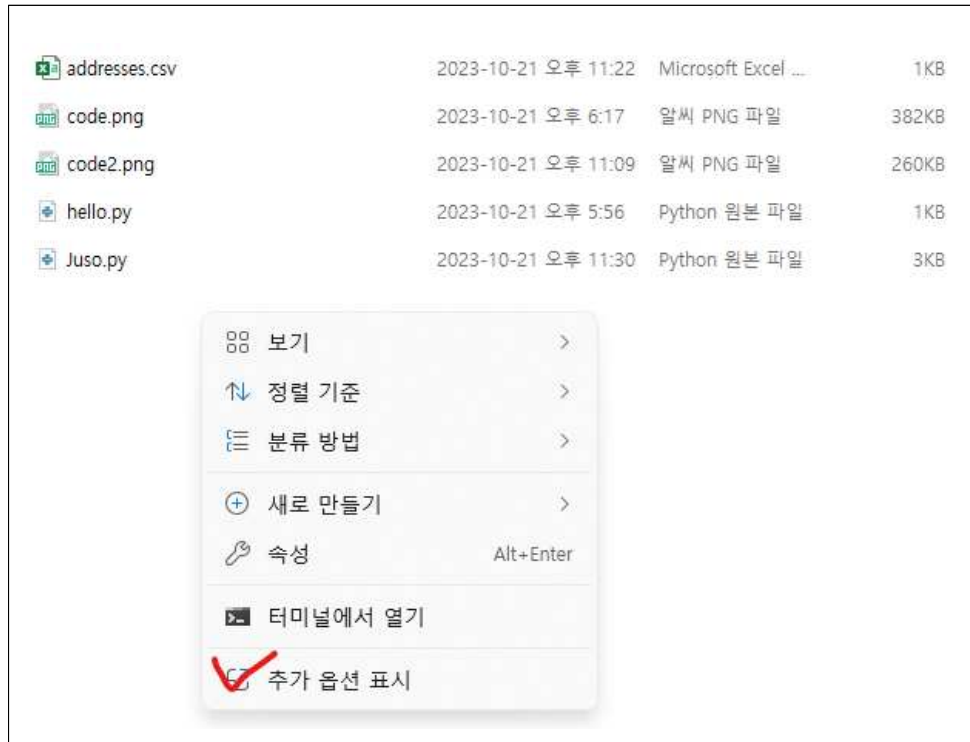
1. 작업 폴더로 이동

다음 [그림 5-2]은 탐색기를 이용하여 작업 폴더로 이동한 모습이다.



[그림 5-2] 탐색기를 이용하여 작업폴더로 이동

2. 작업 폴더에서 오른쪽 클릭 하여 vscode로 열기를 선택
만약 vscode로 열기가 보이지 않는 경우 다음의 [그림 5-3]과 같이
추가 옵션표시를 누른다.



[그림 5-3] 추가 옵션

다음의 [그림 5-4]는 추가 옵션이 표시된 Code로 열기를 선택하기 위한
마지막 단계이다.



[그림 5-4] 추가 옵션과 Code로 열기

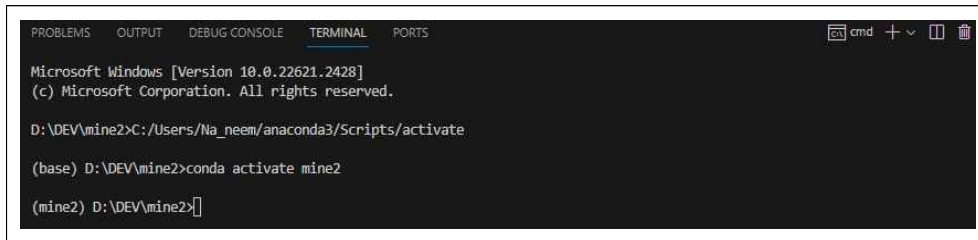
3. 열려진 터미널 창을 닫고 mine2 터미널 창을 활성화 한다.
다음의 [그림 5-5]와 같이 터미널 창의 오른쪽 휴지통 아이콘을 클릭하여 현재 열려 있는 창을 모두 닫는다.



[그림 5-5] 열려진 터미널 창 닫기

4. mine2 터미널 열기

마지막으로 mine2 터미널 창을 열기 위해서 CTRL+SHIFT+~을 눌러 다음 [그림 5-6]과 같은 mine2 터미널 창을 열수 있다.



```
Microsoft Windows [Version 10.0.22621.2428]
(c) Microsoft Corporation. All rights reserved.

D:\DEV\mine2>C:/Users/Na_neem/anaconda3/Scripts/activate

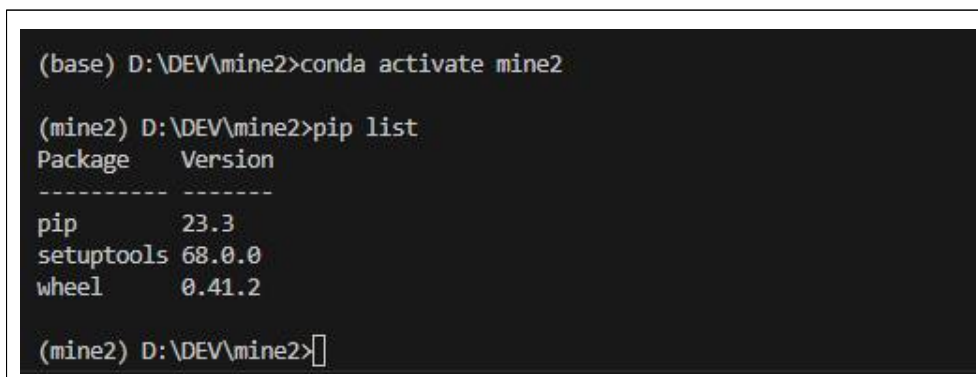
(base) D:\DEV\mine2>conda activate mine2

(mine2) D:\DEV\mine2>
```

[그림 5-6] mine2 터미널 창

5. 현재 설치된 패키지 리스트 보기

이전 작업에서 계속 사용해 오던 mine2 가상 환경에는 별도의 package를 설치 하지 않았으므로 설치된 패키지 리스트를 호출 하면 다음의 [그림 5-7]과 같은 결과가 나온다.



```
(base) D:\DEV\mine2>conda activate mine2

(mine2) D:\DEV\mine2>pip list
Package      Version
-----
pip          23.3
setuptools   68.0.0
wheel        0.41.2

(mine2) D:\DEV\mine2>
```

[그림 5-7] pip list 보기

이제 프로그램 개발을 위한 가상환경을 다시 불러오는 과정까지 복습을 한 셈이다.

5.3 pyInstaller설치

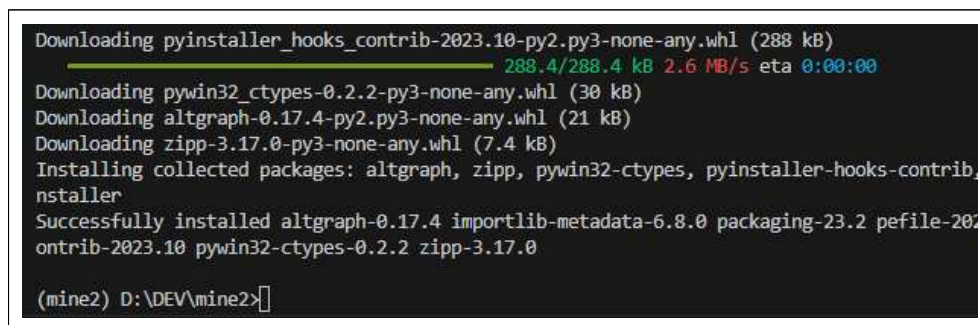
pyinstaller는 pip을 통해서 설치할 수 있다. windows, mac, linux에서 설치가 가능하다. 터미널 창에서 다음의 [그림 5-8]과 같이 입력해보자.



```
pip install pyinstaller
```

[그림 5-8] pyInstaller 의 Install 명령

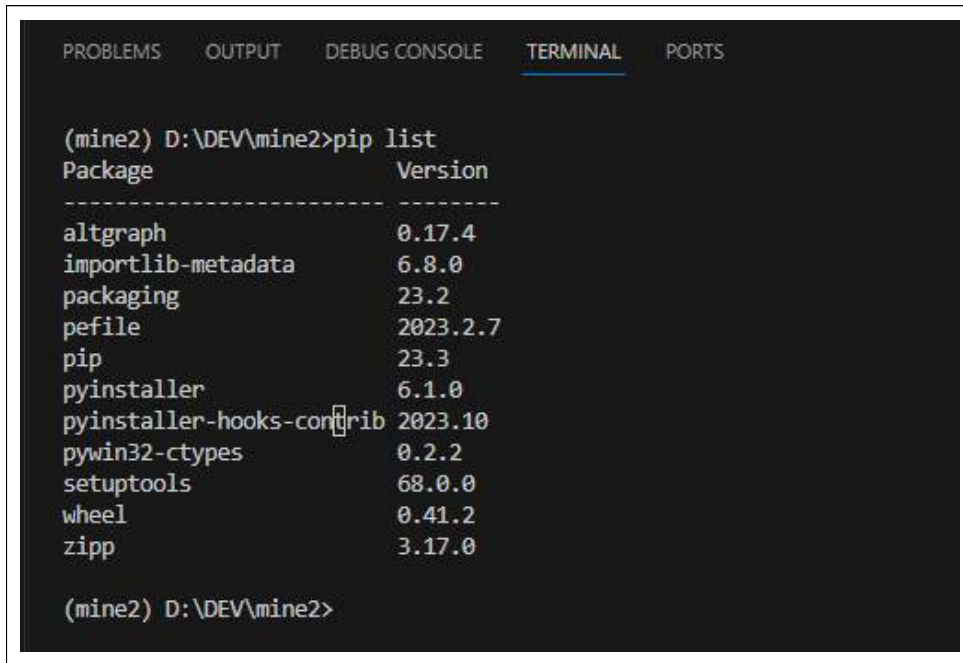
다음의 [그림 5-9]는 인스톨이 완료된 화면이다.



```
Downloading pyinstaller_hooks_contrib-2023.10-py2.py3-none-any.whl (288 kB)
 288.4/288.4 kB 2.6 MB/s eta 0:00:00
Downloading pywin32_ctypes-0.2.2-py3-none-any.whl (30 kB)
Downloading altgraph-0.17.4-py2.py3-none-any.whl (21 kB)
Downloading zipp-3.17.0-py3-none-any.whl (7.4 kB)
Installing collected packages: altgraph, zipp, pywin32-ctypes, pyinstaller-hooks-contrib,
pyinstaller
Successfully installed altgraph-0.17.4 importlib-metadata-6.8.0 packaging-23.2 pefile-2023.10
pyinstaller-hooks-contrib-2023.10 pywin32-ctypes-0.2.2 zipp-3.17.0
(mine2) D:\DEV\mine2>
```

[그림 5-9] pyInstaller Install 결과

다시 한번 터미널에서 pip list를 입력해서 설치된 패키지를 살펴보자. 다음의 [그림 5-10]은 pip list의 결과이다.



```
(mine2) D:\DEV\mine2>pip list
Package                Version
-----
altgraph               0.17.4
importlib-metadata     6.8.0
packaging              23.2
pefile                 2023.2.7
pip                    23.3
pyinstaller            6.1.0
pyinstaller-hooks-contrib 2023.10
pywin32-ctypes         0.2.2
setuptools             68.0.0
wheel                  0.41.2
zipp                   3.17.0

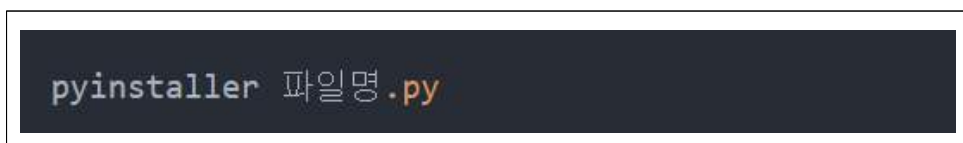
(mine2) D:\DEV\mine2>
```

[그림 5-10] pip list 결과

pyInstaller와 종속된 라이브러리가 설치된 것을 볼 수 있다. 그럴 리가 없겠지만 우리는 가상환경에서 새로 시작한 경우이기 때문에 다음과 같은 경우는 생기지 않을 것이다. 만약 pathlib과 같은 다른 패키지가 설치된 경우 아주 가끔 상호 충돌로 인해서 설치가 안되는 경우도 발생한다. 그래서 늘 가상환경은 매우 중요한 것이라는 것을 다시 한번 강조한다.

