DATAKUBWA

작성자 - 고우주 / 데이터쿱와(주)

Markdown 알아보기

마크다운은 웹에서 텍스트의 스타일을 지정하는 방법으로 문서의 표시를 제어한다. 굵게 또는 기울임 꼴로 단어 서식을 지정하고, 이미지를 추가하고, 목록을 만드는 작업은 Markdown에서 제공한다. 대부분 Markdown은 # 또는 * 와 같이 일부 알파벳이 아닌 문자가 포함 된 일반 텍스트이다.

https://guides.github.com/features/mastering-markdown/ (https://guides.github.com/features/mastering-markdown/)

This is an

This is an

This is an

This text will be italic This will also be italic

This text will be bold This will also be bold

*You **can** combine them*

Unordered List

- Item 1
- Item 2
  - Item 2a
  - Item 2b

Ordered List

1. Item 1
2. Item 2
3. Item 3
   A. Item 3a
   B. Item 3b

http://github.com (http://github.com) GitHub (http://github.com)

As Kanye West said:

> We're living the future so the present is our past.

I think you should use an `<addr>` element here instead.

- [x] @mentions, #refs, links (), **formatting**, and ~~tags~~ supported
- [x] list syntax required (any unordered or ordered list supported)
- [x] this is a complete item
- [ ] this is an incomplete item

First Header	Second Header
Content from cell 1	Content from cell 2
Content in the first column	Content in the second column

파이썬 실습 - Titanic

Features 변수명 설명

- survival: Survival (0 = No; 1 = Yes)
- pclass: Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd)
- name: Name
- sex: Sex
- age: Age
- sibsp: Number of Siblings/Spouses Aboard
- parch: Number of Parents/Children Aboard
- ticket: Ticket Number
- fare: Passenger Fare
- cabin: Cabin
- embarked: Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)

1. package import

In [1]:

```
import pandas as pd
import numpy as np
```

2. Dataset 불러오기

In [2]:

```
titanic = pd.read_csv("data/titanic.csv")
titanic
```

Out[2]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C
10	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.7000	G6	S
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.5500	C103	S
12	13	0	3	Saundercock, Mr. William Henry	male	20.0	0	0	A/5. 2151	8.0500	NaN	S
13	14	0	3	Andersson, Mr. Anders Johan	male	39.0	1	5	347082	31.2750	NaN	S
14	15	0	3	Vestrom, Miss. Hulda Amanda Adolfina	female	14.0	0	0	350406	7.8542	NaN	S
15	16	1	2	Hewlett, Mrs. (Mary D Kingcome)	female	55.0	0	0	248706	16.0000	NaN	S
16	17	0	3	Rice, Master. Eugene	male	2.0	4	1	382652	29.1250	NaN	Q
17	18	1	2	Williams, Mr. Charles Eugene	male	NaN	0	0	244373	13.0000	NaN	S
18	19	0	3	Vander Planke, Mrs. Julius (Emelia Maria Vande...	female	31.0	1	0	345763	18.0000	NaN	S
19	20	1	3	Masselmani, Mrs. Fatima	female	NaN	0	0	2649	7.2250	NaN	C
20	21	0	2	Fynney, Mr. Joseph J	male	35.0	0	0	239865	26.0000	NaN	S
21	22	1	2	Beesley, Mr. Lawrence	male	34.0	0	0	248698	13.0000	D56	S

