NumPy

#01. NumPy 개요

- 다차원 배열을 쉽게 처리하고 효율적으로 사용할 수 있도록 지원하는 파이썬 패키지
- 데이터 구조 외에도 수치 계산을 위해 효율적으로 구현된 기능을 제공
- 데이터 분석에서 Pandas와 함께 자주 사용하는 도구

1) NumPy의 역할

- 실제로 데이터 분석을 수행하기 위한 전제 조건은 컴퓨터가 이해할 수 있도록 데이터를 숫자 형식으로 변환하는 것임
- 파이썬의 list 자료형의 경우 데이터의 크기가 커질수록 저장 및 가공에 효율성을 보장하지 못함
- 이러한 단점을 보완하기 위한 패키지이기 때문에 Data Science에서 핵심적인 도구로 인식되고 있음

2) NumPy 설치하기

```
$ pip install --upgrade numpy
```

3) NumPy 패키지 가져오기

```
import numpy
```

#02. Numpy 패키지 사용하기

1) numpy 배열 생성과 기본 활용

```
# 리스트를 통한 1차원(=1행으로 구성) 배열 만들기
arr = numpy.array([1,3,5,7,9])
arr
```

```
# 배열의 크기와 각 원소에 접근하기
# -> 내장함수 len()은 모든 연속성 데이터(문자열,리스트,튜플 등)에 사용 가능
size = len(arr)
print("배열의 원소는 %d개 입니다." % size)
```

```
# 배열의 원소에 접근하기
#-> 리스트와 마찬가지로 각 원소에 인덱스 번호로 접근 가능.
print(arr[0])
print(arr[1])
print(arr[3])
```

```
# 인덱스가 있기 때문에 반복문을 통해서 제어할 각 원소에 접근 가능
for i, v in enumerate(arr):
```

```
    print("%d번째 원소 >> %d" % (i, v))
```

2) numpy 배열의 특성

```
# 서로 다른 타입의 원소를 갖는 list만들기
# -> 파이썬의 리스트는 서로 다른 타입 허용
arr2 = [1.2, 3, '4']
arr2
```

```
# 정수와 실수가 섞인 리스트를 배열로 변환
# -> 배열은 원소의 타입이 서로 다른것을 허용하지 않음
# -> 가장 포괄적인 형태의 자료형으로 통일함
# -> 여기서는 실수가 범위가 더 크므로 모든 원소가 실수형으로 변환됨
arr3 = numpy.array( [1, 2.4, 3, 4.6] )
arr3
```

```
# 정수,실수,문자열이 포함된 리스트를 배열로 변환
# -> 모든 타입이 문자열로 변환되어 있음
arr4 = numpy.array([1.2, 3, '4'])
arr4
```

```
# 모든 원소의 타입을 강제로 int(정수)로 지정
# -> 소수점 아래 값들은 모두 버려진다.
arr5 = numpy.array( [1, 2.4, 3, 4.6], dtype='int' )
arr5
```

3) numpy 배열의 기초 통계값

```
# 예제를 위한 배열 구성
grade = numpy.array([82, 77, 91, 88])
grade
```

```
# 모든 원소의 합
s1 = numpy.sum(grade)
print("총점: %d" % s1)
```

```
# 모든 원소의 평균
s2 = numpy.average(grade)
print("평균 : %d" % s2)
```

```
# 최대, 최소값
s3 = numpy.max(grade)
s4 = numpy.min(grade)
print("최대값 : %d, 최소값: %d" % (s3, s4) )
```

4) numpy 배열의 각 원소에 대한 연산

```
# 모든 원소에 대한 사칙연산 수행
#-> 모든 원소에 대하여 2씩 더함
# 원본: [82 77 91 88]
new1 = grade + 2
new1
```

```
#-> 모든 원소에 대하여 5씩 뺌
new2 = grade - 5
new2
```

5) numpy 배열끼리의 연산

```
# 연산자를 사용한 배열간의 연산 => 위치가 동일한 각 원소끼리 수행된다.
arr1 = numpy.array([10,15,20,25,30])
arr2 = numpy.array([2,3,4,5,6])
print(arr1)
print(arr2)
```

```
a = arr1 + arr2
a
```

```
b = arr1 - arr2
b
```

```
c = arr1 * arr2
c
```

```
d = arr1 / arr2
d
```

```
# numpy모듈의 함수를 사용한 배열간의 연산 => 연산자와 동일한 결과
arr1 = numpy.array([10,15,20,25,30])
arr2 = numpy.array([2,3,4,5,6])
print(arr1)
print(arr2)
```

```
a = numpy.add(arr1, arr2)
a
```

```
b = numpy.subtract(arr1, arr2)
b
```

```
c = numpy.multiply(arr1, arr2)
```

```
c
```

```
d = numpy.divide(arr1, arr2)
d
```

