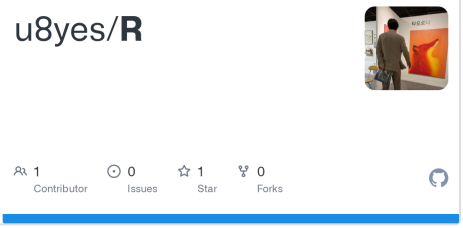
Day43; 20221107(R 보충수업 & python - AI Start)

날짜	@2022년 11월 7일
유형	@2022년 11월 7일
태그	

GitHub - u8yes/R

You can't perform that action at this time. You signed in with another tab or window. You signed out in another tab or window. Reload to refresh your session. Reload to refresh your session.

https://github.com/u8yes/R



https://s3-us-west-2.amazonaws.com/secure.notion-static.com/a769a394-e2e5-4f79-a332-55172e703f73/09._%EC%A0%95%ED%98%95%EA%B3%BC_%EB%B9%84%EC%A0%95%ED%98%95_%EB%8D%B0%EC%9D%B4%ED%84%B0__%EC%9B%B9%ED%81%AC%EB%A1%A4%EB%A7%81.pdf

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/ac728efa-3b56-40cf-a3bd-6329c25db8c6/chap09_FormalInformal.r

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/e0cf093a-fac9-4c06-b90e-5699ad708092/01_%EB%A8%B8%EC%8B%A0%EB%9F%AC%EB%8B%9D_%EA%B0%9C%EB%85%90.pdf

```
#  (3) 파싱(parsing)
#   - 어떤 페이지(문서, HTML등)에서 사용자가 원하는 데이터를 특정 패턴이나 순서로 추출하여 정보를 가공하는 것.
#   - 예를들면 HTML 소스를 문자열로 수집한 후 실제 HTML 태그로 인식할 수 있도록 문자열을 의미있는 단위로 분해하고, 계층적인 트리 구조를 만드는 과정.
```

JAVA) 포장 클래스로 박싱, 포장에서 기본 타입으로 언박싱

기본 타입	포장 클래스
byte	Byte
char	Character
short	Short
int	Integer
long	Long
float	Float
double	Double
boolean	Boolean

■ **문자열 변환(valueOf())**

valueOf() 메소드는 기본 타입의 값을 문자열로 변환하는 기능을 가지고 있습니다. String 클래스에는 매개 변수의 타입별로 valueOf() 메소드가 다음과 같이 오버로딩되어 있습니다.

박싱(Boxing)과 언박싱(Unboxing)

기본 타입의 값을 포장 객체로 만드는 과정을 박싱Boxing이라고 하고, 반대로 포장 객체에서 기본 타입의 값을 얻어내는 과정을 언박싱Unboxing이라고 합니다.

다음은 8개의 기본 타입의 값을 박싱하는 방법을 보여주고 있습니다. 간단하게 포장 클래스의 생성자 매개값으로 기본 타입의 값 또는 문자열을 넘겨주면 됩니다.

기본 타입의 값을 줄 경우	문자열을 줄 경우
Byte obj = new Byte(10);	Byte obj = new Byte("10");
Character obj = new Character('가');	없음
Short obj = new Short(100);	Short obj = new Short("100");
Integer obj = new Integer(1000);	Integer obj = new Integer("1000");
Long obj = new Long(10000);	Long obj = new Long("10000");
Float obj = new Float(2.5F);	Float obj = new Float("2.5F");
Double obj = new Double(3.5);	Double obj = new Double("3.5");
Boolean obj = new Boolean(true);	Boolean obj = new Boolean("true");

```
Integer obj = Integer.valueOf(1000);
Integer obj = Integer.valueOf("1000");
```

이렇게 박싱된 포장 객체에서 다시 기본 타입의 값을 얻어내기 위해서는(언박싱하기 위해서는) 각 포장 클래스마다 가지고 있는 '기본 타입 이름+Value()' 메소드를 호출하면 됩니다.

