데이터프레임과 시리즈

- 나만의 데이터 만들기
- 시리즈 다루기 – 기초
- 시리즈 다루기 – 응용
- 데이터프레임 다루기
- 시리즈와 데이터프레임 데이터 처리하기
- 데이터 저장하고 불러오기

■ 나만의 데이터 만들기

● 시리즈 만들기

- Series() 메소드에 리스트를 전달하여 시리즈 생성

```
s = pd.Series(['Wes McKinney', 'Creator of Pandas'])
print(s)
```

```
0          Wes McKinney
1    Creator of Pandas
dtype: object
```

- 인덱스는 보통 0부터 시작하는 정수형태이지만 문자열로 사용 가능
- index 인자에 문자열을 리스트로 담아서 전달

```
s = pd.Series(['Wes McKinney', 'Creator of Pandas'], index=['Person', 'Who'])
print(s)
```

```
Person          Wes McKinney
Who        Creator of Pandas
dtype: object
```

■ 나만의 데이터 만들기

● 데이터프레임 만들기

- DataFrame() 메소드에 딕셔너리를 전달하여 데이터프레임 생성

```
scientists = pd.DataFrame({
    'Name': ['Rosaline Franklin', 'William Gosset'],
    'Occupation': ['Chemist', 'Statistician'],
    'Born': ['1920-07-25', '1876-06-13'],
    'Died': ['1958-04-16', '1937-10-16'],
    'Age': [37, 61]})
print(scientists)
```

```
                Name    Occupation        Born        Died  Age
0  Rosaline Franklin       Chemist  1920-07-25  1958-04-16   37
1     William Gosset  Statistician  1876-06-13  1937-10-16   61
```

■ 나만의 데이터 만들기

● 데이터프레임 만들기

- DataFrame() 메소드에 딕셔너리를 전달하여 데이터프레임 생성

```
scientists = pd.DataFrame(
    data={'Occupation': ['Chemist', 'Statistician'],
          'Born': ['1920-07-25', '1876-06-13'],
          'Died': ['1958-04-16', '1937-10-16'],
          'Age': [37, 61]},
    index=['Rosaline Franklin', 'William Gosset'],
    columns=['Occupation', 'Born', 'Age', 'Died'])
print(scientists)
```

```
                  Occupation        Born  Age        Died
Rosaline Franklin      Chemist  1920-07-25   37  1958-04-16
William Gosset    Statistician  1876-06-13   61  1937-10-16
```

■ 시리즈 다루기 – 기초

● 데이터프레임에서 시리즈 선택하기

```
scientists = pd.DataFrame(
    data={'Occupation': ['Chemist', 'Statistician'],
          'Born': ['1920-07-25', '1876-06-13'],
          'Died': ['1958-04-16', '1937-10-16'],
          'Age': [37, 61]},
    index=['Rosaline Franklin', 'William Gosset'],
    columns=['Occupation', 'Born', 'Died', 'Age'])

first_row = scientists.loc['William Gosset']
print(type(first_row))
print(first_row)
```

```
<class 'pandas.core.series.Series'>
Occupation    Statistician
Born            1876-06-13
Age                     61
Died            1937-10-16
Name: William Gosset, dtype: object
```

■ 시리즈 다루기 – 기초

● index 속성 사용하기

```
print(first_row.index)
```

```
Index(['Occupation', 'Born', 'Age', 'Died'], dtype='object')
```

● values 속성 사용하기

```
print(first_row.values)
```

```
['Statistician' '1876-06-13' '1937-10-16' 61]
```

● keys() 메소드 사용하기 (== index 속성)

```
print(first_row.keys())
```

```
Index(['Occupation', 'Born', 'Age', 'Died'], dtype='object')
```

■ 시리즈 다루기 – 기초

● 기초 통계 메소드 사용하기

- scientists 데이터프레임의 Age 열 추출

```
ages = scientists['Age']
print(ages)
```

```
Rosaline Franklin    37
William Gosset       61
Name: Age, dtype: int64
```

