

<https://github.com/ARPA-GO/shipcurve>

| Python 을 기반으로 한 선박계산 오픈소스 라이브러리



ARPA-GO

Always Ready for Program Assistance - GO

저작권

Copyright © 2019 ARPA-GO Corp. All Rights Reserved.

라이선스 관련 고지

이 제품은 필요에 따라 특정 소프트웨어를 통합하거나 참조합니다. 라이선스를 호스팅하는 웹 사이트에 대한 링크를 아래에 제공하였습니다. ARPA-GO는 이러한 사이트의 접근성과 내용을 보증하지 않습니다.

PYTHON

버전 : Python 3.7.3

다운로드 : <https://docs.python.org/3/download.html>

Copyright © 2001-2019 Python Software Foundation. All rights reserved.

Copyright © 2000 BeOpen.com. All rights reserved.

Copyright © 1995-2000 Corporation for National Research Initiatives. All rights reserved.

Copyright © 1991-1995 Stichting Mathematisch Centrum. All rights reserved.

문서 정보

문서개요

이 문서는 선박계산을 위한 라이브러리인 ARPA-GO를 설치하고 사용하는 방법을 설명합니다.

독자

이 문서의 독자는 ARPA-GO를 사용해서 별도의 프로그램 없이 선박계산을 하고자 하는 사용자입니다.

문의처

이 문서의 내용에 오류가 있거나 내용과 관련한 의문 사항이 있으면 아래의 연락처로 문의하십시오.

E-mail: ARPAGO.kmou@gmail.com

문서 버전 및 이력

| 버전 | 일자 | 이력 사항 |
|-----|------------|--------|
| 1.0 | 2019.05.14 | 1.0 배포 |

INDEX

| | |
|--------------------------|------------------------|
| 1. ARPA-GO 소개 | |
| 1.1 ARPA-GO 란? | 5 |
| 1.2 기능과 특성 | 5 |
| 2. 실행 및 설치 | |
| 2.1 Pycharm | 7 |
| 2.1.1 Python 설치 | 7 |
| 2.1.2 Python 실행 | 9 |
| 2.1.2 Pycharm 설치 | 10 |
| 2.1.2 Pycharm 실행 | 15 |
| 2.2 exe File | 18 |
| 2.2.2 설치 | 18 |
| 2.2.3 실행 | 오류! 책갈피가 정의되어 있지 않습니다. |
| 3. CODE | |
| 3.1 Codes | 24 |
| 3.1.1 Python 표준 라이브러리 | 24 |
| 3.1.2 함수설명 | 25 |
| 3.2 GUI 형식 | 41 |
| 3.2.1 메인 윈도우 | 41 |
| 3.2.2 Excel 불러오기 | 43 |
| 3.2.3 선박계산표 | 45 |
| 3.2.4 Hydrostatic Curves | 48 |
| 3.2.5 결과 비교하기 | 50 |
| 4. 부록 | |
| 4.1 추가 기능 | 53 |
| 4.1.1 Lisp | 53 |
| 4.1.2 Class | 76 |
| 4.2 용어 | 77 |

1. ARPA-GO 소개

이 장에서는 ARPA-GO에 대한 정의와 기능, 특징을 설명합니다.

1.1 ARPA-GO 란?

[Always Ready for Program Assistance-GO](#)

‘프로그램을 도와주기 위해 항상 준비되어있으니 함께 나아가자’는 저희 프로젝트 의미입니다. Foran, Aveva Marine 과 같이 이미 상용화된 종합 선박 프로그램이 존재합니다. 그러나 학생과 같은 개인이나 start-up 회사는 고가비용인 이 프로그램들을 사용하기가 어렵습니다. 또한, 충분한 숙련도가 갖추어지지 않으면 프로그램을 사용하기 힘듭니다.

이러한 경우들이 존재하기 때문에, 라이브러리를 개발했습니다. ARPA-GO 는 Python 기반의 선박 계산 오픈 소스 라이브러리입니다. 선박 계산은 다양한 분야로 나뉩니다. 그 중에서도 모든 선박 계산에 기초가 되는 hydrostatic curve 를 구하는 것을 기반으로 하였으며, 추가 계산을 위해 종합 선박 계산표가 제공됩니다.

1.2 기능과 특성

ARPA-GO의 기능 및 특성은 다음과 같습니다.

Open source

앞서 언급했듯이 ARPA-GO는 Python 기반의 선박 계산 오픈 소스 입니다. 이는 사용하고자 하는 모든 사람에게 제약 없이 누구나 접근할 수 있습니다. 이 프로젝트는 플랫폼 'github'에 게시되어 사용할 수 있습니다. 사용자는 부담 없이 원하는 정보를 찾을 수 있으며, 다른 사용자와 의견을 공유할 수 있습니다.

GitHub는 프로그램의 소스 코드 관리를 위한 분산 버전 관리 시스템입니다. Git의 작업 폴더는 모두 기록하고 있어서 추적이 가능하고, 다양한 사람들과 협업할 수 있는 플랫폼의 형태입니다. 이 플랫폼에서는 다양한 분야에서의 오픈 소스 개발이 진행 중이지만, 조선분야에서의 활동은 미미합니다. 이 프로젝트를 통해 선박 관련 오픈 소스 개발이 활동적으로 이루어지길 기대합니다.

One click touch

프로젝트 ARPA-GO는 선박의 옵션을 통해 선박 계산을 진행하고, 결과값으로 선박 종합계산표와 Hydrostatic curve를 도출합니다. 저희의 프로젝트를 사용하지 않고 선박 계산 수행 시 폭, 스테이션 간의 간격, 수선간 길이 그리고 워터라인 개수 등 모든 변수들을 offset표에서 찾아야 합니다. 하지만, 저희의 프로젝트는 별도의 변수 입력 없이 복잡한 계산을 가능하게 합니다. 저희 프로젝트 ARPA-GO에서는 선박계산을 위해 선박의 offset 파일만 요구됩니다.

2개의 형식

ARPA-GO는 2가지 형태로 이루어집니다. 하나는 필요에 따라 수정 및 추가 작업이 용이한 코드들과 다른 하나는 실행이 간편한 GUI 형태입니다. 사용자의 선택에 따라 모든 형태의 ARPA-GO를 사용할 수 있습니다. 3장에서는 두 가지 형태에 대해 더 자세히 다루고자 합니다.

CAD: 좌표추출

선박계산을 위해서는 워터라인과 스테이션 별로 나누어진 offset 표가 필요합니다. 이 offset 표는 보통 선박의 정면도에서 수작업을 통해 추출해옵니다. 반면, ARPA-GO에서는 lisp를 통해 자동으로 offset을 엑셀파일로 받아오도록 했습니다. Python에서 명령어를 통해 완성된 offset을 불러온 후, 선박계산을 시작합니다. lisp와 관련하여 부록에서 더 자세히 다룹니다.

비교 값

설계에서는 비슷한 실적선을 찾아 선박계산을 진행한 후, 값을 변경하여 기존의 선박과 비교를 거쳐 최적의 선형을 찾습니다. 이를 위해 기존의 선박계산표와 수정된 선박계산표의 값들을 비교해주는 기능을 추가했습니다. 이 기능은 결과값들을 효율적으로 비교할 수 있도록 합니다. 이 기능에 대해서는 뒤에서 더 자세히 다루고자 합니다.

2. 설치 및 실행

이 장에서는 ARPA-GO를 이용할 수 있는 두 가지 경로에 대해 설명합니다.

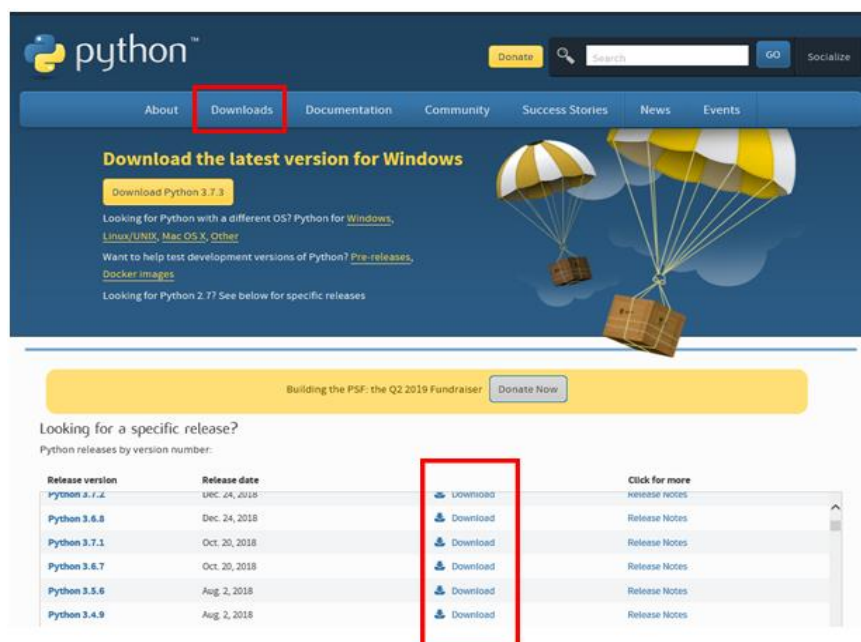
2.1 Pycharm 을 통해 사용

ARPA-GO는 Python을 기반으로 하는 프로그램이기 때문에 (ARPA-GO의) exe 파일 설치 외의 경로로 이용하고자 한다면, 사용자의 컴퓨터에 Python이 준비되어 있어야 합니다. Python은 고급 프로그래밍 언어로, 플랫폼 독립적이며 객체지향적, 동적 타이핑 대화형 언어입니다. 따라서, 저희 라이브러리 개발 도구로 적합하다고 생각하여 이 언어를 사용했습니다.

아래는 Python 설치 과정과 실행여부 확인 과정을 사진과 함께 실어놓았습니다.

2.1.1 Python 설치

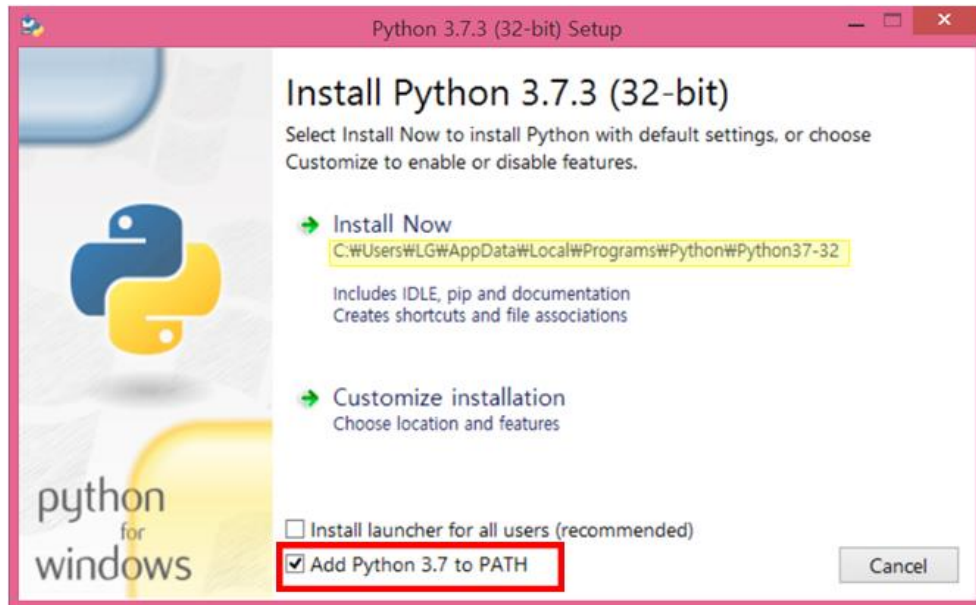
1. Python 공식 홈페이지에 접속합니다. 공식 홈페이지 URL은 (<https://www.python.org/>) 입니다.
홈페이지에서 Python에 대해 더 많은 정보를 얻을 수 있습니다.
2. 상단의 메뉴 중 Download를 클릭하면 다양한 버전의 Python 파일들이 화면에 제시됩니다. 이 중 원하는 버전의 Download를 눌러 파일을 받습니다. ARPA-GO는 Python 3.7.3을 사용하고 있습니다.



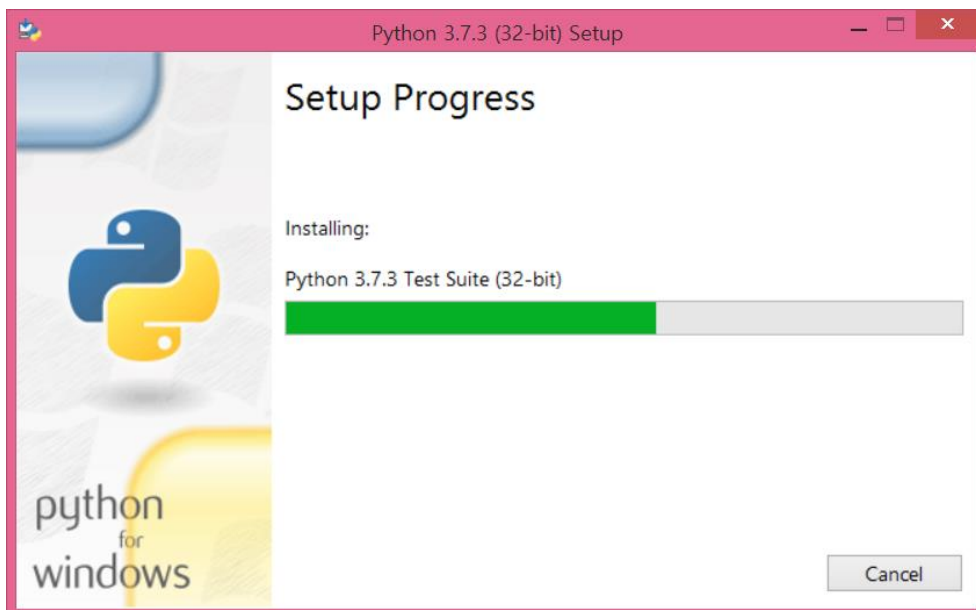
참고

Python2.x 는 2020년까지 지원 예정이므로 python3.x 버전을 추천합니다.

3. 다운받은 파일을 실행하면 아래와 같은 화면이 나옵니다. **Add Python 3.7 to PATH** 를 체크한 뒤 **Install Now** 를 클릭해 설치를 진행합니다. 만약, 설치장소를 바꾸고 싶다면 **Customize Installation**을 선택합니다.



4. 설치가 진행중인 화면입니다.

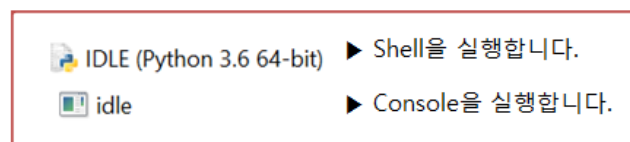


5. 다음과 같은 화면이 나오면 **Close**를 눌러 설치를 완료합니다.

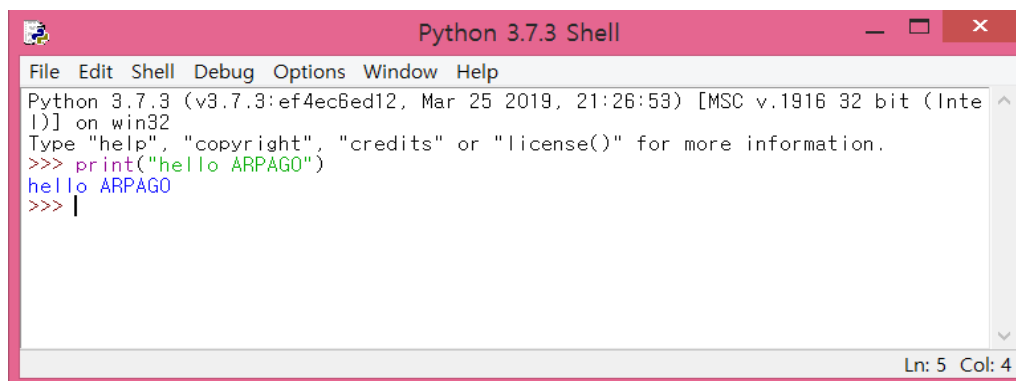


2.1.2 Python 실행

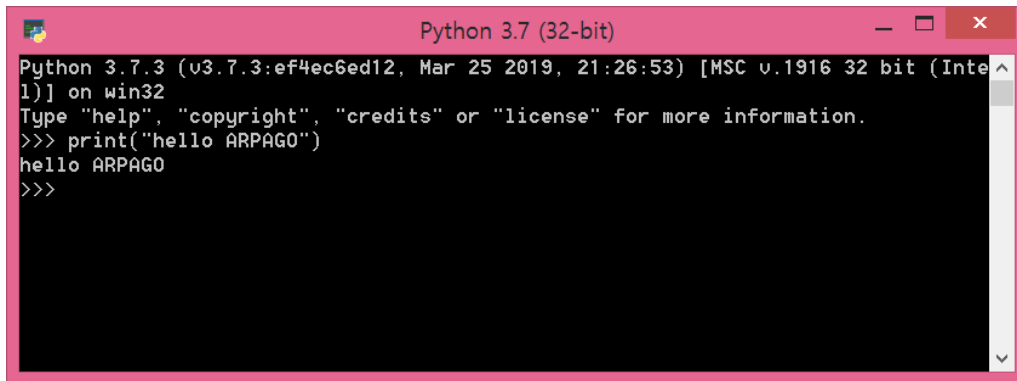
시작메뉴 → 최근 추가한 앱 → IDLE (Python 3.7) 을 클릭하여 Python을 실행합니다. 만약, '최근 추가한 앱' 항목이 없을 경우 '프로그램 및 파일 검색'에 IDLE을 검색하여 실행합니다.



실행여부 확인을 위해 Shell에 'hello ARPAGO' 를 출력한 모습입니다.



실행여부 확인을 위해 Console에 'hello ARPAGO' 를 출력한 모습입니다.



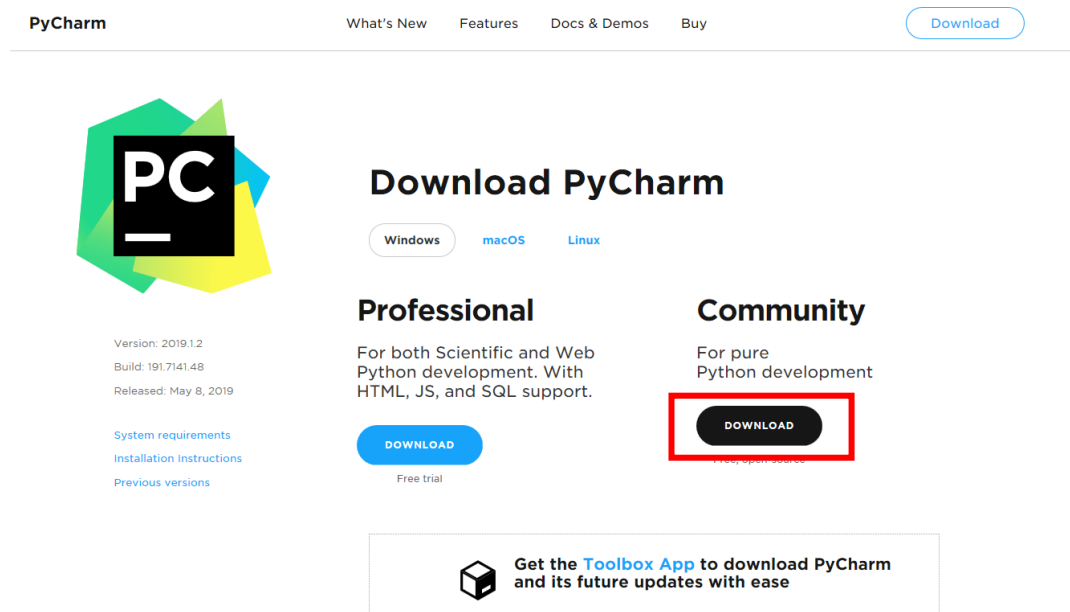
```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("hello ARPAGO")
hello ARPAGO
>>>
```

2.1.3 pycharm 설치

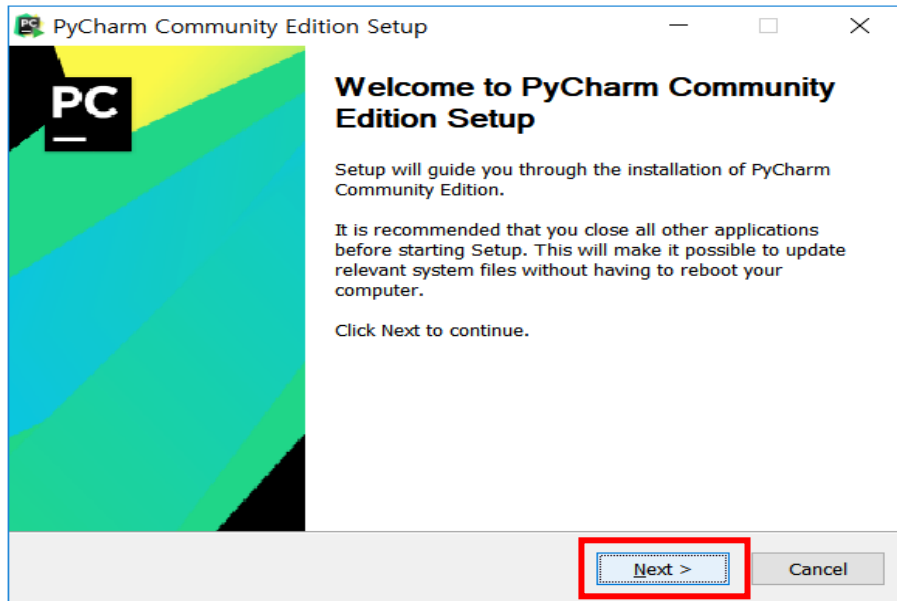
Python을 개발할 수 있는 통합 환경에는 여러 가지가 존재합니다. 그 중에서도 pycharm에 대한 설치방법과 실행방법을 설명합니다. Pycharm은 python의 IDLE창과 달리 코드 별 python 버전 선택을 제공하고 패키지 설치도 쉬울 뿐만 아니라 python 코딩 시에도 파일과 함수를 확인하기 쉽게 보여주기 때문에 관리가 용이하다는 장점이 있습니다.

1. Pycharm 공식 홈페이지에 접속합니다. 공식홈페이지 URL은 (<https://www.jetbrains.com/pycharm/download/#section=windows>) 입니다.

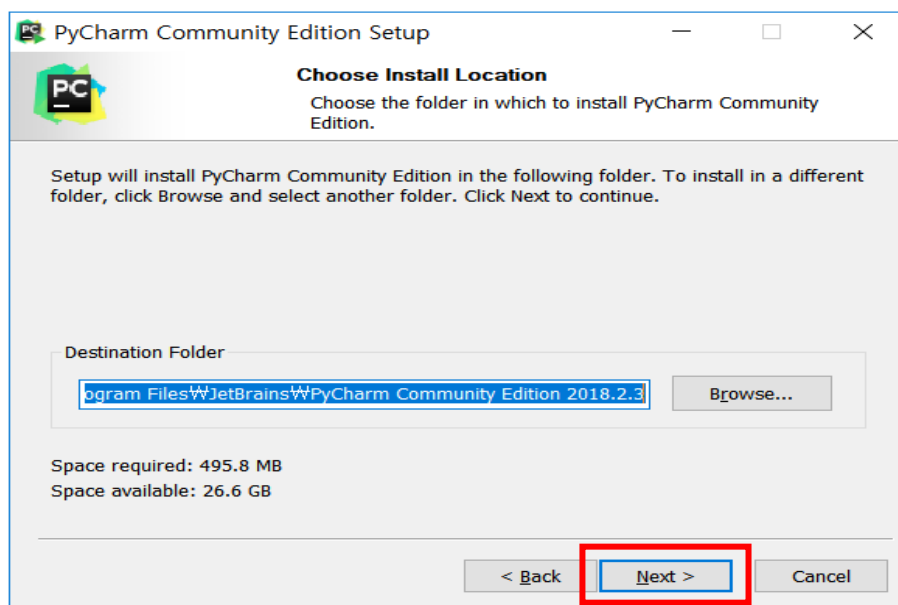
Pycharm은 프로페셔널 버전과 커뮤니티 버전이 존재합니다. 커뮤니티 버전이 무료이므로 커뮤니티 버전을 다운받습니다.



