Numpy를 이용한 데이터 시각화

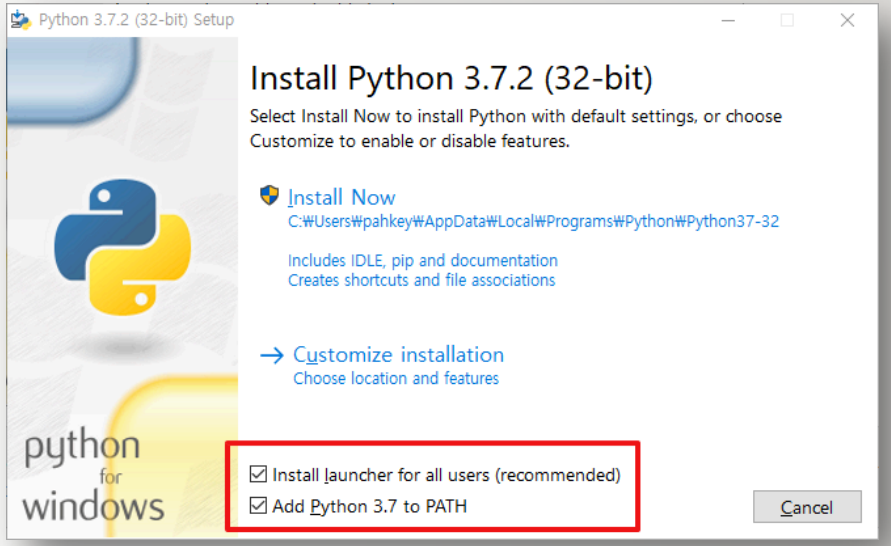
건국대학교 전기전자공학부 기초전자실험 1차시

POWERED BY

python™ NumPy

1 환경 설정

- **Python 3. 7. 3**
  - Windows 이용자
    - 우선 파이썬 공식 홈페이지의 다운로드 페이지(http://www.python.org/downloads)에서 윈도 파이썬 언어 패키지를 다운로드한다. Python 3.x로 시작하는 버전 중 가장 최근의 인스톨러를 다운로드한다.
  - Linux 이용자
    - Terminal에 기본으로 설치되어 있다.
  - macOS 이용자
    - Terminal에 기본으로 설치되어 있으나, 설치되어있지 않은 경우 (http://www.python.org/downloads)에서 macOS용 패키지를 다운로드 및 설치한다.



2 모듈에 대한 이해

- **스크립트** (Script)
  - 프로그램이 길어지면 파이썬 인터프리터 대화창에서 코드 작성하는 것이 유지, 수정 측면에서 힘들다.
  - 인터프리터 창이 아닌 편집기(VS Code, 메모장 등)를 이용해 .py 파일로 코딩한 후에 그 파일을 실행하는 것이 좋은데, 이렇게 하는 것을 "스크립트를 만든다"라고 한다.
- **모듈** (Module)
  - 스크립트 유지를 쉽게 하려고 여러 개의 파일로 나누거나, 여러 프로그램에서 썼던 편리한 함수를 각 프로그램에 사용하고 싶을 수 있다. 이를 위해 파이썬은 정의들을 파일에 넣고 스크립트나 인터프리터의 대화형 모드에서 사용할 수 있는 방법을 제공하는데, 그런 파일을 모듈이라고 한다.

