

Python Module (II)

김준호 / 박수현

모듈 만들기

- 내가 직접 모듈을 만들어 보자

내 모듈을 구현할 Python file 만들기

- Python IDLE에서 file editor 열기
 - File → New File
 - (* 다른 editor를 써도 됨)
- 벡터(vector) 모듈을 구현해 보자
 - my_vector.py 라는 이름으로 새로운 file을 저장

벡터 모듈 구현

- 2차원 벡터 클래스를 구현하고 테스트 하기
 - Instance 초기화, 객체 출력 method, 연산자 오버로딩(operator overloading) 등을 구현

my_vector.py

```
class vec2:
    x = 0.0
    y = 0.0
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def __str__(self):
        msg = "(" + str(self.x) + ", " + str(self.y) + ")"
        return msg
    def __add__(self, other):
        return vec2(self.x+other.x, self.y+other.y)
```

vec2 클래스 테스트

```
p = vec2(3,4)
q = vec2(-1, 2)
r = p + q
print(p)
print(q)
print(r)
```

```

1 class vec2:
2
3     x = 0.0
4     y = 0.0
5
6     def __init__(self, x, y):
7         print("i-1) in init")
8         self.x = x
9         self.y = y
10        print("i-2) self.x = ", self.x)
11        print("i-3) self.y = ", self.y)
12
13    def __str__(self):
14        msg = "(" + str(self.x) + ", " + str(self.y) + ")"
15        print("s-1) msg = ", msg)
16        return msg
17
18    def __add__(self, other):
19        print("a-1) in add")
20        print("a-2) self.x = ", self.x)
21        print("a-3) self.y = ", self.y)
22        print("a-4) other.x = ", other.x)
23        print("a-5) other.y = ", other.y)
24        return vec2(self.x + other.x, self.y + other.y)
25
26 # vec2 클래스 테스트
27 print("n1) vec2 class 테스트")
28 p = vec2(3,4)
29 print("n2) p = ", p)
30
31 q = vec2(-1, 2)
32 print("n3) q = ", q)
33
34 r = p + q
35 print("n4) r = ", r)

```

1) vec2 class 테스트

```

i-1) in init
i-2) self.x = 3
i-3) self.y = 4

```

```

2) p =
s-1) msg = (3, 4)
(3, 4)

```

```

i-1) in init
i-2) self.x = -1
i-3) self.y = 2

```

```

3) q =
s-1) msg = (-1, 2)
(-1, 2)

```

```

a-1) in add
a-2) self.x = 3
a-3) self.y = 4
a-4) other.x = -1
a-5) other.y = 2

```

```

i-1) in init
i-2) self.x = 2
i-3) self.y = 6

```

```

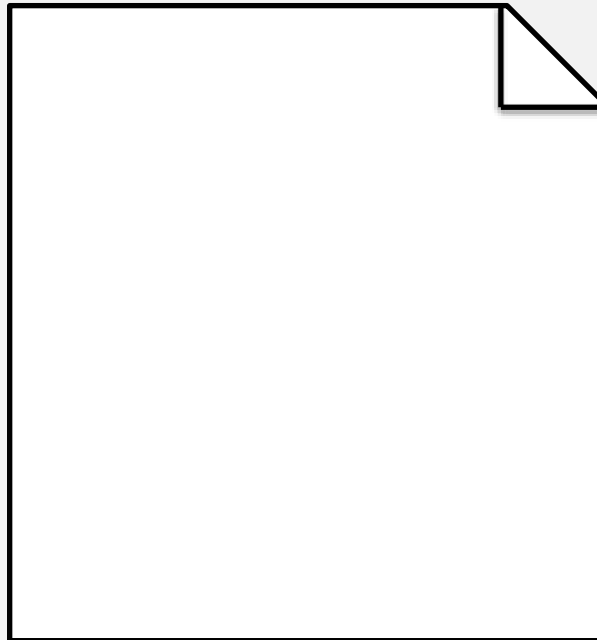
4) r =
s-1) msg = (2, 6)
(2, 6)

```

벡터 모듈 구현

- `my_vector.py` 파일 자체가 이미 `my_vector` 모듈
 - 모듈의 실체: 모듈은 Python code를 갖고 있는 text file (*.py)

`my_vector.py`



내가 만든 모듈 import 하기

- my_vector.py file이 있는 directory를 Python sys.path에 등록
 - sys.path.append() 함수 활용
- my_vector 모듈 사용

예) C:/Python34/MyModules 아래 my_vector.py가 있다면...

```
import sys

sys.path.append("C:/Python34/MyModules")

import my_vector

# my_vector 모듈의 vec2 클래스 활용
a = my_vector.vec2(9,0)
b = my_vector.vec2(-1, -2)
c = a + b

print(c)
```

내가 만든 모듈 import 하기

- 단순히 다음과 같이 처리

```
import my_vector
```



```

my_vector.py
File Edit Format Run Options Window Help
1 class vec2:
2
3     x = 0.0
4     y = 0.0
5
6     def __init__(self, x, y):
7         print("Wni-1) in init")
8         self.x = x
9         self.y = y
10        print("i-2) self.x = ", self.x)
11        print("i-3) self.y = ", self.y)
12
13    def __str__(self):
14        msg = "(" + str(self.x) + ", " + str(self.y) + ")"
15        print("Wns-1) msg = ", msg)
16        return msg
17
18    def __add__(self, other):
19        print("Wna-1) in add")
20        print("a-2) self.x = ", self.x)
21        print("a-3) self.y = ", self.y)
22        print("a-4) other.x = ", other.x)
23        print("a-5) other.y = ", other.y)
24        return vec2(self.x + other.x, self.y + other.y)
25
26    # vec2 클래스 테스트
27    print("Wn1) vec2 class 테스트")
28    p = vec2(3,4)
29    print("Wn2) p = ", p)
30
31    q = vec2(-1, 2)
32    print("Wn3) q = ", q)
33
34    r = p + q
35    print("Wn4) r = ", r)

```

```

import_my_vector.py
File Edit Format Run Options Window Help
1 import my_vector
2
3 # main
4 print("Wnim-1) import_my_vector 시작")
5
6 print("im-2) 객체 a 생성")
7 a = my_vector.vec2(9, 0)
8 print("im-3) a = ", a)
9
10 print("im-4) 객체 b 생성")
11 b = my_vector.vec2(-1, -2)
12 print("im-5) b = ", b)
13
14 c = a + b
15 print("im-6) c = ", c)
16

```

1) vec2 class 테스트

```

i-1) in init
i-2) self.x = 3
i-3) self.y = 4

2) p =
s-1) msg = (3, 4)
(3, 4)

i-1) in init
i-2) self.x = -1
i-3) self.y = 2

3) q =
s-1) msg = (-1, 2)
(-1, 2)

a-1) in add
a-2) self.x = 3
a-3) self.y = 4
a-4) other.x = -1
a-5) other.y = 2

i-1) in init
i-2) self.x = 2
i-3) self.y = 6

4) r =
s-1) msg = (2, 6)
(2, 6)

```

im-1) import_my_vector 시작

```

im-2) 객체 a 생성

i-1) in init
i-2) self.x = 9
i-3) self.y = 0
im-3) a =
s-1) msg = (9, 0)
(9, 0)
im-4) 객체 b 생성

i-1) in init
i-2) self.x = -1
i-3) self.y = -2
im-5) b =
s-1) msg = (-1, -2)
(-1, -2)

a-1) in add
a-2) self.x = 9
a-3) self.y = 0
a-4) other.x = -1
a-5) other.y = -2

i-1) in init
i-2) self.x = 8
i-3) self.y = -2
im-6) c =
s-1) msg = (8, -2)
(8, -2)

```

내가 만든 모듈 import 하기

- 문제점?