기본 타입의 값을 이용			
byte	num	=	obj.byteValue();
char	ch	=	obj.charValue();
short	num	=	obj.shortValue();
int	num	=	obj.intValue();
long	num	=	obj.longValue();
float	num	=	obj.floatValue();
double	num	=	obj.doubleValue();
boolean	bool	=	obj.booleanValue();

자동 박싱과 언박싱

기본 타입 값을 직접 박싱, 언박싱하지 않아도 자동적으로 박싱과 언박싱이 일어나는 경우가 있습니다. 자동 박싱은 포장 클래스 타입에 기본값이 대입될 경우에 발생합니다. 예를 들어 int 타입의 값을 Integer 클래스 변수에 대입하면 자동 박싱이 일어나 힙 영역에 Integer 객체가 생성됩니다.

```
Integer obj = 100;    //자동 박싱
```

자동 언박싱은 기본 타입에 포장 객체가 대입되는 경우와 연산에서 발생합니다. 예를 들어 Integer 객체를 int 타입 변수에 대입하거나, Integer 객체와 int 값을 연산하면 Integer 객체로부터 int 값이 자동 언박싱되어 연산됩니다.

```
Integer obj = new Integer(200);
int value1 = obj;          //자동 언박싱
int value2 = obj + 100;    //자동 언박싱
```

직접 해보는 손코딩

자동 박싱과 언박싱 소스 코드 AutoBoxingUnBoxingExample.java

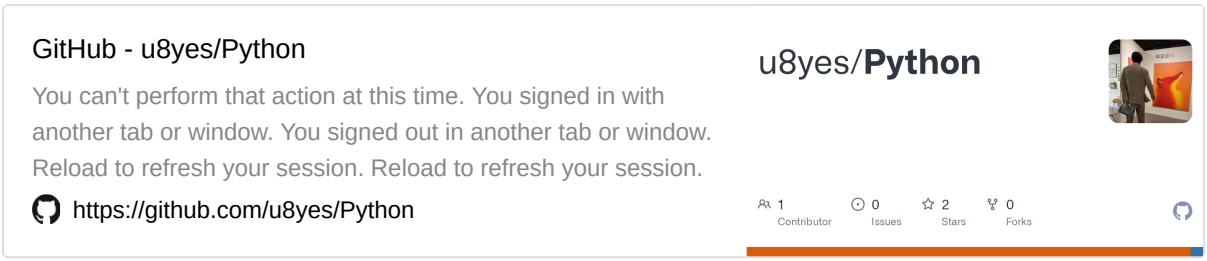
```
01  package sec01.exam23;
02
03  public class AutoBoxingUnBoxingExample {
04    public static void main(String[] args) {
05      //자동 박싱
06      Integer obj = 100;
07      System.out.println("value: " + obj.intValue());
08
09      //대입 시 자동 언박싱
10      int value = obj;
11      System.out.println("value: " + value);
12
13      //연산 시 자동 언박싱
14      int result = obj + 100;
15      System.out.println("result: " + result);
16    }
17  }
```

실행결과

```
value: 100
value: 100
result: 200
```

```
558 # 단계 1: 패키지 로딩과 단어 이름 추출
559 library(wordcloud) # EDA를 대표적으로 시각 표현해줌
560 myNames <- names(wordResult)
561 myNames
562
563
564 # 단계 2: 단어와 단어 빈도수 구하기
565 df <- data.frame(word = myNames, freq = wordResult)
566 head(df)
567
568 # 단계 3: 단어 구름 생성
569 pal <- brewer.pal(12, "Paired")
570 x11()
571 wordcloud(df$word, df$freq, min.freq = 1,
572           random.order = F, scale = c(4, 0.7),
573           rot.per = .1, colors = pal)
574 # min.freq = 1 #1번 언급된 것
575
```

모양은 동그라미, 네모 매번 다름.