5.4 pyInstaller 사용방법

py파일에서 exe로 만드는 방법은 예상외로 매우 간단하다. 터미널에서 다음 그림과 같이 입력한다.



```
pyinstaller 파일명.py
```

[그림 5-11] pyinstaller 사용 원형

다음의 [그림 5-12]는 우리가 작성한 주소록 프로그램을 exe 실행 파일로 만드는 과정을 보여 주고 있다.

```
(mine2) D:\DEV\mine2>pyinstaller Juso.py
770 INFO: PyInstaller: 6.1.0
771 INFO: Python: 3.9.18 (conda)
789 INFO: Platform: Windows-10-10.0.22621-SP0
790 INFO: wrote D:\DEV\mine2\Juso.spec
802 INFO: Extending PYTHONPATH with paths
['D:\DEV\mine2']

.....

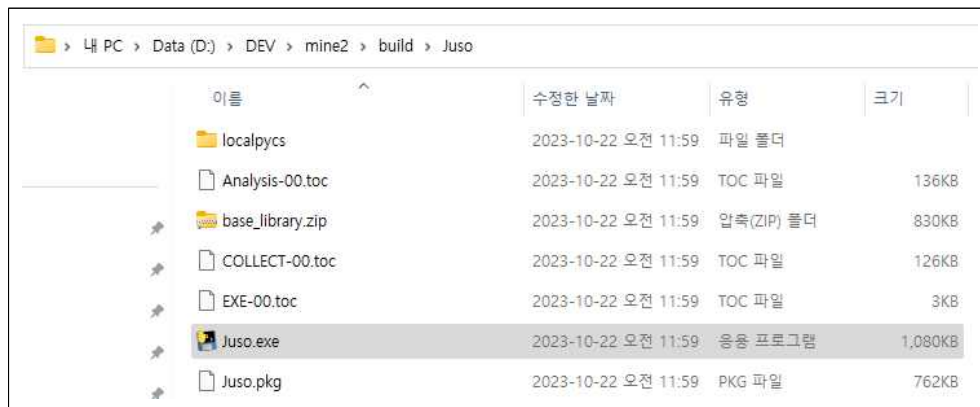
12137 INFO: Building EXE from EXE-00.toc completed successfully.
12155 INFO: checking COLLECT
12156 INFO: Building COLLECT because COLLECT-00.toc is non existent
12157 INFO: Building COLLECT COLLECT-00.toc
14210 INFO: Building COLLECT COLLECT-00.toc completed successfully.

(mine2) D:\DEV\mine2>
```

[그림 5-12] 주소록 실행 파일 만들기

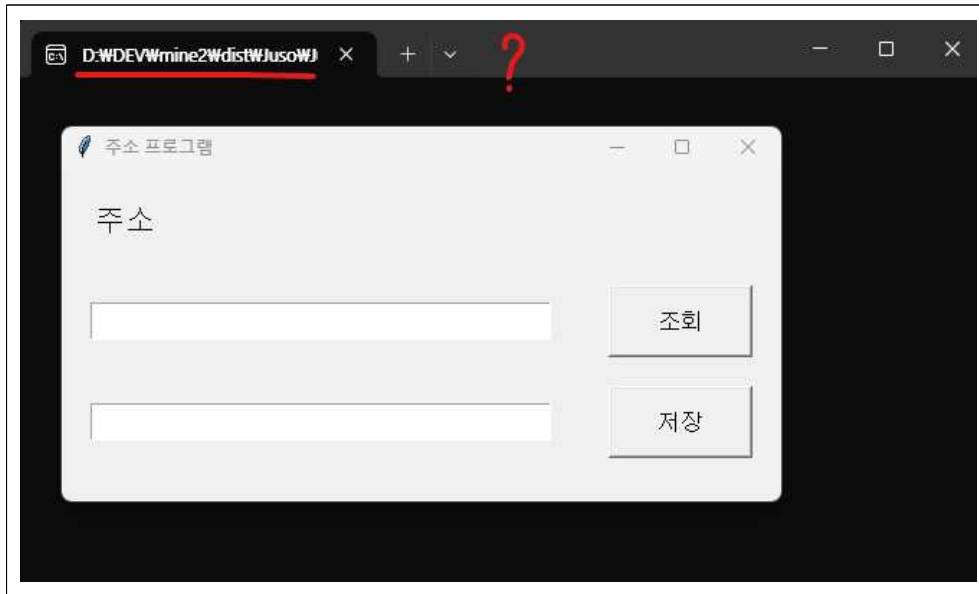
그렇다면 이것으로 끝난 것인가? 물론 현재 작업하던 폴더에 가면 다음 그림과 같은 폴더가 새로 생성된 것을 볼수 있을 것이다. 이 폴더를 통째로 복사해서 다른 컴퓨터에 옮긴다면 아무런 설치 과정 없이 실행은 가능하다.

다음의 [그림 5-13]은 최초 생성된 실행 파일이다.



[그림 5-13] 최초 생성

이렇게 최초 생성된 파일을 탐색기에서 더블 클릭해서 실행시켜 보자. 다음 그림과 같이 실행은 잘 되었으나 터미널 창이 함께 보이는 그런 불편함이 있다. 다음의 [그림 5-14]는 최초 생성 프로그램의 실행 결과이다.



[그림 5-14] 최초 생성 프로그램의 실행 결과

그 뿐만이 아니라 사실 폴더 내부를 들여다 보면 더욱 복잡하다. 다음의 [그림 5-15] 에서 보는 것과 같이 _internal 폴더에 들어가면 수도 없이 많은 dll과 기타 여러 가지 파일들이 있는 것을 볼 수 있다. 이들은 실행 파일 juso.exe가 실행되기 위해 필요한 여러 패키지들로서 이 중에서 한 두 개만 빠지더라도 역시 juso.exe 파일은 실행이 되지 않는다. 이러한 불편을 감수할 것인가? 다음의 [그림 5-15]는 실행 파일 폴더 내부 구조이다.

이름	수정된 날짜	유형	크기
localpycs	2023-10-22 오전 11:59	파일 폴더	
Analysis-00.toc	2023-10-22 오전 11:59	TOC 파일	136KB
base_library.zip	2023-10-22 오전 11:59	압축(ZIP) 폴더	830KB
COLLECT-00.toc	2023-10-22 오전 11:59	TOC 파일	126KB
EXE-00.toc	2023-10-22 오전 11:59	TOC 파일	3KB
Juso.exe	2023-10-22 오전 11:59	응용 프로그램	1,080KB
Juso.pkg	2023-10-22 오전 11:59	PKG 파일	762KB
PKG-00.toc	2023-10-22 오전 11:59	TOC 파일	2KB
PYZ-00.pyz	2023-10-22 오전 11:59	PYZ 파일	748KB
PYZ-00.toc	2023-10-22 오전 11:59	TOC 파일	9KB
Tree-00.toc	2023-10-22 오전 11:59	TOC 파일	109KB
Tree-01.toc	2023-10-22 오전 11:59	TOC 파일	11KB
Tree-02.toc	2023-10-22 오전 11:59	TOC 파일	1KB
warn-Juso.txt	2023-10-22 오전 11:59	텍스트 문서	2KB
xref-Juso.html	2023-10-22 오전 11:59	Microsoft Edge H...	214KB

[그림 5-15] 실행 파일 폴더 내부

5.5 pyInstaller 설정 옵션

모든 옵션에 대한 설명은 [pyinstaller^{21\)} 홈페이지](https://pyinstaller.org/en/stable/usage.html)를 참조 하거나 chatGPT의 도움을 받도록 하자. 또한 Python 3.9를 기준으로 작성 되었으므로 만약 이후 버전을 사용하거나 현재까지 낮은 버전을 사용하는 경우 예기치 않은 오류가 발생 할 수 있으니 주의 해야 한다.

몇가지 유용한 옵션만 설명하기로 한다.

1. 콘솔창 출력되지 않도록 하기

옵션 명 `-w`를 추가함으로써 콘솔 창은 사라진다.

```
(mine2) D:\DEV\mine2>pyinstaller -w juso.py
```

[그림 5-16] -w 옵션

21) <https://pyinstaller.org/en/stable/usage.html>

2. 1개의 파일로 만들기

옵션 명 `-F`를 추가함으로써 여러 가지 패키지와 인터프리터를 포함한 모든 파일을 1개의 `exe` 파일로 모아서 만들어 준다. 장점은 설명하지 않아도 될 것 같고 단점은 이렇게 하면 실행 시작시간이 길어진다. 10초 20초 30초 이상의 시간이 소요 될수 있다.