2. Pycharm 설치 파일을 실행한 뒤 [Next] 버튼을 누릅니다.

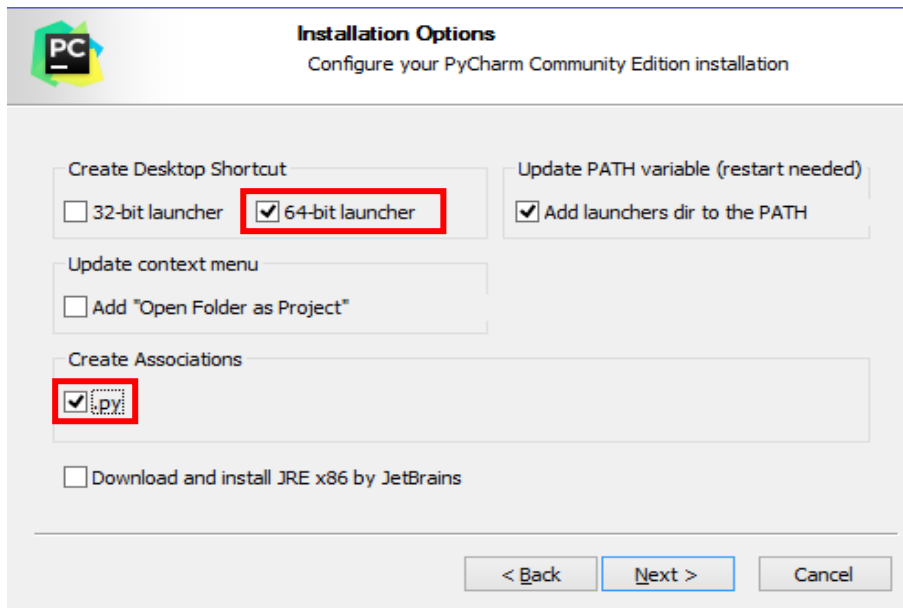


3. Pycharm 설치 경로를 설정합니다. 설치 경로를 바꾸고 싶다면 원하는 경로로 바꿀 수 있습니다. 바꾸고 싶지 않다면 [Next] 버튼을 누릅니다.

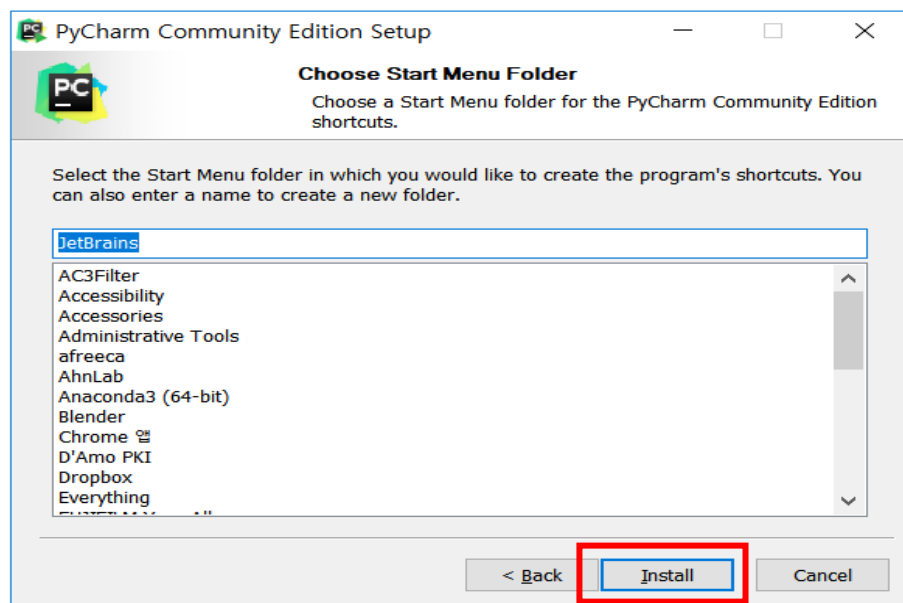


4. 설치 옵션을 선택합니다. 바탕화면 아이콘 생성여부, 환경 변수 업데이트, 프로젝트 열기메뉴, .py파일 생성 등을 선택할 수 있습니다.

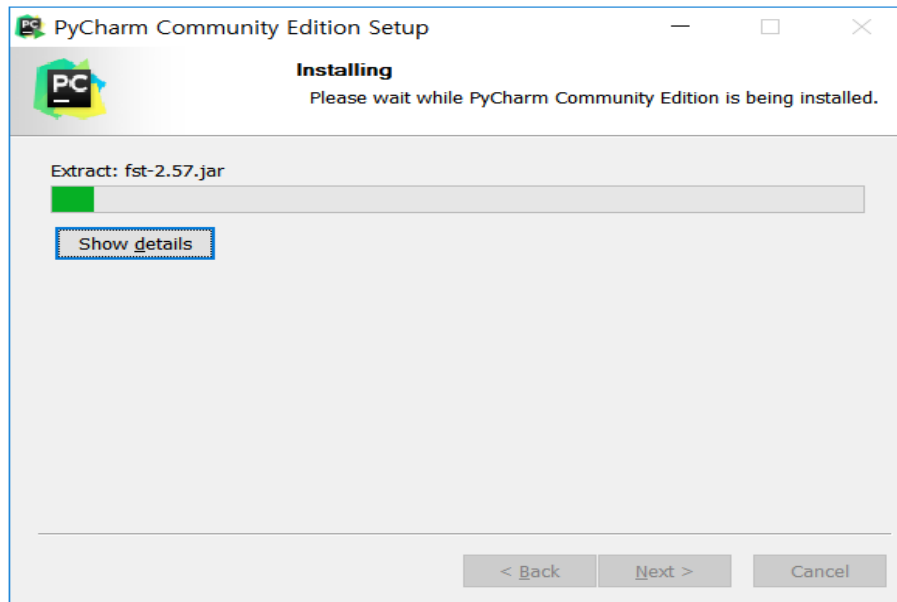
Create Desktop Shortcut은 현재 사용중인 윈도우가 32bit체제인지 64bit체제인지 확인 후 선택하는 것을 추천합니다.



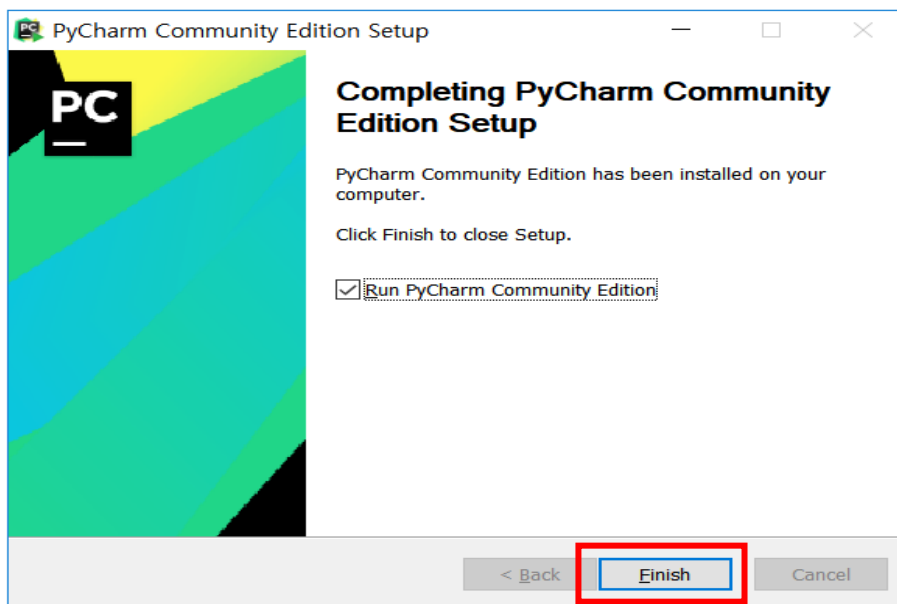
5. 시작 메뉴 폴더를 선택하는 화면입니다. 별다른 설정 없이 [Install] 버튼을 눌러줍니다.



6. 설치가 진행중인 화면입니다. 그렇게 오랜 시간이 걸리지 않습니다.



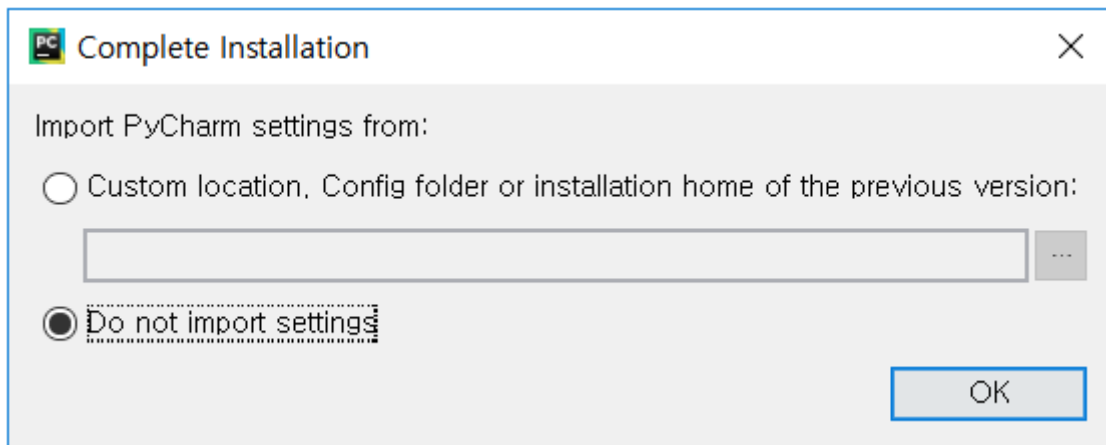
7. Run Pycharm Community Edition 을 체크한 후 설치를 완료합니다.



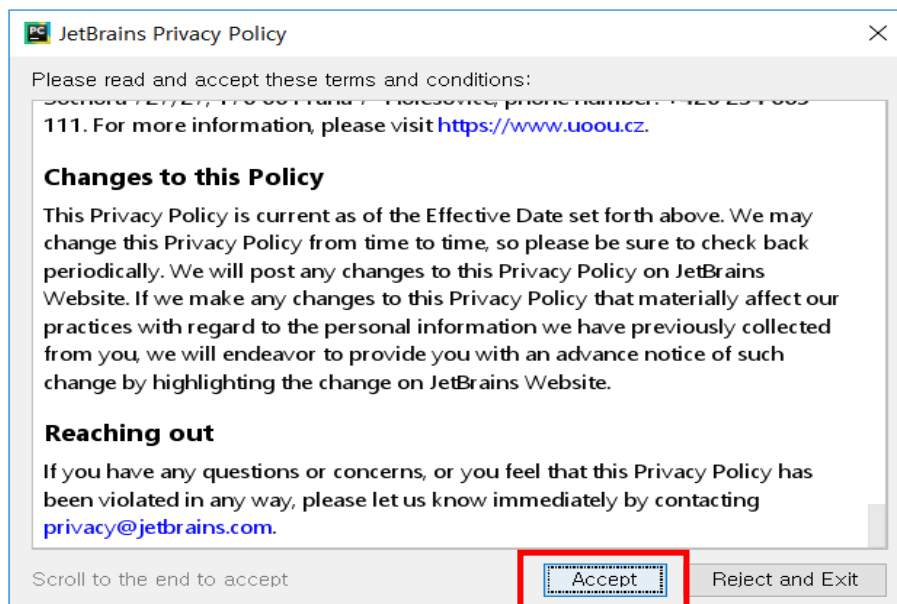
이로써 pycharm의 설치는 완료하였습니다. Pycharm을 실행시켜 환경을 설정하여 줍니다.

8. Pycharm을 실행하면 아래와 같은 화면이 나옵니다.

Do not import settings를 선택한 뒤 [OK]버튼을 누릅니다.



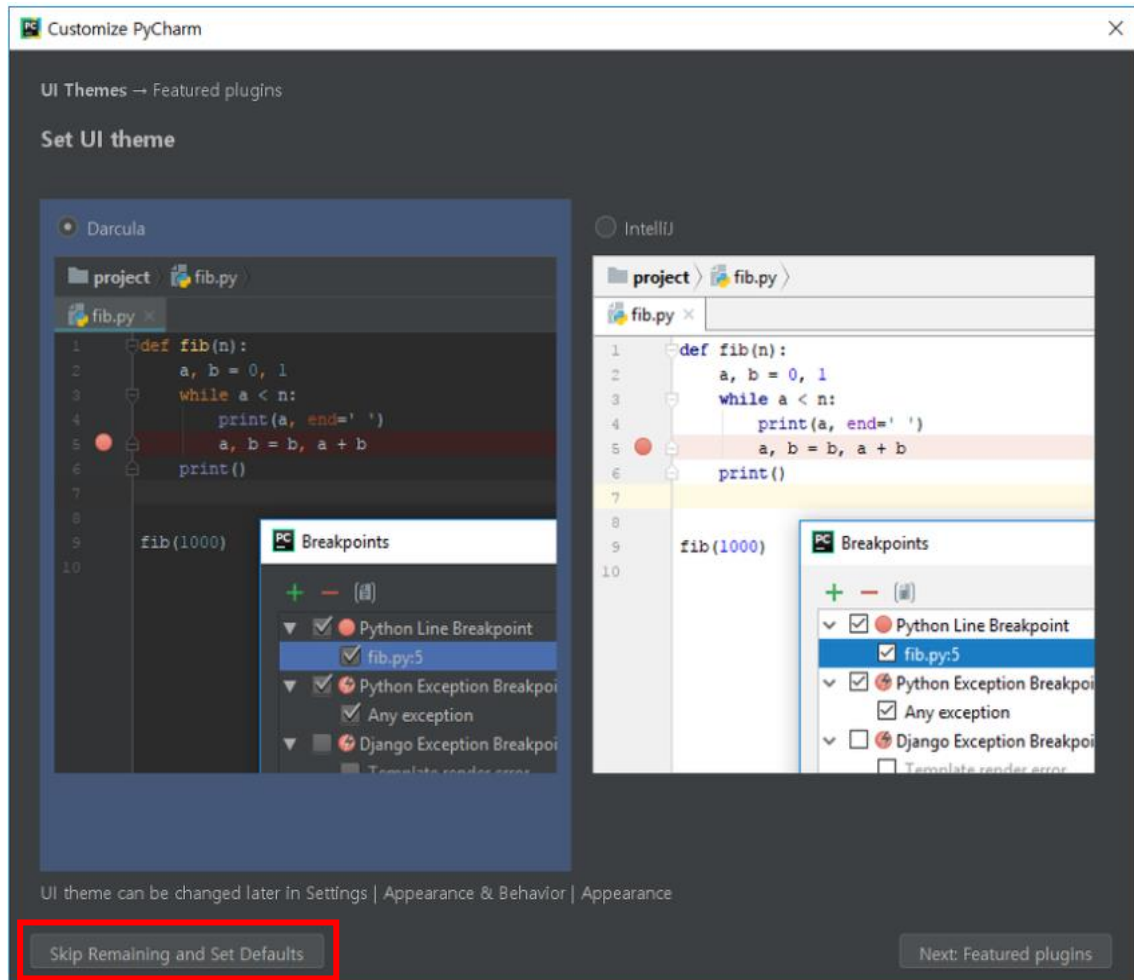
9. 계속 continue를 눌러줍니다. 아래 화면은 JetBrains 개인 정보 정책에 동의하는 창입니다. 동의한다는 **Accept**를 체크한 뒤 계속 진행합니다.



10. Pycharm UI 테마를 선택합니다.

검은색화면, 흰색 화면 중 기호에 따라 pycharm UI를 선택할 수 있습니다.

선택한 뒤 **Skip Remaining and Set Defaults**를 눌러줍니다. Next Featured Plugins를 선택해도 상관 없습니다.

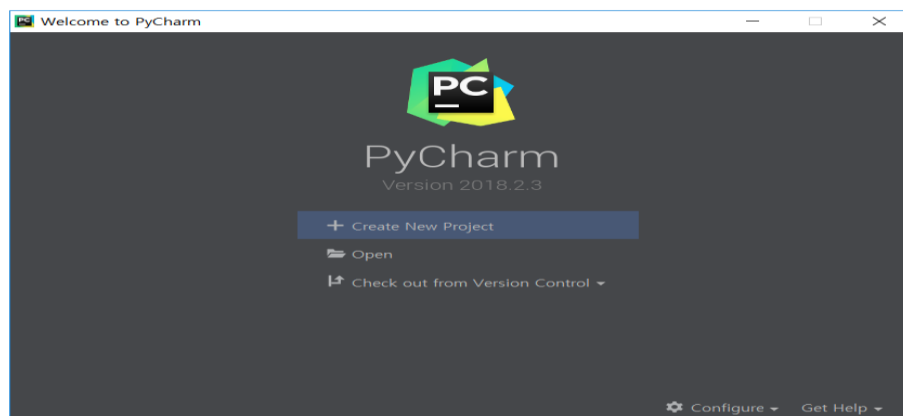


2.1.4 Pycharm 실행

pycharm을 실행하여 선박계산표와 hydrostatic curve를 그리는 것을 설명합니다.

해당 프로젝트의 Github 사이트 (<https://github.com/ARPA-GO/shipcurve>)에서 code file 인 'shipcurve_code.py' 파일을 다운받습니다.

1. Pycharm 을 실행합니다. Open 을 클릭해 다운받은 .py파일을 열어줍니다.



2. Excel 파일 불러오기

선박계산표와 Hydrostatic curve를 알고 싶은 offset표를 excel 파일의 형태로 불러옵니다. 이때 excel 파일이 저장되어 있는 정확한 위치까지 명시해야 합니다.

Offset을 불러오는 라이브러리(pd.read_excel)는 pandas를 이용하여 만든 라이브러리입니다.

```
'''
This command lines make Python import offset file from Excel.
'''
```

```
dataoffset = pd.read_excel('your excel file name.xlsx')
```

3. Pycharm의 실행에 앞서 project interpreter를 통해 코딩에 필요한 패키지를 추가해줍니다. 상단 바의 File > Settings > Project : 'project name' > project Interpreter 의 순으로 진행합니다. 오른쪽 상단에 '+' 버튼을 눌러 필요한 패키지를 추가할 수 있습니다.

DataFrame을 만들고 다루기 위해서는 Pandas라는 패키지가 필요합니다. Numpy는 벡터형 데이터와 matrix를 다루기 위해 사용하고 Matplotlib는 데이터를 시각화하기 위해 사용됩니다. 사용자는 데이터 분석을 도와주는 패키지인 **matplotlib , pandas , numpy** 를 검색해 **Install Package** 합니다.

이제, Pycharm을 통해 선박계산표와 hydrostatic curve를 구해보겠습니다.

4. 선박계산표 구하기

pycharm으로 선박계산표를 구하는 명령어를 입력하기 위해 Pycharm 창 아래 python console을 이용해야 합니다. Run 메뉴는 실행이 끝나면 python이 종료되기 때문에 dataframe을 조회할 수 없습니다. 아래 python console을 클릭하면 ipython이 실행됩니다. 코드에 커서를 놓거나 선택하여 alt+shift+E를 입력하면 아래 console 창에 해당 구문이 실행됩니다. 하지만 이와 같은 방법은 한 줄씩 실행해야 하기 때문에 시간이 오래 걸립니다.

다른 방법으로는 print 함수를 이용합니다. Print(Calculation_sheet()) 코드를 이용하여 선박계산표를 도출할 수 있습니다.

```

Run - shipcurve
Run: Untitled18 x
C:\Users\LG\Desktop\shipcurve\venv\Scripts\python.exe C:/Users/LG/Desktop/shipcurve/Untitled18.py
0 배수율적 (▽m/d.) 배수할 (△m/d.) ... 연직 주형 계수(Cvp) It II
0 1 3426.470485 3512.132248 ... 0.917736 241497.231567 5.358584e+06
1 3 11407.048469 11692.224680 ... 0.914004 291819.746571 6.806208e+06
2 5 19961.968642 20461.017858 ... 0.918658 315837.305528 7.501713e+06
3 7 28844.650244 29565.766500 ... 0.917535 333586.576572 8.052490e+06
4 9 38024.474220 38975.086076 ... 0.909157 348897.595743 8.708164e+06
5 11 47516.053440 48703.954776 ... 0.897667 362174.210419 9.610552e+06
6 13 57311.471112 58744.257890 ... 0.896344 374737.213476 1.042776e+07
7 15 67371.170435 69055.449696 ... 0.894256 387912.236476 1.119037e+07
8 17 77683.449843 79625.536089 ... 0.891503 402125.476191 1.191075e+07
9 19 88252.750236 90459.068992 ... 0.888837 417281.981614 1.266017e+07

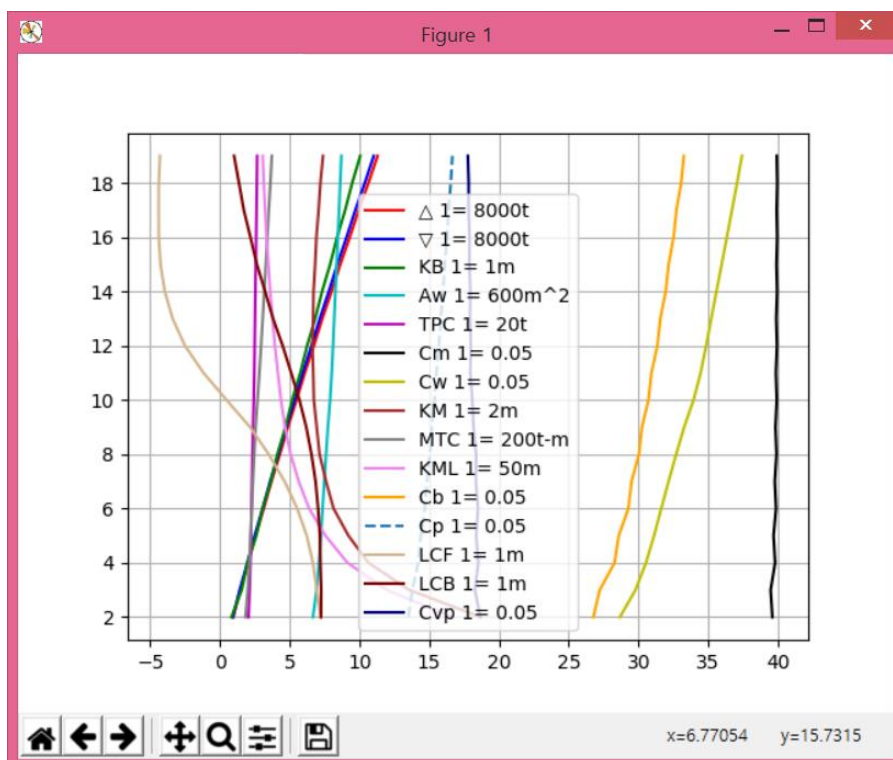
[10 rows x 21 columns]

Process finished with exit code 0

```

5. Hydrostatic curve 구하기

Pycharm 상단의 Run 메뉴를 통해 **run 'file_name'** 을 클릭해 코딩을 실행하면 hydrostatic curve가 그려진 것을 확인할 수 있습니다.