- Import_my_vector.py 파일에서 my_vector 모듈을 import하면 my_vector .py가 자동으로 동작함
- 모듈로 작동할 때는 해당코드가 동작하지 않도록 하기

my_vector.py

```
class vec2:
    # ....

# vec2 클래스 테스트
p = vec2(3,4)
q = vec2(-1, 2)
r = p + q
print(p)
print(q)
print(r)
```

내가 만든 모듈 import 하기

- my_vector.py가 모듈로 import 되는 경우가 아니라, Script로 실행되는 경우에만 해당 부분이 동작하도록 변경
 - `if __name__ == "__main__":`

```
my_vector.py
class vec2:
    # ....

# vec2 클래스 테스트
if __name__ == "__main__":
    p = vec2(3,4)
    q = vec2(-1, 2)
    r = p + q
    print(p)
    print(q)
    print(r)
```

my_vector.py 파일이 Script로 실행될 때만
해당 부분이 실행되도록 변경

main

- 참고) http://hashcode.co.kr/questions/3/if-__name__-__main__%EC%9D%80-%EC%99%9C%EC%93%B0%EB%82%98%EC%9A%94 (2020/06/01 현재)
- Script가 Python interpreter 명령어로 passing되어 실행되면(python a.py같이) 다른 언어들과는 다르게 Python은 자동으로 실행되는 main 함수가 없음
- Python은 main 함수가 없는 대신 들여쓰기 하지 않은 모든 코드 (level 0 code)를 실행
- 다만, function이나 class는 정의되었지만, 실행되지는 않음
- `__name__`은 현재 모듈의 이름을 담고있는 내장 변수임
- `python a.py` 같이 이 모듈이 직접 실행되는 경우에만, `__name__`은 `"__main__"`으로 설정됨

```

1 def func():
2     print("5) 여기는 function func() in a.py")
3
4 print("6) top-level A.py")
5
6 if __name__ == "__main__":
7     print("7) a.py 직접 실행")
8 else:
9     print("8) a.py가 임포트되어 사용됨")
10

```

```

1 import a
2
3 print("1) top-level in B.py")
4 print("2) a.py에 있는 a.func() 호출 ")
5 a.func()
6
7 if __name__ == "__main__":
8     print("3) b.py가 직접 실행")
9 else:
10    print("4) b.py가 임포트되어 사용됨")
11

```

main

6) top-level A.py
7) a.py 직접 실행

a.py를 실행 시

6) top-level A.py
8) a.py가 임포트되어 사용됨
1) top-level in B.py
2) a.py에 있는 a.func() 호출
5) 여기는 function func() in a.py
3) b.py가 직접 실행

b.py를 실행 시

```

1 class vec2:
2
3     x = 0.0
4     y = 0.0
5
6     def __init__(self, x, y):
7         print("i-1) in init")
8         self.x = x
9         self.y = y
10        print("i-2) self.x = ", self.x)
11        print("i-3) self.y = ", self.y)
12
13    def __str__(self):
14        msg = "(" + str(self.x) + ", " + str(self.y) + ")"
15        print("i-4) msg = ", msg)
16        return msg
17
18    def __add__(self, other):
19        print("i-5) in add")
20        print("a-2) self.x = ", self.x)
21        print("a-3) self.y = ", self.y)
22        print("a-4) other.x = ", other.x)
23        print("a-5) other.y = ", other.y)
24        return vec2(self.x + other.x, self.y + other.y)
25
26 # vec2 클래스 테스트
27 if __name__ == "__main__":
28
29     print("nm-1) vec2 class 테스트")
30     p = vec2(3,4)
31     print("nm-2) p = ", p)
32
33     q = vec2(-1, 2)
34     print("nm-3) q = ", q)
35
36     r = p + q
37     print("nm-4) r = ", r)

```

```

1 import my_vector2
2
3 # main
4 print("im-1) import_my_vector 시작")
5
6 print("im-2) 객체 a 생성")
7 a = my_vector2.vec2(9, 0)
8 print("im-3) a = ", a)
9
10 print("im-4) 객체 b 생성")
11 b = my_vector2.vec2(-1, -2)
12 print("im-5) b = ", b)
13
14 c = a + b
15 print("im-6) c = ", c)
16

```

im-1) import_my_vector 시작
im-2) 객체 a 생성

i-1) in init
i-2) self.x = 9
i-3) self.y = 0
im-3) a =
s-1) msg = (9, 0)
(9, 0)
im-4) 객체 b 생성

i-1) in init
i-2) self.x = -1
i-3) self.y = -2
im-5) b =
s-1) msg = (-1, -2)
(-1, -2)

a-1) in add
a-2) self.x = 9
a-3) self.y = 0
a-4) other.x = -1
a-5) other.y = -2

i-1) in init
i-2) self.x = 8
i-3) self.y = -2
im-6) c =
s-1) msg = (8, -2)
(8, -2)

```

*my_vector_only_class.py
File Edit Format Run Options Window Help
1 class vec_3:
2
3     x = 0.0
4     y = 0.0
5
6     def __init__(self, x, y):
7         print("\nin init")
8         self.x = x
9         self.y = y
10        print("in init, self.x = ", self.x)
11        print("in init, self.y = ", self.y)
12
13    def __str__(self):
14        msg = "(" + str(self.x) + ", " + str(self.y) + ")"
15        print("\nin str, msg = ", msg)
16        return msg
17
18    def __add__(self, other):
19        print("\nin add")
20        print("self.x = ", self.x)
21        print("self.y = ", self.y)
22        print("other.x = ", other.x)
23        print("other.y = ", other.y)
24        return vec_3(self.x + other.x, self.y + other.y)
25

```

```

*import_my_vector_only_class.py
File Edit Format Run Options Window Help
1 import my_vector_only_class
2
3 # main
4 print("\n1) import_my_vector_only_class 시작")
5
6 print("\n2) 객체 a 생성")
7 a = my_vector_only_class.vec_3(9, 0)
8 print("\n3) a = ", a)
9
10 print("\n4) 객체 b 생성")
11 b = my_vector_only_class.vec_3(-1, -2)
12 print("\n5) b = ", b)
13
14 c = a + b
15 print("\n6) c = ", c)
16

```

1) import_my_vector_only_class 시작
2) 객체 a 생성

in init
in init, self.x = 9
in init, self.y = 0
3) a =
in str, msg = (9, 0)
(9, 0)
4) 객체 b 생성

in init
in init, self.x = -1
in init, self.y = -2
5) b =
in str, msg = (-1, -2)
(-1, -2)