다음 그림은 두 개의 옵션의 사용 예이다.

```
(mine2) D:\DEV\mine2>pyinstaller -w -F Juso.py
```

[그림 5-17] 2개의 옵션

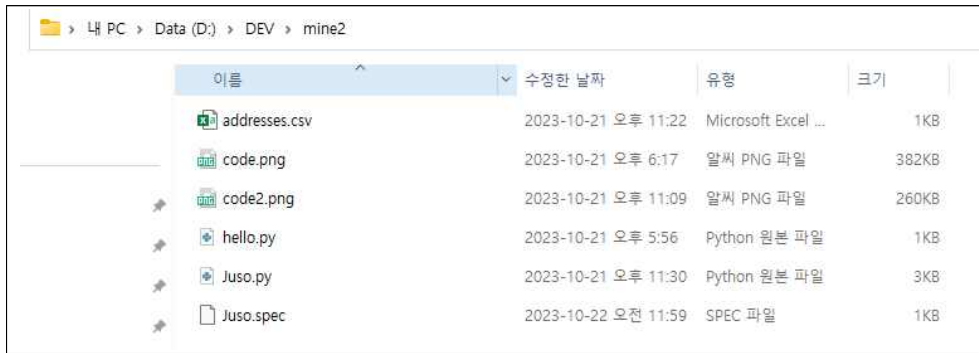
3. 프로그램을 대표하는 아이콘 지정하기

프로그램을 대표하는 아이콘을 지정하고자 할 때 사용하는 옵션이다. 다음 그림은 3개의 옵션의 사용 예이다. 여기서 내아이콘.ico는 실제 아이콘의 파일명을 입력 해주면 된다.

```
(mine2) D:\DEV\mine2>pyinstaller -w -F --icon=내아이콘.ico Juso.py
```

[그림 5-18] 아이콘 지정

이상으로 몇 가지 옵션을 사용하는 방법을 알아보았다. 이미 만들어진 파일들을 제거하기 위해서 작업 폴더 아래 `dist`와 `build` 폴더를 삭제하고 최종적으로 콘솔과 1파일로 만드는 옵션을 사용해서 `exe` 파일을 만들어 보자. 다음 그림은 이전 생성된 폴더들을 삭제한 모습이다.



[그림 5-19] 폴더 삭제

다음 그림은 콘솔을 사용하지 않도록 하고 1개의 파일로 생성하는 실행 결과이다.

```
(mine2) D:\DEV\mine2>pyinstaller -w -F juso.py
877 INFO: PyInstaller: 6.1.0
877 INFO: Python: 3.9.18 (conda)
896 INFO: Platform: Windows-10-10.0.22621-SP0
899 INFO: wrote D:\DEV\mine2\juso.spec
908 INFO: Extending PYTHONPATH with paths
['D:\DEV\mine2']
```

[그림 5-20] 최종 명령

다음 그림은 1개의 파일로 만들어진 실행 파일이다.



[그림 5-21] 1개의 실행 파일

이전 실행 파일에 비해 크기가 대폭 증가한 것을 알 수 있다. 아이콘은 별도로 지정하지 않아 Python Default 아이콘이 사용되었다.

다음의 [그림 5-22]는 최종 실행 결과를 나타내고 있다. 콘솔 창이 사라지고 멋지게 실행 파일이 동작하는 것을 확인하자.



[그림 5-22] 최종 완성된 exe 파일 실행 결과

이상으로 Pyinstaller 패키지를 이용한 1개의 실행 파일을 콘솔 없이 실행 되도록 하는 작업을 하였다. 내가 만든 프로그램을 누군가에게 배포해야 한다면 최소한 이렇게 패키징을 해서 전달하면 될 것이다.


5.6 pyInstaller 자주 나타나는 오류

1. 실행 파일이 작동하지 않는 경우

pyinstaller로 생성한 실행 파일이 정상적으로 작동하지 않는 경우가 있다. 이 경우, 대부분의 경우 실행 파일이 필요로 하는 파일이나 라이브러리가 빠져 있어서 발생하는 문제이다.

이 경우, 실행 파일을 생성할 때 `-D` 옵션을 사용하여 실행 파일과 함께 필요한 파일들을 함께 배포한다. `-D` 옵션은 `--onedir` 옵션으로 대체하여 사용할 수도 있다.

다음 그림은 `-D` 옵션을 사용하는 예이다.



```
(mine2) D:\DEV\mine2>pyinstaller -D juso.py
```

[그림 5-23] -D 옵션으로 오류 수정

-D 로 해결되면 좋겠지만 그렇지 않으면 골치 아파진다. 오류를 직접 찾아가야 한다. py파일에 오류가 있으면 해결하면 그만이지만 그렇지 않으면 버전 문제일 수 있다.

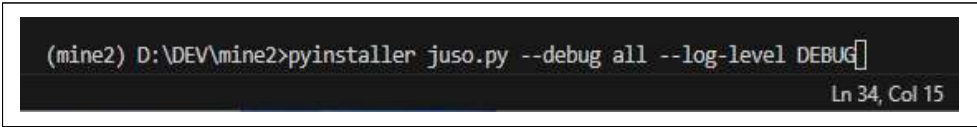
버전 문제인 경우는 해당 가상환경의 패키지 업그레이드를 하여야 한다. 특히, numpy, pandas는 pyinstaller 필수 패키지라서 업데이트가 되어있어야 한다.

2. 실행 파일 크기가 너무 큰 경우

pyinstaller로 생성된 실행 파일의 크기가 너무 커서 배포에 어려움을 느끼는 경우가 있다. 이 경우, 실행 파일에서 사용하지 않는 라이브러리나 파일들을 제거하여 크기를 줄일 수 있다.

먼저, 실행 파일에서 사용되는 라이브러리와 파일들을 확인한다.

다음의 [그림 5-24]는 사용 라이브러리 보기로서 명령어의 예이다.




```
(mine2) D:\DEV\mine2>pyinstaller juso.py --debug all --log-level DEBUG
```

Ln 34, Col 15

[그림 5-24] 사용 라이브러리 보기

다음의 [그림 5-25]는 모듈과 파일 제거화면으로, 모듈과 파일을 제거하기 위한 명령어이다. 지금 만들고 있는 주소록 프로그램에서는 여러 가지 패키지를 사용한 것이 아니기 때문에, 불필요하고, 제외할 수도 없지만 나중에 위해서 다음과 같이 예를 남겨 놓는다.



```
(mine2) D:\DEV\mine2>pyinstaller juso.py --debug all --exclude-module 제외할모듈명 --exclude 제외할 파일명
```

Ln 34, Col 15 Spaces: 4 UTF-8 CRLF Python 3


[그림 5-25] 모듈과 파일 제거

앞의 명령어를 실행하면, 실행 파일에서 사용되는 모든 라이브러리와 파일들이 출력된다. 이 중에서 사용하지 않는 라이브러리와 파일들을 제거하고 지정한 모듈과 파일을 제외하여 실행 파일을 생성한다.

pyInstaller 이외에도 cx_Freeze, py2exe, PyQxidizer 등 파이썬 프로그램을 실행 파일로 변환 해주는 도구들은 여러 가지가 있으니 참고 하기로 한다.

특히 pyInstaller를 GUI 프로그램을 구현한 프로그램이 나와 있다. 이는 여러 가지 옵션을 지정 하는 것을 편리하게 해주는 것 같다. 간단하게 설치와 실행 화면을 남겨 놓겠다. 사용법은 각자 연구해 보기 바란다.

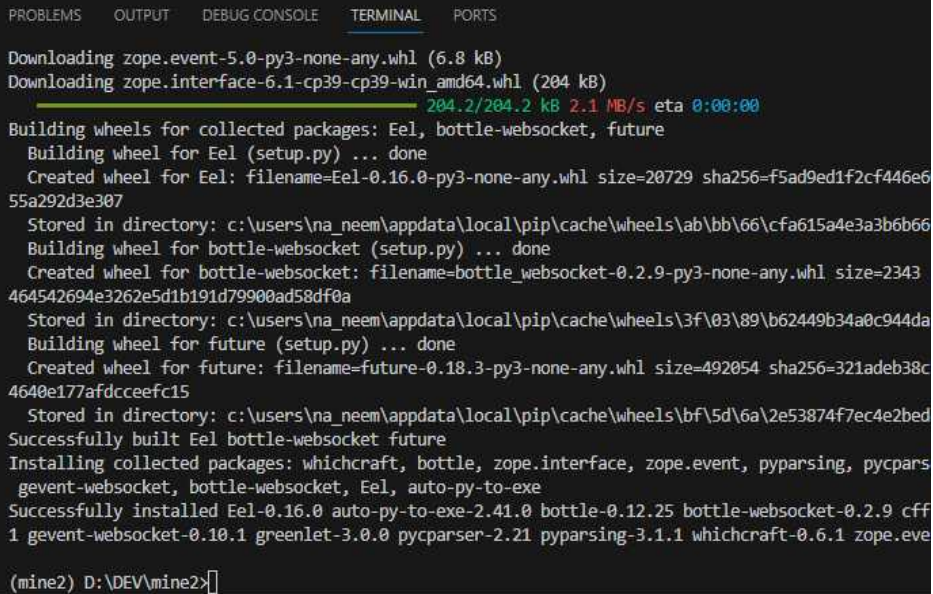
다음의 [그림 5-26]은 인스톨 명령을 보여주고 있다.



```
(mine2) D:\DEV\mine2>pip install auto-py-to-exe
```