2.2 Exe File 을 통한 경로

사용자가 프로그램의 형태로 이용하고자 한다면 exe파일을 설치하여 사용하면 됩니다.

이는 사용자가 사용하기 더욱 편리할 것입니다.

2.2.1 Exe 파일 설치

해당 프로젝트의 Github사이트(<https://github.com/ARPA-GO/shipcurve>)에서 README.md에 보면 공유 문서함의 주소(<https://drive.google.com/open?id=1QNXKc0-pwct7xsZWkVGubp9pcGPgKCwM>)가 나와있습니다. 공유 문서함 → GUI form → shipcurve.exe 파일 다운로드 순으로 진행합니다.

2.2.2 Exe 파일 실행

① 설치된 **shipcurve** 파일을 실행시킵니다.

(프로그램 창이 뜨는데 약간의 시간이 소요됩니다.)



② Offset 파일 불러오기

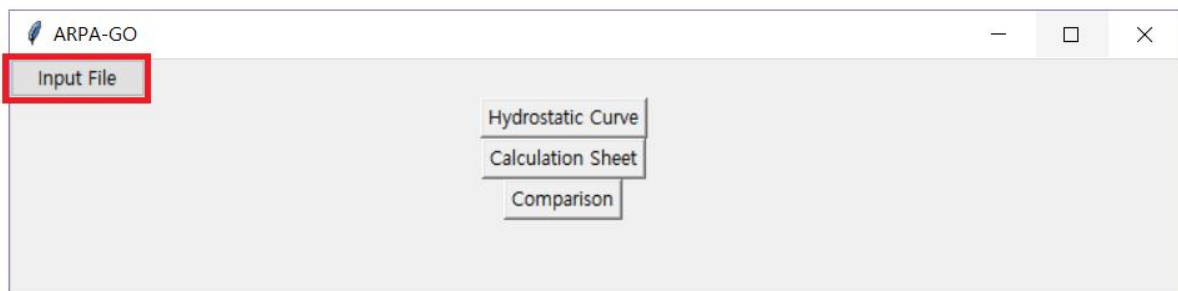
✓ offset 파일이 없을 때

도면만 있는 사용자는 부록 4.1.1 Lisp을 참고하여 쉽게 offset 파일을 만들 수 있습니다.

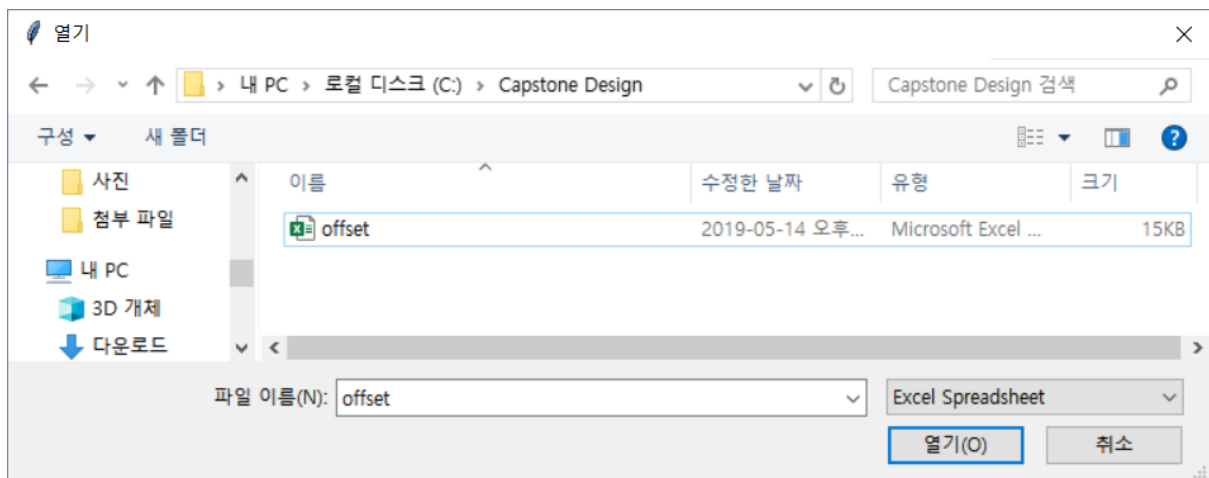
✓ offset 파일이 있을 때

Input File 버튼을 클릭합니다.

계산 값을 구하고자 하는 offset file을 선택하고 열기를 누릅니다.

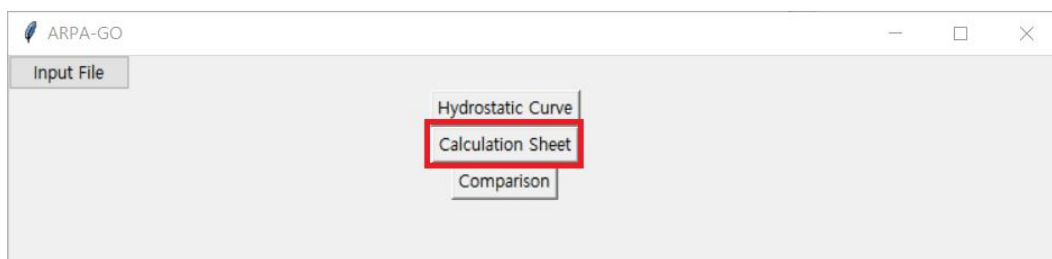


Input File 버튼을 클릭합니다.

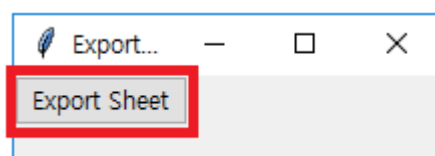


offset 파일을 선택하고 열기를 누릅니다.

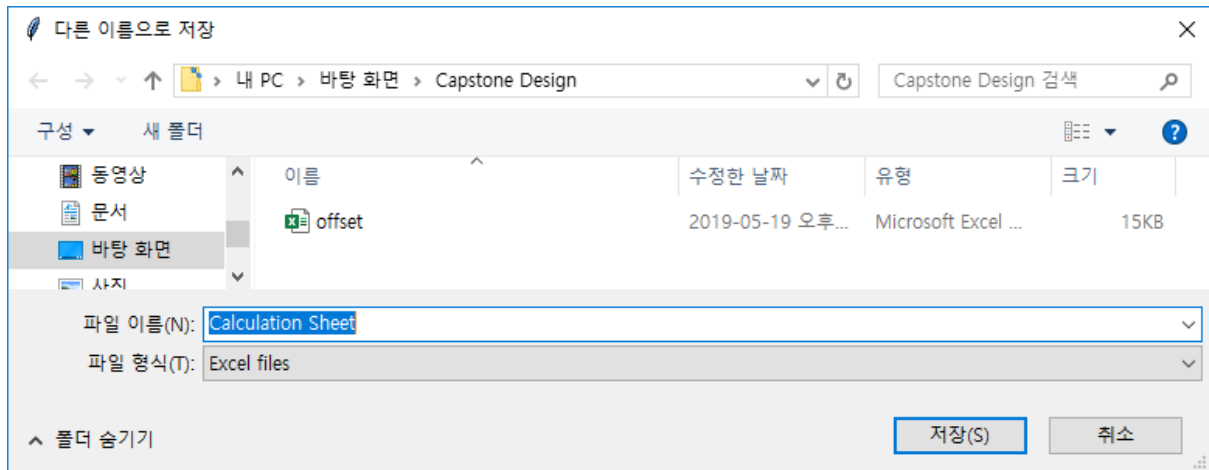
③ 선박계산표 구하기



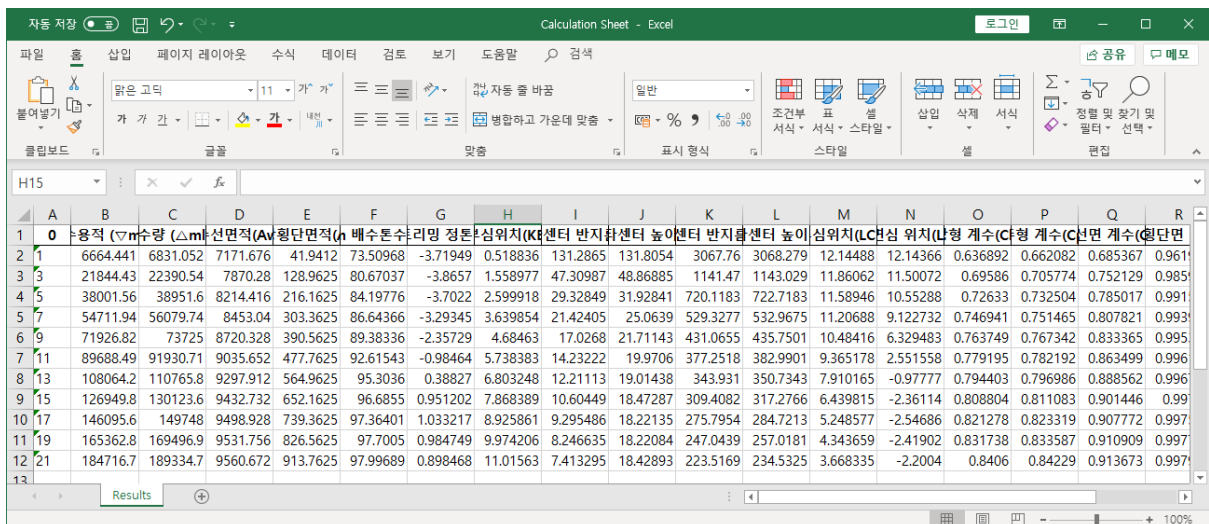
Calculation Sheet를 누르면 다음과 같은 화면이 나옵니다.



Export Sheet를 누르면

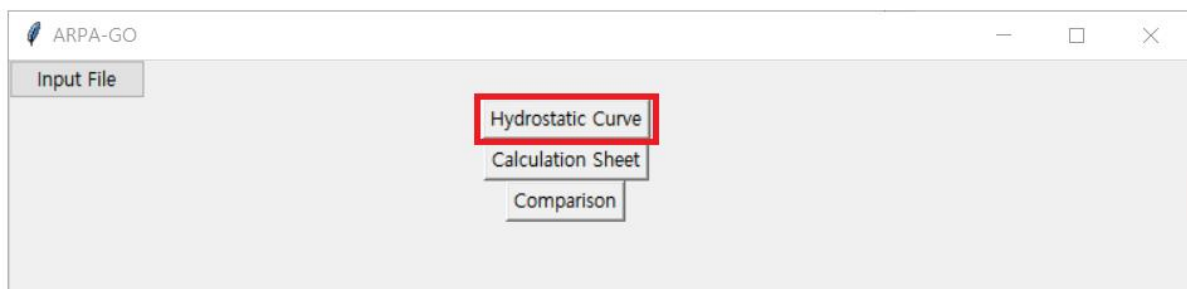


선박계산표를 저장할 경로와 이름을 정하는 창이 나옵니다.

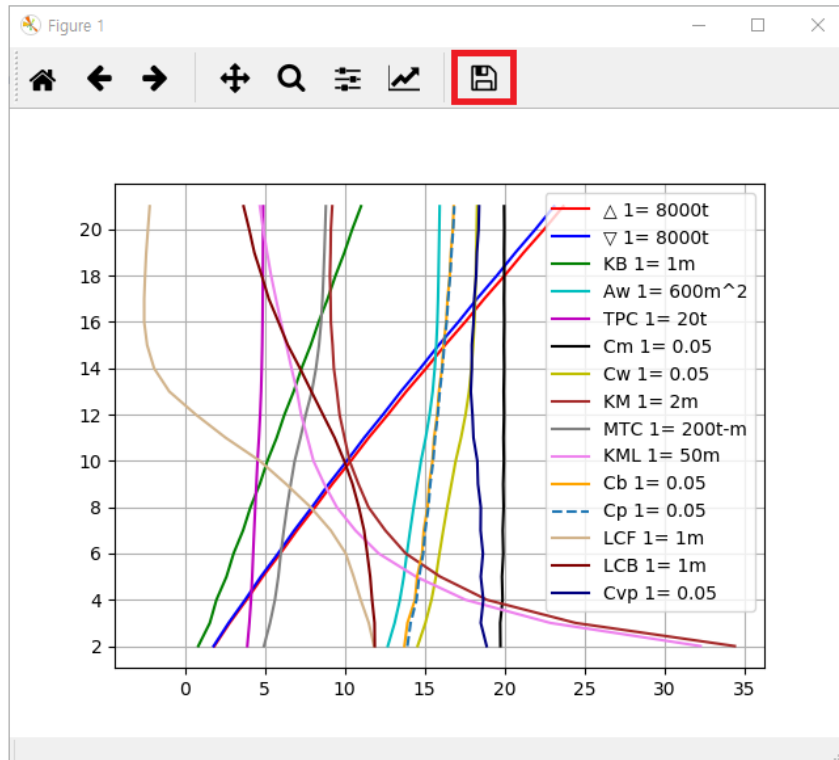


위와 같이 선박계산표가 완성된 것을 볼 수 있습니다.

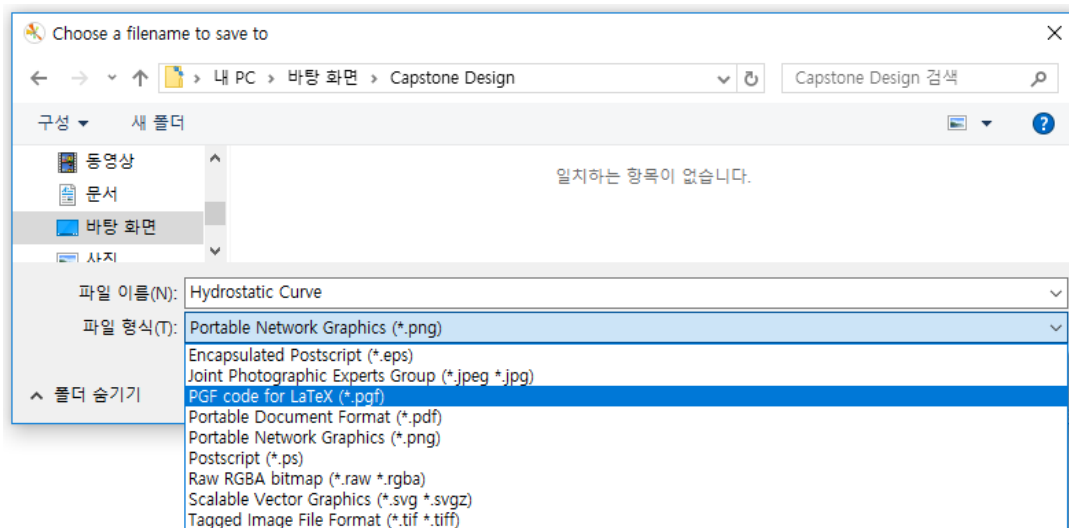
④ Hydrostatic Curve 구하기



Hydrostatic Curve 버튼을 누르면 Hydrostatic Curve가 그려진 것을 보실 수 있습니다.

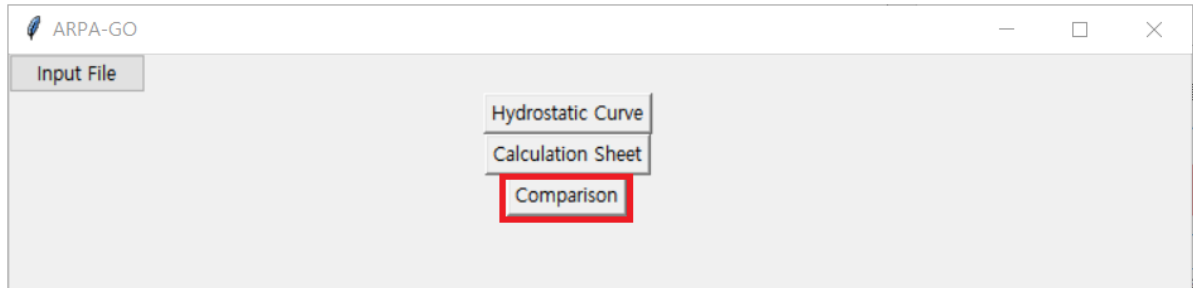


저장버튼을 누르면 원하는 형태로 Curve사진을 저장할 수 있습니다.

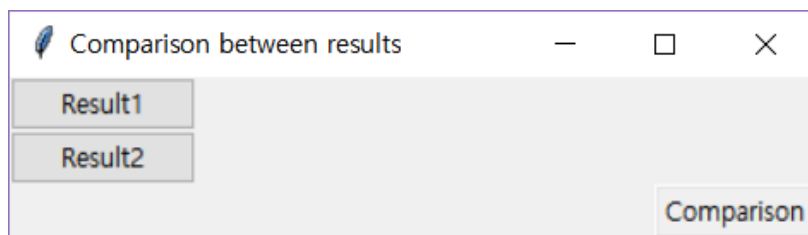


⑤ Excel 비교하기

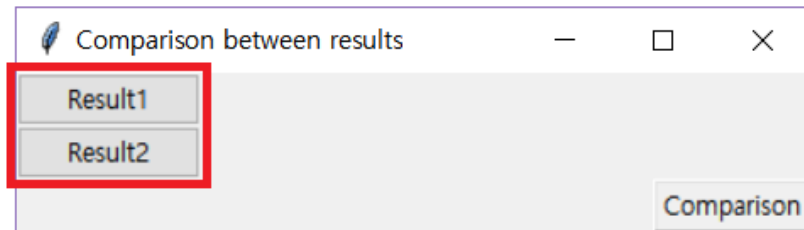
도면을 수정한 후 offset 파일이나 선박계산표가 변경됩니다. 수정전후 파일의 변경 값을 보여줍니다.



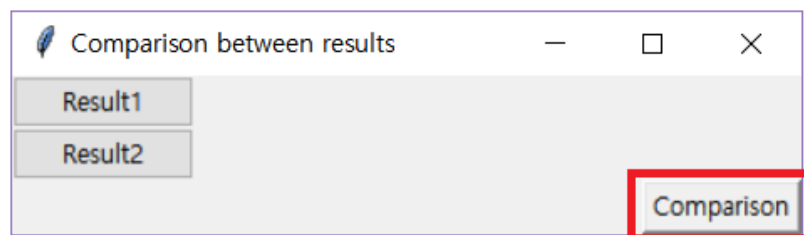
Comparison 버튼을 누르면 다음 창이 나옵니다.



Result1, Result2 을 눌러서 비교할 Excel 파일을 불러옵니다.



그리고 Comparison 버튼을 눌러 달라진 셀이 표시된 Excel 을 저장합니다.



‘ Result1 → Result2 ’ 로 표시됩니다.

Excel window: test - Excel, 이슬미

File: 자동 저장, 홈, 삽입, 페이지 레이아웃, 수식, 데이터, 검토, 보기, 도움말, 검색

Clipboard: 붙여넣기, 클립보드

Font: 맑은 고딕, 11, 가, 가, 간, 가, 가, 글꼴

Alignment: 맞춤

Number: %, 표시 형식

Styles: 조건부 서식, 표 서식, 셀 스타일, 스타일

Cells: 셀

Edit: 편집

Formula Bar: A1, X, ✓, fx

| | E | F | G | H | I | J | K | L | M |
|----|---------|--------|--------|--------|-----------|--------|-----------|-----------|--------|
| 1 | 0.5 W.L | 1 W.L | 2 W.L | 3 W.L | 4 W.L | 5 W.L | 6 W.L | 7 W.L | 8 W.L |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0.615→0.7 | 0.644 | 0 | 0 | 0 |
| 6 | 0.369 | 0.964 | 1.688 | 1.95 | 1.879 | 1.341 | 0.874 | 0.885 | 1.446 |
| 7 | 1.721 | 2.348 | 2.99 | 3.237 | 3.274 | 3.25 | 3.447 | 4.018 | 5.185 |
| 8 | 2.738 | 3.42 | 4.186 | 4.671 | 5.072 | 5.546 | 6.251→7.2 | 7.268→7.3 | 8.791 |
| 9 | 5.392 | 6.367 | 7.671 | 8.69 | 9.659 | 10.702 | 11.889 | 13.295 | 14.858 |
| 10 | 8.91 | 10.186 | 11.956 | 13.364 | 14.652 | 15.899 | 17.097 | 18.222 | 19.251 |
| 11 | 12.788 | 14.122 | 15.947 | 17.354 | 18.557 | 19.568 | 20.332 | 20.906 | 21.336 |
| 12 | 16.506 | 17.704 | 19.248 | 20.283 | 20.921 | 21.337 | 21.603 | 21.75 | 21.8 |
| 13 | 19.372 | 20.244 | 21.207 | 21.611 | 21.762 | 21.8 | 21.8 | 21.8 | 21.8 |

Sheet: DIFF

Status Bar: 100%

3. code

이장에서는 우리의 프로젝트인 ARPA-GO를 구성하는 code에 대한 설명을 합니다.

3.1 codes

컴퓨터 프로그램을 (사람이 읽을 수 있는) 프로그래밍 언어로 기술한 글을 말합니다.

3.1.1 Python 표준 라이브러리

```
import matplotlib.pyplot as plt
from matplotlib.figure import Figure
import pandas as pd
from pandas import DataFrame
```

Pycharm 은 기본적으로 제공하는 수많은 라이브러리가 존재합니다. ARPA – GO 를 만들기 위해 사용한 기본 라이브러리는 Hydrostatic Curve 를 보여주기위한 'matplotlib' 라이브러리 , offset 인 excel 파일을 가져오고 선박계산시트를 표현하기 위한 'pandas' 라이브러리를 사용하였습니다.

<matplotlib>

Matplotlib 는 python 에서 자료를 차트(chart)나 플롯(plot)으로 시각화(visualization)하는 패키지입니다.

<pandas>

Pandas 는 python 에서 사용하는 데이터 분석 라이브러리로, 행과 열로 이루어진 데이터 객체를 만들어 다룰 수 있게 되며 보다 안정적으로 대용량의 데이터들을 처리하는데 매우 편리한 도구입니다.

3.1.2 함수설명

```
"""
This command lines make Python import offset file from Excel.
"""
dataoffset = pd.read_excel('<your_excel_file_name>.xlsx')
```

'pd.read_excel'은 Python 표준 라이브러리인 pandas가 가지고 있는 함수이며, 선박의 offset표(excel file)를 Python 언어를 사용해서 불러들이는 함수입니다. pd는 pandas의 약자이며, 사용자가 지정할 수 있습니다. <your_excel_file_name> 부분에 알고 싶은 선박의 offset file이름을 저장 되어있는 경로를 포함하여 입력하고, 'dataoffset'를 Pycharm에 입력하게 되면 offset파일을 불러드릴 수 있습니다.

```
def WL(self, s, w):
    """
    In WL(s,w), 's' is for station number and 'w' for the number of waterline.
    You can command any station and waterline you want.
    This will show how far it is located from baseline and centerline of ship.
    """
    result = dataoffset.iloc[s+2, w+2]
    return result
```

'def' 는 함수를 만들 때 사용하는 예약어이며, 함수명은 함수를 만드는 사람이 임의로 만들 수 있습니다. 함수명 뒤 괄호 안의 매개변수는 이 함수에 입력으로 전달되는 값을 받는 변수입니다. 함수를 정의한 다음 if, while, for문 등과 마찬가지로 함수에서 수행할 문장들을 입력할 수 있습니다.

'return' 함수의 정의를 하고 return 뒤에 있는 명령어를 결과값으로 받아올 수 있습니다.

'iloc'는 pandas가 가지고 있는 함수로써 '[행,열]'의 순서대로 excel file의 행과 열을 입력하면, 그 좌표에 입력되어 있는 값을 추출해주는 함수입니다.