6) numpy 배열의 기본 인덱싱, 슬라이싱

```
# 철수의 1학년~4학년까지의 평균 점수
grade = numpy.array([82, 77, 91, 88])
grade
```

```
# 인덱싱
# -> 2열(0부터 카운트)의 데이터 접근
grade[2]
```

```
# 슬라이싱
# -> 1열부터 3열 전까지 범위를 추출
grade[1:3]
```

```
# -> 처음부터 2열 전까지 범위를 추출
grade[:2]
```

```
# -> 1열~끝까지 범위를 추출
grade[1:]
```

7) 조건에 맞는 값 추출하기

```
# -> 추출하고자 하는 원본과 같은 사이즈의 배열을 생성한다.
# -> 추출할 값은 True, 그렇지 않으면 False
bool_array = numpy.array([True, False, True, False])
bool_array
```

```
# -> 조건에 맞는 항목만 1차 배열로 추출
# [82 77 91 88]
result1 = grade[bool_array]
result1
```

```
# 80점 이상인지 판별된 조건에 맞는 데이터만 추려냄
result2 = grade[grade >= 80]
result2
```

```
# logical_and 함수를 사용하여 80점 이상이고 90점 이하인 데이터만 추려냄
result3 = grade[numpy.logical_and(grade >= 80, grade <= 90)]
result3
```

```
# logical_or 함수를 사용하여 80점 미만이거나 90점 초과인 데이터만 추려냄
result4 = grade[numpy.logical_or(grade < 80, grade > 90)]
result4
```

#03. 2차 배열

1) numpy 2차 배열 생성 및 기본 정보 조회

```
# 철수의 학년-과목별 점수
grade = numpy.array([
    # 열 ->   0   1   2   3
            [98, 72, 80, 64],   # 0행 - kor
            [88, 90, 80, 72],   # 1행 - eng
            [92, 88, 82, 76]    # 2행 - math
        ])
grade
```

```
# 차원 크기
grade.ndim
```

```
# 각 차원의 원소수
grade.shape
```

```
# 각 원소의 타입
grade.dtype
```

2) 기본 인덱싱과 슬라이싱

```
# 정수형 인덱싱 -> 1행 2열의 데이터 접근 방법(1)
grade[1,2]
```

```
# 정수형 인덱싱 -> 1행 2열의 데이터 접근 방법(2)
grade[1][2]
```

```
# 슬라이싱
# -> 1~3행 전까지,1~4열 전까지 범위를 추출
grade[1:3, 1:4]
```

```
# -> 0~2행 전까지, 0~3열 전까지 범위를 추출
grade[:2, :3]
```

```
# -> 1~끝행, 2~끝열 범위를 추출
grade[1:, 2:]
```

3) 기초 통계 산출 (max,min,sum,average)

```
# 각 열끼리 더함 (세로로 덧셈) -> 결과 타입은 numpy의 1차 배열
s1 = numpy.sum(grade, axis=0)
print(type(s1))
s1
```

```
# 각 행끼리 더함 (가로로 덧셈)
s2 = numpy.sum(grade, axis=1)
type(s2)
s2
```

4) 조건에 맞는 값 추출하기

```
# bool 형 인덱싱
# -> 추출하고자 하는 원본과 같은 사이즈의 배열
# -> 추출할 값은 True, 그렇지 않으면 False
bool_array = numpy.array([
            [True, False, True, False],
            [True, True, True, False],
            [True, True, True, False]
        ])

# 조건에 맞는 항목만 1차 배열로 추출
result1 = grade[bool_array]
result1
```

```
# 80점 이상인지 판별하여 조건에 맞는 데이터만 추려냄
# -> 추출된 결과는 1차 배열
result2 = grade[grade >= 80]
result2
```

```
# logical_and 함수를 사용하여 80점 이상이고 90점 이하인 데이터만 추려냄
result3 = grade[numpy.logical_and(grade >= 80, grade <= 90)]
result3
```

```
# logical_or 함수를 사용하여 80점 미만이거나 90점 초과인 데이터만 추려냄
result4 = grade[numpy.logical_or(grade < 80, grade > 90)]
result4
```