[그림 5-26] 인스톨 명령

다음 [그림 5-37]은 [그림 5-26]의 인스톨 명령에 대한 실행 결과 이다.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Downloading zope.event-5.0-py3-none-any.whl (6.8 kB)
Downloading zope.interface-6.1-cp39-cp39-win_amd64.whl (204 kB)
204.2/204.2 kB 2.1 MB/s eta 0:00:00
Building wheels for collected packages: Eel, bottle-websocket, future
Building wheel for Eel (setup.py) ... done
Created wheel for Eel: filename=Eel-0.16.0-py3-none-any.whl size=20729 sha256=f5ad9ed1f2cf446e655a292d3e307
Stored in directory: c:\users\na_neem\appdata\local\pip\cache\wheels\ab\bb\66\cfa615a4e3a3b6b66
Building wheel for bottle-websocket (setup.py) ... done
Created wheel for bottle-websocket: filename=bottle_websocket-0.2.9-py3-none-any.whl size=2343464542694e3262e5d1b191d79900ad58df0a
Stored in directory: c:\users\na_neem\appdata\local\pip\cache\wheels\3f\03\89\b62449b34a0c944da
Building wheel for future (setup.py) ... done
Created wheel for future: filename=future-0.18.3-py3-none-any.whl size=492054 sha256=321adeb38c4640e177afdcceefc15
Stored in directory: c:\users\na_neem\appdata\local\pip\cache\wheels\bf\5d\6a\2e53874f7ec4e2bed
Successfully built Eel bottle-websocket future
Installing collected packages: whichcraft, bottle, zope.interface, zope.event, pyparsing, pycparser, gevent-websocket, bottle-websocket, Eel, auto-py-to-exe
Successfully installed Eel-0.16.0 auto-py-to-exe-2.41.0 bottle-0.12.25 bottle-websocket-0.2.9 cffi-1.14.6 gevent-websocket-0.10.1 greenlet-3.0.0 pycparser-2.21 pyparsing-3.1.1 whichcraft-0.6.1 zope.event-5.0