in add
self.x = 9
self.y = 0
other.x = -1
other.y = -2

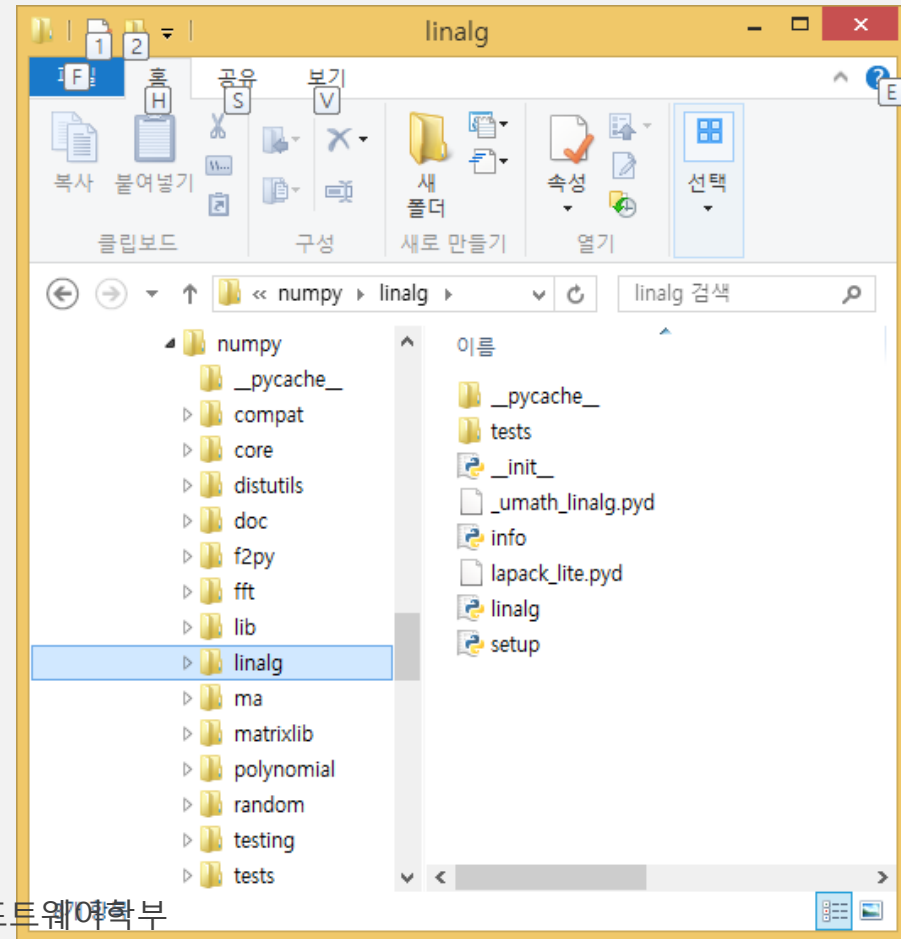
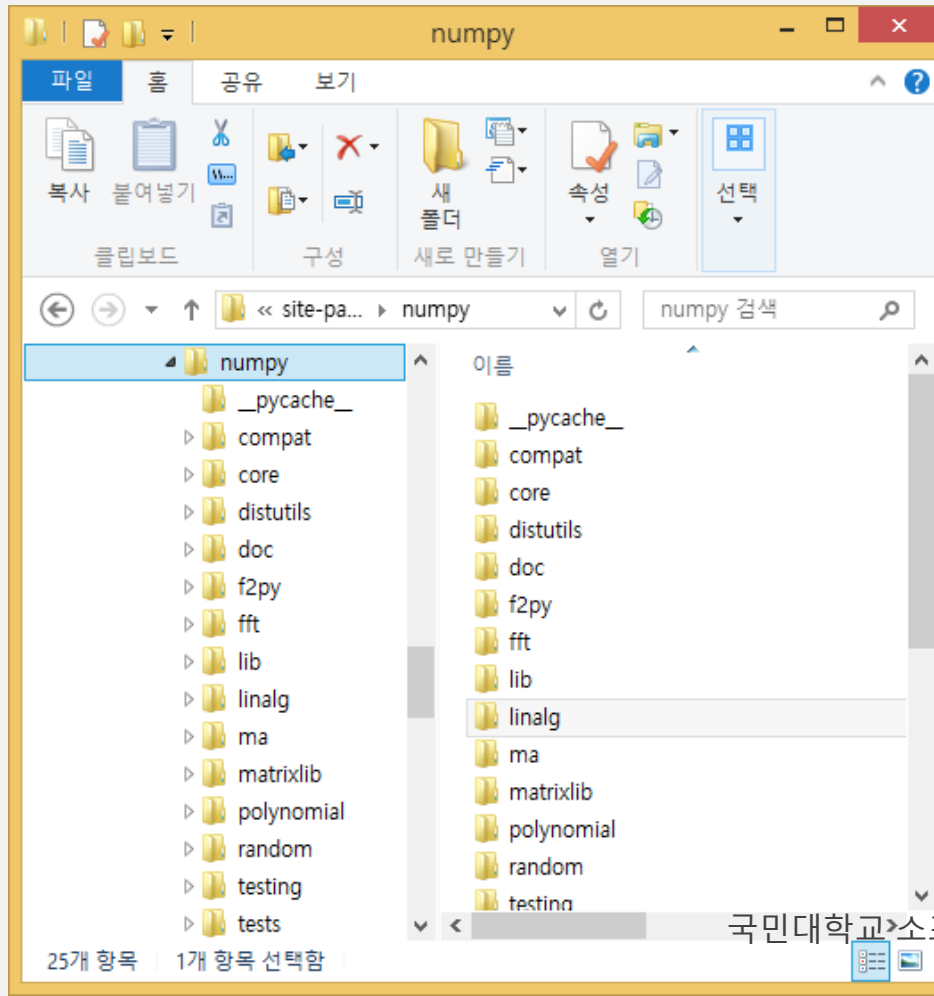
in init
in init, self.x = 8
in init, self.y = -2
6) c =
in str, msg = (8, -2)
(8, -2)

Package 만들기

- 여러 모듈들이 Directory 계층적으로 구성된 package를 만들어 보자

Package

- 닷(.)을 이용하여 계층적으로 구성된 Python 모듈들
 - 예) numpy package numpy.linalg package



Package 만들기

- Directory 계층구조를 자신이 원하는 대로 만듦
- Package로 만들 Directory에 `__init__.py` 파일을 만듦
- 각 Directory 마다 원하는 *.py 파일을 위치 시킴

| | |
|-------------|-------------------------------------|
| sound/ | sound package (최상위 package) |
| __init__.py | sound package 초기화 |
| formats/ | sound.formats package (sub package) |
| __init__.py | sound.formats package 초기화 |
| auread.py | |
| auwrite.py | |
| ... | |
| effects/ | sound.effects package(sub package) |
| __init__.py | sound.effects package 초기화 |
| echo.py | |
| surround.py | |
| ... | |

Package 만들기

- `__init__.py` 파일은 내용이 없어도 무방
 - 대부분의 경우, `__init__.py` 파일은 해당 패키지가 `import *`에 대해 어떤 함수들이 자동으로 `import` 될 것인가를 설명하는 `__all__` 변수 세팅

| | | |
|-------------|--|----------------------------|
| sound/ | | sound 패키지 (최상위 패키지) |
| __init__.py | | sound 패키지 초기화 |
| formats/ | | sound.formats 패키지 (서브 패키지) |
| __init__.py | | sound.formats 패키지 초기화 |
| auread.py | | |
| auwrite.py | | |
| ... | | |
| effects/ | | sound.effects 패키지 (서브 패키지) |
| __init__.py | | sound.effects 패키지 초기화 |
| echo.py | | |
| surround.py | | |
| ... | | |

Package 활용

- 패키지 등록

- 만일 sound 디렉터리의 위치가 "C:/Python34/MyModules/sound/"면, "C:/Python34/MyModules/"를 sys.path에 추가

```
import sys
```

```
sys.path.append("C:/Python34/MyModules")
```

Package 활용

- 패키지 import
 - sound 패키지, sound.formats 패키지, sounds.effects 패키지 импорт

```
import sys
```

```
sys.path.append("C:/Python34/MyModules")
```

```
import sound
```

```
import sound.formats
```

```
import sound.effects
```

Package 활용

- 패키지에 포함된 함수 호출
 - sound/effects/echo.py 파일에 echo_test()라는 함수를 호출하려면
 - 1) sound.effects.echo 모듈 import
 - 2) sound.effects.echo.echo_test() 함수 호출

```
import sys

sys.path.append("C:/Python34/MyModules")

import sound.effects.echo

sound.effects.echo.echo_test()
```