Hydrostatic curve를 만들기 위해서 먼저 그 curve를 이루는 값들을 알아야 하고, 그 값들을 알기 위해서는 하나하나의 offset 값을 알아야 합니다. 위에서 우리가 정의한 함수는 알고 싶은 offset값의 station과, waterline을 매개변수 값에 넣고 실행하면 그 값을 얻을 수 있게 됩니다.

```

def FAP(self,s,w):
    """
    The command lines are to get the value of  $\sum f(yH)$  of stern attachments.
     $\sum f(yH)$  is the value to find the area of midship section.
    You can command FAP(s,w) to get it.
    BUT, you have to be very careful about station number.
    Since, These are about stern attachments, station number starts from AP with number 0.
    And now you will know that if you type -1 in 's', you would get the value which is located middle of transom and AP and -2 is transom of stern.
    """
    a=w
    total=0
    while -2 <= a <= w:
        if == w-1:
            total += dataoffset.iloc[s+2,a+2] *4
        else:
            total += dataoffset.iloc[s+2,a+2]
        a =a-1
    total=total
    return total

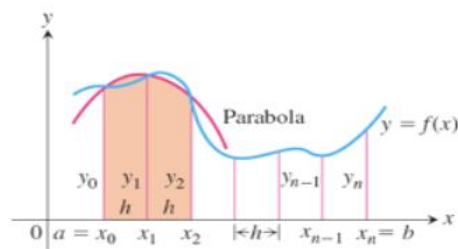
```

선박의 수선면적을 알기 위해 Simpson 공식을 적용한 것으로써, 3개씩 나누어 계산합니다. 따라서 waterline을 3개씩 나눕니다. 'while'은 Python에 내장된 기본함수로서 어떤 것을 일정한 조건까지 반복하는 함수입니다. 한 $\sum f(yH)$ 를 구하기 위해서는 3개의 waterline을 사용하기 때문에 3개가 될 때까지 반복합니다. 또한 'if'함수는 조건을 걸어주는 함수입니다.

심슨의 법칙 (Simpson's Rule)

심슨의 법칙은 선형 다항식(직선)이 아닌 2차 다항식(포물선)에 근거하여 $\int_a^b f(x)dx$ 의 근사값을 구하는 방법이다. 즉, 함수 f 의 그래프를 선분으로 근사시키는 것이 아니라 포물선의 호를 이용하여 근사시키는 방법이다. 아래 그림 1의 어두운 부분의 넓이는 다음과 같다.

$$A_p = \frac{h}{3}(y_0 + 4y_1 + y_2)$$



Simpson 공식은 첫 번째 값에 (x1), 두 번째 값에 (x4) 세 번째 값에 (x1) 해줍니다.

'if' 조건문을 사용하여 각각에 값에 맞게 적용될 수 있도록 코딩을 했습니다.

```

def FAm(self,s,w):
    """
    These are to know the area of midship section of the stern.
    Just same as above, AP starts with number 0.
    If you want to know about transom or middle of transom and AP, command -2 , -1 each in 's'.
    It is applicable in other value also, if you are to get the values of something in the place of stern attachments.
    Type the station number and waterline number you want to know.

    """
    a=s
    if a == -1:
        total = self.FAP(a,w)*4*3/8
    else:
        total = self.FAP(a,w)*3/8
    total=total
    return total

```

위의 정의된 함수는 선미부가부 station별로의 f(AM)입니다. Simpson 공식을 사용하기 때문에 'if'조건문을 사용해서 station별로 조건을 부여하였습니다.

```

def SFAm(self,w):
    """
    These command lines are to get the value of  $\sum f(Am)$  of stern attachments.
     $\sum f(Am)$  is the value, using to know the displacements of stern attachments part of ship.
    You can command SFAm(w) to get this value.
    And SFAm(w) configures the  $\sum f(Am)$  of stern attachments, which has waterline 'w-1' to 'w-2'.

    """

    a = -2
    total = 0
    while a <=0:
        total += self.FAm(a,w)
        a = a +1
    return total

```

위에 정의된 함수 'SFAm'은 선미부가부의 $\sum f(AM)$ 로써, 위에서 구한 3개의 waterline씩 f(AM)을 더한 함수입니다. 'while' 반복문을 사용하여 3개씩 더하도록 하였습니다.

```

def VAP(self,w,t):
    """
    Now, you can get the displacements of stern attachments part of ship.
    Displacement is weight of water that a ship pushes aside when it is floating, which in turn is the weight of a ship
    To get this value, we divided ship by station and waterline.
    As you know, you can just type waterline in the place of 'w', and 't' means the space between two station.

    """
    result = self.SFam(w)*self.t*2*3/8
    return result

```

위에 정의된 함수는 선미부가부의 어느 특정한 waterline 세 부분의 부피입니다. 위에서 구한 f(AM)은 선박의 반쪽을 고려한 x2를 하고, 부가물의 값이기 때문에 1/3이 아닌 3/8을 곱했습니다.

```

def WAP(self,w,t):
    """
    If you got the displacements of ship, you can definitely know the weight.
    These command lines help to get the weight.
    We got the displacements from just above command function VAP(w-1,t).
    What we have to do is just multiply density of water with its result.
    Since, ARPA-GO helps calculation of ship, the density is 1.025 which is sea water.
    You can change the density freely, if you are necessary to.
    Mentioned before, w is for waterline number, t is for the space between two stations.

    """

    if self.VAP(w-1,self.t) ==0:
        result = 0
    else:
        result = self.VAP(w,self.t) * 1.025
    return result

```

위에 정의된 함수는 선미부가부의 어느 특정한 waterline 세 부분의 무게입니다.

앞에서 구한 부피에서, 해수이기 때문에 밀도를 1.025(ton/m³)를 곱해주었습니다.

```

def FyAP(self,s,w):
    """
    FyAP(s,w) is the necessary value to get the water plane area.
    's' is for station number and 'w' is for the number of waterline.
    Don't confuse : Transom of ship is located with station number '-2' and AP is '0'.

    """
    if s == -1:
        result = dataoffset.iloc[s+2,w+2] *4
    else:
        result = dataoffset.iloc[s+2,w+2]
    return result

```

위에 정의된 함수는 $f(y)$ 를 구하기 위한 함수로써, 앞에서는 한 station에서 waterline 으로 3등분을 했다면, 이 함수는 한 waterline에서 station을 3등분 한 것입니다. 이 또한 Simpson법칙을 사용하기 때문에, 'if'조건문을 사용해 각 값에 맞는 조건을 지정해줍니다.

```

def AwAP(self,w,t):
    """
    AwAP(w,t) is the command function for water plane area.
    'w' for water line and 't' for the space between stations.

    """

    s=0
    result=0
    result1=0
    while -2<=s:
        result+= self.FyAP(s,w)
        s=s-1
        result1=result*self.t*2/3
    return result1

```

위에 정의된 함수는 선미부가부의 수선면적인 A_w 를 구하기 위한 함수입니다. 이 또한 앞에서 구한 station 별 $f(y)$ 를 더합니다. 반쪽이기 때문에 $x/2$ 와 Simpson 법칙을 사용하였기 때문에 $x/3$ 을 합니다. 3개 station씩 묶어 구하기 때문에 'while'을 사용하여 3개씩 더하게끔 반복합니다.

```

def OB(self,w,t):
    """
    The command lines help to get the value of OB.
    To prove the stability of ship, you might heard of KB,GM ...
    We put the value of OB to get KB, the distance between keel and the center of buoyancy.
    OB means the distance from center of buoyancy to waterline.
    Thus, KB+OB is the distance between keel and waterline.
    Since it is about stern attachments, the waterline starts with w-1 to w-2.

    """

    if self.AWAP(w,self.t) ==0:
        result = 0
    else:
        result = (2/2+self.VAP(w,self.t)/self.AWAP(w,self.t))/3
    return result

```

위에 정의된 함수는 선미부가부의 선박의 좌표중심으로부터 부력의 위치까지의 거리를 의미합니다.

```

def KbAP(self,w,t):
    """
    KB is the centre of buoyancy which is the height above the keel.
    KbAP(w,t) is to get the value of it.

    """

    if self.OB(w,self.t) ==0:
        result=0
    else:
        result=(w-1)-self.OB(w,self.t)
    return result

```

위에 정의된 함수는 선미부가부의 Kb를 의미합니다. Kb는 waterline – OB의 값으로 정의됩니다.

```

def xfAm(self,s,w):
    """
     $x' * f(AM)$  is the value to get the longitudinal center of gravity (LCB) of ship.
    Here, 'w' indicates 'waterline - 1' which means :
    if you want to get the value about waterline number 10, you have to enter 11 in 'w' place.
    's' is same as before.
    Station number of AP is '0'.

    """
    if s == -2:
        result = (-2)*self.FAm(s,w)
    elif s == -1:
        result = (-1)*self.FAm(s,w)
    else:
        result = 0
    return result

```

위에 정의된 함수는 각 station을 한 칸 씩 나누고 그 위치를 x' 라고 했을 때 선미부가부의 $x' * f(AM)$ 을 의미합니다. Station의 위치가 달라지면 x' 의 값도 변하기 때문에, 'if'조건문을 사용해서 각각 조건을 다르게 주었습니다.

```

def SxfAm(self,w):
    """
    The command function SxfAm(w) is the sum of  $x' * f(AM)$ .
    If you enter waterline you want to know, you will get all station's sum of  $x' * f(AM)$ .
    This function shows the value of w-1 to w-2's  $\sum x' * f(AM)$ .

    """

    s=0
    total=0
    while -2<=s:
        total += self.xfAm(s,w)
        s =s-1
    return total

```

위에 함수는 앞에서 구한 선미부가부의 $x' * f(AM)$ 의 합입니다. 전체의 합이 아닌 waterline별로 3개씩의 합을 구하기 때문에 'while' 반복문을 사용하여 3개씩 더하도록 했습니다.


```

def APb(self,w,t):
    """
    These are for APb of stern attachments of ship.
    The inputs are 'w' & 't'.
    Just same as before the waterline expresses w-1 to w-2.
    Enter the waterline what you want to know + 1 in the place of 'w'.
    't' is the distance between two stations.

    """
    if self.SFam(w) == 0:
        result = 0
    else:
        result = self.t*self.SxfAm(w)/self.SFam(w)
    return result

```

위에 정의된 함수는 선미부가부만의 무게중심을 구한 값입니다. 앞에서 구한 $\sum x' * f(AM)$ 을 $\sum f(AM)$ 으로 나누어 부가물의 무게중심을 구할 수 있습니다.

```

def lcbAP(self,w,t,L):
    """
    The centroid of the underwater volume of the ship expressed as a longitudinal location.
    We call that centroid point as LCB (longitudinal center of gravity) and it is connected with stability of ship.
    To get the value, enter the command function, lcbAP(w,t,L).
    'L' means LBP, in full form Length between perpendiculars.

    """
    if self.APb(w,self.t) == 0 or self.APb(w-1,self.t) == 0 or self.APb(w-2,self.t)==0:
        result = 0
    else:
        result = -(self.L/2)+self.APb(w,self.t)
    return result

```

위에 정의된 함수는 선박전체로 봤을 때 3개의 waterline별 LCB를 의미합니다. LCB는 주요 제원인 LBP에서 나눈 반에 앞에서 구한 선미부가부의 무게중심위치를 더하면 됩니다.

```

def lxAP(self,w,t):
    """
    BM is the metacentric radius of ship.
    Transverse metacentric height (BM) = Transverse moment of inertia of waterplane / volume displacement of ship
    To get the parameter BM, transverse moment of inertia is necessary value.
    Enter lxAP(w,t).
    DO NOT FORGET : the water line to find should be added by 1.
    """

    s=0
    result=0
    result1=0
    while -2<=s:
        if s == -1:
            result += pow(dataoffset.iloc[s+2,w+2],3)*4
        else:
            result += pow(dataoffset.iloc[s+2,w+2],3)
        s = s-1
        result1=result*self.t*2/3*1/3
    return result1

```

위에 정의된 함수는 선미부가부의 이차모멘트인 I_x 를 구하는 함수입니다. Simpson공식을 사용하여 구하기 때문에 'if'조건문을 사용해서 각 위치마다 다른 조건을 부여하고 반복문 'while'을 사용하여 3개의 waterline의 I_x 를 구할 수 있게 하였습니다.

```

def SfyAP(self,w):
    """
    The command lines are to get the value of  $\sum f(y)$  of stern attachments.
    You can command SfyAP(w) to get it.
     $\sum f(y)$  adds all the values of station and expresses with waterline form.
    Here, 'w' indicates 'waterline - 1' which means :
    if you want to get the value about waterline number 10, you have to enter 11 in 'w' place.
    """

    s=0
    result=0
    while -2<=s:
        if s == -1:
            result += (dataoffset.iloc[s+2,w+2])*4
        else:
            result += dataoffset.iloc[s+2,w+2]
        s= s - 1
    return result

```

위에 정의된 함수는 선미부가부의 waterline별 3개의 station의 $\sum f(y)$ 입니다. Simpson 법칙을 사용하기 때문에 조건문 'if'을 사용하여 각각 station이 바뀔 때마다 다른 조건을 부여하고, 'while' 반복문을 사용하여 3개의 station별로 $\sum f(y)$ 를 구하도록 했습니다.

```

def SxfyAP(self,w):
    """
    The command function SxfyAP(w) is the sum of  $x' * f(y)$ .
    If you enter waterline you want to know, you will get all station's sum of  $x' * f(y)$ .
    This function shows the value of w-1 to w-2's  $\sum x' * f(y)$ .
    """

    s=0
    total=0
    while -2<= s:
        if s ==-1:
            total += dataoffset.iloc[s+2,w+2]*(-1)*4
        elif s ==0:
            total += 0
        else:
            total += dataoffset.iloc[s+2,w+2]*(-2)
        s=s-1
    return total

```

위에 정의된 함수는 선미부가부의 $\sum x' * f(y)$ 입니다. . Simpson 법칙을 사용하고 station 별로 x'의 값이 바뀌기 때문에 조건문 'if'을 사용하여 각각 station이 바뀔 때마다 다른 조건을 부여하고, 또한 'while'반복문을 사용하여 3개의 station별로 $\sum x' * f(y)$ 를 구하도록 했습니다.

```

def APf(self,w,t):
    """
    The result of command lines are APf of stern attachments of ship.
    This parameter is necessary to get the longitudinal moment of inertia of stern attachments part.
    'w' : waterline + 1
    't' : distance between two stations in stern attachments part
    """

    if self.SfyAP(w)==0:
        result=0
    else:
        result = self.t*self.SxfyAP(w)/self.SfyAP(w)
    return result

```

위에 정의된 함수는 선미부가부의 무게중심의 높이방향의 위치입니다. $\sum x' * f(y)$ 을 $\sum f(y)$ 을 나눈 x'의 값을 구합니다.

```

def MyAP(self,w,t,L):
    """
    These are command lines for longitudinal moment of stern attachments.
    'w' : waterline + 1
    't' : distance between two stations in stern attachments part
    'L' : Length between perpendiculars
    """

    result = self.AWAP(w,self.t)*((-self.L)/2+self.APf(w,self.t))
    return result

```

위에 정의된 함수는 선미부가부의 1차 면적모멘트 값을 의미합니다.

이렇게 각각의 값들을 선미부가부, 주요부, 선수부가부로 나누어 진행합니다. 그 후 각각의 값을 다 합쳐 선박계산표와 curve에 들어가는 값을 구합니다.

```

def DISLIST(self,w,h,f,t,j):
    """
    Now, you can get the displacements of ship.
    Displacement is weight of water that a ship pushes aside when it is floating.
    To get this value, we divided ship by station and waterline.
    These command lines will show bi-waterline wise.
    'w' : waterline + 1
    'h' : distance between two stations in main part of ship
    'f' : distance between waterlines
    't' : distance between two stations in stern attachments part
    'j' : distance between two stations in bow attachments part
    """

    a=w
    result=[]
    while a>=2:
        result.append(self.DIS(a,self.h,self.f,self.t,self.j))
        a=a-2
    return result

```

위의 정의된 함수는 선미부가부, 주요부, 선수부가부의 구한 배수량을 다 더한 후 waterline 별로의 배수량을 한번에 확인할 수 있습니다. 'append'는 각각의 값들을 list의 형태로 변환시킨 것이며, list를 형성하기 전 '[]'로 정의를 해주어야 합니다. 선박계산표에 들어가는 값들을 list의 형태로 변경시켜 줍니다.

```
def WLlist(w, h, f, L, B):  
    a = 1  
    result = []  
    for a in range(1, w + 1, 2):  
        result.append("%d " % a)  
        a = a + 2  
    return result
```

위에 정의된 함수는 반복문 'for'을 사용하여 선박계산표의 마지막 waterline 숫자 순서대로 2칸 간격으로 표의 가장 왼쪽에 쓰여지게 되어 waterline 별로 값을 알 수 있게 됩니다.

```
def Calculation_sheet():
```

```
w = dataoffset.shape[1] - 3  
h = WL(7, -1) - WL(6, -1)  
f = 1  
t = -WL(-1, -1)  
j = WL(25, -1) - WL(24, -1)  
L = WL(24, -1)  
B = WL(12, w)  
data = WLlist(w, h, f, L, B)
```

```
result = DataFrame(data)
```

```
result['배수용적 (▽mld.)'] = DISLIST2(w, h, f, t, j)  
result['배수량 (△mld.)'] = DISLIST(w, h, f, t, j)  
result['수선면적(Aw)'] = AwLIST(w, h)  
result['중앙횡단면적(Am)'] = AmLIST(w)  
result['매 Cm 배수튼수(TPC)'] = TPCLIST(w, h)  
result['매 Cm 트리밍 정돈수(Wcm)'] = WcmLIST(w, h, t, j, L)  
result['부심위치(KB)'] = KBLIST(w, h, f, t, j)  
result['횡메타센터 반지름(BM)'] = BMLIST(w, h, f, t, j)  
result['횡메타센터 높이(KM)'] = KMLIST(w, h, f, t, j)  
result['종메타센터 반지름(BML)'] = BMLLIST(w, h, f, t, j, L)  
result['종메타센터 높이(KML)'] = KMLLIST(w, h, f, t, j, L)  
result['부심위치(LCB)'] = LCBLIST(w, h, f, t, j, L)  
result['부면심 위치(LCF)'] = LCFLIST(w, h, t, j, L)  
result['방형 계수(Cb)'] = CbLIST(w, h, f, L, B, t, j)  
result['주형 계수(Cp)'] = CpLIST(w, h, f, L, B, t, j)  
result['수선면 계수(Cw)'] = CwLIST(w, h, L, B)
```

```

result['수선면 계수(Cw)'] = CwLIST(w, h, L, B)
result['중앙 횡단면 계수(Cm)'] = CmLIST(w, B)
result['연직 주형 계수(Cvp)'] = CvpLIST(w, h, f, L, B, t, j)
result['lt'] = SSIXLIST(w, t, j, h)
result['ll'] = lllIST(w, h, t, j, L)

```

```

return result

```

위의 정의된 함수는 offset표를 받아 선박계산표를 보여주는 함수입니다. Pandas의 내장함수인 'dataframe'을 사용하여 표를 만들고, [원하는 그룹의 이름]을 적어 각각의 값이 어떤 것을 의미하는 지 알 수 있게 됩니다. 이 앞에 값들은 각각의 변수를 입력해야만 했지만, 선박계산표와 hydrostatic curve를 구하는 함수는 사용자가 가장 많이 사용하는 함수이므로 편의를 위해 offset 표에서 자동으로 각각의 변수들을 받아 함수 이름만 입력하면 자동으로 선박계산표를 보여주게끔 하였습니다.

'w' = waterline의 마지막 숫자는 'pandas'에 내장 되어있는 함수 '.shape[1]'을 사용하여 마지막 waterline 숫자를 받아옵니다. 받아온 offset의 column의 개수를 받아오는데, 'pandas'에서는 숫자가 '0'부터 시작되고, 각각의 waterline의 숫자에 대한 값이 있기 전 다른 값들이 존재하기 때문에 그 waterline 마지막 숫자에 맞게 값을 빼줍니다.

'h' 는 주요부 station 간격, 't'는 선미부 station 간격 'j'는 선수부 station 간격을 의미합니다. 이 값들도 offset표에서 받아옵니다.

'L'은 선박의 LBP를 의미합니다. 선수부가부까지의 거리를 offset표에서 받아옵니다.

'B' 는 선박의 폭을 의미합니다. 대부분 선박에 '10 station'의 마지막 값과 같습니다.

```

def Hydrostatic_curve():
    w = dataoffset.shape[1]-3
    h = WL(7, -1) - WL(6, -1)
    f = 1
    t = -WL(-1, -1)
    j = WL(25, -1) - WL(24, -1)
    L = WL(24, -1)
    B = WL(12, w)
    data = WLlist(w, h, f, L, B)
    y = range(2, w)
    plt.yticks(range(2, w, 2))
    plt.xticks(range(-5, 150, 5))
    x = [DIS(a + 1, h, f, t, j) / 8000 for a in y]
    z = [DIS(a + 1, h, f, t, j) / 8000 / 1.025 for a in y]
    e = [KB(a + 1, h, f, t, j) for a in y]
    g = [Aw(a + 1, h) / 600 for a in y]
    p = [TPC(a + 1, h) / 20 for a in y]
    u = [Cm(a + 1, B) / 0.05 for a in y]
    c = [Cw(a + 1, h, L, B) / 0.05 for a in y]
    q = [KM(a + 1, h, f, t, j) / 2 for a in y]
    m = [MTC(a + 1, h, f, t, j, L) / 200 for a in y]
    v = [KML(a + 1, h, f, t, j, L) / 50 for a in y]
    ww = [Cb(a + 1, h, f, L, B, t, j) / 0.05 for a in y]
    ii = [Cp(a + 1, h, f, L, B, t, j) / 0.05 for a in y]
    kk = [SSLCF(a + 1, h, t, j, L) for a in y]
    hh = [LCB(a + 1, h, f, t, j, L) for a in y]
    ll = [Cvp(a + 1, h, f, L, B, t, j) / 0.05 for a in y]

```



```

plt.plot(x, y, 'r')
plt.plot(z, y, 'b')
plt.plot(e, y, 'g')
plt.plot(g, y, 'c')
plt.plot(p, y, 'm')
plt.plot(u, y, 'k')
plt.plot(c, y, 'y')
plt.plot(q, y, color='brown')
plt.plot(m, y, color='grey')
plt.plot(v, y, color='violet')
plt.plot(ww, y, color='orange')
plt.plot(ii, y, '—')
plt.plot(kk, y, color='tan')
plt.plot(hh, y, color='maroon')
plt.plot(ll, y, color='navy')
plt.grid()
plt.legend(
    ['△ 1= 8000t', '▽ 1= 8000t', 'KB 1= 1m', 'Aw 1= 600m^2', 'TPC 1= 20t', 'Cm 1= 0.05', 'Cw 1= 0.05', 'KM 1= 2m',
     'MTC 1= 200t-m', 'KML 1= 50m', 'Cb 1= 0.05', 'Cp 1= 0.05', 'LCF 1= 1m', 'LCB 1= 1m', 'Cvp 1= 0.05'])
return plt.show()

```

위의 정의된 함수는 hydrostatic curve 를 보여줍니다. 'matplotlib.pyplot'라이브러리를 활용해 정의되었습니다. 이 라이브러리를 'plt'로 불러 사용할 수 있게끔 지정했습니다.

이 라이브러리에 있는 함수인 'yticks'와 'xticks'는 그래프의 각각의 (시작점, 끝점, 간격)의 순으로 적어 지정할 수 있습니다. 'range'는 2 부터 waterline 마지막 모양) 숫자까지의 배열로, y 축을 이루게 지정하고 각각에 대한 x 축은 '변수 = [함수]'로 만 no.2 들어 '[']안에 'for' 반복문을 이용하여 마지막 waterline 까지의 값을 waterline no.2 부터 누적으로 넣을 수 있도록 지정하였습니다.