- mean(), min(), max(), std()

```
print(ages.mean())
print(ages.min())
print(ages.max())
print(ages.std())
```

```
49.0
37
61
16.97056274847714
```

■ 시리즈 다루기 – 기초

● 시리즈 관련 메소드

시리즈 메소드	설명
append	2개 이상의 시리즈 연결
describe	요약 통계량 계산
drop_duplicates	중복값이 없는 시리즈 반환
equals	시리즈에 해당 값을 가진 요소가 있는지 확인
get_values	시리즈 값 구하기 (values 속성과 동일)
isin	시리즈에 포함된 값이 있는지 확인
min	최소값 반환
max	최대값 반환
mean	산술 평균 반환

■ 시리즈 다루기 – 응용

● 평균 나이보다 나이가 많은 사람 데이터 구하기

- 최대값, 평균 메소드로 값 확인

```
scientists = pd.read_csv('data/scientists.csv')
ages = scientists['Age']
print(ages.max(), ages.mean())
```

```
90 59.125
```

- Boolean Array 추출

```
print(ages > ages.mean())
```

```
0    False
1     True
2     True
3     True
4    False
5    False
6    False
7     True
Name: Age, dtype: bool
```

1, 2, 3, 7 인덱스의 데이터가 참인 것을 확인

■ 시리즈 다루기 – 응용

● 평균 나이보다 나이가 많은 사람 데이터 구하기

- 리스트 형태로 참이나 거짓을 담아 시리즈에 전달하면 해당 데이터만 추출

```
manual_bool_values = [False, True, True, True, False, False, False, True]
print(ages[manual_bool_values])
```

```
1    61
2    90
3    66
7    77
Name: Age, dtype: int64
```

- Boolean Array 추출 후 시리즈로 전달

```
print(ages[ages > ages.mean()])
```

```
1    61
2    90
3    66
7    77
Name: Age, dtype: int64
```

■ 시리즈 다루기 – 응용

● 브로드캐스팅 (Broadcasting)

- 시리즈와 같이 여러 개의 값을 가진 데이터(벡터)와
  단순 크기를 나타내는 데이터(스칼라) 간의 연산을 지원하는 것

```
print(ages + ages)
```

```
0     74
1    122
2    180
3    132
4    112
5     90
6     82
7    154
Name: Age, dtype: int64
```

벡터 + 벡터

```
print(ages + 100)
```

```
0    137
1    161
2    190
3    166
4    156
5    145
6    141
7    177
Name: Age, dtype: int64
```

벡터 + 스칼라

■ 시리즈 다루기 – 응용

- 길이가 서로 다른 벡터를 연산하는 경우 ➔ 같은 인덱스의 값만 계산

```
print(ages + pd.Series([1, 100]))
```

```
0     38.0
1    161.0
2      NaN
3      NaN
4      NaN
5      NaN
6      NaN
7      NaN
dtype: float64
```

0, 1 인덱스만 계산되고 나머지는 누락값(NaN)으로 처리

■ 시리즈 다루기 – 응용

● sort_index([ascending=False])

- 인덱스 정렬

```
rev_ages = ages.sort_index(ascending=False)
print(rev_ages)
```

```
7    77
6    41
5    45
4    56
3    66
2    90
1    61
0    37
Name: Age, dtype: int64
```

인덱스 역순 정렬

```
print(ages * 2)
```

```
0     74
1    122
2    180
3    132
4    112
5     90
6     82
7    154
Name: Age, dtype: int64
```

결과 동일

```
print(ages + rev_ages)
```

같은 인덱스끼리 연산

```
0     74
1    122
2    180
3    132
4    112
5     90
6     82
7    154
Name: Age, dtype: int64
```