Package 활용

- **from ... import *** 로 Sub Package를 한꺼번에 import 하려면

- sound/__init__.py 파일에 __all__ 변수를 다음과 같이 세팅

__all__ = ['format', 'effects']

sound/__init__.py

```
# from sound import * 동작 원리 설정  
__all__ == ['format', 'effects']
```

```
import sys
```

```
sys.path.append("C:/Python34/MyModules")
```

```
from * import sound  
임포트
```

```
# sound 패키지, sound.format 패키지, sound.effects 패키지 모두
```

Package 활용

- Package에 포함된 함수 호출을 위해 typing이 길다고 생각되면
 - **from ... import ...** 활용

```
import sys
```

```
sys.path.append("C:/Python34/MyModules")
```

```
from sound.effects.echo import echo_test
```

```
echo_test()          # sound.effects.echo.e 패키지, sound.format 패키지, sound.effects 패키지 모  
두 импорт
```


모듈/ Package 활용에 대한 모든 것

- 모듈
 - <https://docs.python.org/3/tutorial/modules.html#>
- Package
 - <https://docs.python.org/3/tutorial/modules.html#packages>

다른 사람들이 작성해 놓은 유용한 모듈 사용해 보기

- pip를 이용한 Python Package 관리
다른 사람들이 작성해 놓은 유용한 패키지 검색/설치/삭제
- NumPy 사용하기
- 행렬/벡터 연산을 쉽게 할 수 있는 Python 모듈을 사용해 보자
- matplotlib 사용하기
데이터를 다양한 그래프를 이용하여 효과적으로 표현하자
- Curses 사용하기
텍스트 기반 사용자 인터페이스를 만들어 보자

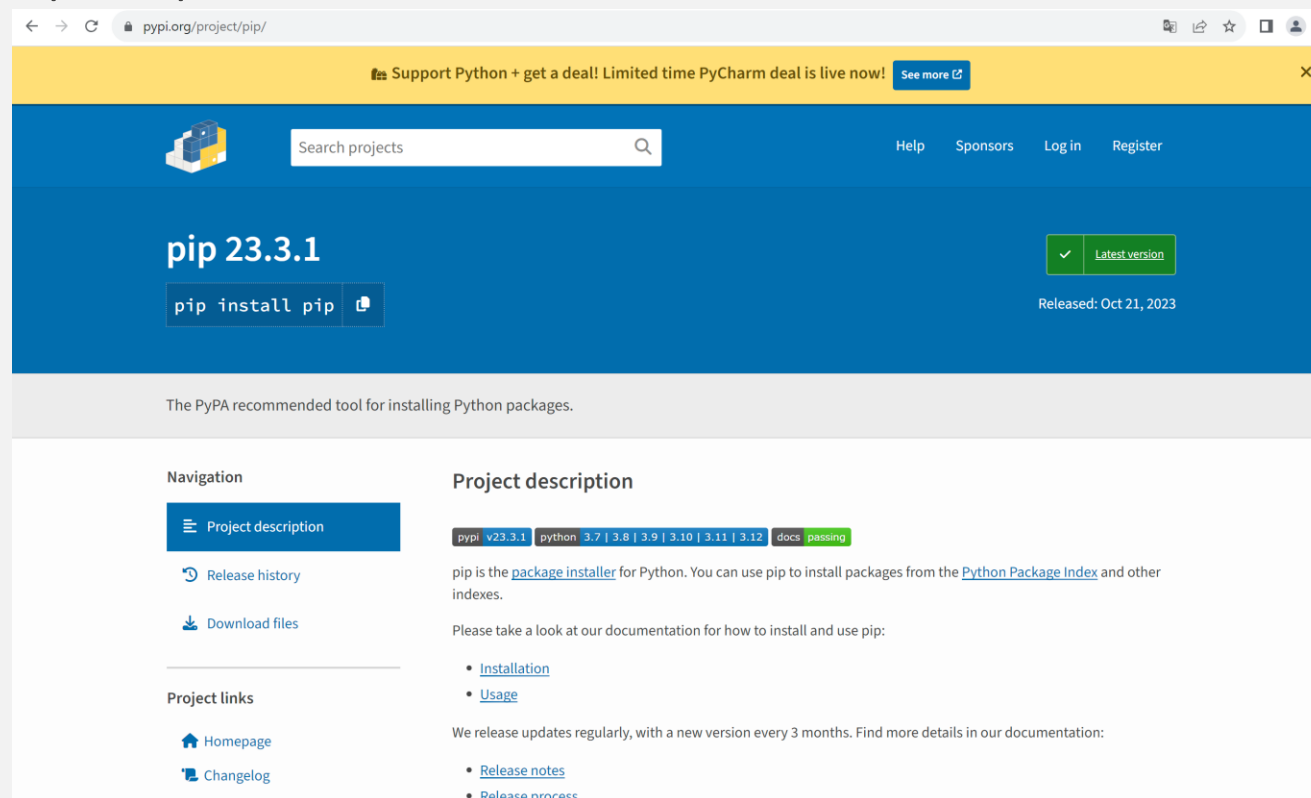
pip를 이용한 Python Package 관리

- pip (Python Package 매니저, **p**ip **i**nstalls **p**ackage)

- <https://pypi.org/project/pip/>
- 인터넷을 통해, Python 패키지를 검색/설치/삭제/업데이트 할 수 있음
- Python 3.4 버전부터 기본으로 포함

- 사용법

- 검색: pip search 키워드 → 서비스 중지
- 설치: pip install 패키지명
- 삭제: pip uninstall 패키지명
- 업데이트: pip install -u 패키지명



pip를 이용한 Python Package 관리

- 예) numpy (벡터/행렬 관련 Package) install
 - pip install numpy
- **주의사항:** Windows의 경우, C/C++ compiler의 문제 등으로 단순한 pip 활용으로 install되지 않은 Package가 꽤 있음
 - <http://www.lfd.uci.edu/~gohlke/pythonlibs/> 에서 해당 Package의 Wheels (*.whl 파일)을 받아 install
 - 예) Python 3.4 이상 버전, Windows 64 비트인 경우 numpy package install
 - 1) numpy-1.21.4+mkl-pp38-pypy38_pp73-win_amd64.whl 파일 다운 받기
 - 2) pip install numpy-1.21.4+mkl-pp38-pypy38_pp73-win_amd64.whl

실습

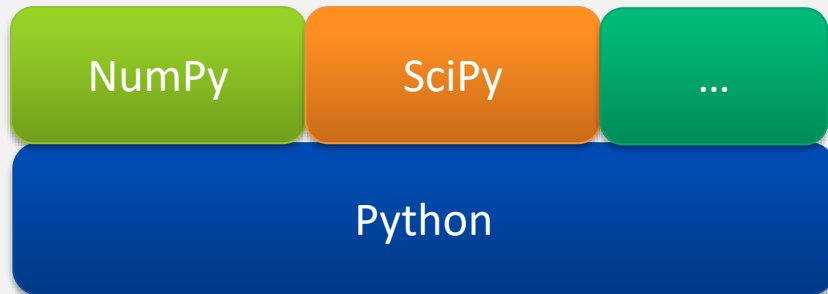
- pip를 이용해서 다음 Package들을 install해보자
 - numpy
 - matplotlib

NumPy 사용하기

- 데이터 수치해석 / visualization

What is NumPy?

- NumPy
 - Numerical Python
 - 데이터, 수치 분석을 위한 Python 패키지 중 하나
 - 효율적인 선형대수 프로그래밍 가능
 - 사용하기 쉬움



What is NumPy?