또한, 'matplotlib.pyplot' 라이브러리에 내장되어 있는 함수인, 'plot'을 사용하여 (x 축 값, y 축 값, 선의 색깔순으로 지정해줍니다. 한 창에 여러 개의 그래프가 가시화됨으로 그래프마다 색깔을 달리하였습니다. 또한 'grid()'함수를 사용하여 사용자가 쉽게 값을 읽을 수 있도록 격자를 넣고, 각 그래프가 의미하는 것을 설명해 주기 위해 'legend()'함수를 이용하였습니다.

마지막으로 'show()'함수로 그래프를 가시화할 수 있게 됩니다. 앞의 선박계산함수와 같게 각각의 주요 제원을 자동으로 offset 표에서 얻어옵니다.

3.2 GUI 형식

선박계산표와 hydrostatic curve를 구하기 위한 다른 방법은 ARPA-GO의 GUI형태를 사용하면 됩니다. 이 형태는 실행이 편하고, 뚜렷한 결과물을 얻을 수 있다는 장점이 있습니다.

3.2.1 메인 윈도우

Python은 GUI(Graphical User Interface)형식 개발을 위해 다양한 모듈을 제공합니다. 다양한 GUI 방법 중에서 'tkinter'가 가장 많이 사용되는 방법입니다. 이는 Python 내부의 라이브러리입니다. 'tkinter'를 이용한 Python의 GUI 애플리케이션은 가장 빠르고 쉽게 생성하는 방법입니다.

```
import tkinter
```

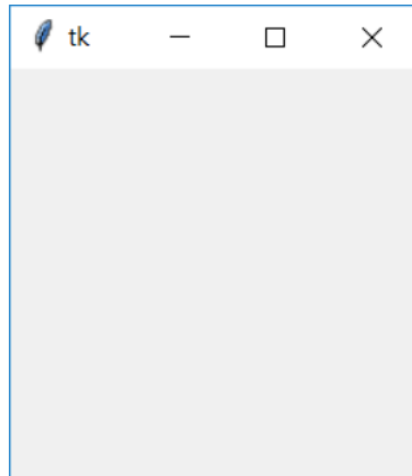
상단 명령어를 사용하여 GUI 모듈을 포함시킵니다. 'tkinter' 함수의 사용방법은 'tkinter.*'를 이용하여 사용이 가능합니다.

```
import tkinter
window=tkinter.Tk()
window.mainloop()
```

이 명령어에서 window는 실행 창 이름을 의미하며, 사용자가 원하는 이름대로 선언할 수 있습니다. 표시된 명령어를 통해 가장 상위 레벨의 윈도우 창을 생성할 수 있습니다. 경우에 따라 'tkinter'를 tk(혹은 ttk)로 줄여 window = tk.Tk()와 같이 응용도 가능합니다.

마지막 명령어의 의미는 window라는 실행창이 종료될 때 까지 실행시키는 명령어함수입니다.

마찬가지로, window자리에 사용자가 원하는 실행 창 이름을 적용시키면 됩니다.

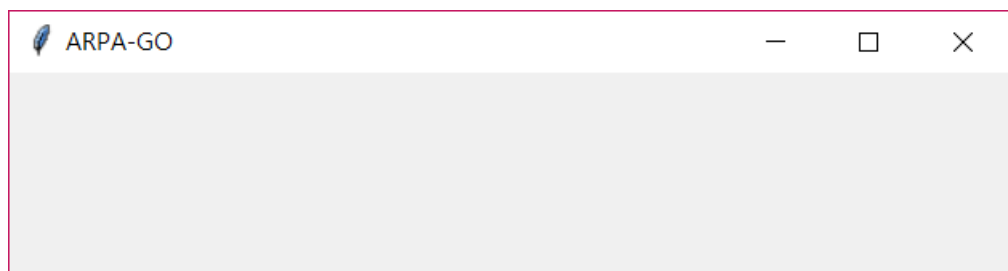


이를 실행할 경우, 가장 기본적인 윈도우 창이 생성됩니다.

```
class ARPAGO(object):  
  
    def __init__(self):  
  
        root = tk.Tk()  
  
        root.title("ARPA-GO") #window 이름 설정  
  
        root.geometry("500x100") #window 크기 설정  
  
        root.mainloop()
```

프로젝트의 윈도우 창에 대한 코딩입니다.

우리의 프로젝트명인 "ARPA-GO"를 실행창의 이름으로 설정하였으며, 적절한 크기를 설정했습니다.



이러한 코딩을 마치면, 프로젝트의 메인 윈도우 창을 생성할 수 있습니다.

3.2.2 Excel 불러오기

코딩과 마찬가지로 구하고자 하는 선박의 옵션이 저장된 엑셀파일을 불러오는 작업을 요구합니다. GUI 형식에서는 옵션을 불러오기 위해 파일 열기를 통해 해당 엑셀파일을 열어줍니다.

이를 위해서는 버튼 위젯을 사용해야 합니다. 버튼 위젯은 기본적인 'tkinter' 위젯입니다. 버튼은 사용자가 상호 작용하도록 설계된 위젯으로 즉, 마우스로 버튼을 누를 경우 일부 작업을 시작할 수 있습니다. 또한, 이 버튼들은 라벨과 같은 텍스트와 이미지를 포함할 수 있습니다. 라벨은 텍스트를 다양한 글꼴로 표시할 수 있지만 버튼은 텍스트만 단일 글꼴로 표시할 수 있으며, 텍스트는 한 줄 이상 가능합니다.

Python 기능이나 방법은 버튼과 연결될 수 있습니다. 버튼을 어떤 식으로든 누르면 해당 기능이 실행됩니다. 우리의 프로젝트 ARPA-GO는 원하는 결과값들을 얻기 위해 버튼 위젯의 이 기능을 사용했습니다.

파일 불러오기 버튼

```
frm = ttk.Frame(root)

frm.grid(column=0, row=0, sticky=(tkinter.N, tkinter.W, tkinter.E, tkinter.S))

frm.columnconfigure(0, weight=1)

frm.rowconfigure(0, weight=1)
```

self.filename_var = tkinter.StringVar()

버튼을 만들기 위해서는 가장 먼저 실행창의 위치에 대한 함수를 선언합니다. 빨간 박스의 부분이 앞서 생성한 실행창의 동,서,남,북 위치 별로 함수를 선언한 코딩입니다.

```
openbt_inputoffset = ttk.Button(frm, text="Input File", command=self.select_offsetfile) # name
openbt_inputoffset_ttp = CreateToolTip(openbt_inputoffset, "Select offset (Excel file)")
openbt_inputoffset.grid(column=1, row=1, sticky=tkinter.W)
openbt_inputoffset['command'] = self.select_offsetfile
display_box = ttk.Label(frm, textvariable=self.filename_var, width=30)
display_box.grid(column=2, row=1, sticky=((tkinter.W, tkinter.E)))
```

메인 윈도우를 만드는 것과 유사한 방식으로 버튼 이름을 먼저 정하고, 위젯을 이용하여 버튼을 생성하며, 버튼의 크기와 위치도 모두 설정 가능합니다. 이 버튼에 해당하는 명령함수는 open_button으로 설정하였으며, 버튼 내에 적힐 글은 "Input File"입니다.

빨간 박스에는 `command = self.select_file` 이라는 명령어가 있습니다. 이는 이 버튼을 클릭 시 `self.select_file`이라는 class 명령을 따른다는 것입니다. Python의 Class 기능에 대해서는 부록에서 더 자세히 다루겠습니다.

버튼의 기능을 효율적으로 이용하기 위해서는 버튼의 툴팁(tooltip)을 작성하는 방법이 있습니다. 해당 요소에 마우스를 올리면 추가적인 정보가 나타나게 하는 효과를 툴팁(tooltip) 효과라고 합니다. 이 효과를 이용하여 버튼을 설명하도록 했습니다. 해당 명령어는 위 코딩의 파란 박스입니다.

파일을 불러오는 버튼을 완성적으로 만들었다면, 버튼 클릭 시 수행 될 기능을 추가하는 작업이 필요합니다.

```
# Import offset file
```

```
def select_offsetfile(self):
```

```
    global dataoffset
```

```
    filename = filedialog.askopenfilename(
```

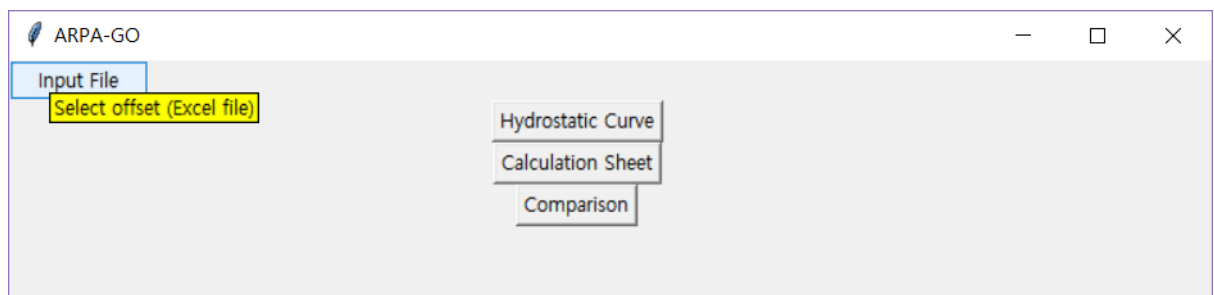
```
        filetypes=[('Excel Spreadsheet', '*.xlsx'), ('Excel Spreadsheet', '*.xls'), ('All files', '*.*')])
```

```
    self.filename_var.set(filename)
```

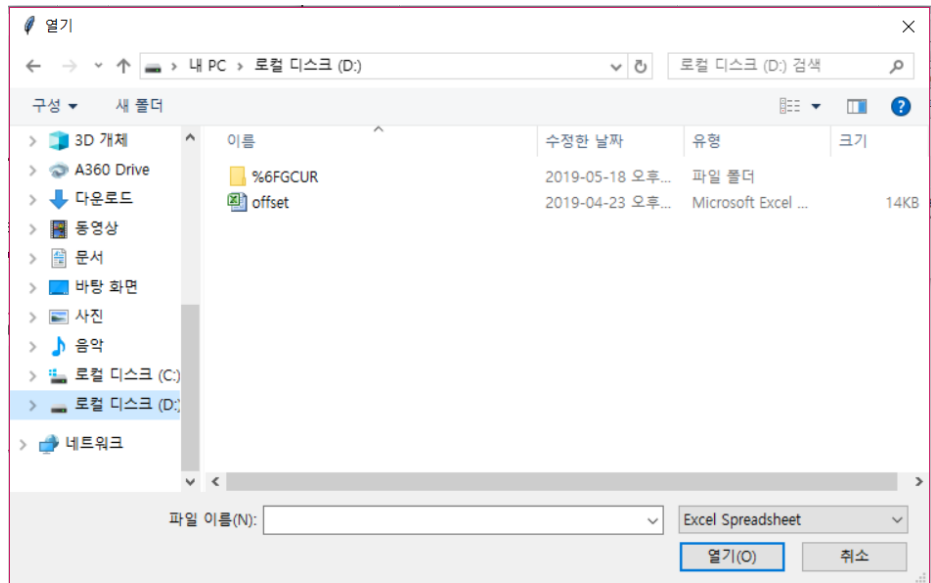
```
    dataoffset = pd.read_excel(filename)
```

이 버튼에 해당하는 기능은 파일을 불러오는 작업입니다. 'filename'은 엑셀을 불러오기 위해 선언한 명령어입니다. 이때, 파일의 종류에 따른 확장자 또한 선택할 수 있습니다. 저희 프로젝트는 구하고자 하는 선박의 엑셀이 포함된 엑셀 파일을 통해 계산을 진행하므로, 엑셀파일을 우선적으로 작성하였습니다.

위와 같은 코딩에 대한 결과값은 다음과 같습니다.



생성된 Input File 버튼에 대한 설명이 Tooltip을 통해 간략하게 적혀있습니다.



버튼을 클릭함으로써 사용 기기에 저장되어 있는 파일을 불러오게 됩니다.

3.2.3 선택계산표

파일 불러오기와 유사한 방식입니다. 버튼을 먼저 생성하고, 해당 버튼과 작동할 기능을 추가하면 됩니다.

Calculation Sheet Button

```
bt_CalcSheet = tk.Button(root, text="Calculation Sheet", command=self.get_window_CalcSheet)
bt_CalcSheet.grid(column=1, row=6)
```

‘Calculation Sheet’ 라는 이름의 버튼을 생성하고, 위치를 설정해 줍니다.

```
def get_window_CalcSheet(self):
    root = tk.Tk()
    root.title('Export Calculation Sheet')

    frm = ttk.Frame(root)
    frm.grid(column=0, row=0, sticky=(tkinter.N, tkinter.W, tkinter.E, tkinter.S))
    frm.columnconfigure(0, weight=1)
    frm.rowconfigure(0, weight=1)
```

이전과 유사하게 윈도우 창을 동,서,남,북 4방향으로 함수를 선언합니다.

```
openbt_Inputoffset = ttk.Button(frm, text="Export Sheet", command=self.load_file) # name
openbt_Inputoffset_ttp = CreateToolTip(openbt_Inputoffset,
                                         "Click this button to save calculation sheet in Excel form")
# When click button, show explanatory window
openbt_Inputoffset.grid(column=1, row=1, sticky=tkinter.W)
openbt_Inputoffset['command'] = self.load_file

display_box = ttk.Label(frm, textvariable=self.filename_var, width=30)
display_box.grid(column=1, row=2, sticky=((tkinter.W, tkinter.E)))
root.mainloop()
```

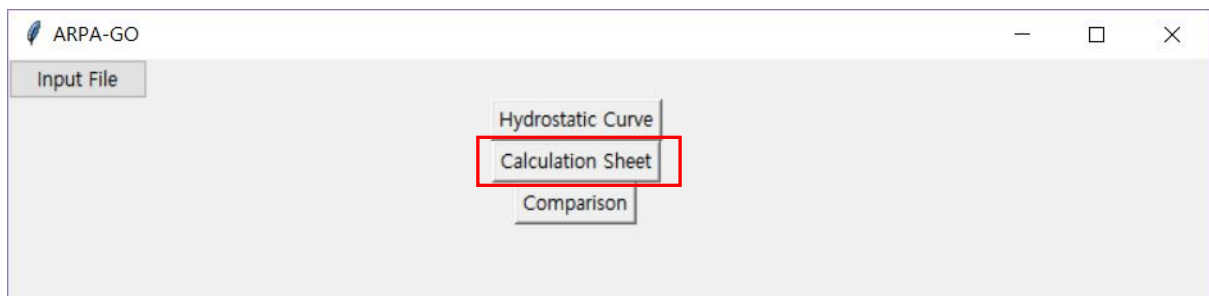
마찬가지로 버튼의 이름과 명령을 따른 클래스를 선정합니다. 그 후, 버튼에 대한 설명을 위한 툴팁을 작성하며 버튼 위치를 결정합니다.

```
# Export Calculation Sheet
```

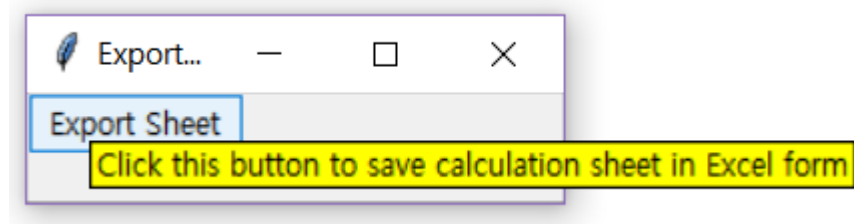
```
def export_calcsheet(self):  
    file = Calculation_sheet()  
  
    savefile = filedialog.asksaveasfilename(filetypes=(("Excel files", "*.xlsx"), ("All files", "*.*")))   
    file.to_excel(savefile + ".xlsx", index=False, sheet_name="Results")
```

다음은 기능을 추가할 차례입니다. 옵셋을 통해 얻은 자료로 계산된 선박 종합계산표를 기기에 엑셀파일로 저장해줍니다. 계산 값들을 한눈에 볼 수 있도록 엑셀파일로 작성하게끔 했지만, 파일의 확장자를 변화함으로써 다른 파일의 형태로도 저장 가능합니다.

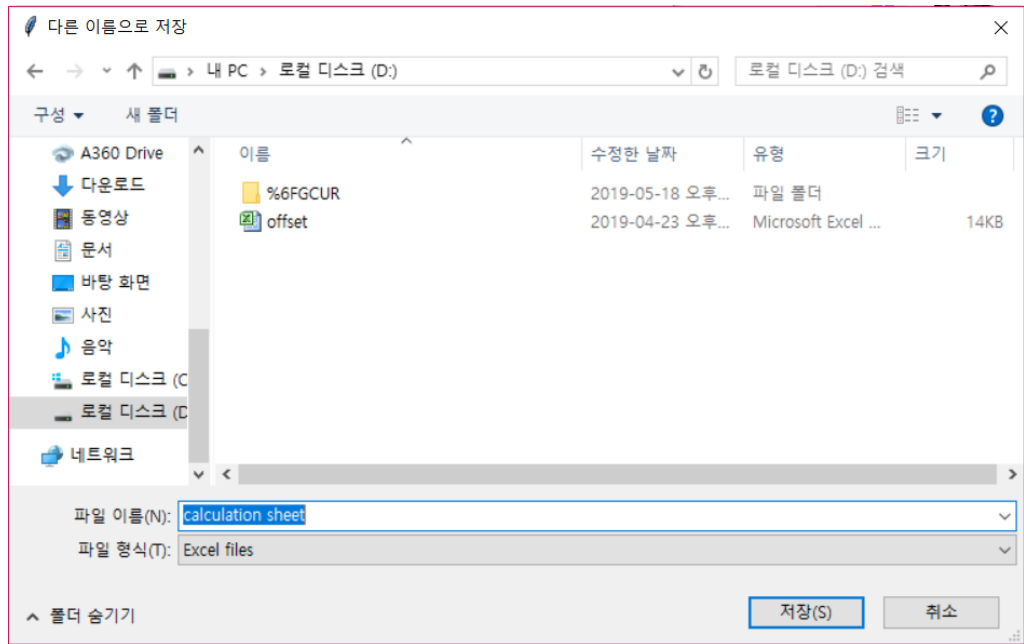
버튼에 대한 기능을 추가하면 코딩은 완성합니다. 이에 대한 결과는 다음과 같습니다.



'Calculation Sheet'라는 버튼이 생성됩니다.



이를 클릭하면, 생성된 Export Sheet 버튼에 대한 설명이 Tooltip을 통해 간략하게 적혀있습니다.



버튼을 클릭함으로써 선박계산표를 엑셀 파일로 저장하게 됩니다.

| 0 | 배수용적 (▽mld.) | 배수량 (△mld.) | 수선면적(Aw) | 중양철단면적(Am) | 매 Cm 배수톤수(TPC) | 매 Cm 트리밍 정돈수(Wcm) | 부심위치(KB) | 필메타센터 반지름(BM) |
|----|--------------|-------------|----------|-------------|----------------|-------------------|-------------|---------------|
| 1 | 6664.440743 | 6831.051761 | 7171.676 | 41.9412 | 73.509679 | -3.719486481 | 0.518835885 | 131.2865393 |
| 3 | 21844.43151 | 22390.5423 | 7870.28 | 128.9625333 | 80.67037 | -3.865696848 | 1.558977416 | 47.30987283 |
| 5 | 38001.5644 | 38951.60351 | 8214.416 | 216.1625333 | 84.197764 | -3.702202824 | 2.599918436 | 29.32848755 |
| 7 | 54711.94286 | 56079.74143 | 8453.04 | 303.3625333 | 86.64366 | -3.293445372 | 3.639854162 | 21.42404683 |
| 9 | 71926.82473 | 73724.99535 | 8720.328 | 390.5625333 | 89.383362 | -2.357293708 | 4.684629648 | 17.02680237 |
| 11 | 89688.49348 | 91930.70581 | 9035.652 | 477.7625333 | 92.615433 | -0.984640294 | 5.738383016 | 14.23221511 |
| 13 | 108064.1937 | 110765.7986 | 9297.912 | 564.9625333 | 95.303598 | 0.388269597 | 6.803248181 | 12.21113467 |
| 15 | 126949.8328 | 130123.5786 | 9432.732 | 652.1625333 | 96.685503 | 0.951201964 | 7.868389203 | 10.60448544 |
| 17 | 146095.5675 | 149747.9567 | 9498.928 | 739.3625333 | 97.364012 | 1.03321711 | 8.925860868 | 9.295486072 |
| 19 | 165362.8492 | 169496.9205 | 9531.756 | 826.5625333 | 97.700499 | 0.984748896 | 9.974206197 | 8.246635082 |
| 21 | 184716.7413 | 189334.6598 | 9560.672 | 913.7625333 | 97.996888 | 0.898468235 | 11.01563021 | 7.413295449 |

실행창에서 'Calculation Sheet' 버튼을 클릭하면, 선박의 종합계산표가 엑셀파일로 저장합니다. 이를 통해 선박계산표를 자세히 검토하고, 한눈에 해당 스테이션과 워터라인 값들을 확인 할 수 있습니다. 위에 보이는 계산식은 일부에 불과하며 "ARPA-GO"를 통해 총 20개의 선박 계산표를 확인 할 수 있습니다.

3.2.4 Hydrostatic curves

선박의 Hydrostatic curve를 구하는 명령어는 비교적 간단합니다.

```
# Hydrostatic Curve Button
```

```
bt_HydroC = tk.Button(root, text="Hydrostatic Curve", command=self.get_window_hydrostaticC)
```

```
bt_HydroC.grid(column=1, row=5) # Location
```

이전 기능과 동일한 방식으로 버튼의 이름과 기능이 이어질 Class, 위치를 지정합니다.

```
# 'Hydrostatic Curve' button click event
```

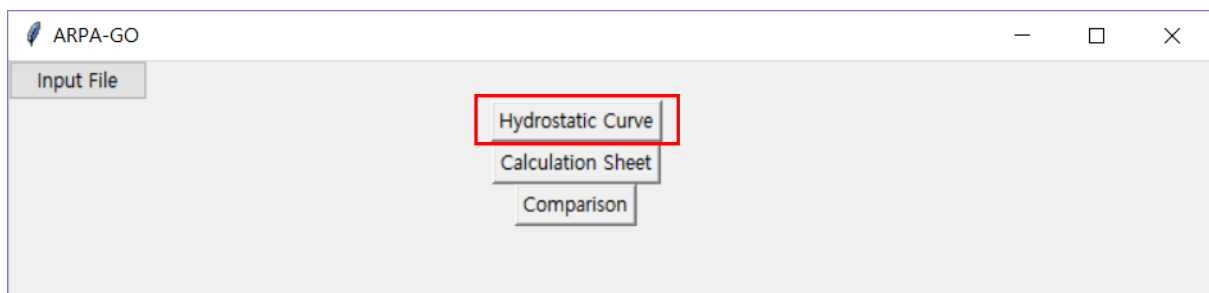
```
def get_window_hydrostaticC(self):
```

```
    fig = Figure()
```

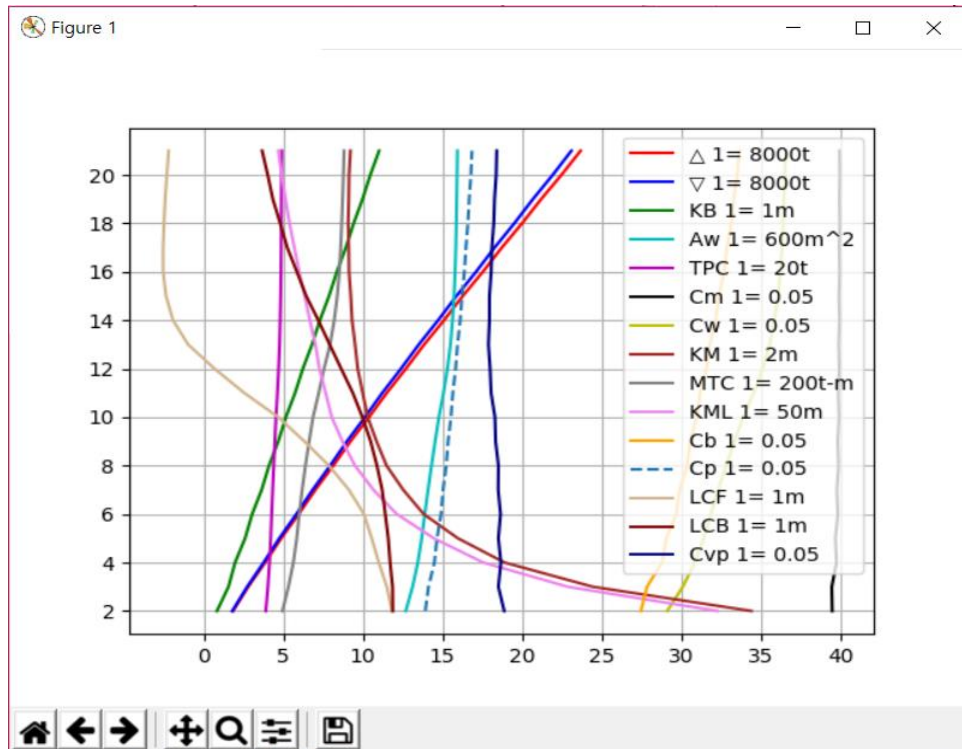
```
    Hydrostatic_curve()
```

그래프를 나타내는 창을 만드는 명령함수는 Figure() 입니다. 이 창에서 나타낼 그래프를 작성합니다. 이 창에서는 Hydrostatic curve 가 나타나며, 저장도 할 수 있습니다.