(mine2) D:\DEV\mine2>
```

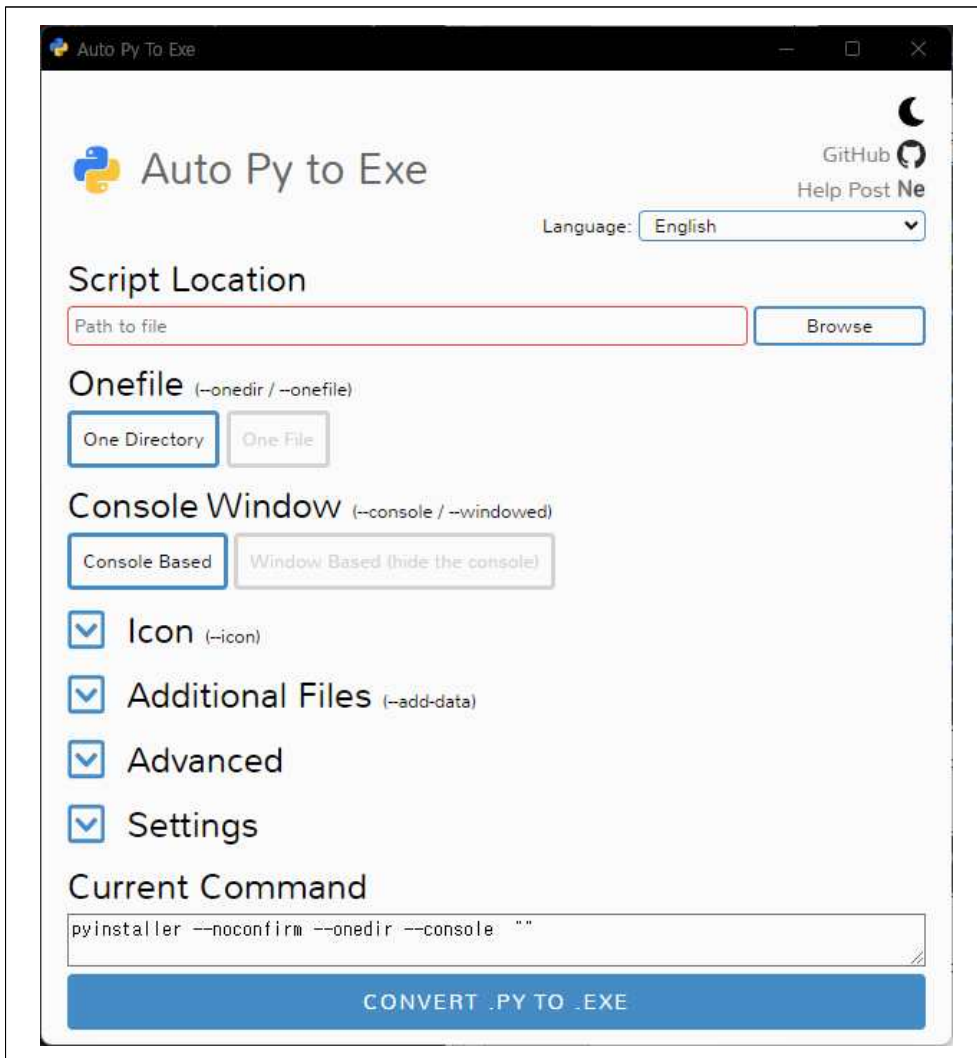
[그림 5-27] 인스톨 결과

다음의 [그림 5-28]은 패키지 인스톨이 완료 된후 실제 실행 명령이다.



[그림 5-28] 실행 명령

다음의 [그림 5-29]는 [그림 5-28]의 실행 명령으로 실행된 프로그램의 실행 화면이다.



[그림 5-29] 실행 화면

주요 학습 목표

- 배포용 setup파일 만들기
- 프로그램 개발의 최종 목적지 배포
- 안정성 있는 셋업 프로그램

Chapter 06. 배포용 setup파일 만들기

6.1 Install Factory는 무엇인가?

이 프로그램이 출시된 것은 오래 되었지만 매우 가볍고 사용하기 편리함 때문에 이제껏 사용되고 있다. setup 파일을 만드는 방법은 다음과 같은 파일 들이 있다. 이것은 참고로 알아 두기로 한다.

"InstallShield"나 "Inno Setup", "NSIS (Nullsoft Scriptable Install System)"와 같은 인스톨러 생성 도구도 있다.

프로그램 실행 파일 (exe 등)와 프로그램 보조 파일 (여러 음성, 사진, jre 등)이 같은 폴더에 있는 상태여야만 실행이 되고 그 상태로 배포해야만 하는 상황이 생길 때 유용하게 사용할 수 있는 프리웨어이다.

InstallFactory를 사용하면 자동으로 원하는 경로에 프로그램 파일을 설치하는 setup파일을 만들 수 있다.

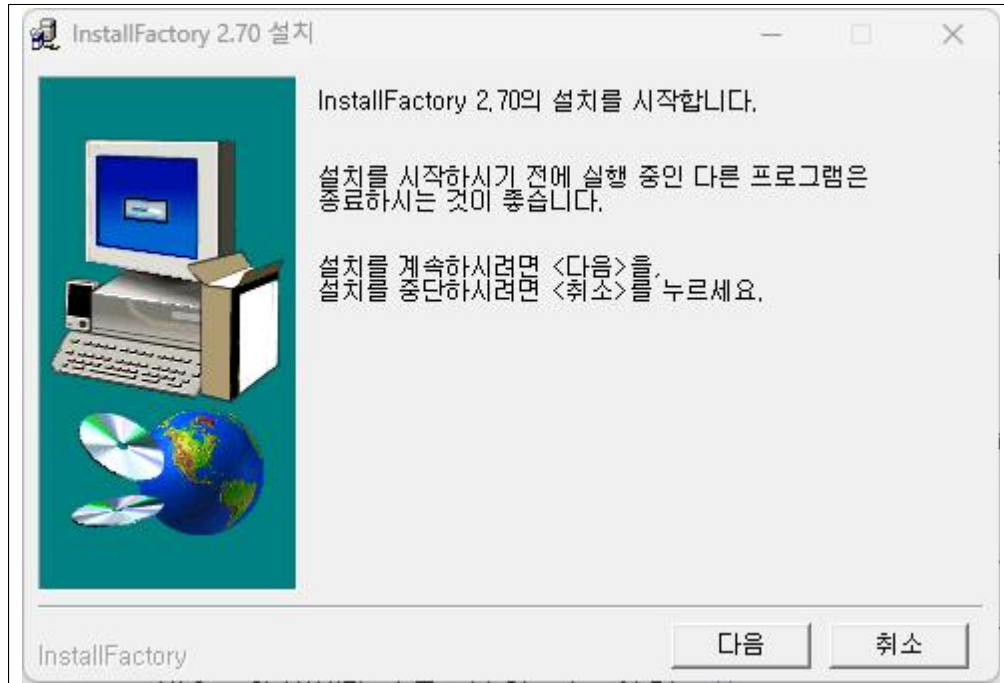
프로그램 설치 파일과 사용 매뉴얼은 다음 링크를 따라 가면 올려져 있다.

https://drive.google.com/drive/folders/1ExMxLS4Umya5kqgkqRTYLjmKSWSiF4fY?usp=share_link

[그림 6-1] install Factory 2.7 다운로드

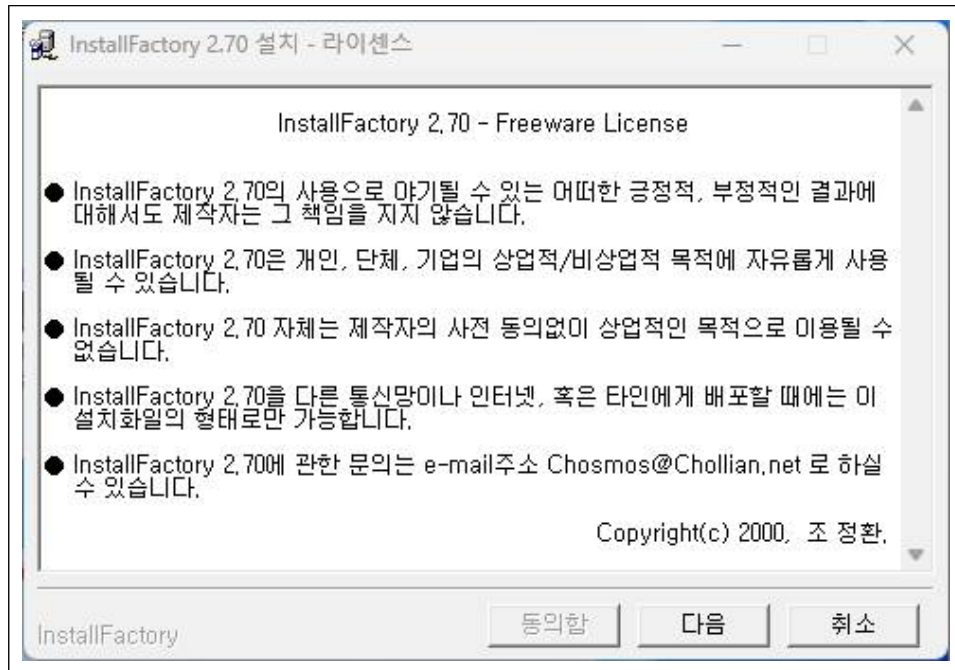
6.2 InstallFactory 2.7 설치

다음의 [그림 6-2] InstallFactory 프로그램 설치 화면이다.



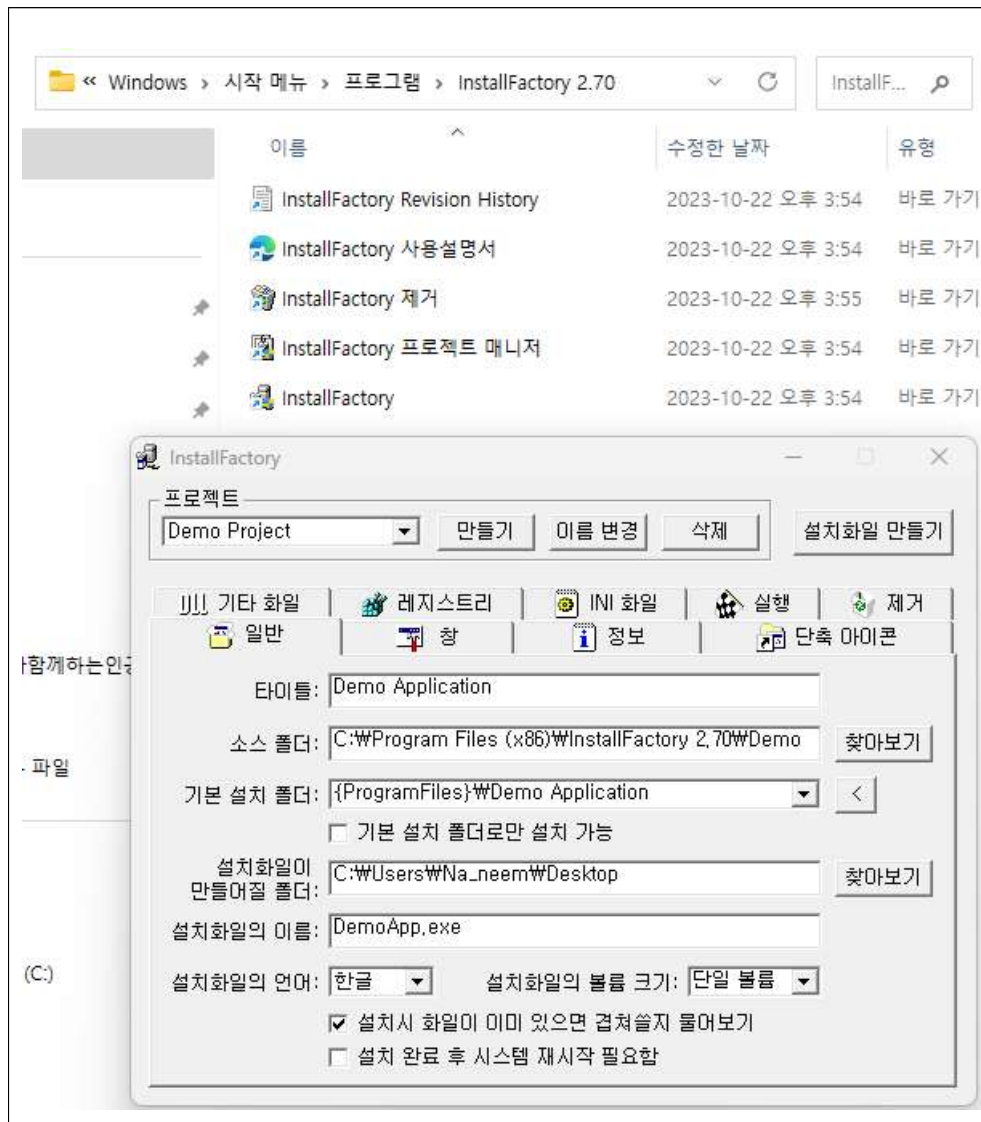
[그림 6-2] InstallFactory 프로그램 설치

다음의 [그림 6-3]은 라이선스가 프리이며 2000년도에 생산된 프로그램인 것을 알 수 있다. 지금은 2023년 23년전에 배포된 프로그램이 이렇게 쓰이고 있는 것에 감사하며 제작자에게 감사의 말씀을 드린다.



[그림 6-3] 프로그램 라이선스와 날자

설치가 완료되고 난 후의 폴더 파일에 사용 설명서가 들어 있으니 함께 활용해도 되겠다. 다음의 [그림 6-4]는 폴더 내용과 실행된 install Factory 프로그램의 초기 화면을 보여 주고 있다.



[그림 6-4] 폴더 내용과 프로그램 실행 화면

6.3 InstallFactory 2.7 사용 방법

앞서 개발한 주소록 프로그램은 pyInstaller를 통해 1개의 파일로 만들어졌다고 가정하고 다음 작업을 진행 하도록 한다.