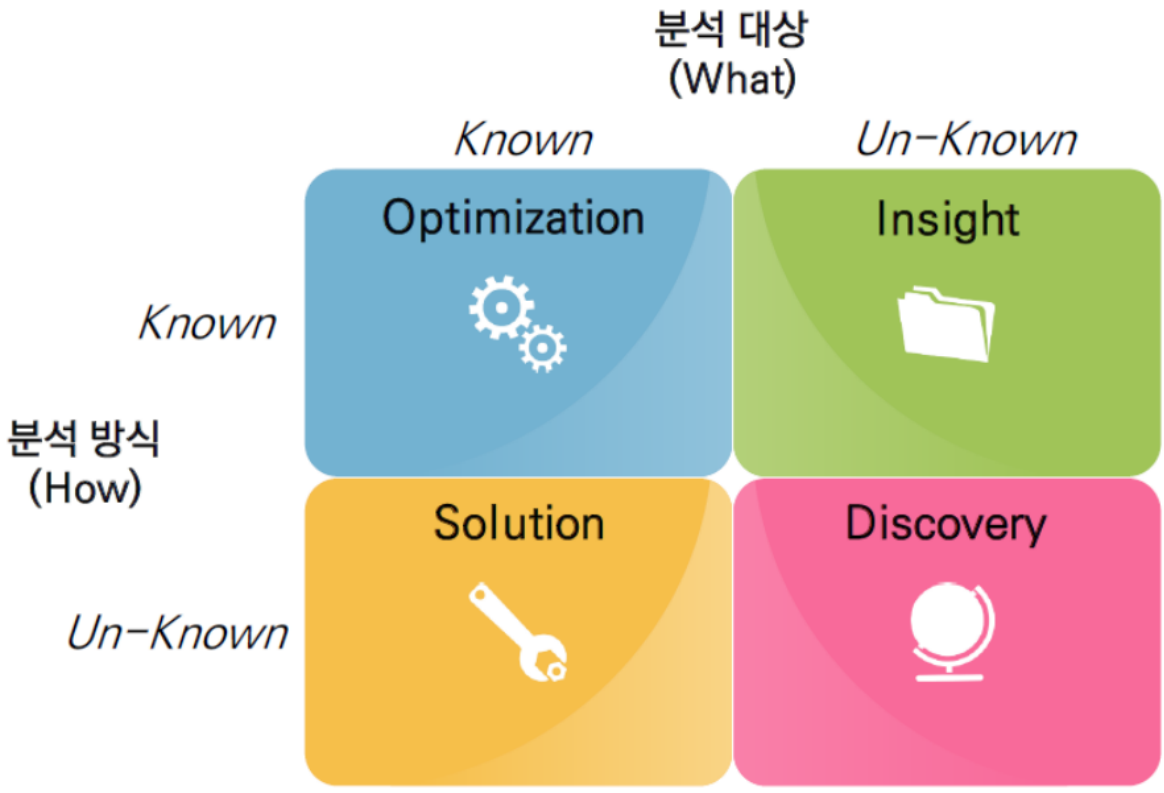


https://s3-us-west-2.amazonaws.com/secure.notion-static.com/14ce4ea5-02d4-4c4c-870f-05adde430686/01_%EB%A8%B8%EC%8B%A0%EB%9F%AC%EB%8B%9D_%EA%B0%9C%EB%85%90.pdf

---

(분석)방(식)알(지)못(할때)솔(루션).

Insight 분석 대상을 모르기에 전처리의 역할이 크다는 것. (R에서 전처리를 통해 많이 살펴봄)



Solution과 Discovery는 교수진들을 통해서 끊임없이 연구 돼지고 있음.

Optimization은 우리가 앞으로 살펴볼 것들.

R에서 데이터 프레임의 형태를 파이썬에서 보여주는 것이 Numpy 패키지

머신러닝, 딥러닝(2차원 배열의 중요성 대두)

시각화(이산형&범주형)

암기약자) 파돗파돗바

```
# 세로 막대 차트
help("barplot")
help(barplot)
barplot(chart_data, y
2020년도 분기별 매출현
```

```
#  2) 점 차트 시각화 - dotchart()
help("dotchart")

par(mfrow=c(1,1)) # 1행1열 그래프 보

dotchart(chart_data, color = c("bla
         xlab = "매출액(단위:억원)"
```

```
#  3) 원형 차트 시각화 - pie() 함수
help(pie)

pie(chart_data, labels = names(chart_data),
    border = 'blue', col=rainbow(8), cex=4)
title("2019~2020년도 분기별 매출현황")
```

