

# Round 3

PRESS START





New Assignment







Let's Go





#### Data Science Project\_1 =



+



It's OK, but...



Data Science Project\_1 =



+



Deep Learning Project\_1



+

TensorFlow matpletlib

Data Science Project\_2 =



+



\_

So sad...







- Anaconda는 파이썬의 배포판
- 특정 용도의 소프트웨어 패키지를 묶어서 제공
  - 각종 가상환경 구축 가능













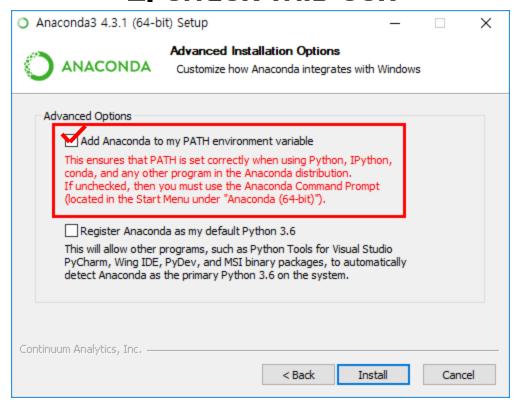




#### 1. Go to

#### https://www.anaconda.com/distribution/#download-section

#### 2. check this box





#### 3. how to check my Anaconda is installed well?



#### 4. Creating virtual environments with conda

conda create --name test python=3.7

5. Activating the virtual environments

conda activate test / conda deactivate

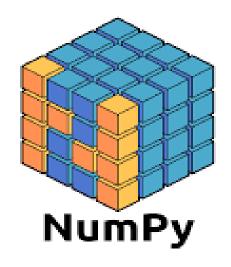




Let's Go







- 대표적인 파이썬 기반 수치해석 라이브러리
  - 다양한 행렬과 배열 연산 함수 지원



### matpletlib

- 자료를 차트나 플롯으로 시각화(visualization)하는 패키지
- 판다스와 연계하여 다양한 방식으로 시각화하는 기능 제공





- 과학용 연산에 필요한 다양한 패키지를 모아 놓은 라이브러리
- 미적분, 선형대수, 행렬연산, 방정식 계산 등 다양한 함수 지원





- 데이터 분석 및 조작을 위한 라이브러리
- R과 비슷한 자료형을 사용함(Dataframe)



# 이렇게 다양한 라이브러리를 하나하나 관리해야 한다면…?



So sad...





- 파이썬 라이브러리 패키지를 손쉽게 관리 할 수 있게 해주는 시스템
  - 파이썬 3.4이후 버전부터 기본 탑재!
  - 짧은 명령어 한 줄로 손쉽게 설치 가능



#### 1. 패키지를 설치해줄 가상환경 활성화

C:\Users\user>conda activate test

#### 2. pip install numpy로 numpy 설치

```
(test) C:\Users\user>pip install numpy
Collecting numpy
Using cached https://files.pythonhosted.org/packages/a9/3
f/numpy-1.18.1-cp37-cp37m-win_amd64.whl
Installing collected packages: numpy
Successfully installed numpy-1.18.1
```

#### 3. pip freeze로 현재 설치된 패키지 확인

(test) C:\Users\user>pip freeze certifi==2019.11.28 numpy==1.18.1 wincertstore==0.2





Let's Go







#### 데이터 분석을 하기 위해서 필요한 것?

각종 통계적 기법??

다량의 데이터를 처리할 만한 풍부한 컴퓨팅 자원??

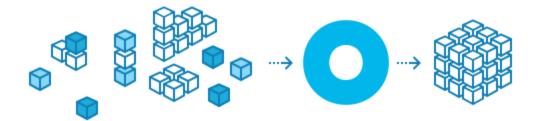
편리한 라이브러리와 시각화 도구??





데이터 그 자체가 가장 중요!

#### 데이터분석 과정의 80%는 데이터를 다듬고 분석에 적합한 형태로 만드는 것





#### **Series**



데이터 값(value)



#### Series

- index와 value가 일대일 대응(dictionary와 비슷한 구조)

```
# pandas 불러오기 Alias pd
import pandas as pd
# Dictionary 생성
dict_data = {'a' _: 1, 'b' _: 2, 'c' _: 3}
# Series로 변환
sr = pd.Series(dict_data)
# sr의 자료형 출력
print(type(sr))
# sr에 저장된 시리즈 객체 출력
print(sr)
```



#### Series

- INOIM의 모든 선형 자료형을 series화 가능

```
tuple_data = ('광종', '1997-07-10', 3, True)

tsr = pd.Series(tuple_data, index_=_["이름", "생년월일", "학년", "재학여부"])

print(tsr)
```

(tuple을 series화 해서 이름형 index를 붙혀주는 모습)