■ 데이터프레임 다루기

● Boolean Array

- 데이터프레임의 Age 열의 평균보다 높은 데이터 출력

```
scientists = pd.read_csv('data/scientists.csv')
print(scientists[scientists['Age'] > scientists['Age'].mean()])
```

```
                   Name        Born        Died  Age     Occupation
1         William Gosset  1876-06-13  1937-10-16   61   Statistician
2  Florence Nightingale  1820-05-12  1910-08-13   90          Nurse
3           Marie Curie  1867-11-07  1934-07-04   66        Chemist
7          Johann Gauss  1777-04-30  1855-02-23   77  Mathematician
```

- 2, 6번 인덱스를 제외한 데이터 출력

```
print(scientists[[True, True, False, True, True, True, False, True]])
```

```
                   Name        Born        Died  Age     Occupation
0  Rosaline Franklin  1920-07-25  1958-04-16   37        Chemist
1     William Gosset  1876-06-13  1937-10-16   61   Statistician
3        Marie Curie  1867-11-07  1934-07-04   66        Chemist
4      Rachel Carson  1907-05-27  1964-04-14   56      Biologist
5          John Snow  1813-03-15  1858-06-16   45      Physician
7       Johann Gauss  1777-04-30  1855-02-23   77  Mathematician
```

■ 데이터프레임 다루기

● 브로드캐스팅 (Broadcasting)

- 시리즈와 같이 모든 요소를 대상으로 연산
- 2를 곱하면 숫자는 2를 곱하고 문자열은 2배로 늘어남

```
print(scientists * 2)
```

```
                                      Name                   Born  \
0        Rosaline FranklinRosaline Franklin  1920-07-251920-07-25
1              William GossetWilliam Gosset  1876-06-131876-06-13
2  Florence NightingaleFlorence Nightingale  1820-05-121820-05-12
3                    Marie CurieMarie Curie  1867-11-071867-11-07
4                Rachel CarsonRachel Carson  1907-05-271907-05-27
5                      John SnowJohn Snow  1813-03-151813-03-15
6                    Alan TuringAlan Turing  1912-06-231912-06-23
7                  Johann GaussJohann Gauss  1777-04-301777-04-30

                    Died  Age                          Occupation
0  1958-04-161958-04-16   74                    ChemistChemist
1  1937-10-161937-10-16  122              StatisticianStatistician
2  1910-08-131910-08-13  180                          NurseNurse
3  1934-07-041934-07-04  132                    ChemistChemist
4  1964-04-141964-04-14  112                BiologistBiologist
5  1858-06-161858-06-16   90                PhysicianPhysician
6  1954-06-071954-06-07   82  Computer ScientistComputer Scientist
7  1855-02-231855-02-23  154            MathematicianMathematician
```

■ 시리즈와 데이터프레임의 데이터 처리하기

● 열의 자료형 바꾸기

- scientists 데이터프레임의 Born과 Died 열의 자료형 확인

```
print(scientists['Born'].dtype)
print(scientists['Died'].dtype)
```

```
object
object
```

판다스에서 문자열은 오브젝트로 취급

- 'Born' 열의 날짜 형식 문자열을 datetime 자료형으로 변경

```
born_datetime = pd.to_datetime(scientists['Born'], format='%Y-%m-%d')
print(born_datetime)
```

```
0   1920-07-25
1   1876-06-13
2   1820-05-12
3   1867-11-07
4   1907-05-27
5   1813-03-15
6   1912-06-23
7   1777-04-30
Name: Born, dtype: datetime64[ns]
```

■ 시리즈와 데이터프레임의 데이터 처리하기

● 열의 자료형 바꾸기

- 'Died' 열의 날짜 형식 문자열을 datetime 자료형으로 변경

```
died_datetime = pd.to_datetime(scientists['Died'], format='%Y-%m-%d')
print(died_datetime)
```

```
0   1958-04-16
1   1937-10-16
2   1910-08-13
3   1934-07-04
4   1964-04-14
5   1858-06-16
6   1954-06-07
7   1855-02-23
Name: Died, dtype: datetime64[ns]
```