22	23	1	3	McGowan, Miss. Anna "Annie"	female	15.0	0	0	330923	8.0292	NaN	Q
23	24	1	1	Sloper, Mr. William Thompson	male	28.0	0	0	113788	35.5000	A6	S
24	25	0	3	Palsson, Miss. Torborg Danira	female	8.0	3	1	349909	21.0750	NaN	S
25	26	1	3	Asplund, Mrs. Carl Oscar (Selma Augusta Emilia...	female	38.0	1	5	347077	31.3875	NaN	S
26	27	0	3	Emir, Mr. Farred Chehab	male	NaN	0	0	2631	7.2250	NaN	C
27	28	0	1	Fortune, Mr. Charles Alexander	male	19.0	3	2	19950	263.0000	C23 C25 C27	S
28	29	1	3	O'Dwyer, Miss. Ellen "Nellie"	female	NaN	0	0	330959	7.8792	NaN	Q
29	30	0	3	Todoroff, Mr. Lalio	male	NaN	0	0	349216	7.8958	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
861	862	0	2	Giles, Mr. Frederick Edward	male	21.0	1	0	28134	11.5000	NaN	S
862	863	1	1	Swift, Mrs. Frederick Joel (Margaret Welles Ba...	female	48.0	0	0	17466	25.9292	D17	S
863	864	0	3	Sage, Miss. Dorothy Edith "Dolly"	female	NaN	8	2	CA. 2343	69.5500	NaN	S
864	865	0	2	Gill, Mr. John William	male	24.0	0	0	233866	13.0000	NaN	S
865	866	1	2	Bystrom, Mrs. (Karolina)	female	42.0	0	0	236852	13.0000	NaN	S
866	867	1	2	Duran y More, Miss. Asuncion	female	27.0	1	0	SC/PARIS 2149	13.8583	NaN	C
867	868	0	1	Roebling, Mr. Washington Augustus II	male	31.0	0	0	PC 17590	50.4958	A24	S
868	869	0	3	van Melkebeke, Mr. Philemon	male	NaN	0	0	345777	9.5000	NaN	S
869	870	1	3	Johnson, Master. Harold Theodor	male	4.0	1	1	347742	11.1333	NaN	S
870	871	0	3	Balkic, Mr. Cerin	male	26.0	0	0	349248	7.8958	NaN	S
871	872	1	1	Beckwith, Mrs. Richard Leonard (Sallie Monypeny)	female	47.0	1	1	11751	52.5542	D35	S
872	873	0	1	Carlsson, Mr. Frans Olof	male	33.0	0	0	695	5.0000	B51 B53 B55	S
873	874	0	3	Vander Cruyssen, Mr. Victor	male	47.0	0	0	345765	9.0000	NaN	S
874	875	1	2	Abelson, Mrs. Samuel (Hannah Wizosky)	female	28.0	1	0	P/PP 3381	24.0000	NaN	C
875	876	1	3	Najib, Miss. Adele Kiamie "Jane"	female	15.0	0	0	2667	7.2250	NaN	C
876	877	0	3	Gustafsson, Mr. Alfred Ossian	male	20.0	0	0	7534	9.8458	NaN	S
877	878	0	3	Petroff, Mr. Nedelio	male	19.0	0	0	349212	7.8958	NaN	S
878	879	0	3	Laleff, Mr. Kristo	male	NaN	0	0	349217	7.8958	NaN	S
879	880	1	1	Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)	female	56.0	0	1	11767	83.1583	C50	C
880	881	1	2	Shelley, Mrs. William (Imanita Parrish Hall)	female	25.0	0	1	230433	26.0000	NaN	S
881	882	0	3	Markun, Mr. Johann	male	33.0	0	0	349257	7.8958	NaN	S
882	883	0	3	Dahlberg, Miss. Gerda Ulrika	female	22.0	0	0	7552	10.5167	NaN	S
883	884	0	2	Banfield, Mr. Frederick James	male	28.0	0	0	C.A./SOTON 34068	10.5000	NaN	S
884	885	0	3	Sutehall, Mr. Henry Jr	male	25.0	0	0	SOTON/OQ 392076	7.0500	NaN	S
885	886	0	3	Rice, Mrs. William (Margaret Norton)	female	39.0	0	5	382652	29.1250	NaN	Q
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

In [3]:

```
titanic.head()
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

3. Data 확인하기

In [4]:

```
# 데이터프레임의 행과 열 갯수 확인
titanic.shape
```

Out[4]:

```
(891, 12)
```

In [5]:

```
# 전반적인 데이터프레임의 정보 확인
titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
```

In [6]:

```
# 기본 통계치 확인
titanic.describe()
```

Out[6]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

4. Feature Data 탐색

In [7]:

```
# values_counts로 Pclass의 각 값의 갯수를 확인
value_counts = titanic["Pclass"].value_counts()
print(value_counts)
```

```
3    491
1    216
2    184
Name: Pclass, dtype: int64
```

In [8]:

```
# 데이터프레임에서 한개의 열을 선택시 데이터클래스 확인 -> 시리즈
titanic_pclass = titanic["Pclass"]
print(type(titanic_pclass))
```

```
<class 'pandas.core.series.Series'>
```

In [9]:

```
titanic_pclass.head()
```

Out[9]:

```
0    3
1    1
2    3
3    1
4    3
Name: Pclass, dtype: int64
```

In [10]:

```
# 한번에 확인하기
titanic_plcass = titanic["Pclass"].value_counts()
print(type(value_counts))
print(value_counts)
```

```
<class 'pandas.core.series.Series'>
3    491
1    216
2    184
Name: Pclass, dtype: int64
```

5. DataFrame의 컬럼 데이터 엑세스

In [11]:

```
# Age_0 컬럼에 0을 채워 추가하기
titanic["Age_0"]=0
titanic.head()
```

Out[11]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Age_0
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	0
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	0
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	0
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	0

In [12]:

```
# Age_by_10 열을 생성하고 Age * 10 값을 넣기
titanic["Age_by_10"] = titanic["Age"] * 10
```

In [13]:

```
# Family_No 열을 생성하고 여성과 아이 승객을 합해 남자 본인을 합하여 가족수 생성
titanic["Family_No"] = titanic["SibSp"] + titanic["Parch"] + 1
titanic.head()
```

Out[13]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Age_0	Age_by_10	Fami
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	0	220.0	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	0	380.0	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	0	260.0	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	0	350.0	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	0	350.0	

In [14]:

```
# 생성한 Age_by_10에 100을 더해보자
titanic["Age_by_10"] = titanic["Age_by_10"] + 100
titanic.head()
```

Out[14]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Age_0	Age_by_10	Fa
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	0	320.0	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	0	480.0	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	0	360.0	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	0	450.0	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	0	450.0	

6. DataFrame 데이터 삭제

In [15]:

```
# 데이터이프레임에서 drop 함수로 Age_0을 삭제, 이때 axis=1 인자를 넣어 열을 삭제
titanic_drop = titanic.drop('Age_0', axis=1 )
titanic_drop.head()
```

Out[15]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Age_by_10	Family_No
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	320.0	2
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	480.0	2
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	360.0	1
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	450.0	2
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	450.0	1

In [16]:

```
titanic.head()
```

Out[16]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Age_0	Age_by_10	Fa
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	0	320.0	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	0	480.0	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	0	360.0	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	0	450.0	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	0	450.0	

In [17]:

```
# drop_result 변수를 생성하고 복수의 열을 삭제한다. inplace=True는 반환받지 않고, 기존의 데이터프레임을 변경
drop_result = titanic.drop(['Age_0', 'Age_by_10', 'Family_No'], axis=1, inplace=True)
print(' inplace=True 로 drop 후 반환된 값:', drop_result)
titanic.head()
```

```
 inplace=True 로 drop 후 반환된 값: None
```

Out[17]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [18]:

```
# 행을 삭제시 행 index를 지정
titanic.drop([0,1,2], axis=0, inplace=True)
titanic.head()
```

Out[18]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S

7. Index 객체

In [19]:

```
# 원본 파일 재 로딩
df = pd.read_csv("data/titanic.csv")