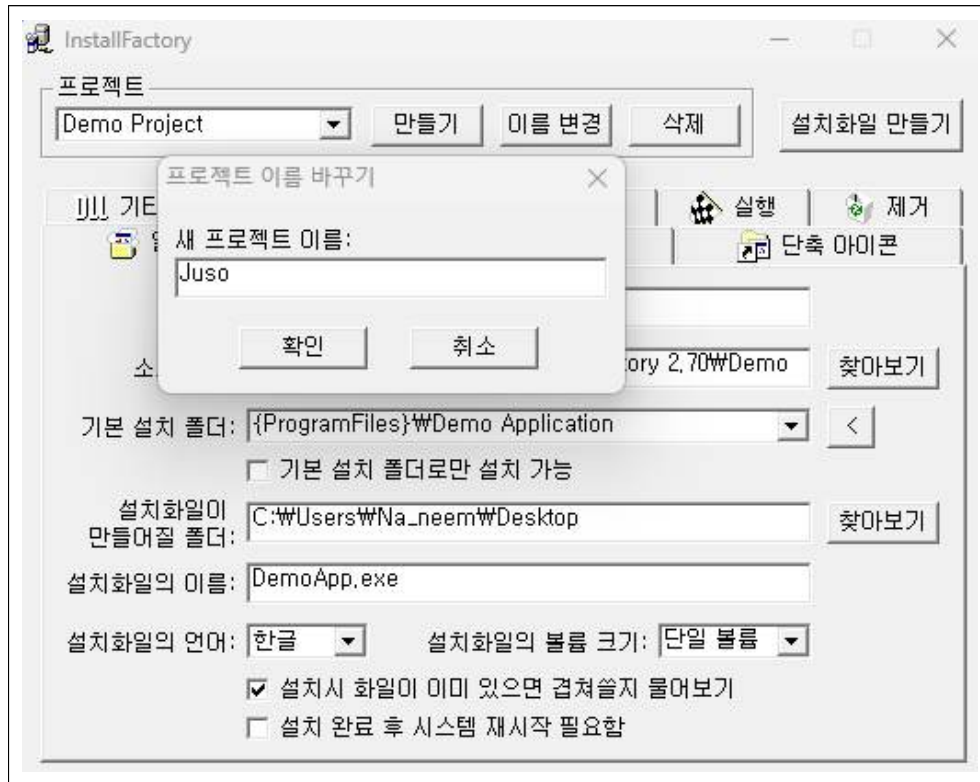


[그림 6-5] 1파일 실행 파일

앞의 [그림 6-5]는 해당 폴더의 dist 폴더 아래에 1파일로 만들어진 exe 파일을 보여 주고 있다.

1. 프로젝트 이름 바꾸기

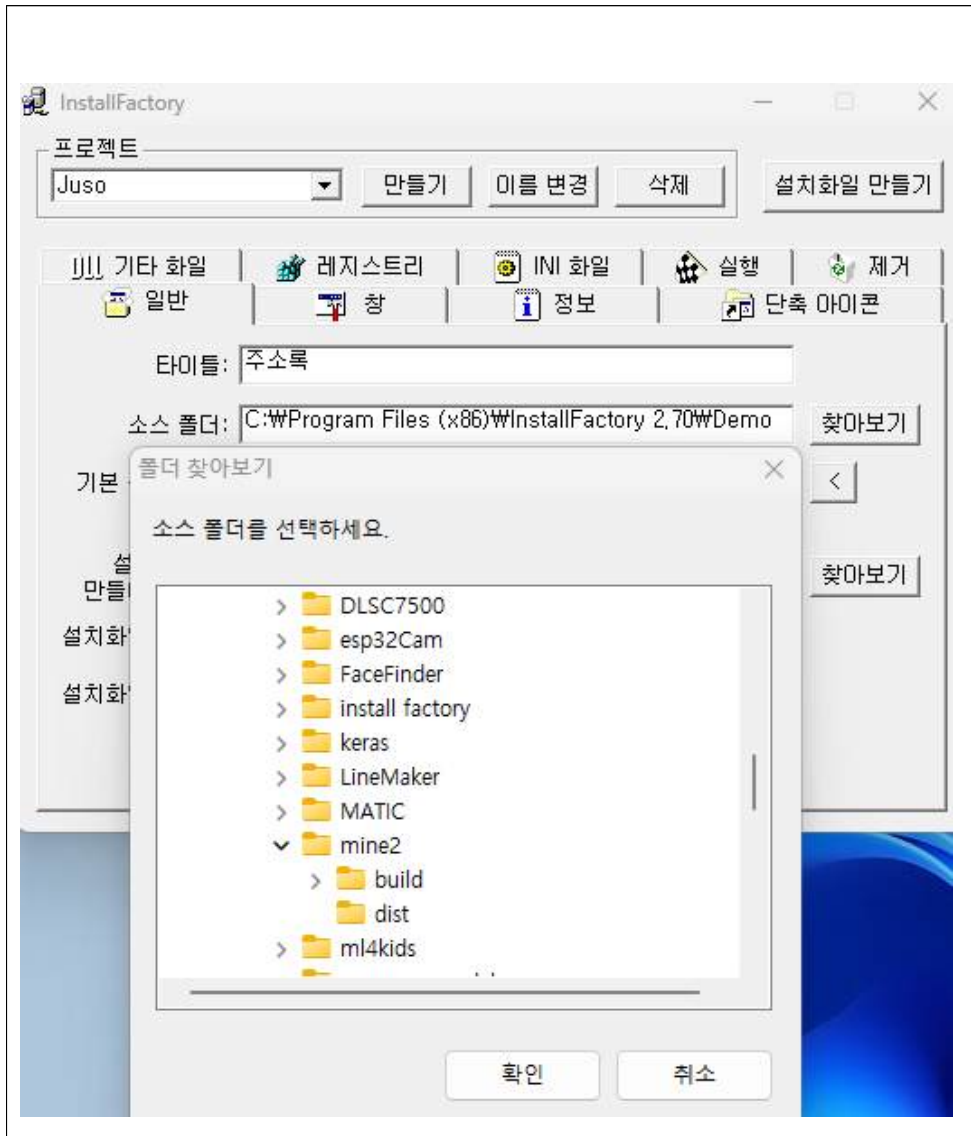
다음의 [그림 6-6]은 프로젝트 이름을 바꾸는 과정을 보여 준다. 먼저 이름변경을 누르고 새 프로젝트 이름을 입력한 후 확인 버튼을 누른다.



[그림 6-6] 프로젝트 이름 바꾸기

다음의 [그림 6-7]은 설치 프로그램의 타이틀과 소스 폴더의 위치를 정하는 과정을 보여 준다.

소스폴더의 찾아보기 버튼을 누른 후 우리가 작업했던 mine2 폴더의 최종 완성된 exe 파일이 있는 위치 dist까지 선택하고 확인을 눌러 준다.



[그림 6-7] 타이틀과 소스 폴더 정하기

다음의 [그림 6-7]은 나머지 옵션 기본 설치 폴더와 설치파일이 만들어진 폴더 설치파일의 이름을 정한다.



[그림 6-8] 나머지 옵션 정리 하기

기본 설치 폴더의 {ProgramFiles}는 컴퓨터 시스템에서 예약된 것으로 윈도우가 설치될 때 정해지도록 되어 있다. 그리고 뒷부분의 \Juso는 ProgramFiles 폴더 아래에 새로 만들어질 우리 프로그램이 위치할 폴더의 이름이다. 이 폴더는 사전에 만들어져 있지 않아도 되고 여기에 기술해줌으로서 설치 과정에서 만들어 지도록 되어 있다.

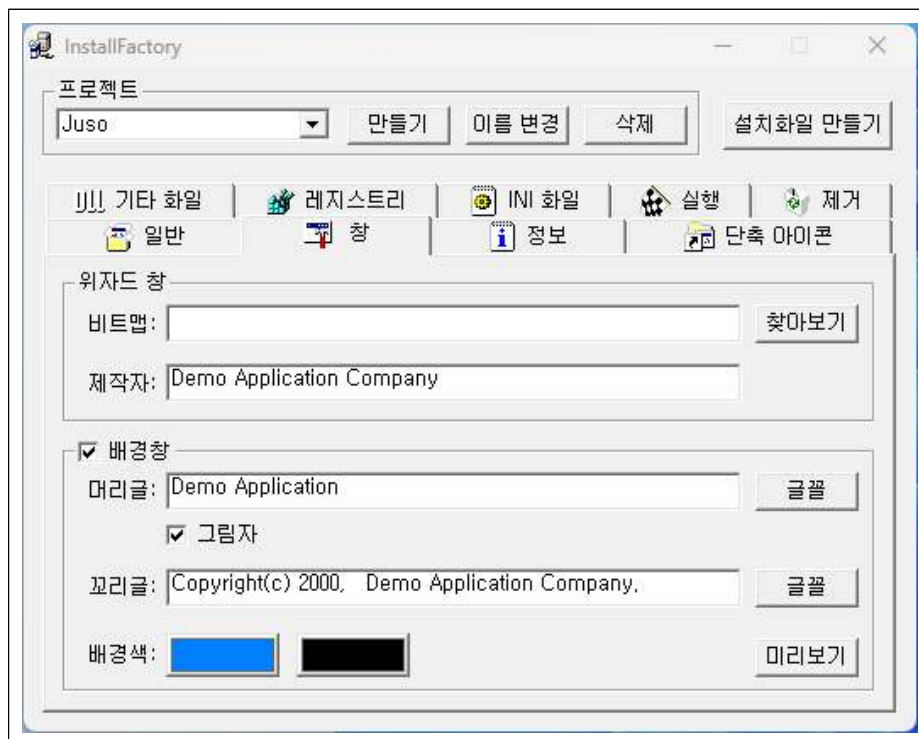
설치파일이 만들어질 폴더는 본 installFactory 프로그램을 실행해서 최종적으로 만들어질 Juso_setup.exe 파일이 어디에 만들어 질 것인가를 나타내고 있다. 앞의 [그림 6-8]의 내용대로 한다면 바탕 화면에 만들어지게 될 것이다.

설치파일의 이름은 Juso_setup.exe라는 이름으로 만들어지도록 하였다.

기본 설치 폴더 예약어	
{WinDrive}	사용자 시스템의 윈도우즈가 설치된 드라이브
{WinDir}	사용자 시스템의 윈도우즈 디렉토리
{WinSysDir}	사용자 시스템의 윈도우즈 시스템 디렉토리
{ProgramFile}	사용자 시스템의 "Program Files" 폴더가 있는 절대경로
{CommonFiles}	사용자 시스템의 "Common Files" 폴더가 있는 절대경로
{Reg}	설치시 사용자 시스템의 레지스트리에서 읽어들이는 문자열이나 확장 문자열(Expand String) 값에 의해 설치 폴더가 결정

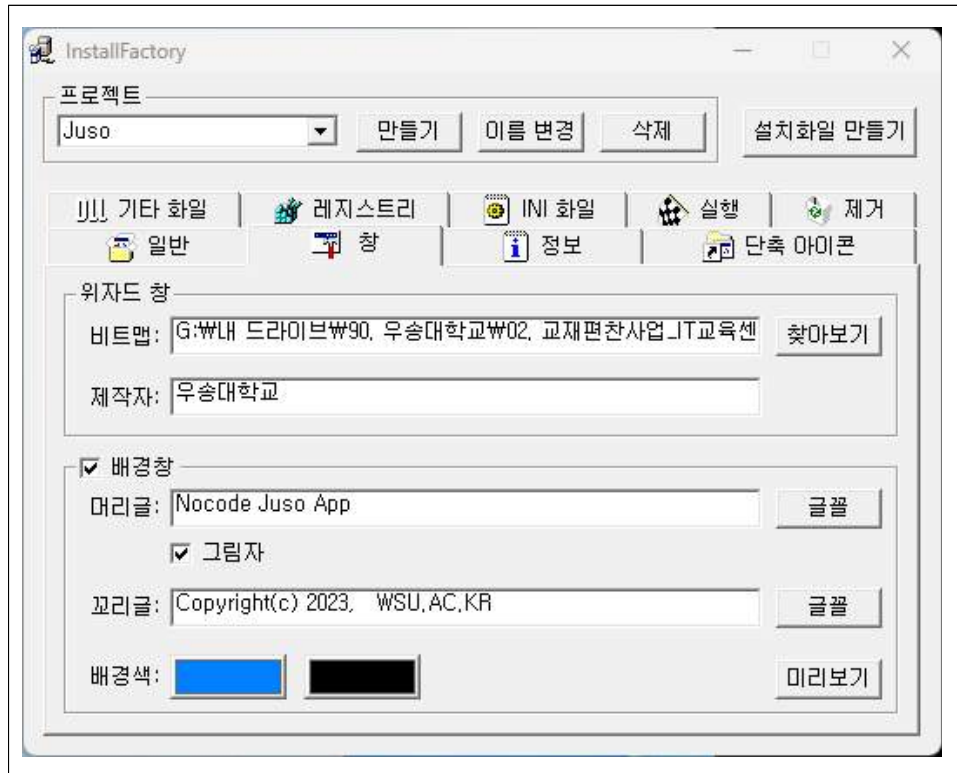
[그림 6-9] 기본 설치 폴더 예약어

이렇게 하게 되면 일반 탭에 대한 셋팅은 끝났다. 다음은 창 탭에 대한 설명을 이어 나가겠다. 이 창은 프로그램의 설치 과정에서 화면에 보여지는 몇 가지 정보를 포함하고 있다.



[그림 6-10] 창 탭

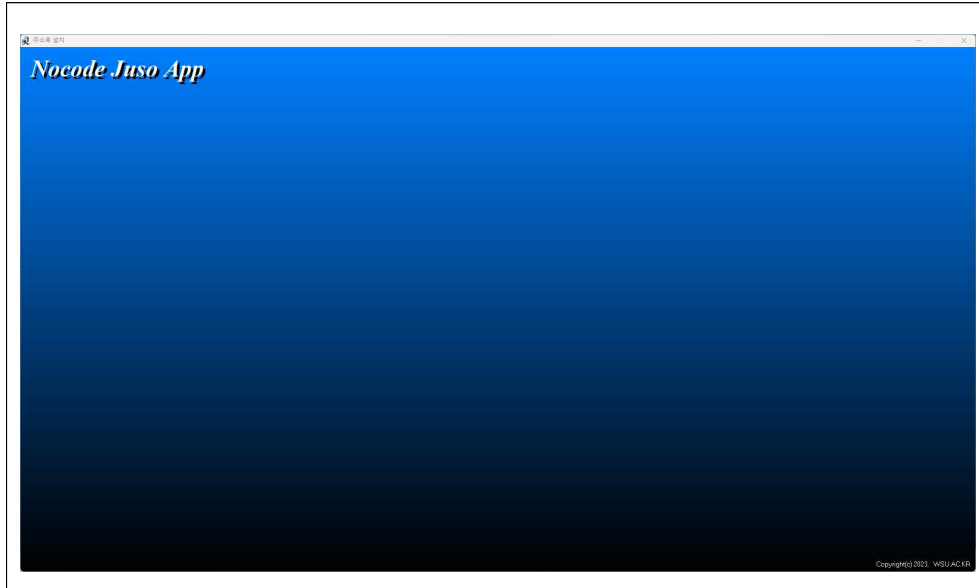
앞의 [그림 6-10]의 창탭에 대한 설정은 중요한 내용은 아니므로 각자의 이름과 머릿글 꼬릿글등을 지정해 보기 바란다. 별도의 설명은 하지 않기로 한다.