시각화(연속형)

플박히

```
# 2. 연속변수(Continuous quantitative data) 시각화 # Day25; 20221011
#   - 시간, 길이 등과 같은 연속성을 가진 변수.

#  1) 상자 그래프 시각화 : 요약정보를 시각화하는데 효과적. 특히 데이터의 분포
정도와 이상치 발견을 목적으로 하는 경우 유용.
help(boxplot)
par(mfrow=c(1,2))
boxplot(VADeaths) # 상자그래프 시각화.
boxplot(VADeaths, range=0)
# range=0:최소값과 최대값 사이를 점선으로 연결하는 역할.
```

```r
#  2) 히스토그램 시각화
#    - 측정값의 범위(구간)를 그래프의 x축으로 놓고,
빈도수를 y축으로 나타낸 그래프 형태.

data("iris") # iris 데이터 셋 가져오기
head(iris)
table(iris$Species)
#setosa versicolor  virginica
#    50         50         50

names(iris)
# "Sepal.Length" "Sepal.Width"  "Petal.Length" "Pet

summary(iris$Sepal.Width) # 기술 통계량

hist(iris$Sepal.Width, xlab = "꽃받침의 너비",
     col="green", xlim=c(2.0, 4.5),
```

```r
#  3) 산점도 시각화
#    - 두 개 이상의 변수들 사이의 분포를 점으로 표시한 차트를 의미.

# 기본 산점도 시각화
price <- runif(10, min = 1, max = 100) # 1~100 사이의 10개 난수 발생.
price # 난수를 확인 가능
plot(price)

# 대각선 추가
par(new=T) # 차트 추가
line_chart <- c(1:100)
line_chart
plot(line_chart, type = "l", col="red", axes = F, ann = F)
```

고급 시각화

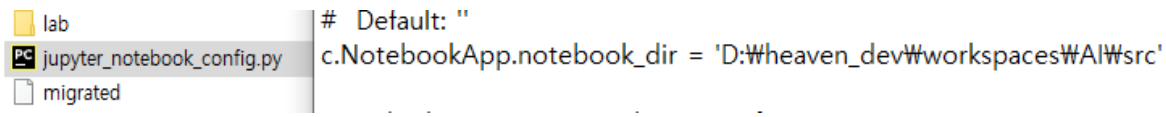
지지라

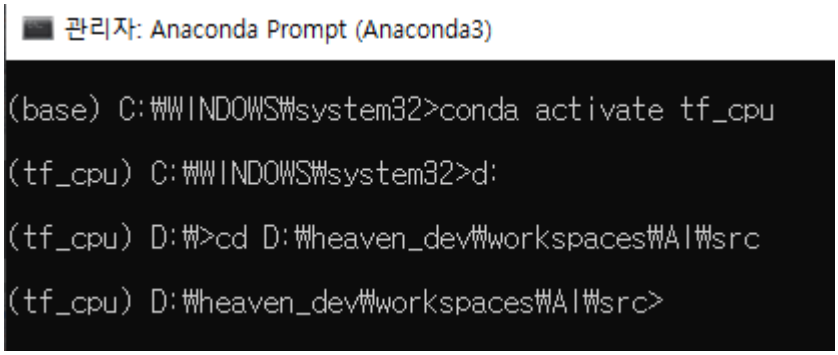
```r
# 3. 기하학적 기법 시각화(ggplot2 package) # Day31; 20221019

# 3.1 qplot() 함수

install.packages("ggplot2") # 필수 패키지(ggplot, lattice 등)
library(ggplot2)
data("mpg")
View(mpg)

str(mpg) # 234 obs. of  11 variables:
class(mpg) # "data.frame"
summary(mpg)
```

```r

## 1. R 고급 시각화 도구
# 제공 패키지- graphics/lattice/ggplot2 등...

## 2. 격자형(lattice) 기법 시각화(lattice package)
# 패키지 설치와 실습 데이터 셋 가져오기
install.packages("lattice") # 격자형은 셀 단위로 구분해서 시각화한 것.
library(lattice)

```

AI

(jupyter)다시 연결을 변경해줌. C:\Users\tjouen-jr\.jupyter (Users는 사용자)





데이터 분석에 유용한 기능들...