# Index 객체 추출
indexes = df.index
print(indexes)

# Index 객체를 실제 값 arrray로 변환
print('Index 객체 array값:\n',indexes.values)
```

```
RangeIndex(start=0, stop=891, step=1)
Index 객체 array값:
 [  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35
  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53
  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71
  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89
  90  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107
 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143
 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161
 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179
 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197
 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215
 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233
 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251
 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269
 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287
 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305
 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323
 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341
 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359
 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377
 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395
 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413
 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431
 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449
 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467
 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485
 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503
 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521
 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539
 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557
 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575
 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593
 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611
 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629
 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647
 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665
 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683
 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701
 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719
 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737
 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755
 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773
 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791
 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809
 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827
 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845
 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863
 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881
 882 883 884 885 886 887 888 889 890]
```

In [20]:

```
# 인덱스의 데이터타입 확인 및 인덱싱
print(type(indexes.values))
print(indexes.values.shape)
print(indexes[:5].values)
print(indexes.values[:5])
print(indexes[6])
```

```
<class 'numpy.ndarray'>
(891,)
[0 1 2 3 4]
[0 1 2 3 4]
6
```

In [21]:

```
# 열의 시리즈 반환하기
series_fair = df['Fare']
print('Fair Series max 값:', series_fair.max())
print('Fair Series sum 값:', series_fair.sum())
print('sum() Fair Series:', sum(series_fair))
print('Fair Series + 3:\n',(series_fair + 3).head(3) )
```

```
Fair Series max 값: 512.3292
Fair Series sum 값: 28693.9493
sum() Fair Series: 28693.949299999967
Fair Series + 3:
 0    10.2500
1    74.2833
2    10.9250
Name: Fare, dtype: float64
```

In [22]:

```
# 인덱스를 reset 하고 inplace=False 인자로 titanic_reset에 index 별도로 넣기
df_reset = df.reset_index(inplace=False)
df_reset.head()
```

Out[22]:

	index	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [23]:

```
# reset_index 전후 비교하기
print('### before reset_index ###')
value_counts = df['Pclass'].value_counts()
print(value_counts)
print('value_counts 객체 변수 타입:',type(value_counts))

new_value_counts = value_counts.reset_index(inplace=False)
print('### After reset_index ###')
print(new_value_counts)
print('new_value_counts 객체 변수 타입:',type(new_value_counts))
```

```
### before reset_index ###
3    491
1    216
2    184
Name: Pclass, dtype: int64
value_counts 객체 변수 타입: <class 'pandas.core.series.Series'>
### After reset_index ###
   index  Pclass
0      3     491
1      1     216
2      2     184
new_value_counts 객체 변수 타입: <class 'pandas.core.frame.DataFrame'>
```

7. 데이터 셀렉션 및 필터링

In [24]:

```
# 하나의 열만 선택하여 보기 -> 시리즈
df[ 'Pclass' ].head()
```

Out[24]:

```
0    3
1    1
2    3
3    1
4    3
Name: Pclass, dtype: int64
```

In [25]:

```
# 두개 이상의 열만 선택하여 보기 -> 데이터프레임
df[ ['Survived', 'Pclass'] ].head()
```

Out[25]:

	Survived	Pclass
0	0	3
1	1	1
2	1	3
3	1	1
4	0	3

In [26]:

```
# 행 슬라이싱
df[0:2]
```

Out[26]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C

In [27]:

```
# 불리언 인덱싱으로 3등급 데이터만 보기
df[ df['Pclass'] == 3].head()
```

Out[27]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S

In [29]:

```
# 나이 60 이상의 승객을 df_boolean에 넣고 실행하기
df_boolean = df[df['Age'] > 60]
print(type(df_boolean))
df_boolean
```