- 효율적인 선형대수 연산 제공
 - 행렬(Matrix)과 벡터(Vector) 사용
 - 차수가 높은 행렬/벡터 연산도 손쉽게 가능

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax + by \\ cx + dy \end{bmatrix}$$
$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by + c \\ dx + ey + f \\ 1 \end{bmatrix}$$
$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} ax + by + cz \\ dx + ey + fz \\ gx + hy + iz \end{bmatrix}$$
$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by + cz + d \\ ex + fy + gz + h \\ ix + jy + kz + l \\ 1 \end{bmatrix}$$



What is NumPy?

- Python 수치해석 프로그램의 기본 패키지로 널리 쓰임
 - NumPy + SciPy == MATLAB

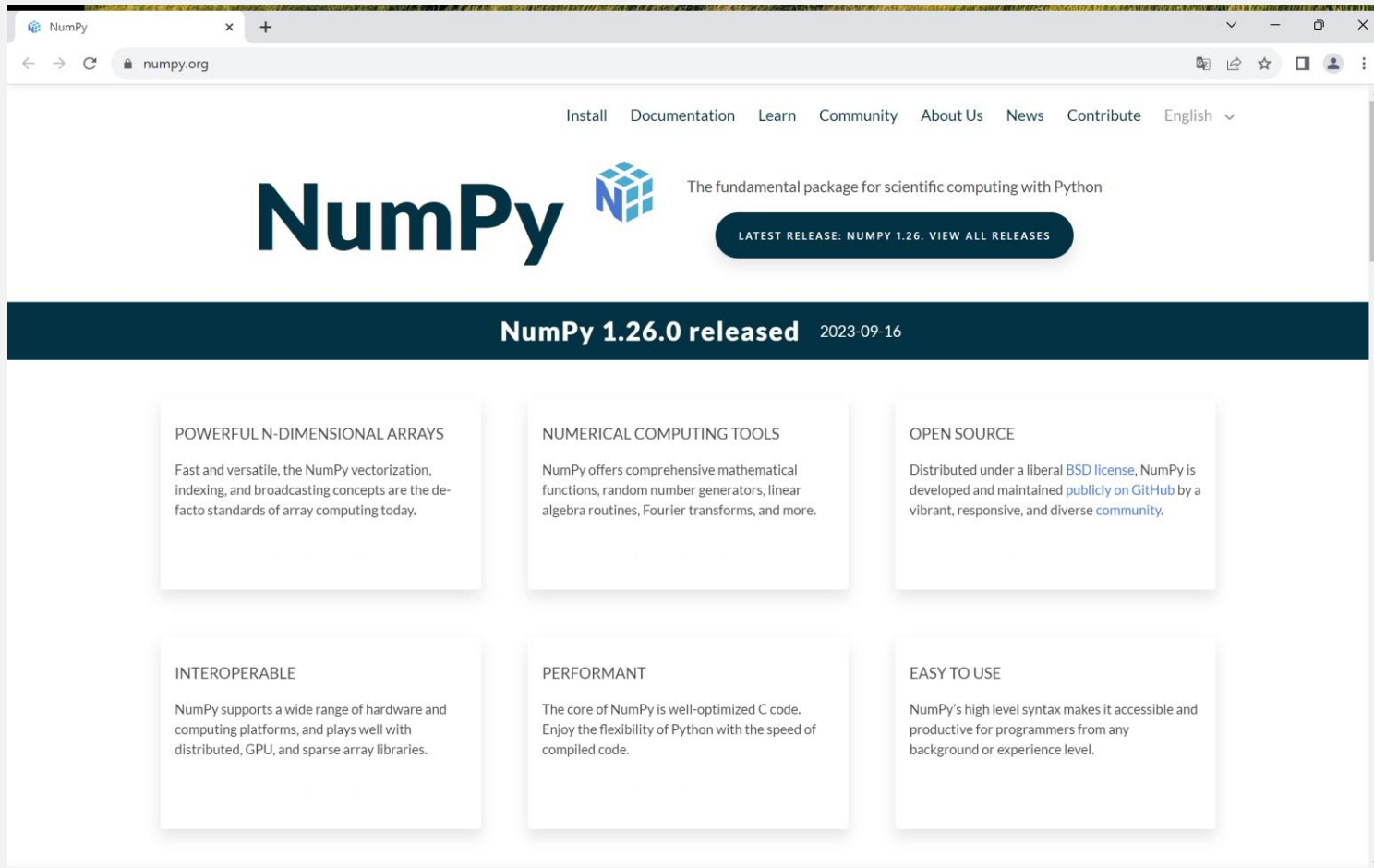


NumPy 설치

- Python 3.4 이상
 - > pip install numpy
 - Windows 운영체제의 경우, <http://www.lfd.uci.edu/~gohlke/pythonlibs/> 에서 numpy 패키지의 Wheels (*.whl 파일)을 받아 인스톨
 - 예) Python 3.4 이상 버전, Windows 64 비트인 경우 numpy 패키지 인스톨
 - 1) numpy-1.21.4+mkl-pp38-pypy38_pp73-win_amd64.whl 파일 다운 받기
 - 2) pip install numpy-1.21.4+mkl-pp38-pypy38_pp73-win_amd64.whl- NumPy web page
 - <http://www.numpy.org/>

NumPy 설치

- NumPy web page
 - <http://www.numpy.org/>



NumPy 설치

```
PS C:\Users\SooHyunPark> pip install numpy
Collecting numpy
  Downloading numpy-1.26.2-cp311-cp311-win_amd64.whl.metadata (61 kB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 61.2/61.2 kB 809.0 kB/s eta 0:00:00
  Downloading numpy-1.26.2-cp311-cp311-win_amd64.whl (15.8 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 15.8/15.8 MB 11.9 MB/s eta 0:00:00
Installing collected packages: numpy
Successfully installed numpy-1.26.2
PS C:\Users\SooHyunPark>
```

Example

행렬(matrix), 벡터(vector) 만들기

```
>>> import numpy as np
```

```
>>> A = np.mat([[1,0,0],[0,1,0],[0,0,1]])
```

```
>>> B = np.mat([[1],[0],[0]])
```

```
>>> import numpy as np
```

```
>>> A = np.mat([[1,0,0],[0,1,0],[0,0,1]])
```

```
>>> A = np.mat([[1,0,0],[0,1,0],[0,0,1]])
```

```
>>> B = np.mat([[1],[0],[0]])
```

```
>>> A  
matrix([[1, 0, 0],  
        [0, 1, 0],  
        [0, 0, 1]])
```

```
>>> B  
matrix([[1],  
        [0],  
        [0]])
```

```
>>>
```

Example

행렬과 벡터의 연산

matrix_vector.py

File Edit Format Run Options Window Help

```
1 import numpy as np
2
3 A = np.mat([[1,0,0],[0,1,0],[0,0,1]])
4 B = np.mat([[1],[0],[0]])
5
6 print("Wn1) A ")
7 print(A)
8
9 C = A + A
10 print("Wn2) A + A ")
11 print(C)
12
13 C = A - A
14 print("Wn3) A - A ")
15 print(C)
16
17 C = A * A
18 print("Wn4) A * A ")
19 print(C)
20
21
22 print("Wn5) B ")
23 print(B)
24
25 C = B + B
26 print("Wn6) B + B ")
27 print(C)
28
29 C = B - B
30 print("Wn7) B - B")
31 print(C)
32
33
```

1) A
 $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ identity matrix
(항등행렬)

2) A + A
 $\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$ diagonal matrix
(대각행렬)

3) A - A
 $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ zero matrix
(영행렬)