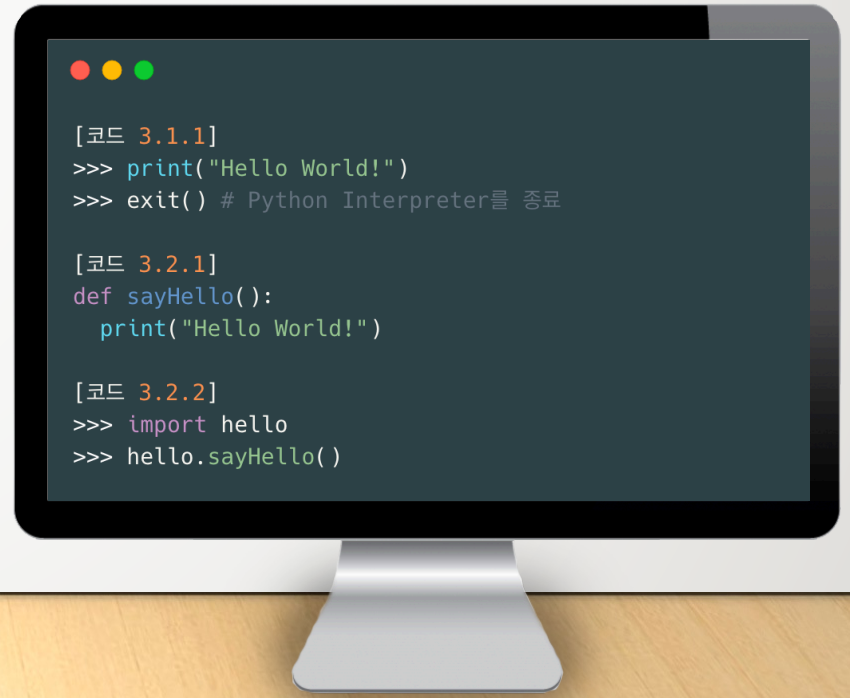
3

“Hello World!” 출력하기

- 인터프리터 대화창에서 출력하기
  - cmd, powershell 등 python 실행가능한 콘솔창에서 'python' 또는 'py'를 친다.
  - [코드 3.1.1]를 작성한다.

- 모듈을 작성하여 출력하기
  - 메모장(또는 기타코드편집기)을 열고 [코드 3.2.1]을 친다.
  - C:\hello.py로 저장한다.
  - cmd, powershell 등 python 실행가능한 콘솔창에서 'python' 또는 'py'를 친다.
  - [코드 3.2.2]를 작성한다.

>>>는 파이썬 인터프리터 대화창을 의미한다.



4

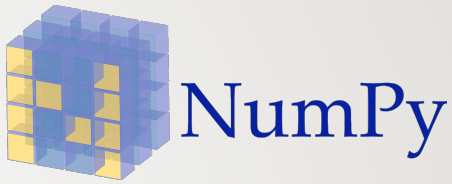
패키지 (Package)

- 모듈이 많아지면 모듈 간 이름이 중복되어 충돌이 일어날 수 있는데, "점으로 구분된 모듈 이름"을 써서 이름 중복을 방지할 수 있는데, 이를 패키지라고 한다.
- 여러 개의 모듈을 하나로 묶어놓은 것이다.
- 패키지 사용자는 다음과 같이 패키지로부터 개별 모듈을 임포트할 수 있다.