```
<class 'pandas.core.frame.DataFrame'>
```

Out[29]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
33	34	0	2	Wheadon, Mr. Edward H	male	66.0	0	0	C.A. 24579	10.5000	NaN	S
54	55	0	1	Ostby, Mr. Engelhart Cornelius	male	65.0	0	1	113509	61.9792	B30	C
96	97	0	1	Goldschmidt, Mr. George B	male	71.0	0	0	PC 17754	34.6542	A5	C
116	117	0	3	Connors, Mr. Patrick	male	70.5	0	0	370369	7.7500	NaN	Q
170	171	0	1	Van der hoef, Mr. Wyckoff	male	61.0	0	0	111240	33.5000	B19	S
252	253	0	1	Stead, Mr. William Thomas	male	62.0	0	0	113514	26.5500	C87	S
275	276	1	1	Andrews, Miss. Kornelia Theodosia	female	63.0	1	0	13502	77.9583	D7	S
280	281	0	3	Duane, Mr. Frank	male	65.0	0	0	336439	7.7500	NaN	Q
326	327	0	3	Nysveen, Mr. Johan Hansen	male	61.0	0	0	345364	6.2375	NaN	S
438	439	0	1	Fortune, Mr. Mark	male	64.0	1	4	19950	263.0000	C23 C25 C27	S
456	457	0	1	Millet, Mr. Francis Davis	male	65.0	0	0	13509	26.5500	E38	S
483	484	1	3	Turkula, Mrs. (Hedwig)	female	63.0	0	0	4134	9.5875	NaN	S
493	494	0	1	Artagaveytia, Mr. Ramon	male	71.0	0	0	PC 17609	49.5042	NaN	C
545	546	0	1	Nicholson, Mr. Arthur Ernest	male	64.0	0	0	693	26.0000	NaN	S
555	556	0	1	Wright, Mr. George	male	62.0	0	0	113807	26.5500	NaN	S
570	571	1	2	Harris, Mr. George	male	62.0	0	0	S.W./PP 752	10.5000	NaN	S
625	626	0	1	Sutton, Mr. Frederick	male	61.0	0	0	36963	32.3208	D50	S
630	631	1	1	Barkworth, Mr. Algernon Henry Wilson	male	80.0	0	0	27042	30.0000	A23	S
672	673	0	2	Mitchell, Mr. Henry Michael	male	70.0	0	0	C.A. 24580	10.5000	NaN	S
745	746	0	1	Crosby, Capt. Edward Gifford	male	70.0	1	1	WE/P 5735	71.0000	B22	S
829	830	1	1	Stone, Mrs. George Nelson (Martha Evelyn)	female	62.0	0	0	113572	80.0000	B28	NaN
851	852	0	3	Svensson, Mr. Johan	male	74.0	0	0	347060	7.7750	NaN	S

In [30]:

```
# 60이상의 승객 중 'Name', 'Age'  열만 선택해서 데이터프레임 반환
df[df['Age'] > 60][['Name','Age']].head()
```

Out[30]:

	Name	Age
33	Wheadon, Mr. Edward H	66.0
54	Ostby, Mr. Engelhart Cornelius	65.0
96	Goldschmidt, Mr. George B	71.0
116	Connors, Mr. Patrick	70.5
170	Van der hoef, Mr. Wyckoff	61.0

In [31]:

```
# loc로 불리언 인덱싱하여 소환하기 [행, 열]
df.loc[df['Age'] > 60, ['Name','Age']].head()
```

Out[31]:

	Name	Age
33	Wheadon, Mr. Edward H	66.0
54	Ostby, Mr. Engelhart Cornelius	65.0
96	Goldschmidt, Mr. George B	71.0
116	Connors, Mr. Patrick	70.5
170	Van der hoef, Mr. Wyckoff	61.0

In [32]:

```
# 60이상의 나이와 1등급, 성이 여성만의 승객만 보기
df[ (df['Age'] > 60) & (df['Pclass']==1) & (df['Sex']=='female')]
```

Out[32]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
275	276	1	1	Andrews, Miss. Kornelia Theodosia	female	63.0	1	0	13502	77.9583	D7	S
829	830	1	1	Stone, Mrs. George Nelson (Martha Evelyn)	female	62.0	0	0	113572	80.0000	B28	NaN

In [32]:

```
# 시리즈 불리언으로 저장하고 이를 합쳐서 보기
cond1 = df['Age'] > 60
cond2 = df['Pclass']==1
cond3 = df['Sex']=='female'
```

In [33]:

```
df[ cond1 & cond2 & cond3]
```

Out[33]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
275	276	1	1	Andrews, Miss. Kornelia Theodosia	female	63.0	1	0	13502	77.9583	D7	S
829	830	1	1	Stone, Mrs. George Nelson (Martha Evelyn)	female	62.0	0	0	113572	80.0000	B28	NaN

8. 정렬, Aggregation , GroupBy

8-1 sort_values 함수 적용

In [33]:

```
# 'Name' 열에서 이름으로 정렬하기
df_sorted = df.sort_values(by=['Name'])
df_sorted.head()
```

Out[33]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
845	846	0	3	Abbing, Mr. Anthony	male	42.0	0	0	C.A. 5547	7.55	NaN	S
746	747	0	3	Abbott, Mr. Rossmore Edward	male	16.0	1	1	C.A. 2673	20.25	NaN	S
279	280	1	3	Abbott, Mrs. Stanton (Rosa Hunt)	female	35.0	1	1	C.A. 2673	20.25	NaN	S
308	309	0	2	Abelson, Mr. Samuel	male	30.0	1	0	P/PP 3381	24.00	NaN	C
874	875	1	2	Abelson, Mrs. Samuel (Hannah Wizosky)	female	28.0	1	0	P/PP 3381	24.00	NaN	C

In [34]:

```
# 'Pclass'와 Name'으로 내림차순으로 정렬, ascending=False
df_sorted = df.sort_values(by=['Pclass', 'Name'], ascending=False)
df_sorted.head()
```

Out[34]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
868	869	0	3	van Melkebeke, Mr. Philemon	male	NaN	0	0	345777	9.5	NaN	S
153	154	0	3	van Billiard, Mr. Austin Blyler	male	40.5	0	2	A/5. 851	14.5	NaN	S
282	283	0	3	de Pelsmaeker, Mr. Alfons	male	16.0	0	0	345778	9.5	NaN	S
286	287	1	3	de Mulder, Mr. Theodore	male	30.0	0	0	345774	9.5	NaN	S
559	560	1	3	de Messemaeker, Mrs. Guillaume Joseph (Emma)	female	36.0	1	0	345572	17.4	NaN	S

8-2 count, sum 등 Aggregation 함수 적용

In [35]:

```
# count 메서드로 values count
df.count()
```

Out[35]:

```
PassengerId    891
Survived       891
Pclass         891
Name           891
Sex            891
Age            714
SibSp          891
Parch          891
Ticket         891
Fare           891
Cabin          204
Embarked       889
dtype: int64
```

In [36]:

```
# 'Age'와 'Fare' 열의 평균값
df[['Age', 'Fare']].mean()
```

Out[36]:

```
Age     29.699118
Fare    32.204208
dtype: float64
```

8-3 groupby( ) 적용

In [38]:

```
# 'Pclass'로 groupby
df_groupby = df.groupby(by='Pclass')
print(type(df_groupby))
```

```
<class 'pandas.core.groupby.groupby.DataFrameGroupBy'>
```

In [39]:

```
# 'Pclass'로 그룹 지었을 때 class에 따른 values를 카운트하기
df_groupby = df.groupby('Pclass').count()
df_groupby
```

Out[39]:

Pclass	PassengerId	Survived	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	216	216	216	216	186	216	216	216	216	176	214
2	184	184	184	184	173	184	184	184	184	16	184
3	491	491	491	491	355	491	491	491	491	12	491

In [40]:

```
# 'Pclass'로 그룹짓기후에 'PassengerId'와 'Survived' 열의 values만 카운팅하기
df_groupby = df.groupby('Pclass')[['PassengerId', 'Survived']].count()
df_groupby
```

Out[40]:

Pclass	PassengerId	Survived
1	216	216
2	184	184
3	491	491

In [41]:

```
df.groupby('Pclass')['Age'].agg([max, min])
```

Out[41]:

Pclass	max	min
1	80.0	0.92
2	70.0	0.67
3	74.0	0.42

In [42]:

```
# 딕셔너리로 행과 열 인덱스 부여하기
agg_format = {'Age':'max', 'SibSp':'sum', 'Fare':'mean'}
df.groupby('Pclass').agg(agg_format)
```

Out[42]:

Pclass	Age	SibSp	Fare
1	80.0	90	84.154687
2	70.0	74	20.662183
3	74.0	302	13.675550

9. 결측 데이터 처리하기

9-1 isna( )로 결손 데이터 확인

In [43]:

```
# isna() 메서드로 불리언 데이터프레임 생성, True는 결측치
df.isna().head()
```

Out[43]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	False	False	False	False	False	False	False	False	False	False	True	False
1	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	True	False
3	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	True	False

In [45]:

```
# sum()으로 결측치 갯수 합계보기
df.isna( ).sum( )
```

Out[45]:

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

9-2 fillna( ) 로 Missing 데이터 대체하기

In [46]:

```
# 'Cabin' 열에 결측치에 C000 채우기
df['Cabin'] = df['Cabin'].fillna('C000')
df.head()
```

Out[46]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	C000	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	C000	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	C000	S

In [47]:

```
# 'Age' 열의 결측치에 mean 값 채우기
df['Age'] = df['Age'].fillna(df['Age'].mean())
```

In [49]:

```
# 'Embarked'' 열의 결측치에 문자열 'S' 채우기
df['Embarked'] = df['Embarked'].fillna('S')
```

In [50]:

```
df.isna().sum()
```

Out[50]:

```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
dtype: int64
```

10. apply, lambda로 데이터 가공

In [51]:

```
def get_square(a):
    return a**2

print('3의 제곱은:',get_square(3))
```

3의 제곱은: 9

In [52]:

```
lambda_square = lambda x: x ** 2
print('3의 제곱은:',lambda_square(3))
```

3의 제곱은: 9

In [53]:

```
# 'Name_len' 열을 생성하고, 'Name'의 문자열 길이를 넣기
df['Name_len']= df['Name'].apply(lambda x: len(x))
df[['Name','Name_len']].head()
```

Out[53]:

	Name	Name_len
0	Braund, Mr. Owen Harris	23
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	51
2	Heikkinen, Miss. Laina	22
3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	44
4	Allen, Mr. William Henry	24

In [54]:

```
#' 'Child_Adult' 열을 생성하고 Child를 넣어주고, 만약 15세 이상이면 Adult 문자열 삽입
df['Child_Adult'] = df['Age'].apply(lambda x: 'Child' if x <=15 else 'Adult' )
df[['Age','Child_Adult']].head(10)
```

Out[54]:

	Age	Child_Adult
0	22.000000	Adult
1	38.000000	Adult
2	26.000000	Adult
3	35.000000	Adult
4	35.000000	Adult
5	29.699118	Adult
6	54.000000	Adult
7	2.000000	Child
8	27.000000	Adult
9	14.000000	Child

In [57]:

```
# 'Age_Cat' 열을 생성하고 15세 이상은 "adult", 15세 이하는 "Child", 60세 이상은 "Elderly" 채우기
df['Age_cat'] = df['Age'].apply(lambda x: 'Child' if x <= 15 else ('Adult' if x <= 60 else 'Elderly'))
```

In [59]:

```
# value_counts()로 갯수 확인
df['Age_cat'].value_counts()
```

Out[59]:

```
Adult      786
Child       83
Elderly     22
Name: Age_cat, dtype: int64
```

In [ ]:

```

```