4) A * A
 $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

5) B
 $\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

6) B + B
 $\begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}$

7) B - B
 $\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

>>>

Example

역행렬(inverse matrix) 구하기

```
inverse_matrix.py
File Edit Format Run Options Window Help
1 import numpy as np
2
3 A = np.mat([[2,1,1],[3,2,1],[2,1,2]])
4 print()
5 print(A)
6
7 print()
8 inv_A = np.linalg.inv(A)
9 print(inv_A)
10
11 print()
12 print(A*inv_A)
13
```

```
[[2 1 1]
 [3 2 1]
 [2 1 2]]

[[ 3. -1. -1.]
 [-4.  2.  1.]
 [-1.  0.  1.]]

[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

Example

행렬에서 요소 접근하기

```
>>> import numpy as np
>>> A = np.mat([[1,2,3],[4,5,6],[7,8,9]])
>>> A[1,1]
>>> A[1,1] = 0
>>> A[0,0] = A[1,0] + A[2,0]
>>> A[4,4] = 0
```

```
>>>
>>> import numpy as np
>>> A = np.mat([[1,2,3],[4,5,6],[7,8,9]])
>>> A
matrix([[1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]])
>>> A[1,1]
5
>>> A[1,1] = 0
>>> A
matrix([[1, 2, 3],
        [4, 0, 6],
        [7, 8, 9]])
>>> A[0,0] = A[1,0] + A[2,0]
>>> A
matrix([[11, 2, 3],
        [4, 0, 6],
        [7, 8, 9]])
>>> A[4,4] = 0
Traceback (most recent call last):
  File "<pyshell#19>", line 1, in <module>
    A[4,4] = 0
IndexError: index 4 is out of bounds for axis 0 with size 3
>>>
```


Example

행렬에서 행 또는 열 추출하기

```
extract_from_matrix.py
File Edit Format Run Options Window Help
1 import numpy as np
2
3 A = np.mat([[1,2,3],[4,5,6],[7,8,9]])
4
5 print("\n0) A = ")
6 print(A)
7
8 print("\n1) A[0, 0] = ", A[0, 0])
9
10 print("\n2) A[0:3, 0]) = ")
11 print(A[0:3, 0])
12
13 print("\n3) A[0, 1:3] = ")
14 print(A[0, 1:3])
15
16 print("\n4) A[0, :] = ")
17 print(A[0, :])
18
19 print("\n5) A[:, 1] = ")
20 print(A[:, 1])
21
```

```
0) A =
[[1 2 3]
 [4 5 6]
 [7 8 9]]

1) A[0, 0] = 1

2) A[0:3, 0]) =
[[1]
 [4]
 [7]]

3) A[0, 1:3] =
[[2 3]]

4) A[0, :] =
[[1 2 3]]

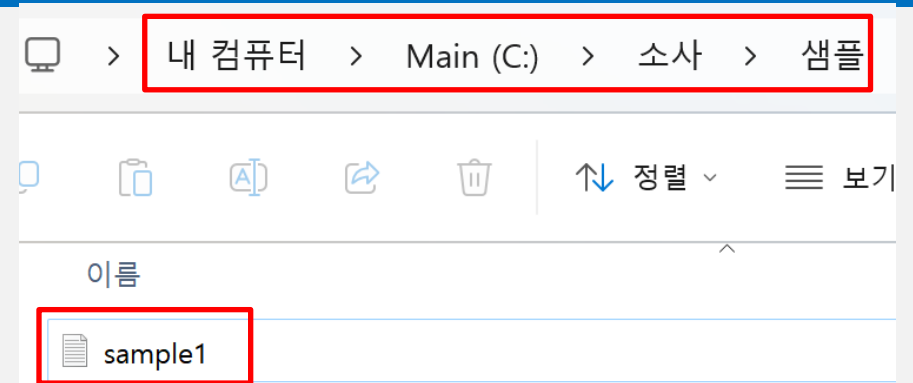
5) A[:, 1] =
[[2]
 [5]
 [8]]

>>>
```

Example

행렬을 텍스트 파일로 저장

```
matrix_to_text.py
File Edit Format Run Options Window Help
1 import os
2
3 os.makedirs("C:/소사/샘플")
4
5 import numpy as np
6
7 A = np.mat([[1,2,3],[4,5,6],[7,8,9]])
8 print("Wn1) A")
9 print(A)
10
11 print("Wn2) C:/소사/샘플/sample1.txt 생성")
12
13 np.savetxt("C:/소사/샘플/sample1.txt", A, fmt="%d")
14
```

A screenshot of a Windows Notepad window titled 'sample1 - Windows 메모장'. The menu bar includes '파일(F)', '편집(E)', '서식(O)', and '보기(V)'. The text content of the file is:
1 2 3
4 5 6
7 8 9

```
1) A
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

```
2) C:/소사/샘플/sample1.txt 생성
```

Example

텍스트 파일로부터 행렬 읽기

 text_to_matrix.py

File Edit Format Run Options Window Help

```
1 import numpy as np
2
3 print("\n1) C:/소사/샘플/sample1.txt load")
4 data = np.loadtxt("C:/소사/샘플/sample1.txt")
5
6 print("\n2) np.loadtxt(data)")
7 print(data)
8
9 data = np.mat(data)
10 print("\n3) np.mat(data)")
11 print(data)
12
```

1) C:/소사/샘플/sample1.txt load

2) np.loadtxt(data)

```
[[1. 2. 3.]
 [4. 5. 6.]
 [7. 8. 9.]]
```

3) np.mat(data)

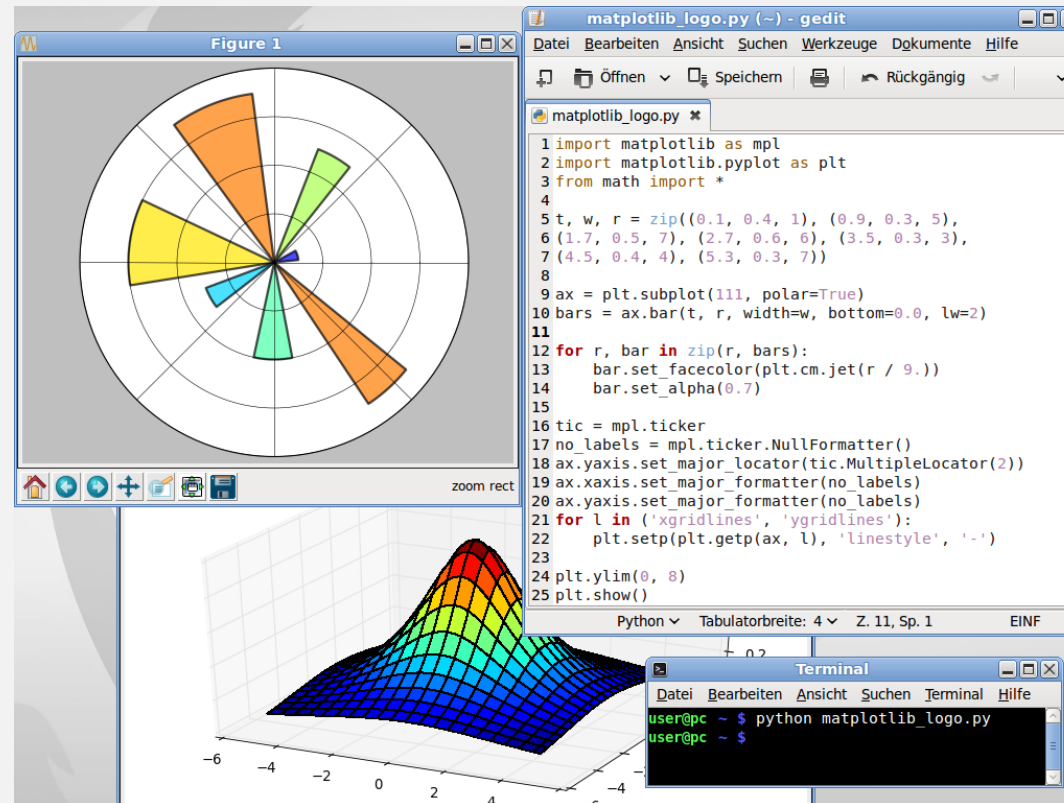
```
[[1. 2. 3.]
 [4. 5. 6.]
 [7. 8. 9.]]
```