- list comprehension 기본 구조

In [7]:
```
numbers = [1,2,3,4,5]
square1 = []
```

In [8]:
```
for i in numbers:
    square1.append(i ** 2)

square1 # 제곱한 값이 출력되기 시작
```

Out[8]:
```
[1, 4, 9, 16, 25]
```

In [10]:
```
square2 = [i ** 2 for i in numbers] # list comprehension 이라고 불리는 구조
square2
```

Out[10]:
```
[1, 4, 9, 16, 25]
```

In [11]:
```
square3 = []

for i in numbers:
    if i >= 3:
        square3.append(i ** 2)

square3
```

Out[11]:
```
[9, 16, 25]
```

In [12]:
```
square4 = [i ** 2 for i in numbers if i >= 3] # append와 for문과 if문이 섞여있는 짬짜면
square4
```

Out[12]:
```
[9, 16, 25]
```

*** List comprehension : 리스트 변환하는 표현식으로 유용한 기능**

In [14]:
```
day = ['mon', 'tue','wed','thu','fri','sat','sun']
print([x for x in day]) # x에 리스트 자료형으로 데이터를 담아줌.
```

```
['mon', 'tue', 'wed', 'thu', 'fri', 'sat', 'sun']
```

In [17]:
```
data = [35, 56, -53, 45, 27, -28, 0, -12]
print([x for x in data if x >= 0]) # [x(리스트 자료형으로 바꿔줌) for x(값을 담아줌) in data if x >= 0]
```

```
[35, 56, 45, 27, 0]
```

In [18]:
```
data = [35, 56, -53, 45, 27, -28, 0, -12]
print([x ** 2 for x in data if x >= 0])
```

```
[1225, 3136, 2025, 729, 0]
```

- split(구분자) : 구분자로 구분, 기본값은 공백

In [23]:
```
test_text = "the-joeun-Computer_academy-with-python"
result = test_text.split('-') # 바(-)를 기준으로 데이터들을 구분해주고 리턴해준다.
result # 리스트 자료 구조로 리턴해줌
```

Out[23]:
```
['the', 'joeun', 'Computer_academy', 'with', 'python']
```

In [24]:
```
test_text = ['the', 'joeun', 'Computer_academy', 'with', 'python']
test_text
```

Out[24]:
```
['the', 'joeun', 'Computer_academy', 'with', 'python']
```

In [26]:
```
result = '-'.join(test_text) # -를 기준으로 데이터를 조합해준다.
result
```

Out[26]:
```
'the-joeun-Computer_academy-with-python'
```

- split()와 join()의 응용

In [30]:
```
result = '-'.join('345.234'.split('.'))
result
```

Out[30]:
```
'345-234'
```

- enumerate(list): 인덱스와 값을 함께 반환

In [33]:
```
for i, name in enumerate(['a','b','c','d','e']): # 2개의 변수를 선언해줄 수 있는 프로그램은 python밖에 없다. Check!!!
    print(i, name) # 리스트에 있는 데이터들을 for문을 통해 name변수에다가 매칭 시켜준다.
```