이 기능에 대한 결과물은 다음과 같습니다.



생성된 [Hydrostatic Curve] 버튼을 클릭합니다.



그 결과, 원하는 선박의 Hydrostatic curve가 보여집니다.

3.2.5 결과 비교하기

먼저, 두 개의 excel파일을 받아 달라진 값이 나타나는 새로운 excel을 만드는 명령어입니다.

```
def excel_diff(path_OLD, path_NEW, index_col):
    df_OLD = pd.read_excel(path_OLD, index_col=index_col).fillna(0)
    df_NEW = pd.read_excel(path_NEW, index_col=index_col).fillna(0)

    # 차이점 나타내기
    dfDiff = df_NEW.copy()
    newRows = []

    cols_OLD = df_OLD.columns
    cols_NEW = df_NEW.columns
    sharedCols = list(set(cols_OLD).intersection(cols_NEW))

    for row in dfDiff.index:
        if (row in df_OLD.index) and (row in df_NEW.index):
            for col in sharedCols:
                value_OLD = df_OLD.loc[row, col]
                value_NEW = df_NEW.loc[row, col]
                if value_OLD == value_NEW:
                    dfDiff.loc[row, col] = df_NEW.loc[row, col]
                else:
                    dfDiff.loc[row, col] = ('{}→{}'.format(value_OLD, value_NEW))
            else:
                newRows.append(row)
```

이때, df_OLD와 df_NEW는 Result1과 Result2에서 받은 excel 파일의 경로입니다.

```
# First, Initialize an object to use global function
df_OLD = 0

# Import 1st Excel file to comparison

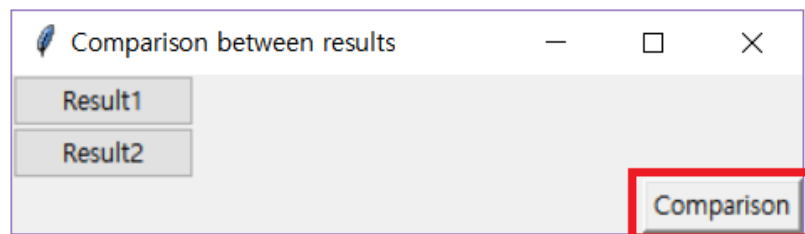
def select_1stxl(self):
    global df_OLD
    filename1 = filedialog.askopenfilename(
        filetypes=[('Excel Spreadsheet', '*.xlsx'), ('Excel Spreadsheet', '*.xls'), ('All files', '*.*')])
    self.filename_var1.set(filename1)
    df_OLD = pd.read_excel(filename1)

# Same reason that df_OLD is defined df_OLD=0
df_NEW = 0

# Import 2nd Excel file to comparison

def select_2ndxl(self):
    global df_NEW
    filename2 = filedialog.askopenfilename(
        filetypes=[('Excel Spreadsheet', '*.xlsx'), ('Excel Spreadsheet', '*.xls'), ('All files', '*.*')])
    self.filename_var2.set(filename2)
    df_NEW = pd.read_excel(filename2)
```

위의 코드에서 df_OLD=0, df_NEW=0을 해준 이유는 global 함수로 사용하기 위해서 먼저 선언해야 하기 때문입니다.



[Result1]과 [Result2]를 사용하여 Excel파일을 받고 [Comparison] 버튼을 누르면 다음 코드가 실행됩니다.

```
# Export excel file that show difference two excel

def load_diffxl(self):
    global df_OLD, df_NEW
    dfdiff = excel_diff1(df_OLD, df_NEW)

    """
    dx : differencexl
    This code use differencexl.py , so certainly import 'differencexl.py' file
    """

    savefile1 = filedialog.asksaveasfilename(filetypes=[("Excel files", "*.xlsx"), ("All files", "*.*")])

    writer = pd.ExcelWriter(savefile1 + '.xlsx', engine='xlsxwriter')

    dfdiff.to_excel(writer, index=True, sheet_name="DIFF")
```

```

# workbook : Create New Excel
workbook = writer.book
worksheet = writer.sheets['DIFF']
worksheet.hide_gridlines(2)
worksheet.set_default_row(15)

# Definition Excel Format
center_fmt = workbook.add_format({'align': 'center'})
number_fmt = workbook.add_format({'align': 'center', 'num_format': '#,##0.00'})
cur_fmt = workbook.add_format({'align': 'center', 'num_format': '$#,##0.00'})
grey_fmt = workbook.add_format({'font_color': '#E0E0E0'})
highlight_fmt = workbook.add_format({'font_color': '#FF0000', 'bg_color': '#B1B3B3'})
new_fmt = workbook.add_format({'font_color': '#32CD32', 'bold': True})

# Highlight different cell
worksheet.conditional_format('A1:ZZ1000', {'type': 'text',
                                             'criteria': 'containing',
                                             'value': '→',
                                             'format': highlight_fmt})

writer.save()
# This code shows that comparison work has been carried out
print('\nDone.\n')

```

Writer.save() : Definition Excel Format과 Highlight different cell에서 지정한 excel 형식을 저장합니다.

Print('\nDone.\n') : Excel 파일이 생성되었다는 것을 사용자가 알 수 있습니다.

```

def excel_diff1(df_OLD, df_NEW):
    # 차이점 나타내기
    dfDiff = df_NEW.copy()
    newRows = []

    cols_OLD = df_OLD.columns
    cols_NEW = df_NEW.columns
    sharedCols = list(set(cols_OLD).intersection(cols_NEW))

    for row in dfDiff.index:
        if (row in df_OLD.index) and (row in df_NEW.index):
            for col in sharedCols:
                value_OLD = df_OLD.loc[row, col]
                value_NEW = df_NEW.loc[row, col]
                if value_OLD == value_NEW:
                    dfDiff.loc[row, col] = df_NEW.loc[row, col]
                else:
                    dfDiff.loc[row, col] = ('{}→{}'.format(value_OLD, value_NEW))
            else:
                newRows.append(row)

```

Excel_diff1은 Excel_diff가 3개의 변수를 사용하기 때문에 load_diff1에서 인식하지 못합니다.

이를 해결하기 위해 변수 2개의 함수 excel_diff1 함수를 만들어줍니다.

4. 부록

이 장에서는 ARPA-GO 에 사용 된 추가기능을 설명합니다.

4.1 추가기능

4.1.1 Lisp

AutoCAD에 내장된 VisualLisp(AutoLisp)은 사용자가 필요한 기능을 스스로 개발하여 사용할 수 있도록 CAD에서 지원하는 언어입니다. 이는 프로그램 언어인 LISP에서 파생되었으며, LISP이 실용적으로 사용되는 대표적인 예입니다. VisualLisp파일은 CAD의 상단 탭 중 '관리 - 프로그램 - Visual Lisp 편집기' 를 이용해 생성(수정)하거나, 메모장에 작성한 코드를 '.LSP' 확장자로 저장하여 파일을 생성(수정)할 수 있습니다.

우리 프로젝트에서는 사용자가 ARPA-GO를 이용해 선박계산을 수행할 시, 작업을 더 수월하게 진행할 수 있도록 좌표추출기능을 제공합니다. 좌표추출기능인 ARPA-GO Lisp은 Visual Lisp 편집기를 이용해 작성하였으며, 사용자의 편의에 따라 수정이 가능합니다.

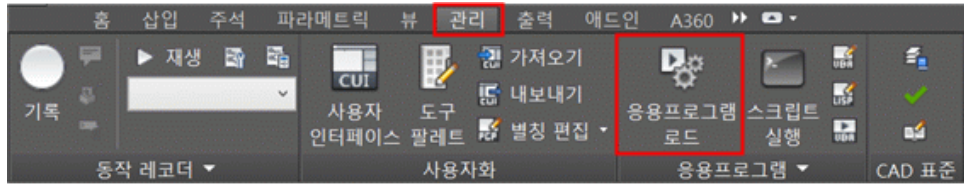
"ARPA-GO Lisp"은 실질적으로 Lisp을 실행시켜주는 메인 함수와 이를 보조하는 여러 개의 서브함수들로 구성 되어있습니다. (서브 함수는 다른 서브 함수로 구성될 수 있습니다.) 사용자의 편의에 맞게 Visual Lisp을 수정하고자 할 시 각 함수 구문의 의미를 알아야만 가능 하지만, Visual Lisp을 공부하기 위한 국내의 자료는 해외에 비해 많이 부족합니다. 따라서, 국내 Visual Lisp 활성화에 조금이나마 도움이 되고자 하는 바람으로 ARPA-GO Lisp의 각 함수 구문들에 대한 상세 설명을 첨부 하였습니다.

"ARPA-GO Lisp" 사용방법

- 1) 선박 계산을 수행하고자 하는 선박의 Body Plan을 활성화합니다.
- 2) 명령 창에 'Appload'를 입력하거나 '관리-응용프로그램-응용프로그램 로드' 를 선택합니다.

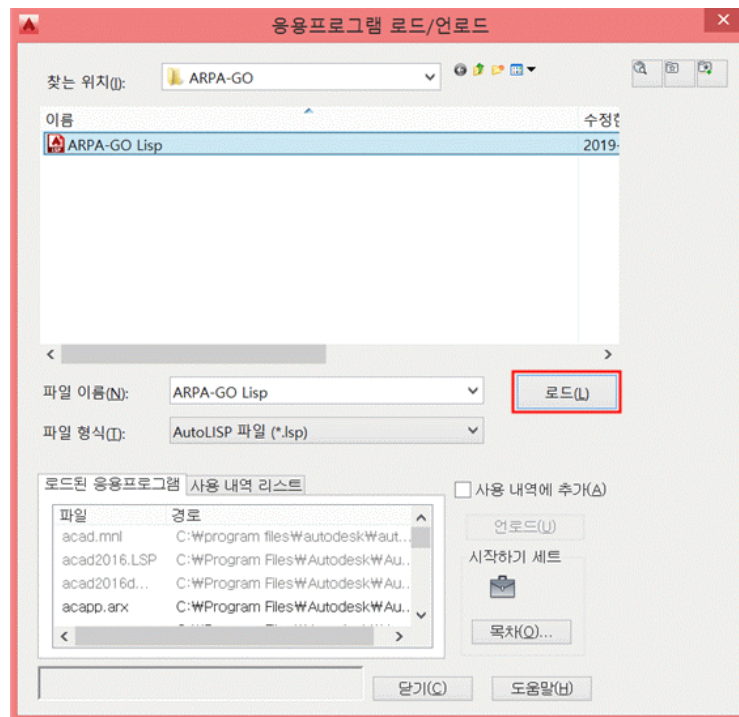


[명령창 이용]



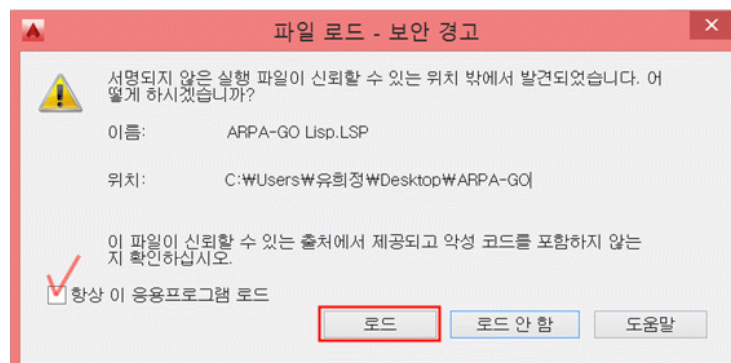
[관리 탭 이용]

- 3) '응용프로그램 로드/언로드' 창이 뜨면 'ARPA-GO Lisp' 파일을 선택하고 로드를 눌러줍니다.

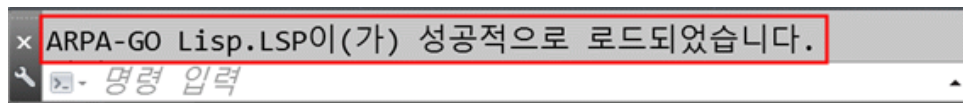


- 4) 보안경고창이 뜨는 경우 '로드' 를 클릭해줍니다.

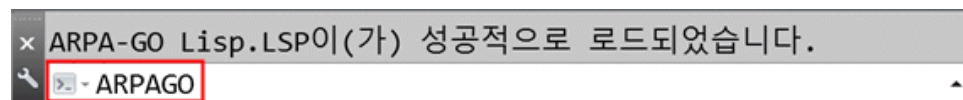
(항상 이 응용프로그램 로드를 체크해도 좋습니다.)



- 5) 닫기를 누른 후 명령 창을 보면, ARPA-GO Lisp이 성공적으로 로드 된 것을 확인 할 수 있습니다.



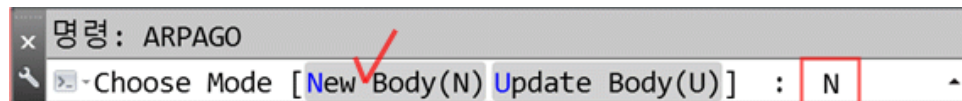
- 6) 명령 창에 ARPA-GO Lisp의 명령어인 'arpago' (혹은, 'ARPAGO') 를 입력합니다.



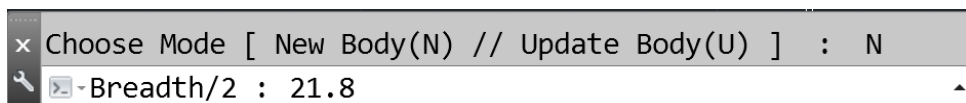
- 7) 두 개의 모드 중 원하는 모드를 고릅니다. 새로운 선형의 좌표를 추출하는 'New Body(N)' 모드와, 선형 수정 후 좌표를 추출하는 'Update Bode(U)' 모드가 있습니다.

- 'New Body' 모드에 대해 알아보겠습니다.

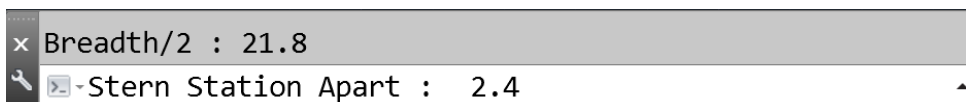
01. 명령 창에 뜨는 'New Body(N)' 를 클릭하거나 'N' 을 입력합니다.



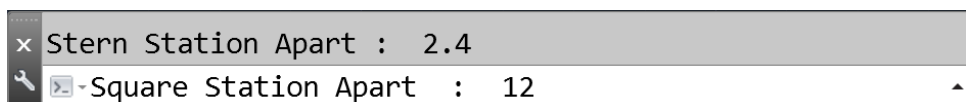
02. 명령 창에 제시되는 순서대로 각각의 값을 입력합니다.



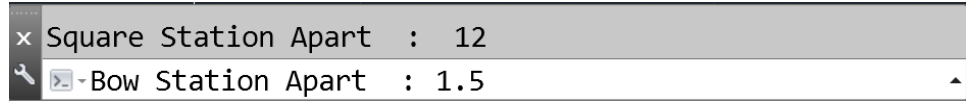
[반폭]



[선미부 STATION 간격]

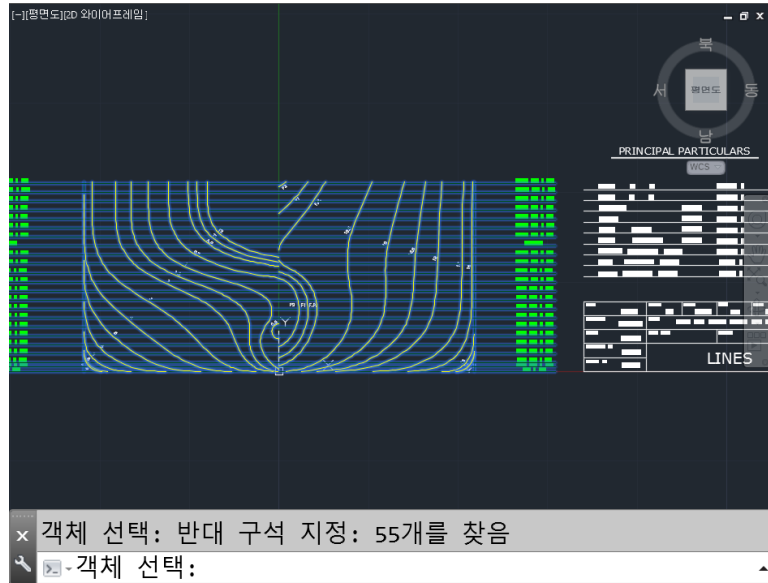


[주요부 STATION 간격]



[선수부 STATION 간격]

03. Body Plan 전체 (Station별 선형 및 워터라인) 를 드래그해 선택합니다. 이때, 텍스트 및 수직선이 포함 되어있어도 상관 없습니다.



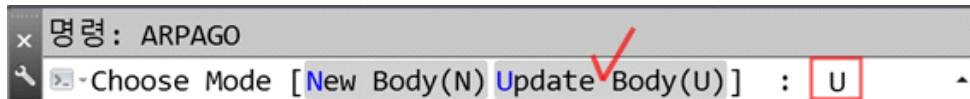
04. 객체를 모두 선택한 후 Enter키를 누르면 알림 창이 뜨며, 추출된 좌표 값을 엑셀을 통해 확인할 수 있습니다.

05. ARPA-GO Lisp을 통해 완성된 offset표는 아래 사진과 같이 생성됩니다.

| | A | B | C | D | E | F | G | H | I |
|----|------------|---------|--------|--------|--------|--------|--------|--------|--------|
| 1 | NO.Station | BOTTOM | 0.5 WL | 1 W.L | 2 W.L | 3 W.L | 4 W.L | 5 W.L | |
| 2 | T2 | -4.800 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 3 | T1 | -2.400 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 4 | A.P. | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 5 | S.T 0.5 | 6.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.615 | 0.644 |
| 6 | S.T 1 | 12.000 | 0.000 | 0.369 | 0.964 | 1.688 | 2.038 | 1.985 | 1.531 |
| 7 | S.T 1.5 | 18.000 | 0.148 | 1.658 | 0.000 | 0.000 | 3.239 | 3.274 | 3.250 |
| 8 | S.T 2 | 24.000 | 0.694 | 2.738 | 0.000 | 4.186 | 4.671 | 5.072 | 5.546 |
| 9 | S.T 3 | 36.000 | 2.533 | 5.392 | 6.367 | 7.671 | 8.690 | 9.659 | 10.702 |
| 10 | S.T 4 | 48.000 | 5.228 | 8.910 | 10.186 | 11.956 | 13.364 | 14.652 | 15.899 |
| 11 | S.T 5 | 60.000 | 8.762 | 12.788 | 14.122 | 15.947 | 17.354 | 18.557 | 19.568 |
| 12 | S.T 6 | 72.000 | 12.632 | 16.506 | 17.704 | 19.248 | 20.283 | 20.920 | 21.337 |
| 13 | S.T 7 | 84.000 | 16.478 | 19.372 | 20.244 | 21.207 | 21.610 | 21.762 | 21.800 |
| 14 | S.T 8 | 96.000 | 19.184 | 20.842 | 21.358 | 21.780 | 21.800 | 21.800 | 21.800 |
| 15 | S.T 9 | 108.000 | 19.800 | 21.118 | 21.529 | 21.800 | 21.800 | 21.800 | 21.800 |
| 16 | S.T 10 | 120.000 | 19.800 | 21.118 | 21.529 | 21.800 | 21.800 | 21.800 | 21.800 |
| 17 | S.T 11 | 132.000 | 19.800 | 21.118 | 21.529 | 21.800 | 21.800 | 21.800 | 21.800 |
| 18 | S.T 12 | 144.000 | 19.766 | 21.107 | 21.495 | 21.783 | 21.800 | 21.800 | 21.800 |
| 19 | S.T 13 | 156.000 | 19.766 | 21.107 | 21.495 | 21.783 | 21.800 | 21.800 | 21.800 |
| 20 | S.T 14 | 168.000 | 19.766 | 21.107 | 21.495 | 21.783 | 21.800 | 21.800 | 21.800 |
| 21 | S.T 15 | 180.000 | 18.210 | 20.349 | 20.903 | 21.466 | 21.696 | 21.791 | 21.800 |
| 22 | S.T 16 | 192.000 | 14.360 | 17.613 | 18.776 | 19.993 | 20.633 | 21.028 | 21.281 |

- 'Update Bode' 모드에 대해 알아보겠습니다.

01. 명령 창에 뜨는 'Update Body(U)' 를 클릭하거나 'U' 를 입력합니다.



02. 다시 객체를 선택해줄 필요 없이 바로 완료 알림 창이 뜨며, 변형된 선형에 대한 엑셀 파일이 생성됩니다.

- 8) 이렇게 생성된 엑셀 파일을 사용자가 원하는 경로와 이름으로 저장함으로써 ARPA-GO Lisp을 통한 좌표 추출이 완료됩니다.