matplotlib 사용하기

- 데이터를 다양한 그래프를 이용하여 효과적으로 표현하자

matplotlib이란

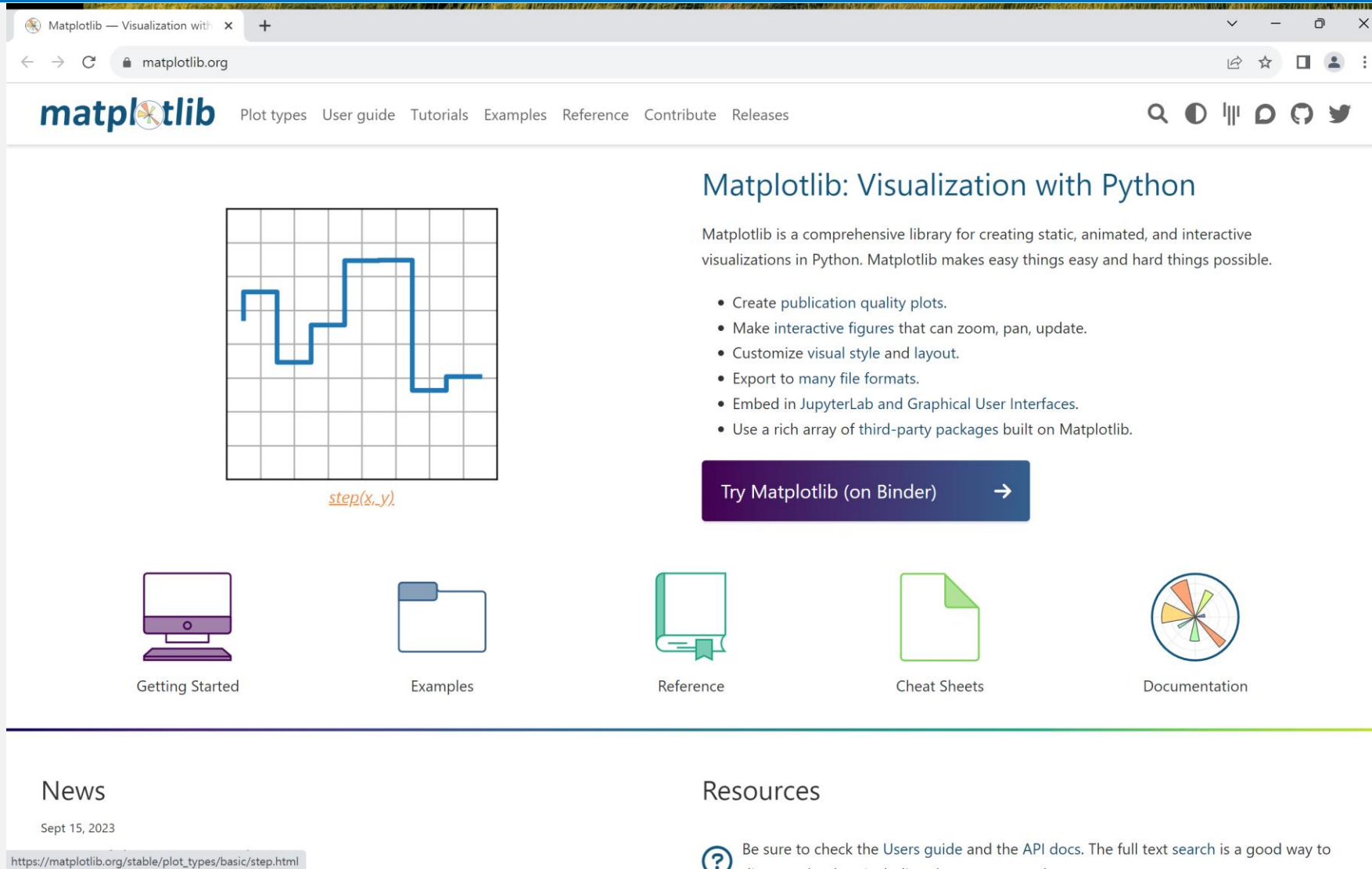
- Python plotting 패키지
 - 데이터, 통계 자료 분석 및 가시화 패키지(visualization package)



matplotlib 설치

- Python 3.4 이상
 - > pip install matplotlib
 - numpy 패키지와 의존성이 있음
 - Windows 운영체제의 경우, <http://www.lfd.uci.edu/~gohlke/pythonlibs/> 에서 numpy 패키지의 Wheels (*.whl 파일)을 받아 인스톨
 - 예) Python 3.4 이상 버전, Windows 64 비트인 경우 numpy 패키지 인스톨
 - 1) [matplotlib-3.5.0-pp38-pypy38_pp73-win_amd64.whl](#) 파일 다운 받기
 - 2) pip install matplotlib-3.5.0-pp38-pypy38_pp73-win_amd64.whl
- matplotlib web page
 - <http://matplotlib.org/>

matplotlib 설치



The screenshot shows the Matplotlib website homepage. At the top, there's a navigation bar with links: Plot types, User guide, Tutorials, Examples, Reference, Contribute, and Releases. The main heading is "Matplotlib: Visualization with Python". Below this, a paragraph describes Matplotlib as a comprehensive library for creating static, animated, and interactive visualizations in Python. A list of features includes: Create publication quality plots, Make interactive figures that can zoom, pan, update, Customize visual style and layout, Export to many file formats, Embed in JupyterLab and Graphical User Interfaces, and Use a rich array of third-party packages built on Matplotlib. A prominent button says "Try Matplotlib (on Binder)". Below the main content, there are five icons representing different sections: Getting Started (computer icon), Examples (folder icon), Reference (book icon), Cheat Sheets (document icon), and Documentation (wheel icon). At the bottom, there's a "News" section with a date "Sept 15, 2023" and a "Resources" section with a link to the Users guide and API docs.

Matplotlib — Visualization with Python


matplot**lib** Plot types User guide Tutorials Examples Reference Contribute Releases

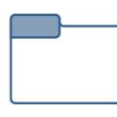
Matplotlib: Visualization with Python


Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.


- Create publication quality plots.
- Make interactive figures that can zoom, pan, update.
- Customize visual style and layout.
- Export to many file formats.
- Embed in JupyterLab and Graphical User Interfaces.
- Use a rich array of third-party packages built on Matplotlib.