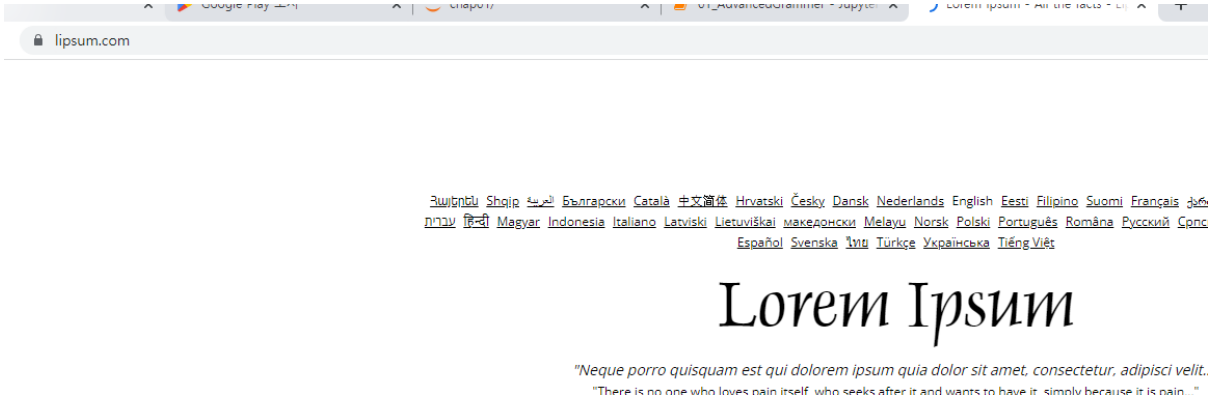
```
0 a
1 b
2 c
3 d
4 e
```

In [35]:
```
seq = ['mon', 'tue','wed','thu','fri','sat','sun']
dict(enumerate(seq)) # key, value형태로 dict가 알아서 출력해주게 됨.
```

Out[35]:
```
{0: 'mon', 1: 'tue', 2: 'wed', 3: 'thu', 4: 'fri', 5: 'sat', 6: 'sun'}
```

In [38]:
```
key_seq = 'abcdefg'
value_seq = ['mon', 'tue','wed','thu','fri','sat','sun']
dict(zip(key_seq, value_seq)) # 두 변수가 하나씩 물려가는 구조로 zipper 데이터를 만들어주게 됨.
# dict 형태로 강제형변환해줌.
```

Out[38]:
```
{'a': 'mon',
 'b': 'tue',
 'c': 'wed',
 'd': 'thu',
 'e': 'fri',
 'f': 'sat',
 'g': 'sun'}
```



의미없는 문장들을 담아둔 사이트

- Counter를 이용한 카운팅

- Count는 아이템의 갯수를 자동으로 카운팅.(단어의 빈도수)

In [40]:
```
from collections import Counter
```

In [41]:
```
message = """
What is Lorem Ipsum?
Lorem Ipsum is simply dummy text of the printing and typesetting industry.
Lorem Ipsum has been the industry's standard dummy text ever since the 1500s,
when an unknown printer took a galley of type and scrambled it to make a type specimen book.
It has survived not only five centuries, but also the leap into electronic typesetting,
remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages,
and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.
What is Lorem Ipsum?
Lorem Ipsum is simply dummy text of the printing and typesetting industry.
Lorem Ipsum has been the industry's standard dummy text ever since the 1500s,
when an unknown printer took a galley of type and scrambled it to make a type specimen book.
It has survived not only five centuries, but also the leap into electronic typesetting,
remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages,
and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.
"""
```

In [44]:
```
counter = Counter(message.split()) # new와 같이 사용하려면 ()를 이용해 호출해주면 생성됨
print(counter) # 단어 빈도수를 key, value형태로 리턴해줌
```

```
Counter({'the': 12, 'Lorem': 10, 'of': 8, 'Ipsum': 6, 'and': 6, 'is': 4, 'dummy': 4, 'text': 4, 'has': 4, 'a': 4, 'type': 4, 'It': 4, 'with': 4, 'What': 2, 'Ipsum?': 2, 'simply': 2, 'printing': 2, 'typesetting': 2, 'industry.': 2, 'been': 2, "industry's": 2, 'standard': 2, 'ever': 2, 'since': 2, '1500s,': 2, 'when': 2, 'an': 2, 'unknown': 2, 'printer': 2, 'took': 2, 'galley': 2, 'scrambled': 2, 'it': 2, 'to': 2, 'make': 2, 'specimen': 2, 'book.': 2, 'survived': 2, 'not': 2, 'only': 2, 'five': 2, 'centuries,': 2, 'but': 2, 'also': 2, 'leap': 2, 'into': 2, 'electronic': 2, 'typesetting,': 2, 'remaining': 2, 'essentially': 2, 'unchanged.': 2, 'was': 2, 'popularised': 2, 'in': 2, '1960s': 2, 'release': 2, 'Letraset': 2, 'sheets': 2, 'containing': 2, 'passages,': 2, 'more': 2, 'recently': 2, 'desktop': 2, 'publishing': 2, 'software': 2, 'like': 2, 'Aldus': 2, 'PageMaker': 2, 'including': 2, 'versions': 2, 'Ipsum.': 2})
```