[그림 6-11] 창탭을 채우기

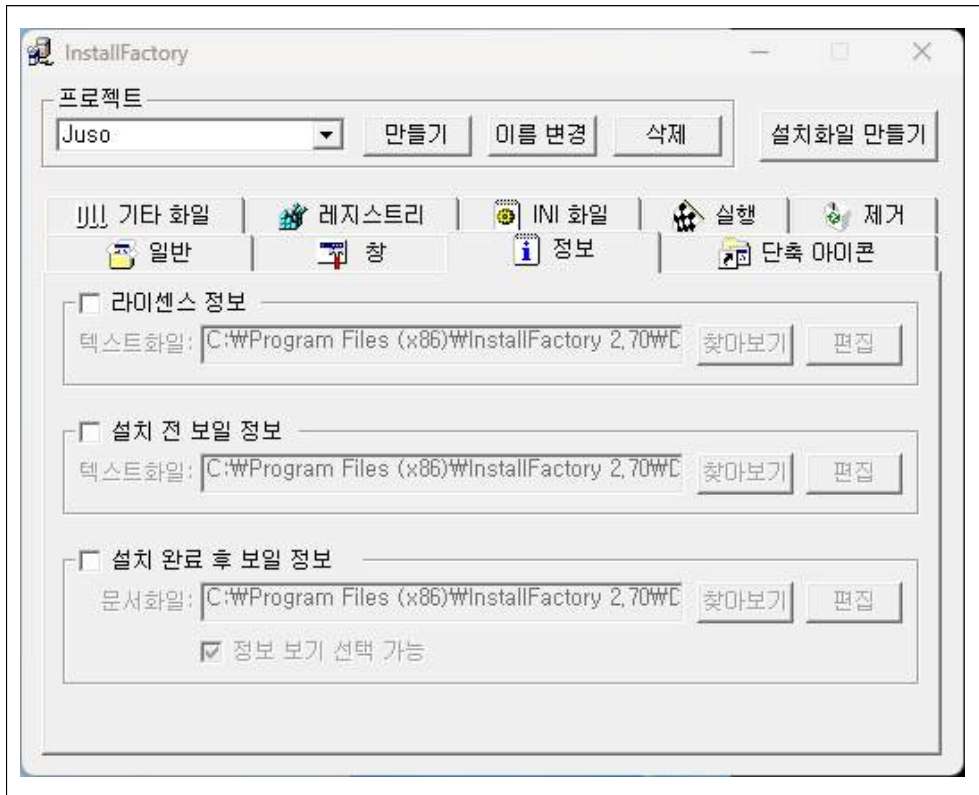
앞의 [그림 6-11]은 앞서 삽화로 제작된 그림을 bmp 파일 포맷으로 바꾸어서 비트맵으로 사용해 보겠다. 그러나 크기의 제한이 있으므로 맞춰줘야 한다.

다음의 [그림 6-12]는 이러한 셋팅을 기반으로 미리보기한 사례이다.



[그림 6-12] 미리보기 결과

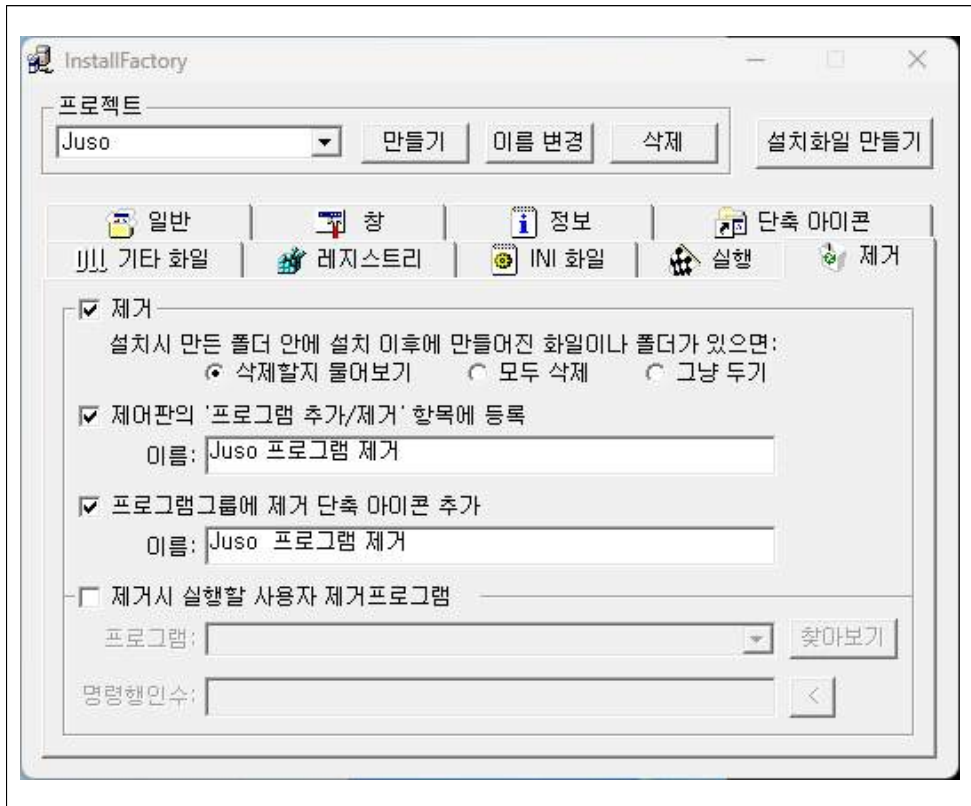
다음의 [그림 6-13]은 정보 탭으로 모든 정보를 체크아웃 함으로서 단순화 하였다. 만약 설치 전 / 후에 주의사항과 설치 전에 라이선스 동의를 받을 수 있는 화면을 텍스트 파일로 보여 주고자 한다면 해당 문서를 지정 하도록 한다.



[그림 6-13] 정보탭

다음은 제거탭에 관한 내용이다. 제거 아이콘 또는 프로그램 추가/제거 항목에 등록 하는 것은 설치 과정에서 만들어진 정보를 바탕으로 만들어 짐으로 제거 이름만 설정하면 된다.

다음의 [그림 6-14]는 제거탭의 제거 항목, 또는 단축 아이콘에 대한 내용을 보여 준다. 제거 단축 아이콘은 선택적으로 사용할 수 있다. 즉 제어판의 프로그램 추가/제거 만을 이용해서 제거가 가능하다는 뜻이다.



[그림 6-14] 제거탭

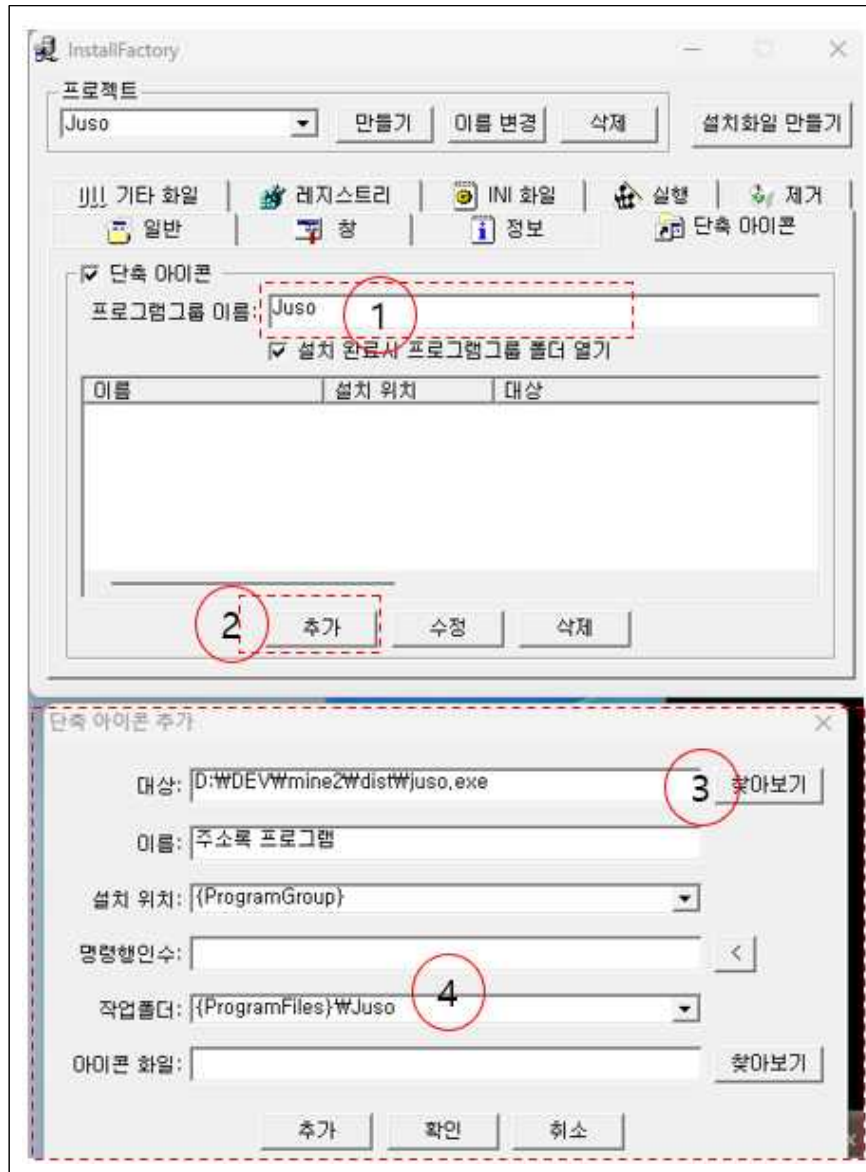
단축 아이콘탭에 관한 설명을 하기에 앞서 단축 아이콘이 어디에 만들어 질지를 정하는데는 예약어가 중요하다.

다음의 [그림 6-15]는 생성 위치를 나타내는 예약어를 보여 준다.

{ProgramGroup}	프로그램그룹에 단축 아이콘을 설치합니다.
{DeskTop}	"바탕 화면"에 단축 아이콘을 설치합니다.
{StartMenu}	"시작 메뉴"에 단축 아이콘을 설치합니다.
{ProgramMenu}	"시작 메뉴 -> 프로그램"메뉴에 단축 아이콘을 설치합니다.
{StartupMenu}	"시작 메뉴 -> 프로그램 -> 시작프로그램"메뉴에 단축 아이콘을 설치합니다. 단축아이콘의 대상프로그램은 윈도우 시작시 실행됩니다.
{QuickLaunch}	시스템 작업 표시줄의 "빠른 실행"에 추가합니다.
{SendTo}	SendTo 폴더에 단축아이콘을 설치하며, 단축 아이콘의 대상 프로그램은 콘텍스트 메뉴의 "보내기"메뉴에 등록합니다.

[그림 6-15] 예약어 목록

이제 단축 아이콘을 만들어 보겠다. 설명이 길어질수 있으니 프로그램 그룹과 바탕화면에 각각 1개의 단축 아이콘을 만드는 것으로 하겠다. 다음의 [그림 6-16]은 단축 아이콘탭의 셋팅 과정을 보여 준다.



[그림 6-16] 단축아이콘 만들기

다음에 그림의 과정을 상세히 설명한다.

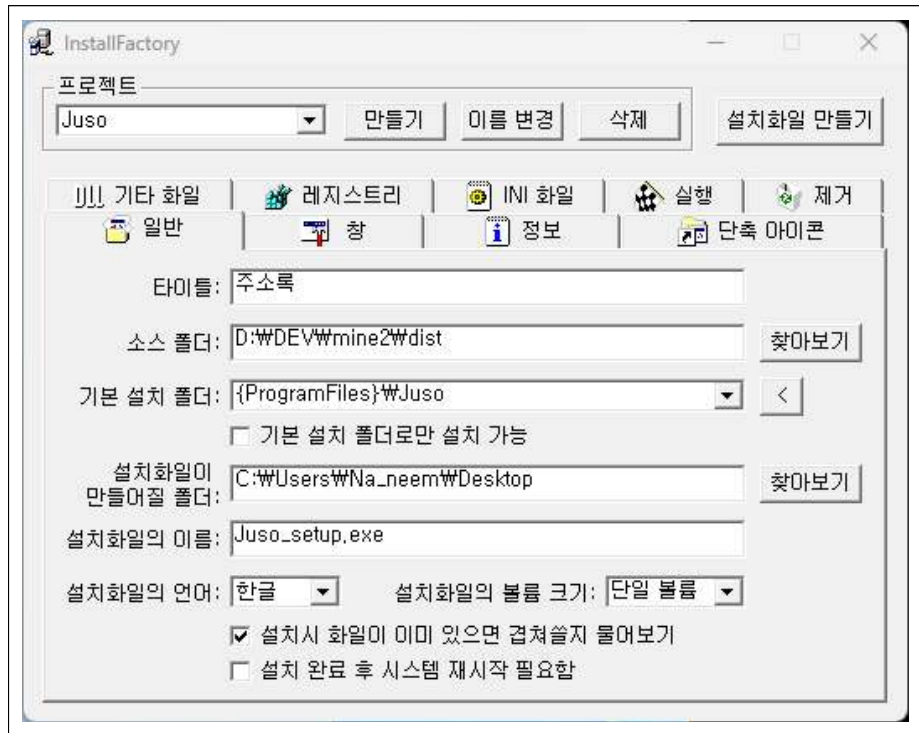
1. 프로그램 그룹 이름을 Juso라고 지정하였다. 이렇게 지정하면 데스크탑에서 모든앱-Juso 라는 프로그램 그룹이 생성된다. 이것은 물리적으로 ProgramFile아래의 juso라는 폴더에 프로그램을 설치하기로 했던 것과는 다른 것이다.