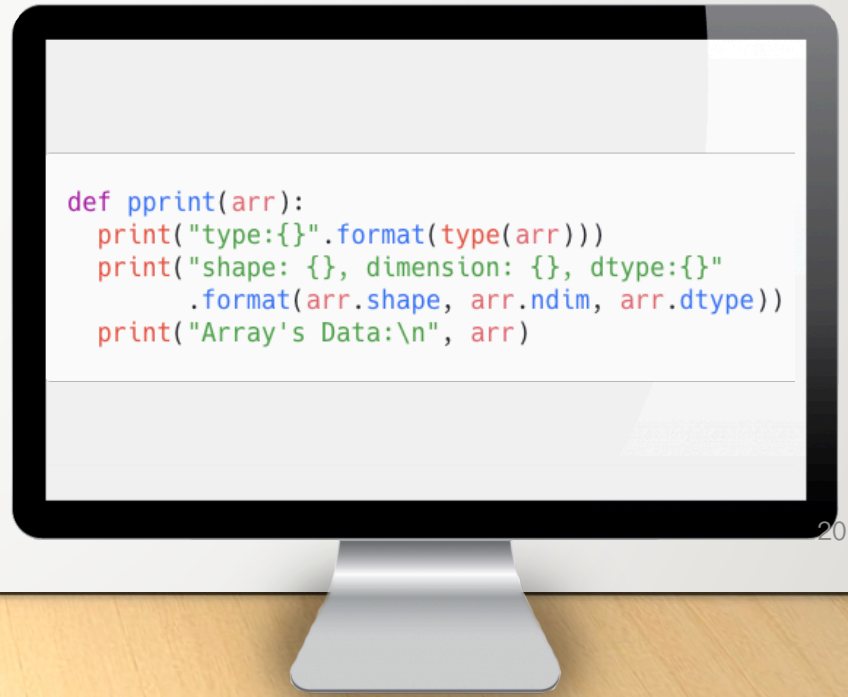
5

Numpy

- Numarray와 Numeric이라는 오래된 패키지를 계승해서 나온 수학 및 과학 연산을 위한 파이썬 패키지
- Py는 파이썬을 나타내기 때문에 "넘파이"라고 읽는다.
- Numpy는 다음과 같은 특징을 갖는다:
  - 강력한 N차원 배열 객체
  - 정교한 브로드캐스팅(Broadcasting) 기능
  - C/C++ 및 포트란 코드 통합 도구
  - 유용한 선형대수, 푸리에 변환 및 난수 기능
  - 범용적 데이터 처리에 사용 가능한 다차원 컨테이너
- NumPy를 사용하기 위해 다음과 같이 numpy 모듈을 "np"로 임포트하여 사용한다.



결과값을 쉽게 확인하기 위해
앞으로 다음 함수를 작성하여 사용한다.



```
def pprint(arr):
  print("type:{}".format(type(arr)))
  print("shape: {}, dimension: {}, dtype:{}"
        .format(arr.shape, arr.ndim, arr.dtype))
  print("Array's Data:\n", arr)
```

6

파이썬 배열로 Numpy 배열 생성

- 파이썬 배열을 인자로 NumPy 배열을 생성할 수 있다. 파라미터로 list 객체와 데이터타입(dtype)을 입력하여 NumPy 배열을 생성한다. dtype를 생략할 경우, 입력된 list 객체의 요소 타입이 설정된다.

실습 01 해당 코드를 실행 후 결과값을 확인하시오

1차원 List로 배열 생성

```
>>> arr = [1, 2, 3]
>>> a = np.array([1, 2, 3])
>>> pprint(a)
```

```
type:<class 'numpy.ndarray'>
shape: (3,), dimension: 1, dtype:int64
Array's Data:
[1 2 3]
```

2차원 배열 생성,
원소 데이터타입 지정

```
>>> arr = [(1, 2, 3), (4, 5, 6)]
>>> a = np.array(arr, dtype = float)
>>> pprint(a)
```

```
type:<class 'numpy.ndarray'>
shape: (2, 3), dimension: 2, dtype:float64
Array's Data:
[[ 1.  2.  3.]
 [ 4.  5.  6.]]
```

3차원 배열 생성, 원소 데이터타입 지정

```
>>> arr = np.array([[[1,2,3], [4,5,6]], [[3,2,1], [4,5,6]]], dtype = float)
>>> a= np.array(arr, dtype = float)
>>> pprint(a)
```

```
type:<class 'numpy.ndarray'>
shape: (2, 2, 3), dimension: 3, dtype:float64
Array's Data:
[[[ 1.  2.  3.]
  [ 4.  5.  6.]]

 [[ 3.  2.  1.]
  [ 4.  5.  6.]]]
```

배열 생성 및 초기화 (1)

- NumPy는 원하는 shape로 배열을 설정하고, 각 요소를 특정 값으로 초기화하는 zeros, ones, full, eye 함수를 제공한다.
- 또한 파라미터로 입력한 배열과 같은 shape의 배열을 만드는 zeros_like, ones_like, full_like 함수도 제공한다. 이 함수를 이용하여 배열을 생성하고 초기화할 수 있다.

```
np.empty(shape, dtype=float, order='C')
```

- 지정된 shape의 배열 생성
- 요소의 초기화 과정에 없고, 기존 메모리값을 그대로 사용
- 배열 생성비용이 가장 저렴하고 빠름
- 배열 사용 시 주의가 필요(초기화를 고려)

```
np.zeros(shape, dtype=float, order='C')
```

- 지정된 shape의 배열을 생성하고, 모든 요소를 0으로 초기화

```
np.ones(shape, dtype=None, order='C')
```

- 지정된 shape의 배열을 생성하고, 모든 요소를 1로 초기화

```
np.full(shape, fill_value, dtype=None, order='C')
```

- 지정된 shape의 배열을 생성하고, 모든 요소를 지정한 "fill_value"로 초기화

```
np.eye(N, M=None, k=0, dtype=<class 'float'>)
```

- (N, N) shape의 단위 행렬(Unit Matrix)을 생성

실습 02 해당 코드를 실행 후 결과값을 확인하시오

```
>>> a = np.zeros((3, 4))
>>> pprint(a)
```

```
>>> a = np.ones((2,3,4),dtype=np.int16)
>>> pprint(a)
```

```
>>> a = np.full((2, 2), 7)
>>> pprint(a)
```

```
>>> a = np.empty((4, 2))
>>> pprint(a)
```

```
>>> np.eye(4)
```