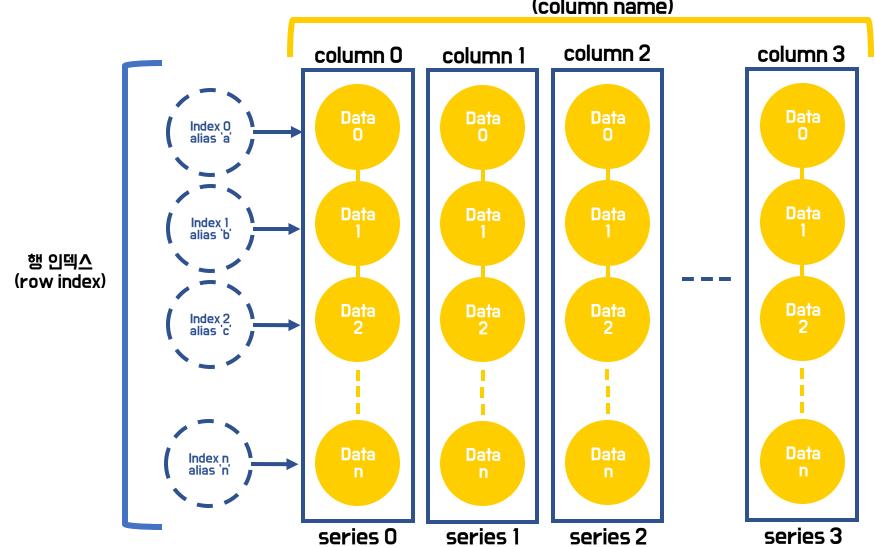
- series는 정수형 인덱스와 이름형 인덱스로 모두 접근 가능

```
print(sr['a'])
print(sr[0])
```

(이름 안 붙혀줄 경우 정수형 index로만 접근가능)



열 이름 (column name)





- dataframe을 만들기 위해선 같은 길이의 1차원 배열 여러 개가 필요

```
dict_data = {'c0'_: [1,2,3], 'c1'_: [4,5,6], 'c2'_: [7,8,9], 'c3'_: [10,11,12]}

df = pd.DataFrame(dict_data)

print(type(df))

print('\n')

print(df)
```

```
<class 'pandas.core.frame.DataFrame'>
    c0 c1 c2 c3
0 1 4 7 10
1 2 5 8 11
2 3 6 9 12
```



#### - 요렇게도 가능

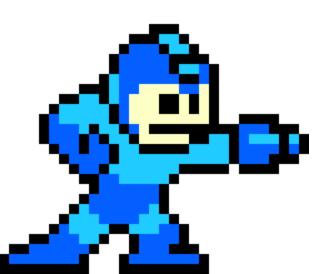
```
나이 성별 학교
팡종 15 남 덕영중
습인 17 여 강남중
주년 19 남 상정고
선 14 여 산곡여중
```



- DataFrame으로 저장된 데이터들에 통계함수를 직접적으로 사용가능
  - 다양한 포맷으로 import/export할 수 있음 (CSV, JSON, Excel···)
- 행에 대한 접근은 DataFrame.loc[인덱스 이름]과 .iloc[정수형 위치 인덱스]
  - 열에 대한 접근은 DataFrame["열이름"] 혹은 ["정수형 위치 인덱스"]
    - DataFrame의 깊은 복사는 s.copy()

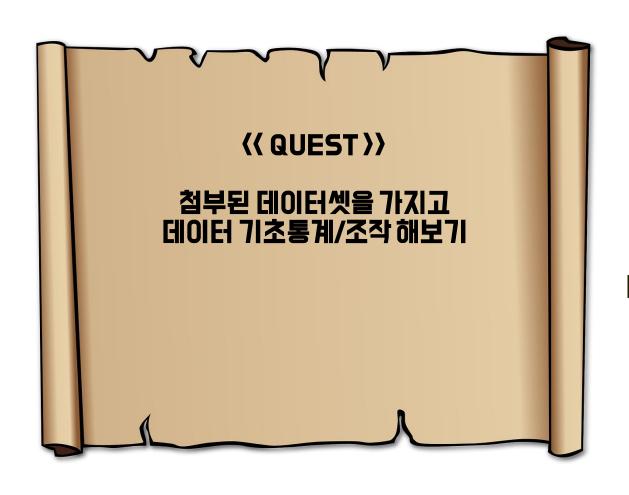


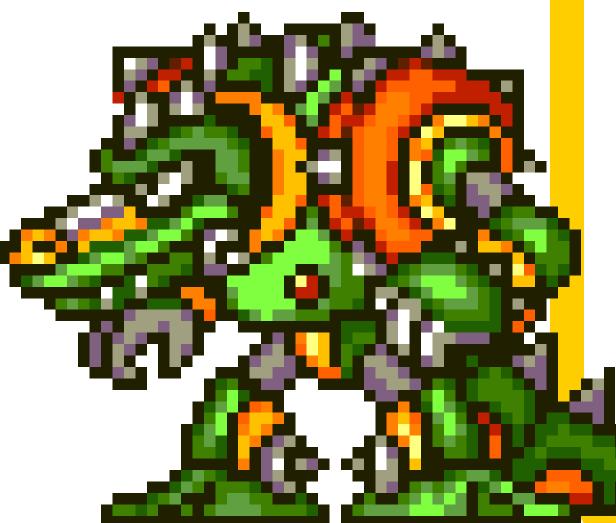
# MARNING





## WARNING







# NEXT STAGE