“ARPA-GO Lisp” 내장 함수 설명

구문 설명에 앞서, 사용된 AutoLisp 내장 함수에 대해 설명하고자 합니다. AutoLisp은 괄호로 표현 되기 때문에 왼쪽 괄호와 오른쪽 괄호의 개수가 다르면 오류가 발생합니다. 또한, 하나의 괄호 안에는 하나의 함수만이 존재 해야 하며, 표현 시 영어의 대소문자를 구분하지 않습니다.

| 언어 요소 | 색상 |
|-----------------|------------|
| 내장함수 및 기호 | 파란색 |
| 문자열 | 분홍색 |
| 정수 | 녹색 |
| 실수 | 청록색 |
| 주석 | 회색 바탕의 분홍색 |
| 괄호 | 붉은색 |
| 변수 / 인식 불가능한 요소 | 검은색 |

.Visual Lisp 편집기에서 자동으로 인식 가능한 요소들과 각 요소들이 표현되는 색상은 위의 표와 같으며, 색상 설정은 '도구 - 윈도우 속성 - 문법 색상 (혹은, 현재 구성)' 에서 가능합니다.

| | |
|---|--|
| (- num1 num2) | <i>num1 - num2 의 값을 구합니다.</i> |
| (+ num1 num2) | <i>num1 + num2 의 값을 구합니다.</i> |
| (= num1 num2) | <i>num1 과 num2 가 같은지 검사합니다.</i> |
| (/= num1 num2) | <i>num1 과 num2 가 다른지 검사합니다.</i> |
| (< num1 num2) | <i>num1 < num2 를 만족하는지 검사합니다.</i> |
| (> num1 num2) | <i>num1 > num2 를 만족하는지 검사합니다.</i> |
| (1+ num) | <i>num 에 1 을 더합니다.</i> |
| (Abs num) | <i>A 에 절댓값을 취합니다.</i> |
| (Alert "알림 내용") | <i>대화상자를 통해 경고 메시지 "알림 내용" 을 출력합니다.</i> |
| (And A B) | <i>A 와 B 모두 참이면 T를 반환합니다.</i> |
| (Append list1 list2) | <i>list1 에 list2 를 추가합니다.</i> |
| (Assoc item setlist) | <i>setlist 에서 지정한 항목(item)을 포함하고 있는 요소(list) 를 출력합니다.</i> |
| (Cadr list1) | <i>List1의 두 번째 요소를 구합니다.</i> |
| (Car list1) | <i>List1의 첫 번째 요소를 구합니다.</i> |
| (Cdr list1) | <i>List1의 첫 번째 요소를 제외한 모든 요소를 구합니다.</i> |
| (Cond ((test-A) (A)) ((test-B) (B))) | <i>여러 함수를 인수로 받아들여 각 조건에 맞는 수식을 수행시키는 함수입니다.</i> |
| (Defun Name) | <i>Name 이라는 이름의 함수를 정의한다. (서브 함수 정의)</i> |
| (Defun c:ARPAGO) | <i>ARPAGO 라는 명령어를 가진 함수를 정의한다. (메인 함수 정의)</i> |
| (Entget entity-name) | <i>Entity-name 의 entity list 를 구한다</i> |
| (Equal A B) | <i>A 와 B 가 같은지 검사합니다.</i> |
| (Getreal "명령창 출력") | <i>명령창에 "명령창 출력" 을 띄우며, 사용자로부터 실수 값을 입력 받습니다.</i> |
| (Getkeyword "명령창 출력") | <i>Initget 으로 지정해 놓은 값 중 단일문자만을 사용자에게 입력 받습니다.</i> |
| (Getvar "varname") | <i>AutoCAD 의 시스템 변수를 구합니다.</i> |
| (If (test-A) (then-A) (else-B)) | <i>test-A 가 nil 이 아니면 then-A 를 실행하고 그 외엔 else-B 를 실행합니다.</i> |
| (Initget "받을 문자열") | <i>Get** 함수에 의해 사용되는 각종 옵션을 설정합니다.</i> |
| (Itoa num) | <i>정수(num) 를 문자로 변환합니다.</i> |
| (Lambda (A) (+3 (/ A 2))) | <i>무명 함수를 정의하며, Defun 과 유사합니다. [보기의 A 는 A/2+3 정의]</i> |
| (Last list1) | <i>List1의 마지막 요소를 응답합니다.</i> |

| | |
|--|--|
| (Length list1) | List1의 요소 개수를 정수로 응답합니다. |
| (List A B C) | A B C 요소들을 묶어 하나의 리스트로 응답합니다. |
| (Mapcar func list1) | List1의 개별 요소에 function 을 실행합니다. |
| (Member A list1) | List1내에서 A 를 찾아 존재하면 T 를, 아니면 nil 을 반환합니다. |
| (Not A) | A 의 결과가 참인지 nil 인지 검사합니다. |
| (Nth num list1) | List1의 요소 중 num 번째에 위치하고 있는 요소를 선택합니다. |
| (Or A B C) | A B C 요소 중 어느 하나라도 nil 이 아니면 T 를 반환합니다. |
| (Princ A File) | A 를 File (화면 또는 파일) 에 출력합니다. |
| (Progn (A) (B) (C)) | 여러 개의 실행문 및 수식을 하나의 문장으로 묶어 처리합니다. |
| (Repeat num A) | A 를 num 만큼 반복합니다. |
| (Setq A B) | B 를 A 에 저장합니다. |
| (Setvar "varname" A) | AutoCAD 의 시스템 변수를 A 로 설정합니다. |
| (SSget) | 하나 이상의 도면요소를 선택합니다. |
| (SSlength selection-set) | selection-set 에 있는 도면요소의 개수를 구합니다. |
| (SSname selection-set num) | selection-set 에서 num 번째의 도면 요소의 이름을 구합니다. |
| (Strcat A B C) | 나열된 A B C 를 연결하여 응답합니다. |
| (Subst A 'B list1) | List1에 존재하는 B 를 A 로 교환합니다. |
| (Terpri) | 화면에 새로운 줄을 출력합니다. |
| (VI-load-com) | Visual LISP 을 AutoLISP 에 확장하여 로드 합니다. |
| (VI-registry-read "경로") | 주어진 레지스트리 키나 디폴트 값 혹은 그곳에 속한 변수의 값을 읽어옵니다. |
| (VI-remove A list1) | A 가 list1내에 있으면 모두 제거합니다. |
| (VI-sort list1 '<) | 리스트 내의 요소들을 오름차순 정렬합니다. |
| (VI-sort list1 '>) | 리스트 내의 요소들을 내림차순 정렬합니다. |
| (Vla-get-angle object) | object(객체) 의 각도를 구합니다. |
| (Vla-curve-getpointatparam object 0.5) | object 가 곡선일 때 parameter 의 곡선 원 중심까지의 vevtor 를 구합니다. |
| (Vla-curve-getstartpoint object) | object 의 시작점을 구합니다. |
| (Vla-ename->vla-object name) | name(도면 요소의 이름)을 객체로 바꿔줍니다. |

| | |
|---|--|
| (Vlax-get-or-create-object "--.App") | 실행중인 --어플리케이션 개체를 반환하거나, 새 개체를 생성합니다. |
| (Vlax-get-property A 'B) | VLA 개체 A 의 B 속성을 얻어옵니다. |
| (Vlax-import-type-library :tlb-filename FileName :methods-prefix "매서드 접두사" :properties-prefix "속성 접두사" :constants-prefix "상수 접두사") | 라이브러리 유형에서 정보를 가져옵니다. filename 은 라이브러리 이름이며, 파일유형은 EXE TLB OLB TLB DLL 가 있습니다. 파일 유형에 따라 매서드 접두사를 지정해줍니다. (엑셀의 경우 msxl-) 파일 유형에 따라 속성 접두사를 지정해줍니다. (엑셀의 경우 msxl-) 파일 유형에 따라 상수 접두사를 지정해줍니다. (엑셀의 경우 msxl-) |
| (Vlax-invoke object1 'intersectwith object2 Extend-Option) | object1과 object2의 교차점을 구하며, Extend-Option 이 나타내는 각 경우는 [0=연장 X, 1=선택 객체 연장, 2=다른 객체 연장, 3=둘 다 연장] 입니다. |
| (Vlax-invoke-method A 'B) | 개체 A 의 B 매소드를 호출하는 함수입니다. |
| (Vlax-put-property A 'B C) | 개체 A 의 B 속성을 C 로 설정합니다. |
| (Vlax-variant-value var) | 특정 값(var)을 객체로 변형합니다. |
| (Zerop num) | num 이 0인지 검사합니다. |

“ARPA-GO Lisp” 구문 설명

- 메인함수

```
(defun c:ARPAGO (/ oldsnap)
```

명령어가 ARPA-GO인 함수를 정의합니다. Defun 함수 뒤에 “c: ****” 오는 경우 ****이 명령어입니다.

지역변수 oldsnap 을 선언해줍니다. 지역변수를 선언 해주기 위한 괄호와 명령어 사이는 반드시 띄워 주어야 합니다.

```
(setq oldsnap(getvar "osmode"))  
(setvar "osmode" 0)
```

현재 설정되어 있는 “오스냅 모드” 를 변수 oldsnap에 저장한 후, 모든 설정을 끕니다.

```
(vl-load-com)
```

Visual Lisp 확장자를 불러옵니다.

```
(initget "N U")  
(setq mode(getkeyword  
"Choose Mode [ New Body(N) // Update Body(U) ] : "))(terpri)
```

새로운 선형의 좌표를 추출하는 ‘New Body’ 모드는 “N”으로, 선형 수정 후 좌표를 추출하는 Update Body’ 모드는 “U”로 설정하여 사용자에게 문자를 입력 받아 변수 mode로 저장합니다.

```
(if (= mode nil)(setq mode "N"))
```

조건 mode = nil 이 참일 경우, 변수 mode에 “N” 을 저장합니다.

```
(if (= mode "N")(modeNB)(modeUB))
```

조건 mode = “N” 이 참일 경우, 서브함수 modeNB를 수행하고, 거짓일 경우 서브함수 modeUB를 수행합니다.

```
(alert "Excel을 확인해주세요 !")
```

```
(setvar "osmode" oldsnap)  
(princ)
```

```
);_메인함수 ARPAGO 종료
```

“Excel을 확인해주세요 !” 라는 내용의 경고 알림 창을 띄우고, 오스냅모드를 oldsnap으로 설정합니다. 함수의 마지막에 (princ) 를 적지 않으면, 실행창에 nil이 표기됩니다.

- 서브함수

1. modeNB

```
(defun modeNB (/ wlspl num ni m objname jpoint Astlist Fstlist Alaylist
```

서브함수 modeNB 를 선언합니다. 이 함수는 새로운 선형일 때 좌표를 추출하기 위해 수행할 서브함수들을 순서대로 모아 놓은 큰 덩어리로 볼 수 있습니다.

하나의 CAD 파일 안에 여러 개의 Body Plan을 놓고 비교해 보려면 New body 모드를 여러 번 사용해야 합니다. 이때, 데이터의 누적을 피하기 위해 서브함수들에서 사용된 모든 변수를 지역 변수로 선언해주었습니다. (서브함수 setWL 예외)

```
(setq halfB(getreal "Breadth/2 : "))(terpri)
```

배의 반폭을 입력 받아 변수 halfB 에 저장합니다.

```
(STapart)
```

서브함수 STapart 를 실행합니다.

```
(setq wlspl(ssget '((0 . "LINE,SPLINE,ARC"))))
```

사용자가 선택한 여러 도면 요소 중 LINE, SPLINE, ARC 만 골라내어 변수 wlspl 에 저장합니다.

```
(setWL)
(setST)
(offsetNB)
(Xposition)
(intersect)
```

```
);_서브함수 modeNB 종료
```

서브함수 setWL, setST, offsetNB, Xposition, intersect 를 수행 한 후, modeNB 함수가 종료됩니다.

2. modeUB

```
(defun modeUB()
```

```
(offsetUB)
(Xposition)
(intersect)
```

```
);_서브함수 modeUB 종료
```

서브함수 modNB 는 기존 선형을 수정했을 경우에 수행할 서브함수들을 순서대로 모은 큰 덩어리로 볼 수 있으며, 이미 modNB 를 통해 필요한 변수들을 받은 상태이므로 서브함수 offsetUB, Xposition, intersect 만을 실행합니다.

3. STapart

```
(defun STapart()  
  (setq stern(getreal "Stern Station Apart : "))(terpri)  
  (setq square(getreal "Square Station Apart : "))(terpri)  
  (setq Bow(getreal "Bow Station Apart : "))(terpri)  
);_서브함수 STapart 종료
```

서브함수 STapart 는 선체의 각 부분별 station 의 간격을 받는 함수입니다.

사용자로부터 입력 받은 선미부의 station 간격을 변수 stern 에 저장하고, 주요부의 station 간격은 변수 square 에, 선수부의 station 간격은 변수 Bow 에 저장합니다.

4. setWL

```
(defun setWL())
```

서브함수 setWL 을 정의합니다. setWL 은 도면 요소 중 LINE, SPLINE, ARC 만 받아 저장해 놓은 wlspl 리스트에서 WaterLine 을 판별해 변수 wllist 에 리스트 형태로 저장합니다. 또한, waterline 개수에 따른 offset 표의 형식을 나타내기 위한 리스트인 wlnumlist 를 구합니다.

```
(setq wllist(list ) wlnumlist(list ))
```

Wllist 와 wlnumlist 를 빈 리스트로 선언해줍니다. 이 구문으로 시작하는 이유는 modeNB 에서 지역변수를 선언해 준 이유와 같습니다.

Wllist 는 waterline별 도면 이름 리스트 이고, wlnumlist 는 waterline 개수에 따른 offset표의 형식을 나타내기 위한 리스트 입니다.

```
(setq num 0)  
(repeat (sslength wlspl)
```

변수 num에 0을 저장하고 wlspl 리스트의 세트 길이 만큼 아래 구문들을 반복시킵니다.


```

(setq name(ssname wlspl num))
(if (and (= "LINE" (cdr(assoc 0 (entget name))))
    (or (zerop (vla-get-angle (vlax-ename->vla-object name)))
        (equal (vla-get-angle (vlax-ename->vla-object name)) pi 0.0001))))
    (setq wllist(append wllist (list name)))
    (setq num(+ num 1)))

```

Wlspl 리스트 중 num 번째에 해당하는 세트의 도면 이름을 변수 name 에 저장합니다.

waterline 판별을 위한 if문의 조건은 ‘도면 요소가 LINE이고, 각도가 0도 혹은 180도 일 때’ 즉, ‘수평선 일 때’ 입니다. Name 에 저장된 도면 이름을 객체로 바꿔 각도를 구해주었습니다.

조건이 참일 경우, 객체의 도면 이름을 wllist 리스트에 추가하며 if문이 종료되고, 변수 num 에 num+1값을 저장한 후 repeat 문이 종료됩니다. 이 구문들을 wlspl 리스트의 세트 길이 만큼 반복합니다.

```

(setq wllist(vl-sort wllist '(lambda (x y)(< (cadr (cdr(assoc 10 (entget x))))
                                             (cadr (cdr(assoc 10 (entget y)))))))

```

최종적으로 얻은 waterline별 도면 요소 이름 리스트인 wllist 를 시작점 좌표(y)를 기준하여 오름차순으로 정렬해줍니다.

```

(setq wllistn(length wllist))

```

Wlnumlist 를 구하기 위해, wllist 의 길이를 변수 wllistn 에 저장합니다.

```

(setq wlnum 1)
(repeat (- wllistn 2)
  (setq wlnumlist(append wlnumlist(list wlnum)))
  (setq wlnum(+ wlnum 1)))
);_서브함수 setWL 종료

```

변수 wlnum 에 1을 저장한 후 wllistn-2 만큼 다음 구문을 반복합니다.

변수 wlnum 을 wlnumlist 에 추가하고 이것을 변수 wlnumlist 라고 저장합니다. 변수 wlnum 에 wlnum+1 값을 저장합니다.

5. separateST

```

(defun separateST()

```

서브함수 separateST 를 정의합니다. 사용자가 modeUB (선형 변경시 offset표 구하는 경우)를 사용 수 있게 하려면, 모든 station 이 각각 다른 변수에 저장되어 있어야 합니다. separateST 함수는 station을 선미부, 선수부로 나누어 리스트 형태로 저장하는 함수입니다.

```
(setq Astlist(list ) Fstlist(list ) Alaylist(list ) Flaylist(list ))
```

앞의 setWL 함수와 마찬가지로 리스트를 비우고 시작합니다.

Astlist : 선미부 station의 도면 이름

Fstlist : 선수부 station의 도면 이름

Alaylist : 선미부 station의 레이어 이름

Flaylist : 선수부 station의 레이어 이름

```
(setq num 0)
(repeat (sslength wlspl)
```

변수 num에 0을 저장하고, wlspl 세트 리스트의 길이만큼 다음 구문들을 반복합니다.

```
(setq name(ssname wlspl num))
(setq layname(cdr (assoc 8 (entget name))))
(setq objname(vlax-ename->vla-object name))
```

wlspl 리스트 중 num번째에 해당하는 세트의 도면 이름을 변수 name 에 저장합니다. 도면 이름을 통해 얻은 레이어 이름을 변수 layname 에 저장한 후 도면 이름을 객체로 바꾸어 변수 objname 에 저장합니다.

```
(if (= "SPLINE" (cdr(assoc 0 (entget name))))
  (progn
    (setq jpoint(car(vlax-curve-getpointatparam objname 0.5)))
    (if (< jpoint 0)
      (progn
        (setq Astlist(append Astlist(list name)))
        (if (= (member layname Alaylist) nil)
          (setq Alaylist(append Alaylist(list layname)))))
      (progn
        (setq Fstlist(append Fstlist(list name)))
        (if (= (member layname Flaylist) nil)
          (setq Flaylist(append Flaylist(list layname)))))))
```

Body Plan에서 Station 은 Spline 또는 ARC로 그려집니다. 먼저, 요소의 종류 = SPLINE 일 때입니다.

요소의 종류 조건이 참일 경우, 실행문이 여러 개 이므로 progn 을 통해 하나의 덩어리로 묶어 주며 시작합니다.

곡선의 중간점 좌표 중 x좌표를 변수 jpoint 에 저장합니다. 이때, jpoint<0 가 참일 경우 (선미부 station임을 알 수 있으므로, 도면 이름을 Astlist 리스트에 저장합니다. 그리고, 레이어 이름이 Alaylist 에 없으면 레이어 이름을 Alaylist 에 추가하며 jpoint<0 가 참일 경우의 실행문을 묶은 progn을 닫습니다.) jpoint<0 이 거짓일 경우, (과정은 참일 경우와 같으며, 선수부 station 이기 때문에 Astlist 와 Alaylist 대신에, Fstlist 와 Flaylist 에 저장합니다.)

```

(progn
  (if (= "ARC" (cdr(assoc 0 (entget name))))
    (progn
      (setq jpoint(car(vlax-curve-getstartpoint objname)))
      (if (< jpoint 0)
        (progn
          (setq Astlist(append Astlist(list name)))
          (if (= (member layname Astlist) nil)
            (setq Alaylist(append Alaylist(list layname))))))))))

```

요소의 종류 조건이 거짓일 경우 또한, 실행문이 여러 개 이므로 progn 을 통해 하나의 덩어리로 묶어주며 시작합니다.

요소 종류 = ARC 인지 한 번 더 조건을 줍니다. 시작점의 x좌표를 jpoint 에 저장합니다. 이하는 SPLINE인 경우와 동일하게 jpoint<0 만족 여부에 따라 Astlist, Fstlist 에 저장합니다.

```

(setq num(+ num 1))

```

Repeat 문 시작 전 0을 저장한 변수 num에 num+1 값을 저장한 후 repeat 문을 닫습니다. 여기까지의 구문들을 wlspl 세트 리스트 길이만큼 반복하여 사용자가 선택한 모든 요소에 대한 조건의 만족 여부를 통해 모든 station 을 선수부와 선미부로 구별합니다.

각 station별 레이어 이름은 Transom부터 순서대로 ST T2, ST T1, ST A.P., ST 0.5, ST 1, ST 1.5, ST 2, ST 3, ST 4, ST 5, ST 6, ST 7, ST 8, ST 9, ST 10, ST 11, ST 12, ST 13, ST 14, ST 15, ST 16, ST 17, ST 18, ST 18.5, ST 19, ST 19.5, ST F.P., ST F1, ST F2 를 기준으로 하였습니다. 엑셀에 표기 되는 station 번호 또한, 이를 기준으로 하였으므로 가급적 같은 이름을 사용하는 것을 추천합니다.

```

(setq Alaylist(vl-sort Alaylist '<))

```

Transom부터 순서대로 좌표를 추출하기 위해 선미부 station의 레이어 이름 리스트인 Alaylist 를 오름차순으로 정렬합니다.

```

(setq Alaylistn(length Alaylist))

```

ST T1, ST T2, ST A.P. 는 문자이므로 뒤에 정렬됩니다. 이를 바로잡기 위해 우선 Alaylist 의 길이를 변수 Alaylistn 에 저장합니다.

```

(setq Alastlist(append Alastlist (list (nth (- Alaylistn 1) Alaylist)
                                       (nth (- Alaylistn 2) Alaylist)
                                       (nth (- Alaylistn 3) Alaylist))))

```

Alaylist 리스트의 (Alaylistn-1), (Alaylistn-2), (Alaylistn-3) 번 째 요소들을 변수 Alastlist 에 추가합니다.

```
(setq Alaylist(vl-remove (nth 0 Alastlist) Alaylist))
(setq Alaylist(vl-remove (nth 1 Alastlist) Alaylist))
(setq Alaylist(vl-remove (nth 2 Alastlist) Alaylist))
```

Alaylist 에서 Alastlist의 0,1,2 번째 요소를 지워줍니다. Alaylist에는 0.5~10 station까지 남아있는 상태입니다.

```
(setq Alaylist(append Alastlist Alaylist))
```

Alastlist에 Alaylist를 추가해주면, Alastlist 요소들 뒤에 Alaylist 가 표시됩니다. 이렇게 Transom부터 순서대로 정렬 되도록 만든 리스트를 다시 변수 Alaylist로 저장해줍니다.

```
(setq Playlist(vl-sort Playlist '>))
(setq Playlistn(length Playlist))
```

```
);_서브함수 separateST 종료
```

Playlist는 선수부 station 이므로 ST 10 이 리스트의 맨 처음에, ST F2 가 리스트의 가장 마지막에 위치해야 합니다. 따라서, 내림차순으로 정렬하고, 이후 station별로 다른 변수에 저장하기 위해 Playlist의 길이를 변수 Playlistn에 기억시킵니다.

6. setAST

```
(defun setAST())
```

서브함수 setAST를 정의합니다. Alaylist와 Astlist를 이용해 선미부 station들을 각각 다른 변수로 분리해주기 위한 함수입니다. 선형이 같으면 레이어 또한 묶어서 표현하기 때문에 레이어 이름 리스트인 Alaylist의 길이를 기준으로 선미부 station을 분리하였으며 레이어 이름이 10개인 경우부터 15개인 경우까지 선언해 주었습니다. 선미부 station을 저장해놓은 변수는 Transome부터 10 station까지 순서대로 sta, stb, stc, std, ste, stf, stg, sth, sti, stj, stk, stl, stm, stn, sto입니다. 이 서브함수는 같은 구문을 여러 번 반복시키는 것이므로 일부만 발췌하여 설명하였습니다.

```
(if (= Alaylistn 15)
  (progn
    (setq nn 0)
    (repeat (length Astlist)
```

Alaylistn = 15인 경우, 즉 레이어 이름이 Transom부터 ST 10까지 모두 존재할 경우입니다. 조건이 참일 경우 실행문이 여러 개이므로 progn으로 시작합니다. 변수 nn에 0을 저장하고, Astlist(선미부 도면 이름 개수) 만큼 아래 구문을 반복합니다.

```
(cond
  ((= (cdr(assoc 8 (entget(nth nn Astlist)))) (nth 0 Alaylist))
    (setq sta (append sta (list (nth nn Astlist)))))
```

cond는 조건문이지만, if처럼 조건이 거짓일 경우에 대한 실행문이 따로 없으며 다음 조건으로 넘어갑니다. 마지막 조건이 거짓일 경우, cond문을 종료합니다. 여기서는 Astlist에 저장된 n 번째 station 요소의 레이어 이름을 구해 Alaylist 리스트 중 n번째 레이어 이름과의 일치 여부를 판별해 station에 따라 변수에 저장합니다.