8

배열 생성 및 초기화 (2)

- empty() 함수
  - 지정된 shape의 배열 생성
  - 요소의 초기화 과정에 없고, 기존 메모리값을 그대로 사용
  - 배열 생성비용이 가장 저렴하고 빠름
  - 배열 사용 시 주의가 필요(초기화를 고려)

```
np.empty(shape, dtype=float, order='C')
```

- numpy는 지정된 배열과 shape이 같은 행렬을 만드는 like 함수를 제공한다.
  - np.zeros_like()
  - np.ones_like()
  - np.full_like()
  - np.empty_like()

실습 03 해당 코드를 실행 후 결과값을 확인하시오

```
a = np.empty((4, 2))
pprint(a)
```

```
type:<class 'numpy.ndarray'>
shape: (4, 2), dimension: 2, dtype:float64
Array's Data:
[[  0.00000000e+000   6.91240343e-310]
[  6.91240500e-310   5.39088070e-317]
[  5.39084907e-317   6.91239798e-310]
[  3.16202013e-322   6.91239798e-310]]
```

```
a = np.array([[1,2,3], [4,5,6]])
b = np.ones_like(a)
pprint(b)
```

```
type:<class 'numpy.ndarray'>
shape: (2, 3), dimension: 2, dtype:int64
Array's Data:
[[1 1 1]
[1 1 1]]
```

9

데이터 생성 함수

- NumPy는 다음과 같이 주어진 조건으로 데이터를 생성한 후, 배열을 만드는 데이터 생성 함수를 제공한다.
  - np.linspace()
  - np.arange()
  - np.logspace()

```
np.linspace(start, stop, num=50, endpoint=True, retstep=False, dtype=None)
```

- start부터 stop의 범위에서 num개를 균일한 간격으로 데이터를 생성하고 배열을 만드는 함수
- 요소 개수를 기준으로 균등 간격의 배열을 생성

```
np.arange([start,] stop[, step,], dtype=None)
```

- start부터 stop 미만까지 step 간격으로 데이터 생성한 후 배열을 만듦
- 범위내에서 간격을 기준 균등 간격의 배열
- 요소의 객수가 아닌 데이터의 간격을 기준으로 배열 생성

```
np.logspace(start, stop, num=50, endpoint=True, base=10.0, dtype=None)
```

- 로그 스케일의 linspace 함수
- 로그 스케일로 지정된 범위에서 num 개수만큼 균등 간격으로 데이터 생성한 후 배열 만듦

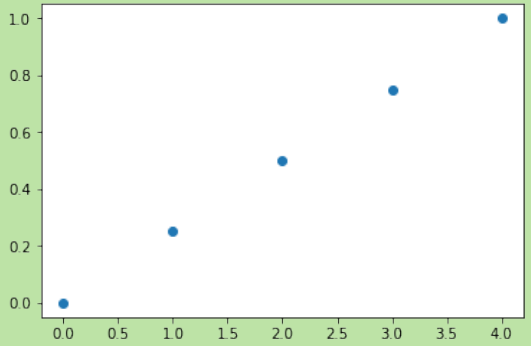
실습 04 다음과 같은 방법대로 값을 확인하시오

1. linspace(0, 1, 5) 의 값을 확인한다.

```
>>> a = np.linspace(0, 1, 5)
>>> pprint(a)
```

2. 다음 코드를 실행하여 시각화된 그래프를 확인한다.

```
# linspace의 데이터 추출 시각화
import matplotlib.pyplot as plt
plt.plot(a, 'o')
plt.show()
```



3. arange(), logspace()에 대해서도 같은 방법으로 그래프를 확인한다.

```
# linspace의 데이터 추출 시각화
import matplotlib.pyplot as plt
plt.plot(a, 'o')
plt.show()
```

10

난수 기반 배열 생성

- NumPy는 난수 발생 및 배열 생성을 생성하는 numpy.random 모듈을 제공한다.
- numpy.random 모듈은 다음과 같은 함수를 제공한다:
  - np.random.normal()
  - np.random.rand()
  - np.random.randn()
  - np.random.randint()
  - np.random.random()

```
np.random.normal(loc=0.0, scale=1.0, size=None)
```