In [45]:
```
print(type(counter))
```

```
<class 'collections.Counter'>
```

*** Counter(dict) -> list 형태로 반환.**

```
In [13]: print(counter.most_common()) # list 형태로 변환
         print(type(counter.most_common()))
```

```
[('the', 12), ('Lorem', 10), ('of', 8), ('Ipsum', 6), ('and', 6), ('is', 4), ('dummy', 4), ('text', 4), ('has', 4), ('a', 4), ('type', 4),
('It', 4), ('with', 4), ('What', 2), ('Ipsum?', 2), ('simply', 2), ('printing', 2), ('typesetting', 2), ('industry.', 2), ('been', 2), ("i
ndustry's", 2), ('standard', 2), ('ever', 2), ('since', 2), ('1500s,', 2), ('when', 2), ('an', 2), ('unknown', 2), ('printer', 2), ('too
k', 2), ('galley', 2), ('scrambled', 2), ('it', 2), ('to', 2), ('make', 2), ('specimen', 2), ('book.', 2), ('survived', 2), ('not', 2),
('only', 2), ('five', 2), ('centuries,', 2), ('but', 2), ('also', 2), ('leap', 2), ('into', 2), ('electronic', 2), ('typesetting,', 2),
('remaining', 2), ('essentially', 2), ('unchanged.', 2), ('was', 2), ('popularised', 2), ('in', 2), ('1960s', 2), ('release', 2), ('Letras
et', 2), ('sheets', 2), ('containing', 2), ('passages,', 2), ('more', 2), ('recently', 2), ('desktop', 2), ('publishing', 2), ('software',
2), ('like', 2), ('Aldus', 2), ('PageMaker', 2), ('including', 2), ('versions', 2), ('Ipsum.', 2)]
<class 'list'>
```

*** list -> dict 형태로 변환**

```
In [15]: dict_msg = dict(counter.most_common())
         print(dict_msg)
```

```
{'the': 12, 'Lorem': 10, 'of': 8, 'Ipsum': 6, 'and': 6, 'is': 4, 'dummy': 4, 'text': 4, 'has': 4, 'a': 4, 'type': 4, 'It': 4, 'with': 4,
'What': 2, 'Ipsum?': 2, 'simply': 2, 'printing': 2, 'typesetting': 2, 'industry.': 2, 'been': 2, "industry's": 2, 'standard': 2, 'ever':
2, 'since': 2, '1500s,': 2, 'when': 2, 'an': 2, 'unknown': 2, 'printer': 2, 'took': 2, 'galley': 2, 'scrambled': 2, 'it': 2, 'to': 2, 'mak
e': 2, 'specimen': 2, 'book.': 2, 'survived': 2, 'not': 2, 'only': 2, 'five': 2, 'centuries,': 2, 'but': 2, 'also': 2, 'leap': 2, 'into':
2, 'electronic': 2, 'typesetting,': 2, 'remaining': 2, 'essentially': 2, 'unchanged.': 2, 'was': 2, 'popularised': 2, 'in': 2, '1960s': 2,
'release': 2, 'Letraset': 2, 'sheets': 2, 'containing': 2, 'passages,': 2, 'more': 2, 'recently': 2, 'desktop': 2, 'publishing': 2, 'softw
are': 2, 'like': 2, 'Aldus': 2, 'PageMaker': 2, 'including': 2, 'versions': 2, 'Ipsum.': 2}
```

```
In [16]: print(dict_msg['dummy']) # 4
```

```
4
```

파이썬의 모듈은 소스파일이다.

__init__ 를 보면 파이썬은 내가 관리해야할 패키지로 인식한다.

모듈의 묶음은 패키지/ 패키지의 묶음은 라이브러리라고 칭한다.