‘Astlist의 nn번째 요소의 레이어 이름 = Alaylist의 0번째 요소’가 참일 경우 Astlist의 nn번째 요소(도면 이름)을 stalist에 추가합니다. 조건이 거짓일 경우, 다음 조건으로 넘어갑니다. Alaylist의 길이가 15인 경우 이므로, 조건은 Alaylist의 14번째 요소까지 선언해주고 cond문을 닫습니다.

```
(setq nn(+ nn 1))))
```

변수 nn에 nn+1 값을 저장한 후 repeat, progn, if문 모두를 닫습니다.

```
(setq #sta sta #stb stb #stc stc #std std #ste ste #stf stf #stg stg
      #sth sth #sti sti #stj stj #stk stk #stl stl #stm stm #stn stn #sto sto)
);_서브함수 setAST 종료
```

각 station이 저장된 변수인 sta, stb... 은 교차점을 찾기 위해 저장된 값을 유지하여야 하지만, 리스트에 append로 요소를 추가 받는 형태이므로 지역변수로 선언해주어야 함수의 재사용 시 리스트 요소가 누적되지 않습니다. 따라서, 다른 변수에 각각의 값들을 저장하여 sta등을 지역변수로 선언하여도 교차점을 구할 수 있도록 하였습니다.

7. setFST

```
(defun setFST()
  (if (= Flaylistn 14)
```

서브함수 setFST 를 정의합니다. Flaylist 와 Fstlist 를 이용해 선수부 station들을 각각 다른 변수로 분리해주기 위한 함수이며, 과정은 setAST 와 같습니다. ST 10 은 선미부에서 구분해주었으므로 레이어 이름이 10개인 경우부터 14개인 경우까지 선언해주었습니다. 선수부 station을 저장해 놓은 변수는 ST F2부터 ST 9까지 순서대로 staa, stbb, stcc, stdd, stee, stff, stgg, sthh, stii, stj, stkk, stll, stmm, stnn입니다.

```
(setq #staa staa #stbb stbb #stcc stcc #stdd stdd
      #stee stee #stff stff #stgg stgg
      #sthh sthh #stii stii #stjj stj #stkk stkk
      #stll stll #stmm stmm #stnn stnn)
);_서브함수 setAST 종료
```

setAST 와 마찬가지로 append로 요소를 추가 받는 형태인 변수들을 지역 변수로 선언하기 위해, 새로운 변수들로 저장하였습니다.

8. intersect

```
(defun intersect()
```

서브함수 intersect를 정의합니다. 오름차순으로 정렬해놓은 waterline 리스트와 station의 교차점을 찾아 엑셀로 좌표를 추출하는 함수입니다. Body Plan을 보면, 한 개의 객체로 그려진 station이 있고 두 개의 객체로 그려진 station이 존재하므로 이에 따라 다른 서브 함수를 실행시켜주었습니다. 각 station에 대한 구문이 동일해, Transom 부분만 발췌하여 설명하였습니다.

```
(if (= (length #sta) 1)
  (progn
    (setq obj2(vlax-ename->vla-object (nth 0 #sta)))
    (caseone)
    (setq mm 2)
    (inputExcel))
  (progn
    (setq obj3(vlax-ename->vla-object (nth 0 #sta)))
    (setq obj4(vlax-ename->vla-object (nth 1 #sta)))
    (casetwo)
    (setq mm 2)
    (inputExcel))))
```

조건 'Transom을 이루는 객체가 1개일 때'가 참인 경우, #sta는 리스트 형태이므로 0번째에 위치하는 도면 이름을 받아 변수 obj2에 객체로 바꾸어 저장합니다. 서브함수 caseone 을 실행해 waterline 리스트와의 교차점을 찾습니다. offset표에서 Transom은 2번째 줄에 위치 하므로 변수 mm에 2를 저장하고, 서브함수 inputExcel 을 실행하여 엑셀에 값을 입력시킵니다.

```
(if (= (length #staa) 1)
  (progn
    (setq obj2(vlax-ename->vla-object (nth 0 #staa)))
    (caseone)
    (setq mm 30)
    (inputExcel))
  (progn
    (setq obj3(vlax-ename->vla-object (nth 0 #staa)))
    (setq obj4(vlax-ename->vla-object (nth 1 #staa)))
    (casetwo)
    (setq mm 30)
    (inputExcel))))
);_서브함수 intersect 종료
```

모든 station에 대해 이와 같은 구문들을 수행하면, 서브함수 intersect 가 종료됩니다.

9. caseone

```
(defun caseone()  
  (setq blist(list ))
```

서브함수 caseone 을 정의합니다. intersect 함수에서 station이 객체 1개로 그려졌을 경우, 실질적으로 waterline 리스트와 station의 교차점을 찾는 함수입니다. blist는 이 함수를 통해 구할 교차점의 x좌표 리스트이며, intersect 함수 내에서 caseone 이 여러 번 사용되므로 데이터가 누적되지 않도록 리스트를 비워주고 시작합니다.

```
(setq ni 0 m 0)  
(repeat wllistn
```

초기값으로 변수 ni와 m에 각각 0을 저장해주고 wllistn (워터라인 리스트 길이) 만큼 다음 구문들을 반복합니다.

```
(setq w1(nth ni wllist))  
(setq obj1(vlax-ename->vla-object w1))  
(setq interp(vlax-invoke obj1 'intersectwith obj2 0))  
(setq selectx(car interp))
```

wllist (waterline의 도면이름 리스트) 의 ni 번째를 변수 w1 에 저장하고, 도면이름인 w1 을 객체로 바꾸어 변수 obj1로 저장합니다. obj1과 intersect 함수에서 지정한 obj2의 교차점을 구해 변수 interp 로 저장하고 그 중 첫 번째 요소인 x좌표만을 변수 select 로 저장합니다.

```
(if (/= selectx nil)  
  (progn  
    (setq selectx(abs selectx))  
    (setq blist(append blist(list selectx))))  
    (setq blist(append blist(list m))))  
(setq ni(+ ni 1)))
```

조건 'selectx 가 nil 이 아닐 때' 가 참일 경우, station과 해당 홀수의 교차점이 존재한다는 뜻이며 x좌표에 절댓값을 씌워 blist 에 추가합니다.

조건이 거짓일 경우, 교차점은 존재하지 않으므로 0을 blist 에 추가합니다. 이 작업을 하지 않을 시, 엑셀 offset표에 0이 아닌 nil이 표시됩니다.

변수 ni 에 ni+1 값을 저장하고 repeat문을 종료합니다. wllistn 만큼 이 구문들을 반복하면, station과 waterline 리스트와의 교차점 리스트인 blist 가 구해집니다.

```
(if (= (last blist) 0)  
  (setq blist(subst halfB '0 blist)))  
);_서브함수 caseone 종료
```

하지만, 대표적으로 ST 10 과 같은 경우 해당 홀수에서 선박의 반폭 길이와 같은 값을 갖지만,

도면상 waterline과 station의 교차점은 존재하지 않아 nil값이 나오게 됩니다. 이를 해결 하기 위해 어느 위치의 station이든 모두 마지막 waterline에서의 값은 존재한다는 것을 이용해 위와 같은 조건문을 적용시켰습니다.

조건 'blist의 마지막 요소가 0 일 때' 가 참이라면, blist에 존재하는 0을 halfB(초반에 사용자에게 입력 받은 반폭)로 대체 합니다. 이로써 해당 station에 대한 blist가 최종적으로 결정됩니다.

10. casetwo

```
(defun casetwo()  
  (setq clist(list ) dlist(list ) blist(list ))
```

서브함수 casetwo 를 정의합니다. intersect 함수에서 station이 객체 2개로 그려졌을 경우,실질적으로 waterline 리스트와 station의 교차점을 찾는 함수입니다. clist는 station을 이루는 첫 번째 요소와 waterline 리스트의 교차점을 구할 리스트이고, dlist는 두 번째 요소와의 교차점을 구할 리스트입니다. blist는 이 둘을 이용해 최종적으로 구할 해당 station의 교차점 리스트입니다. caseone과 마찬가지로 리스트들을 비우고 시작합니다. 교차점을 구해 clist, dlist에 저장하는 과정은 caseone과 같으므로 생략하겠습니다.

```
(setq blist (mapcar '(lambda (x y) (+ x y)) clist dlist))  
);_서브함수 casetwo 종료
```

clist와 dlist의 각각의 요소를 서로 더해 최종적으로 blist를 구해줍니다.

11. LoadExcel

```
(defun LoadExcel ()
```

서브함수 LoadExcel을 정의합니다. 엑셀에 offset 값을 보내기 위해 AutoLisp과 Excel을 연결하는 함수 입니다.

```
(if (and (setq excelPath (vl-registry-read  
                          "HKEY_LOCAL_MACHINE\\SOFTWARE\\Microsoft\\Windows\\Cu  
                          "Path"))  
      (setq excelPath (strcat excelPath "Excel.exe")))
```

엑셀의 실행파일 경로를 excelPath에 저장합니다.


```
(progn
  (if (not msxl-acos)
    (vlax-import-type-library
      :tlb-filename excelPath
      :methods-prefix "msxl-"
      :properties-prefix "msxl-"
      :constants-prefix "msxl-"))
    (setq ExcelApp (vlax-get-or-create-object "Excel.Application"))))
```

excelPath에서 매서드, 속성, 정보를 가져와 엑셀 어플리케이션 개체를 반환하거나 생성하여 변수 ExcelApp에 저장합니다.

ExcelApp

);LoadExcel 종료

ExcelApp을 실행 시키고, 함수LoadExcel을 종료합니다.

12. offsetNB

```
(defun offsetNB())
```

서브함수 offsetNB 를 정의합니다. 사용자가 modeNB를 선택하였을 때, 엑셀에 나타낼 offset표의 형식을 지정하기 위한 함수이며, offset 값의 표시 형식, waterline number의 표시 형식, station number와 station의 실제 x좌표의 표시 형식으로 나눌 수 있습니다.

```
(setq ExcelApp (LoadExcel))
(vlax-put Excelapp "visible" :vlax-false)
(setq Workbooks (vlax-get-property ExcelApp 'Workbooks))
(setq CurrentWBook (vlax-invoke-method Workbooks 'Add))
(setq Sheets (vlax-get-property ExcelApp 'Sheets))
(setq AcSheet (vlax-get-property ExcelApp 'ActiveSheet))
(setq Cells (vlax-get-property AcSheet 'Cells))
```

서브함수 LoadExcel은 엑셀과 연결한 후 이를 실행시키는 함수였습니다. 이 과정을 변수 ExcelApp에 저장합니다. vlax-true일 경우, 엑셀을 화면에 실행시킵니다. 이는 사용자에게 불편함을 줄 수 있으므로 false값을 주었습니다. 열려있는 엑셀파일을 변수 workbooks에 저장하고, 교차점을 구해 엑셀에 데이터를 기록할 것이므로 새로운 엑셀파일을 생성하여 변수 CurrentWBook에 저장합니다. 현재 시트들은 변수 Sheets에 저장하고, 현재 활성화된 시트를 변수 AcSheet에 저장합니다. 활성화된 시트의 셀들을 변수 Cells에 저장합니다.

```
(setq Ran (vlax-get-property Excelapp 'Range "B2:Z100"))
(vlax-put-property Ran 'Numberformat "0.000")
```

offset 값의 표시 형식은 소수점 넷째 자리에서 반올림 하므로 숫자의 형식을 지정해주었습니다. B2 이후 셀들의 property를 변수 Ran에 저장합니다. 숫자형식을 지정해주는 Numberformat을

이용해 소수점 셋째 자리까지로 형식을 지정합니다.

```
(setq colY 1 rowX 3)
(setq val "BOTTOM")
(vlax-put-property Cells 'Item colY rowX val)

(setq colY 1 rowX 4)
(setq val "0.5 WL")
(vlax-put-property Cells 'Item colY rowX val)

(setq colY 1 rowX 5)
(setq wlnumlist(mapcar '(lambda(x)(strcat (itoa x) " W.L"))wlnumlist))
(mapcar
  '(lambda (val)
    (vlax-put-property Cells 'Item colY rowX val)
    (setq rowX(1+ rowX)))wlnumlist)
```

waterline number의 표시 형식입니다. 먼저, 1행 3열인 C1셀에 BOTTOM부터 시작해 가로로 water line의 이름들을 입력해줍니다. 서브함수 setWL에서 생성한 wlnumlist 요소들을 문자로 바꾸고 "W.L"을 붙여 셀에 입력해 줍니다.

```
(setq wx 3)
(repeat wllistn
```

waterline number의 표시 형식 중 셀의 특성을 설정한 부분입니다. 3열인 C1부터 시작하므로 변수 wx에 3을 저장하고, waterline 개수만큼 반복합니다.

```
(setq colY 1 rowX wx)
(setq cel(vlax-variant-value(vlax-get-property Cells 'Item colY rowX)))
```

해당 셀의 속성을 얻기 위해 셀을 객체로 잡아 변수 cel에 저장합니다.

```
(setq int(vlax-get-property cel 'Interior))
(setq Font(vlax-get-property cel 'Font))
(setq borders(vlax-get-property cel 'Borders))
```

셀의 배경색 설정을 위해 Interior 속성을 얻어와 변수 int에 저장합니다. 글꼴의 두께 설정을 위해 Font 속성을 얻어와 변수 Font에 저장합니다. 테두리 형식 지정을 위해 Borders 속성을 얻어와 변수 borders에 저장합니다.

```
(setq bordersB(vlax-get-property borders 'Item 9))
(setq bordersR(vlax-get-property borders 'Item 10))
```

테두리 중 가장 아래 부분의 속성을 얻어와 변수 bordersB에 저장합니다. 가장 오른쪽 부분의 속성을 얻어와 변수 bordersR에 저장합니다. 이렇게 따로 속성을 얻어오지 않으면 부분적으로 테두리의 두께, 종류 등을 설정해 줄 수 없습니다.

```

(vlax-put-property cel 'HorizontalAlignment -4108)
(vlax-put-property int 'Color 14599344)
(vlax-put-property Font 'Bold "true")
(vlax-put-property bordersB 'Weight 4)
(vlax-put-property bordersB 'LineStyle 1)
(vlax-put-property bordersR 'Weight 2)
(vlax-put-property bordersR 'LineStyle 1)
(setq wx(+ wx 1)))

```

셀에 입력된 값의 수평정렬을 가운데로 정렬합니다. 배경색을 변경하고, 글꼴 두께는 굵게 설정합니다. 가장 아래 부분의 테두리는 굵은 연속적 스타일의 테두리로, 가장 오른쪽 부분의 테두리는 얇은 연속적 스타일의 테두리로 설정한 후 변수 wx에 wx+1을 저장하며 repeat문을 종료합니다.

```

(setq Ran(vlax-get-property Excelapp 'Range "A1:B1"))
(vlax-put-property Ran 'MergeCells "true")

```

station number와 station의 실제 x좌표의 표시 형식은 WL 표시 형식 부분과 대부분 동일하므로 다른 부분만 발췌하여 설명하겠습니다. ARPA-GO를 통해 추출한 offset표의 형식을 보면 알 수 있듯이 A1셀과 B1셀은 병합되어 "No.Station"이라고 표시되어있습니다. 이를 위해 A1:B1을 범위로 잡아 변수 Ran으로 지정하고, MergeCells를 이용해 셀들을 병합하였습니다.

```

(setq bordersV(vlax-get-property borders 'Item 11))
(setq bordersH(vlax-get-property borders 'Item 12))

```

하나의 셀이 아닌 범위로 지정된 셀이므로 내부 테두리 중 수직 테두리의 속성을 얻어와 변수 bordersV에 저장하고, 수평 테두리의 속성은 변수 bordersH에 저장합니다.

13. offsetUB

서브함수 offsetUB는 사용자가 modeUB를 선택하였을 때, 엑셀에 나타낼 offset표의 형식을 지정하기 위한 함수입니다. 대부분 offsetNB함수와 동일하나 wlnumlist, backlist등의 리스트들을 다시 생성하지 않고 setWL 함수와 offsetNB 함수에서 생성한 리스트를 이용합니다.

14. InputExcel

```

(defun InputExcel()

```

서브함수 InputExcel 을 정의합니다. 교차점을 구하는 intersect 함수에서 구한 최종적인 좌표 리스트를 엑셀로 보내는 함수입니다.

```

(setq xlist blist)
(setq colY mm rowX 3)
(mapcar
  '(lambda(val)
    (vlax-put-property Cells 'Item colY rowX val)
    (setq rowX (1+ rowX))))xlist)

);_서브함수 InputExcel종료

```

intersect 함수를 통해 구한 x좌표 리스트인 blist를 변수 xlist로 저장합니다. 행의 값은 변수 mm으로 각 station별로 intersect 함수에서 지정해주었습니다. xlist를 해당 셀부터 가로로 삽입해줍니다.

15. Xposition

```

(defun Xposition()

```

서브함수 Xposition 을 정의합니다. STapart를 통해 입력 받은 선미부, 주요부, 선수부의 station 간격을 이용해 실제 x좌표를 엑셀에 입력하는 함수입니다. 각 부분별 station의 개수를 고려하여 위치를 계산해 엑셀에 입력합니다. 동일하거나 앞서 설명한 구문들이 사용되었으므로 선미부만을 발췌하여 설명하였습니다.

```

(setq colY 2 rowX 2)
(setq val (- 0 (* 2 stern)))
(vlax-put-property Cells 'Item colY rowX val)

```

2행2열, 즉 B2에 들어갈 값은 Transom의 x 좌표 이므로, stern 값에 2배를 한 후 0에서 빼준 값을 변수 val에 저장합니다. 셀 2행 2열에 val을 입력합니다.

16. setST

```

(defun setST()

  (separateST)
  (setAST)
  (setFST)

);_서브함수 setST 종료

```

서브함수 setST 를 정의합니다. 이 함수는 모든 station을 구분하기 위해 사용한 서브함수들을 묶어놓은 함수입니다.

4.1.2 Class

Python은 객체지향 프로그래밍 (OOP, Object Oriented Programming)을 기본적으로 지원하고 있습니다.

python에서 객체지향 프로그래밍의 기본단위인 class는 함수뿐만 아니라 다양한 변수를 포함한 큰 프로그램을 하나의 묶음으로 정의합니다.

대부분의 초보자 혹은 몇몇의 Python 프로그래머들은 인스턴스와 클래스 변수의 구별을 하지 못합니다. 이러한 문제들은 다양한 변수들을 틀린 방법으로 사용하도록 합니다.

가장 대표적인 차이점은 다음과 같습니다.

- 1) 인스턴스 변수들은 각각의 객체마다 다른 개개의 데이터를 가집니다.
- 2) 클래스 변수들은 하나의 클래스 내부에서 서로 다른 인스턴스 간 데이터를 공유합니다.

클래스를 작성하는 방법은 첫 행에는 class 예약어, 클래스 이름, 그리고 콜론을 사용합니다. Class 내부에 함수인 메서드(매개변수)를 정의하고 호출하여 결과 값을 도출합니다. 메서드도 클래스에 포함되어 있다는 점만 제외하면 일반함수와 다를 것이 없습니다.

참고

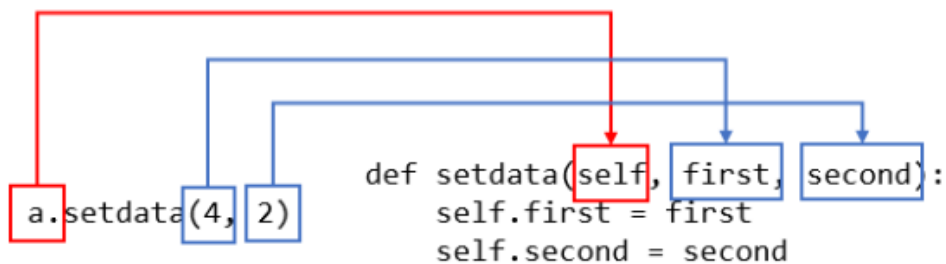
클래스 이름을 지을 때는 파스칼 표기법을 따르는 것이 관례입니다.

Class 호출 방법

- 1) 객체를 통해 메서드 호출하는 방법

a라는 객체를 만들고 이를 통해 setdata 메서드를 호출하는 방법입니다.

메서드 setdata는 총 3개의 매개변수를 필요로 하는데 실제로 (4,2) 2개의 값만 전달된 이유는 첫 번째 매개변수 self에는 setdata메서드를 호출한 객체 a가 자동으로 전달되기 때문입니다.



참고

객체를 통해 클래스의 메서드를 호출하려면 a.setdata(4,2)와 같이 도트(.) 연산자를 이용해야 합니다.

2) 클래스를 통해 메서드 호출하는 방법

“클래스명.메서드” 형태로 호출할 때에는 객체 a를 첫 번째 매개변수인 self에 꼭 전달해 주어야 합니다.

이와 같은 방법은 잘 사용하지 않습니다.

```
>>> a = FourCal()
>>> FourCal.setdata(a, 4, 2)
```

Class 특징

- 1) Class는 객체의 구조와 행동을 정의합니다.
- 2) 객체의 클래스는 초기화를 통해 제어합니다.
- 3) Class는 복잡한 문제를 다루기 쉽도록 만듭니다.

4.2 용어

선박계산을 하기 위해서는 선박관련 용어를 먼저 인지하고 있어야 합니다. 이 장에서는 ARPA-GO에서 사용된 용어들에 대해 설명합니다.

- LOA (Length overall) : 전장, 선체의 고정적으로 부착되는 돌출물을 포함한 최선미 끝 단에서 최선수 끝 단까지의 수평거리
- LWL (Length on the Waterline) : 수선길이, 선박의 저항 또는 운동성과 관련되는 길이로 설계수선과 선체의 선수부 및 선미부가 만나는 두 점 사이의 거리
- LBP (Length Between Perpendiculars) =LPP : 수선간장, 적재화물 측정에 기본이 되는 선박의 등록장으로써 선미수선(A.P) 과 선수수선(F.P) 간의 수평거리
- B : 선박의 폭
- D : 선박의 깊이
- 배수량 : 배가 물에 뜰 적에 그 무게로 인하여 밀려 나가는 물의 중량
- KB : 선저에서 부심까지의 거리, 부심의 기선으로부터의 수직거리는 기선에 관한 수면하부 용적의 모멘트를 그 용적으로 나눔으로써 얻을 수 있다.
- OB : 부력중심(부심) 에서 수선까지의 거리
- LCB (Longitudinal Center of Buoyancy) : midship, 선수 수선 혹은 선미 수선과 부심 간의 거리
- LCF : 부면심의 전후위치
- BM : 메타센터 반경 (BML : 종 메타센터반경, BMT: 횡 메타센터반경)
- KM : 선저에서 메타센터까지의 높이 (KB+BM)
- TPC (ton per cm) : 1cm 침하하는데 필요한 배수톤수 (배수량 증가량)
- MTC : 1cm 트림 일으키는데 필요한 종방향 모멘트
- KML : 선저 위의 종 방향 메타센터의 높이

- C_b : 방형계수, 선박의 배수 용적을 선박의 길이, 폭 및 흘수로 나눈 몫을 말하며, 선박수면 밑의 형상이 넓고 좁음을 나타내는 하나의 지수
- A_w : 수선면적, 수선으로 둘러싸인 부분의 수선면의 면적.
- C_w : 수선면계수, 수선면적과 수선면적을 포위하는 직사각형의 면적과의 비
- C_m : 중앙단면계수, 선박의 특정 흘수에서 중앙 횡단면의 흘수선 아래의 중앙 횡단면적과 그 부분에 외접하는 직사각형의 면적의 비이다
- C_p : 주형계수, 특정 흘수에서 선체의 배수용적과 중앙 횡단면과 같은 단면을 갖고 길이가 선체의 길이와 동일한 기둥형태의 용적과의 비
- C_{vp} : 수직주형계수, 배의 용적에 대한 배의 흘수와 그 흘수에서의 수선면과의 상승적, 즉 수선면적을 가지며 깊이가 흘수와 같은 주상체의 용적의 비