- 정규 분포 확률 밀도에서 표본 추출
- loc: 정규 분포의 평균
- scale: 표준편차

실습 05 해당 코드를 실행 후 결과값을 확인하시오

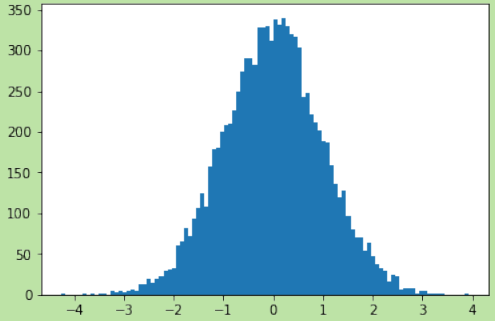
```
>>> mean = 0
>>> std = 1
>>> a = np.random.normal(mean, std, (2, 3))
>>> pprint(a)
```

```
type:<class 'numpy.ndarray'>
shape: (2, 3), dimension: 2, dtype:float64
Array's Data:
[[ 1.4192442  -2.0771293   1.84898108]
[-0.12303317  1.04533993  1.94901387]]
```

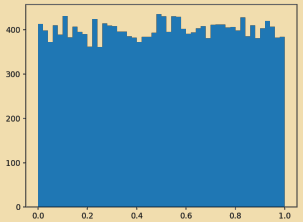
- np.random.normal()이 생성한 난수는 정규 분포의 형상을 갖는다.
- 다음 예제는 정규 분포로 10000개 표본을 뽑은 결과를 히스토그램으로 표현한 예다.
- 표본 10000개의 배열을 100개 구간으로 구분할 때,정규 분포 형태를 보이고 있다.

랜덤 데이터의 히스토그램 확인

```
>>> data = np.random.normal(0, 1, 10000)
>>> import matplotlib.pyplot as plt
>>> plt.hist(data, bins=100)
>>> plt.show()
```



과제 01 해당 문제를 해결하여 레포트에 반영하시오



1. 다음 np.random.rand() 레퍼런스를 참고하여 20000개의 표본을 뽑아 50개의 구간으로 구분한 임의 난수 히스토그램을 그리고, 출력 결과를 해석하시오.

https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand.html

2. 다음 np.random.randint() 레퍼런스를 참고하여 0에서 5까지의 5x5 정수 임의 배열을 생성하시오. 꼬한 10000개의 표본을 뽑아 임의 난수 히스토그램을 그리고, 출력 결과를 해석하시오.

https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.randint.html

Numpy 입출력

- Numpy는 배열 객체를 바이너리 파일 혹은 텍스트 파일에 저장하고 로딩하는 기능을 제공한다.
- 입출력을 위한 함수는 다음과 같다:

함수명	기능	파일포맷
np.save()	NumPy 배열 객체 1개를 파일에 저장	바이너리
np.savez()	Numpy 배열 객체 복수개를 파일에 저장	바이너리
np.load()	Numpy 배열 저장 파일로부터 객체 로딩	바이너리
np.loadtxt()	텍스트 파일로부터 배열 로딩	텍스트
np.savetxt()	텍스트 파일에 NumPy 배열 객체 저장	텍스트

실습 06 해당 코드를 실행 후 결과값을 확인하시오

```
a = np.random.randint(0, 10, (2, 3))
b = np.random.randint(0, 10, (2, 3))
pprint(a)
pprint(b)
# a 배열 파일에 저장
np.save("./my_array1", a)
# a, b 두 개의 배열을 파일에 저장
np.savez("my_array2", a, b);
# 1개 배열 로딩
np.load("./my_array1.npy")
# 복수 파일 로딩
npzfiles = np.load("./my_array2.npz")
npzfiles.files
npzfiles['arr_0']
npzfiles['arr_1']
```

```
type:<class 'numpy.ndarray'>
shape: (2, 3), dimension: 2, dtype:int64
Array's Data:
[[8 7 4]
[4 3 8]]
type:<class 'numpy.ndarray'>
shape: (2, 3), dimension: 2, dtype:int64
Array's Data:
[[6 9 8]
[9 7 7]]
array([[8, 7, 4],
[4, 3, 8]])
['arr_0', 'arr_1']
array([[8, 7, 4],
[4, 3, 8]])
array([[6, 9, 8],
[9, 7, 7]])
```

배열 상태 검사 Inspecting

- NumPy는 배열의 상태를 검사하는 다음과 같은 방법을 제공한다.