2. 추가 버튼

3. 찾아보기는 실제 우리가 만들려고 하는 프로그램의 본체를 지정하는 것이다. 이전 과정에서 소스폴더를 정했던 부분에서 dist 폴더에 있는 최종 exe 파일을 선택하여야 한다.
4. 작업 폴더는 사실상 우리와 같은 딱 1개의 프로그램을 사용하는 경우에는 중요하지 않다. 어느 곳에서나 실행하더라도 부속 파일들이 필요하지 않기 때문이다. 그러나 만약 부속 파일들이 있는 경우는 반드시 해당 폴더를 지정 해주어야 한다. 어찌되었건 만약의 경우를 대비해서 가장 먼저 했던 일반 탭에 대한 셋팅에서 기본설치 폴더 라는 곳을 지정해 준 것이다.

이렇게 해서 1개의 단축 아이콘에 대한 설정을 모두 마쳤다. 나머지 1개의 아이콘은 Desktop 즉 바탕 화면에 만드는 것을 스스로 해보기 바란다.

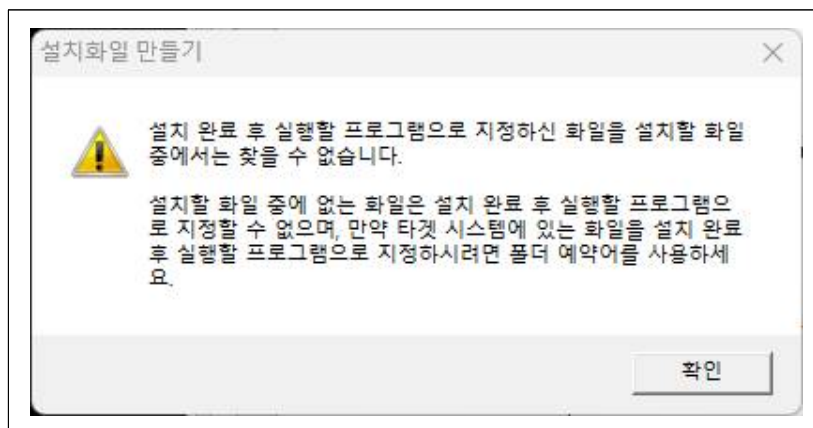
여기까지 설정했으면 대부분의 경우 끝난다. 다음 그림 [6-17]의 ‘설치 파일 만들기’를 눌러 마무리 한다. 개발하는 프로그램의 특성에 따라 레지스트리 편집, INI 파일 설정 등이 필요한 경우 InstallFactory 매뉴얼을 참고한다.



[그림 6-17] 설치파일 만들기

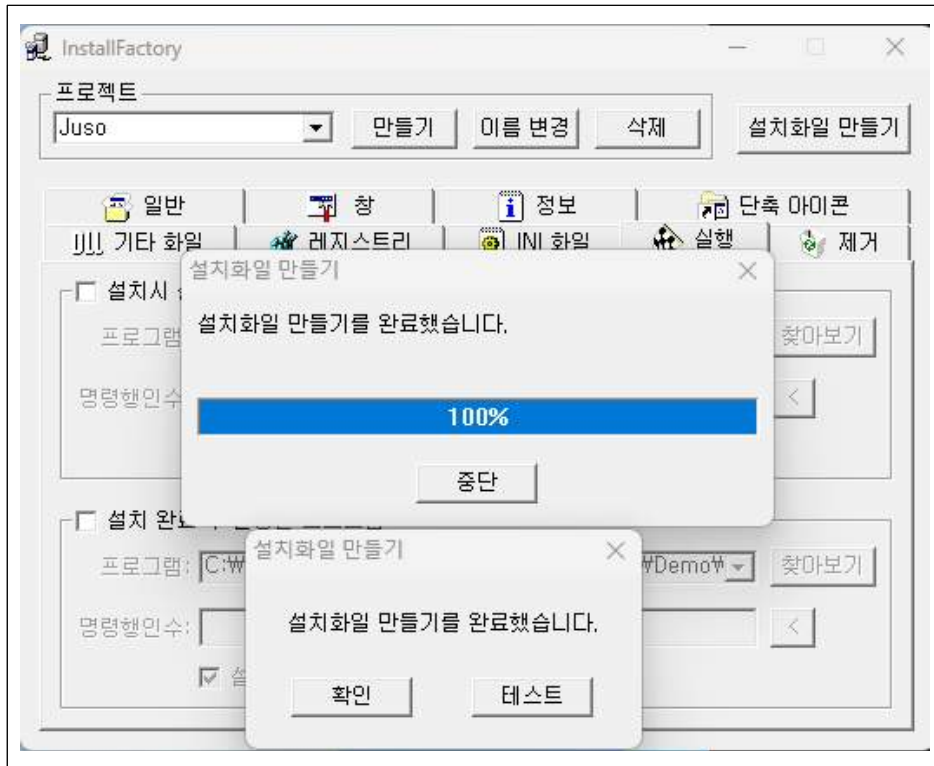
6.4 InstallFactory 2.7 테스트

다음 [그림 6-18]은 설치파일 만들기를 눌렀을 때 오류가 발생한 것이다.



[그림 6-18] 설치파일 만들기 오류

확인 버튼을 누르면 화면이 실행 탭이 활성화되어 나타난다. 프로그램 설치 후에 실행할 파일명을 제대로 지정해주지 않았기 때문이다. 체크 아웃으로 사용하지 않음을 알려주자.



[그림 6-19] 최종 설치 파일 만들기 결과

앞의 [그림 6-19]는 설치 완료후 실행할 파일을 체크 아웃 하고 설치 파일 만들기를 실행 했을 때의 화면이다.

다음의 [그림 6-20]은 테스트 버튼을 눌렀을 때 실행 화면이다.

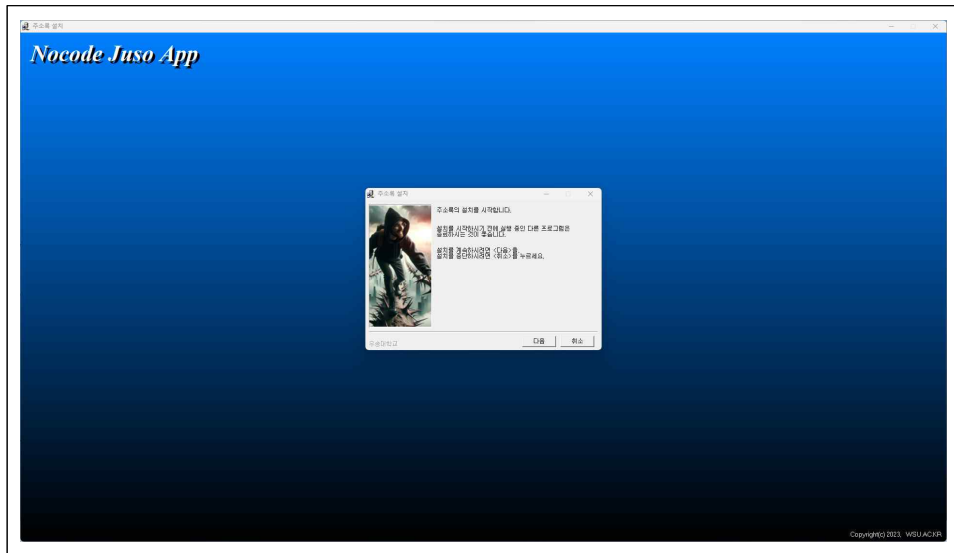
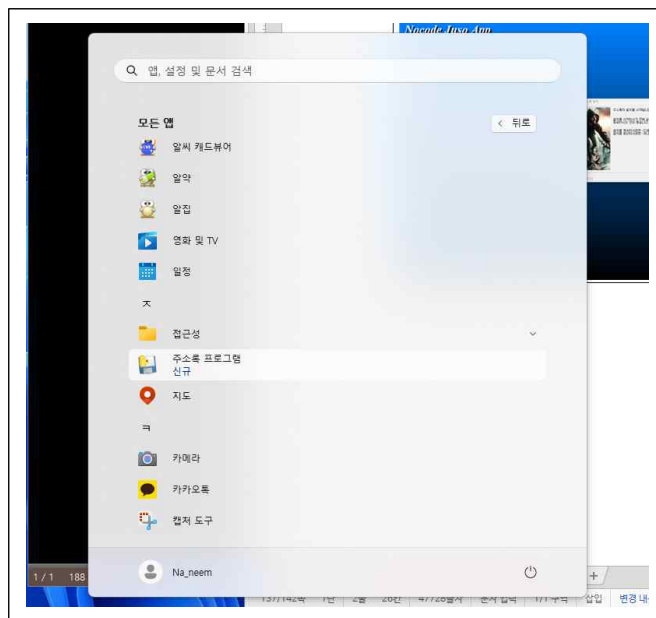


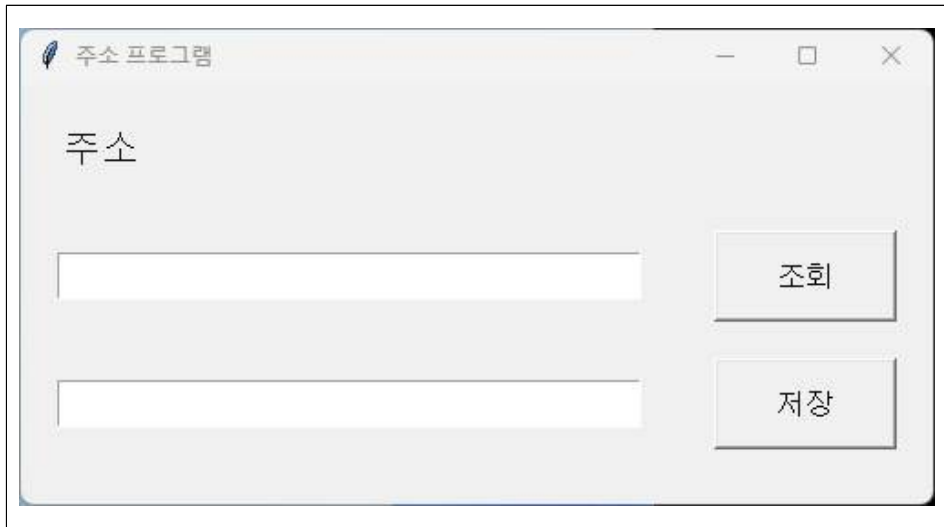
그림 6-20] 테스트 화면

다음 [그림 6-21]은 윈도우버튼과 모든앱에서 새로 생성된 프로그램그룹과 단축아이콘을 볼 수 있다.



[그림 6-21] 생성된 단축아이콘

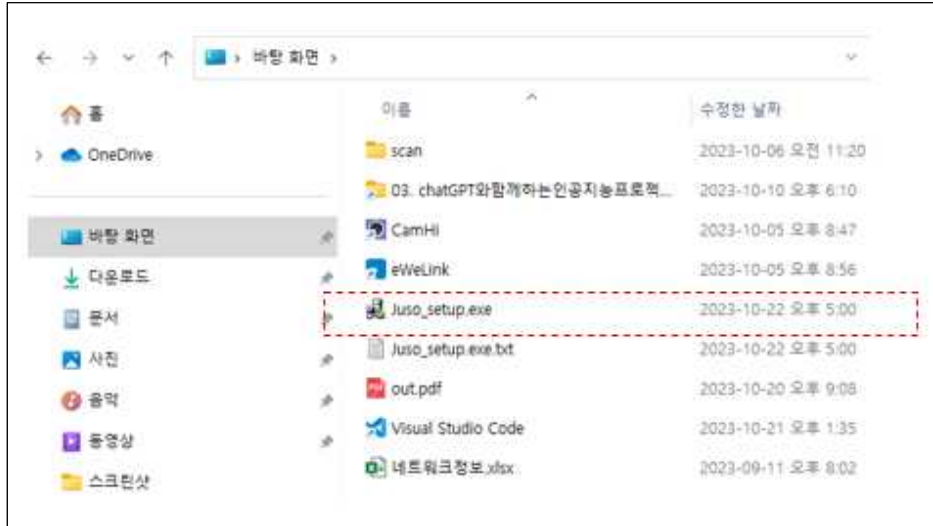
마지막으로 [그림 6-22]에서는 단축 아이콘을 클릭해서 프로그램이 실행된 모습을 볼 수 있다.



[그림 6-22] 단축아이콘으로 실행된 프로그램

여기까지 우리는 chatGPT와 함께 프로그램을 만들고 컴파일(1개의 exe 파일로 만드는 것)하고 배포 가능한 설치 프로그램 juso-setup.exe 파일을 만드는 과정까지 해보았다.

다음의 [그림 6-23]은 바탕화면에 만들어진 배포 파일 juso_setup.exe 파일을 보여 주고 있다.



[그림 6-23] juso_setup.exe 파일

제 4편 실전 : 고급 UI