[Try Matplotlib \(on Binder\)](#)

 Getting Started

 Examples

 Reference

 Cheat Sheets

 Documentation

News

Sept 15, 2023

https://matplotlib.org/stable/plot_types/basic/step.html

Resources

Be sure to check the Users guide and the API docs. The full text search is a good way to

matplotlib 설치

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

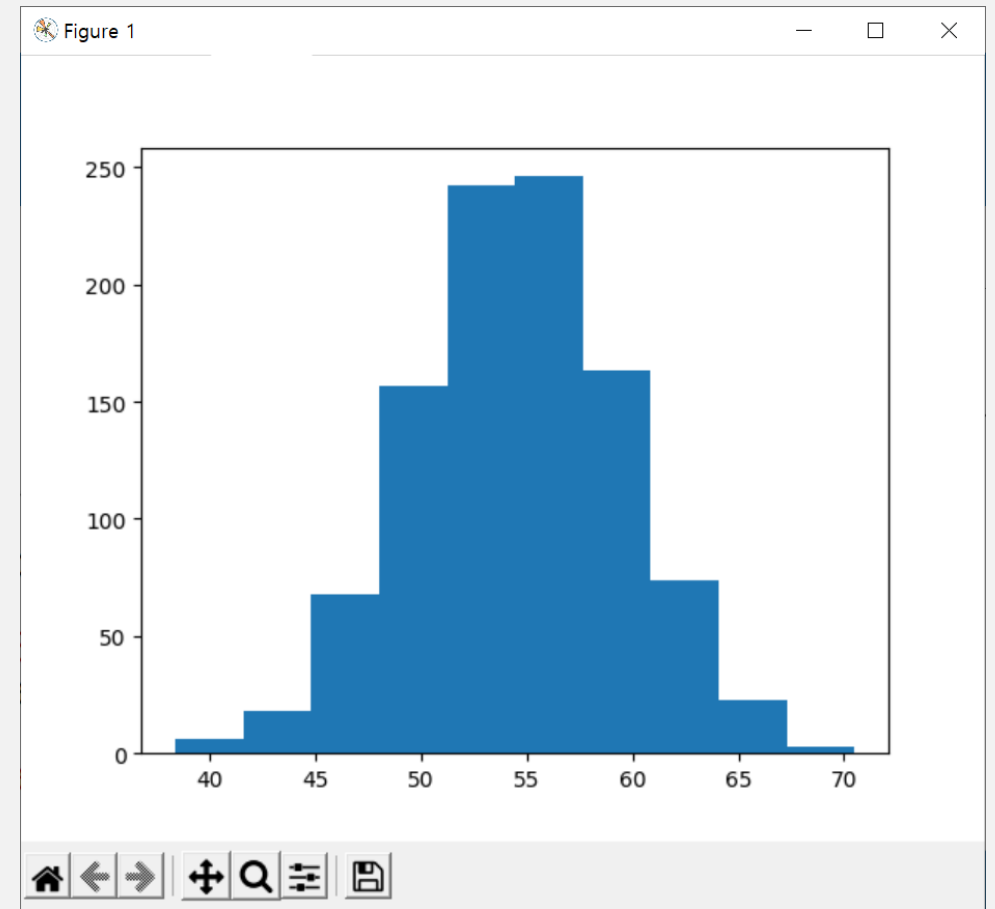
새로운 기능 및 개선 사항에 대한 최신 PowerShell을 설치 하세요! https://aka.ms/PSWindows

PS C:\Users\SooHyunPark> pip install matplotlib
Requirement already satisfied: matplotlib in c:\users\soohyunpark\appdata\local\programs\python\python311\lib\site-packages (3.8.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\soohyunpark\appdata\local\programs\python\python311\lib\site-packages (from matplotlib) (1.1.1)
Requirement already satisfied: cycler>=0.10 in c:\users\soohyunpark\appdata\local\programs\python\python311\lib\site-packages (from matplotlib) (0.12.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\soohyunpark\appdata\local\programs\python\python311\lib\site-packages (from matplotlib) (4.43.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\soohyunpark\appdata\local\programs\python\python311\lib\site-packages (from matplotlib) (1.4.5)
Requirement already satisfied: numpy<2,>=1.21 in c:\users\soohyunpark\appdata\local\programs\python\python311\lib\site-packages (from matplotlib) (1.26.0)
Requirement already satisfied: packaging>=20.0 in c:\users\soohyunpark\appdata\local\programs\python\python311\lib\site-packages (from matplotlib) (23.1)
Requirement already satisfied: pillow>=6.2.0 in c:\users\soohyunpark\appdata\local\programs\python\python311\lib\site-packages (from matplotlib) (10.0.1)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\soohyunpark\appdata\local\programs\python\python311\lib\site-packages (from matplotlib) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\soohyunpark\appdata\local\programs\python\python311\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\soohyunpark\appdata\local\programs\python\python311\lib\site-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
PS C:\Users\SooHyunPark> |
```


Example

히스토그램 그리기

```
histogram.py
File Edit Format Run Options Window Help
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 # 평균 55, 표준편차 5를 가지는 난수 1000개를 발생시킴
5 data = 55 + 5 * np.random.randn(1000)
6
7 # histogram of the data
8 num_bins = 10
9 plt.hist(data, num_bins, density = False)
10 plt.show()
11
```



Example

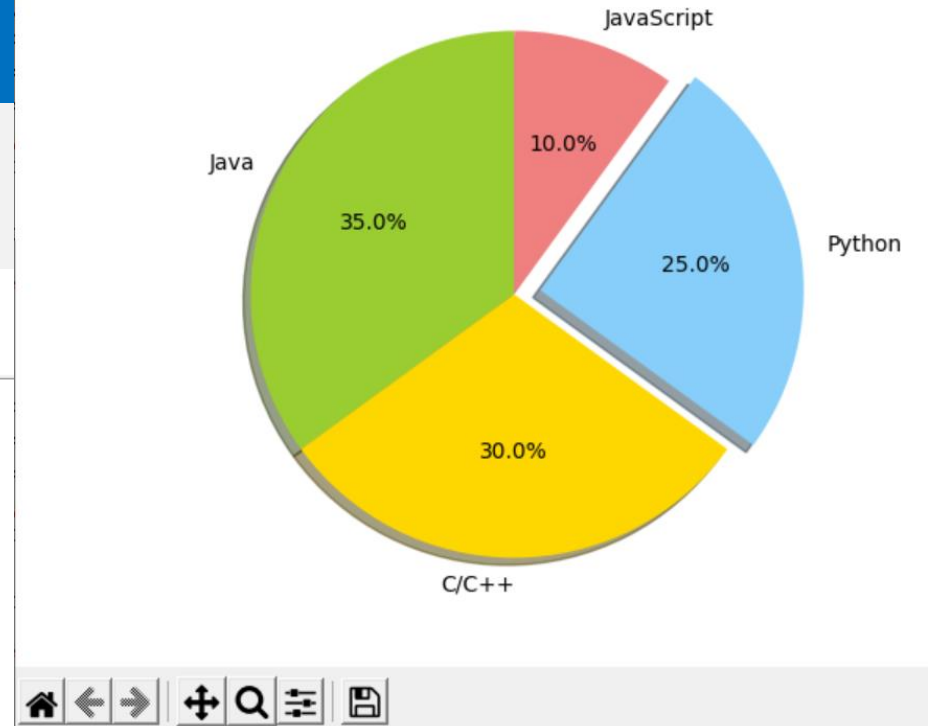
파이차트 만들기

pie-chart.py

File Edit Format Run Options Window Help

```
1 import matplotlib.pyplot as plt
2
3 labels = ['Java', 'C/C++', 'Python', 'JavaScript']
4
5 sizes = [35, 30, 25, 10]
6
7 colors = ['yellowgreen', 'gold', 'lightskyblue', 'lightcoral']
8
9 explode = (0, 0, 0.1, 0) # 2번째 슬라이스만 띄우기
10
11 plt.pie(sizes, explode=explode, labels=labels, W
12         colors=colors, autopct='%1.1f%%', shadow=True, startangle=90)
13
14 plt.axis('equal') # 그림이 찌그러지지 않게 하기
15 plt.show()
16
```

Figure 1



감사합니다

Q & A