배열 속성 검사 항목	배열 속성 확인 방법	예시	결과
배열 shape	np.ndarray.shape 속성	arr.shape	(5, 2, 3)
배열 길이	일차원의 배열 길이 확이	len(arr)	5
배열 차원	np.ndarray.nsim 속성	arr.ndim	3
배열 요소 수	np.ndarray.size 속성	arr.size	30
배열 타입	np.ndarray.dtype 속성	arr.dtype	dtype ('float64')
배열 타입 명	np.ndarray.dtype.name 속성	arr.dtype.name	float64
배열 타입 변환	np.ndarray.astype 함수	np.ndarray.astype	arr.astype (np.int)

실습 07 해당 코드를 실행 후 결과값을 확인하시오

- NumPy 배열 객체는 다음과 같은 방식으로 속성을 확인할 수 있다.

```
# 데모 배열 객체 생성
arr = np.random.random((5, 2, 3))
# 배열 타입 조회
type(arr)
# 배열의 shape 확인
arr.shape
# 배열의 길이
len(arr)
# 배열의 차원 수
arr.ndim
# 배열의 요소 수: shape(k, m, n) ==> k*m*n
arr.size
# 배열 타입 확인
arr.dtype
# 배열 타입 명
arr.dtype.name
# 배열 요소를 int로 변환
# 요소의 실제 값이 변환되는 것이 아님
# View의 출력 타입과 연산을 변환하는 것
arr.astype(np.int)
# np.float으로 타입을 다시 변환하면 np.int 변환 이전 값으로 모든 원소 값이 복원됨
arr.astype(np.float)
```

데이터 타입

- NumPy는 다음과 같은 데이터 타입을 지원한다. 배열을 생성할 때 dtype속성으로 다음과 같은 데이터 타입을 지정할 수 있다.
  - `np.int64` : 64 비트 정수 타입
  - `np.float32` : 32 비트 부동 소수 타입
  - `np.complex` : 복소수 (128 float)
  - `np.bool` : 불린 타입 (True, False)
  - `np.object` : 파이썬 객체 타입
  - `np.string\_` : 고정자리 스트링 타입
  - `np.unicode\_` : 고정자리 유니코드 타입

과제 02 해당 문제를 해결하여 레포트에 반영하시오

1. 0부터 30까지 임의 실수의 난수를 생성하는 5x5 배열을 assignment2.npy에 저장하고, getArray("assignment2.npy") 호출 시 다음과 같이 출력하는 함수 getArray를 작성하시오.

```
def getArrray(filename)
```

- 입력: [string] .npy 형식 파일명
- 출력

```
-------- 배열 정보 --------
타입: ndarray
길이: 5
요소 수: 25
배열 타입: int64
------------------------
array([[17, 29, 15, 10, 27],
[18, 12, 2, 10, 26],
[29, 5, 12, 17, 10],
[ 8, 16, 25, 26, 10],
[18, 29, 26, 19, 13]])
```

2. `np.random.random()`을 이용하여 4x4 배열을 생성하고, 1번 `getArray()`함수를 이용하여 배열 정보를 출력하시오. 그리고 배열의 모든 값에 100을 곱해주고, `np.int`형으로 변환한 뒤 배열 정보와 값을 확인하시오.

도움말

- NumPy의 모든 API는 np.info 함수를 이용하여 도움말을 확인할 수 있다.

실습 08 다음과 같은 방법대로 값을 확인하시오

np.squeeze에 관한 도움말 얻기

```
np.info(np.squeeze)
```

```
squeeze(a, axis=None)

Remove single-dimensional entries from the shape of an array.

Parameters
----------
a : array_like
  Input data.
axis : None or int or tuple of ints, optional
  .. versionadded:: 1.7.0

...skipped
```

np.ndarray.dtype에 관한 도움말 얻기

```
np.info(np.ndarray.dtype)
```

```
Data-type of the array's elements.

Parameters
----------
None

Returns
-------
d : numpy dtype object

See Also
--------
numpy.dtype

Examples
--------
>>> x
array([[0, 1],
   [2, 3]])
>>> x.dtype
dtype('int32')
>>> type(x.dtype)
<type 'numpy.dtype'>
```