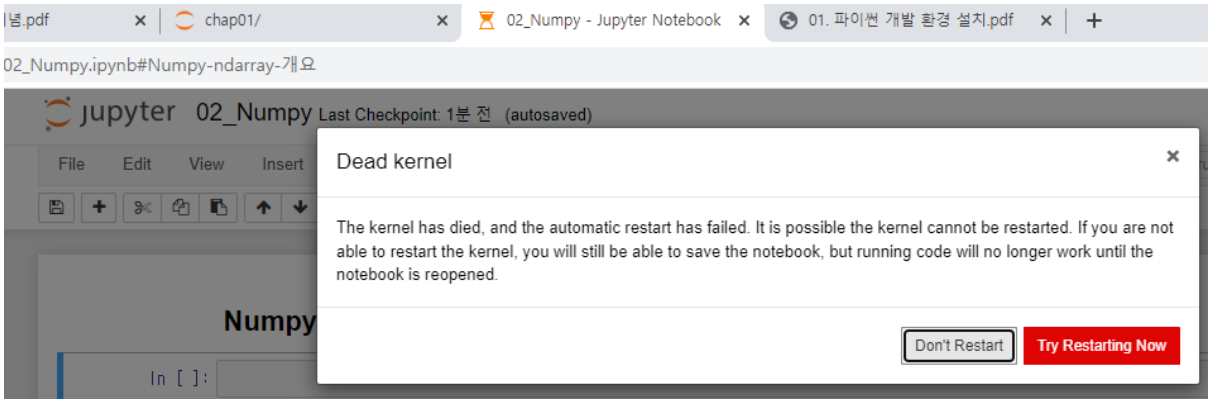
R에서는 파이썬의 라이브러리를 패키지라고 한다. 따라서 파이썬에서는 R의 패키지를 라이브러리라고 한다.

파이썬 ( ) 가 행렬식

➢ ndarray의 크기, 즉 행과 열의 수를 튜플 형태로 제공, 차원 확인까지 가능.

맵리듀스(JAVA) 기술

라이브러리 설치를 위해 잠시 중단 Don’t Restart!



다만 파이썬 3.7버전에 맞춰서 다운을 해야하기 때문에 코랩을 사용해야함.

(항상 최신 버전으로 설치되기 때문에)

가상 환경 구성 및 텐서플로우/conda 커널 설치

```
(tf_cpu) C:₩Users₩user>conda install numpy

(tf_cpu) C:₩Users₩user>conda install pandas

(tf_cpu) C:₩Users₩user>conda install matplotlib

(tf_cpu) C:₩Users₩user>conda install scikit-learn

(tf_cpu) C:₩Users₩user>conda install tensorflow    # 텐서플로우 설치하기.

(tf_cpu) C:₩Users₩user>conda install nb_conda     # conda 커널 설치.
```

Numpy ndarray 개요(nd: n차원, 예) 1차원, 2차원..)

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

- 배열 생성

In [2]:
```
# - 1차원 배열(Vector) 정의
array1 = np.array([1,2,3])
print(array1) # [1 2 3] # ,가 없이 보여주는 것이 numpy
print('array1.type:', type(array1))
print('array1 array 형태:', array1.shape) # 행렬로 오해하지 말고 1차원이더라도 3, 으로 나온다.
```

```
[1 2 3]
array1.type: <class 'numpy.ndarray'>
array1 array 형태: (3,)
```

In [3]:
```
# of) 리스트 자료형
list = [1,2,3]
print(list)
print('list.type:', type(list)) # 파이썬에서 알려주는 list
```

```
[1, 2, 3]
list.type: <class 'list'>
```

In [4]:
```
# - 2차원 배열(Vector) 정의
array1 = np.array([1,2,3])
print(array1) # [1 2 3] # ,가 없이 보여주는 것이 numpy
print('array1.type:', type(array1))
print('array1 array 형태:', array1.shape) # 행렬로 오해하지 말고 1차원이더라도 3, 으로 나온다.
```

```
[1 2 3]
array1.type: <class 'numpy.ndarray'>
array1 array 형태: (3,)
```