■ 시리즈와 데이터프레임의 데이터 처리하기

● 열의 자료형 바꾸기

- datetime 으로 바뀐 2개의 자료를 새로운 열로 추가

```
scientists['born_dt'], scientists['died_dt'] = (born_datetime, died_datetime)
print(scientists.head())
```

```
                  Name        Born        Died  Age    Occupation    born_dt  \
0     Rosaline Franklin  1920-07-25  1958-04-16   37       Chemist 1920-07-25
1        William Gosset  1876-06-13  1937-10-16   61  Statistician 1876-06-13
2  Florence Nightingale  1820-05-12  1910-08-13   90         Nurse 1820-05-12
3          Marie Curie  1867-11-07  1934-07-04   66       Chemist 1867-11-07
4         Rachel Carson  1907-05-27  1964-04-14   56     Biologist 1907-05-27

     died_dt
0 1958-04-16
1 1937-10-16
2 1910-08-13
3 1934-07-04
4 1964-04-14
```

■ 시리즈와 데이터프레임의 데이터 처리하기

● 열의 자료형 바꾸기

- 과학자 삶의 시간 계산하기 (died_dt – born_dt)

```
scientists['age_days_dt'] = (scientists['died_dt'] - scientists['born_dt'])
print(scientists)
```

```
                  Name        Born        Died  Age          Occupation  \
0     Rosaline Franklin  1920-07-25  1958-04-16   37             Chemist
1       William Gosset  1876-06-13  1937-10-16   61         Statistician
2  Florence Nightingale  1820-05-12  1910-08-13   90                Nurse
3          Marie Curie  1867-11-07  1934-07-04   66             Chemist
4        Rachel Carson  1907-05-27  1964-04-14   56            Biologist
5            John Snow  1813-03-15  1858-06-16   45            Physician
6          Alan Turing  1912-06-23  1954-06-07   41  Computer Scientist
7         Johann Gauss  1777-04-30  1855-02-23   77       Mathematician

     born_dt    died_dt age_days_dt
0 1920-07-25 1958-04-16  13779 days
1 1876-06-13 1937-10-16  22404 days
2 1820-05-12 1910-08-13  32964 days
3 1867-11-07 1934-07-04  24345 days
4 1907-05-27 1964-04-14  20777 days
5 1813-03-15 1858-06-16  16529 days
6 1912-06-23 1954-06-07  15324 days
7 1777-04-30 1855-02-23  28422 days
```

■ 시리즈와 데이터프레임의 데이터 처리하기

● 데이터 삭제하기 - drop(['인덱스' 또는 '열 이름'][, axis=n])

- 인덱스(행) 삭제

```
scientists.drop(0).head(2)
```

	Name	Born	Died	Age	Occupation	born_dt	died_dt	age_days_dt
1	William Gosset	1876-06-13	1937-10-16	37	Statistician	1876-06-13	1937-10-16	22404 days
2	Florence Nightingale	1820-05-12	1910-08-13	61	Nurse	1820-05-12	1910-08-13	32964 days

- 여러 개의 인덱스(행) 삭제

```
scientists.drop([0, 1]).head(2)
```

	Name	Born	Died	Age	Occupation	born_dt	died_dt	age_days_dt
2	Florence Nightingale	1820-05-12	1910-08-13	61	Nurse	1820-05-12	1910-08-13	32964 days
3	Marie Curie	1867-11-07	1934-07-04	77	Chemist	1867-11-07	1934-07-04	24345 days

■ 시리즈와 데이터프레임의 데이터 처리하기

● 데이터 삭제하기 - drop(['인덱스' 또는 '열 이름'][, axis=n])

- 열 삭제

```
scientists.drop('Age', axis=1).head(2)
```

	Name	Born	Died	Occupation	born_dt	died_dt	age_days_dt
0	Rosaline Franklin	1920-07-25	1958-04-16	Chemist	1920-07-25	1958-04-16	13779 days
1	William Gosset	1876-06-13	1937-10-16	Statistician	1876-06-13	1937-10-16	22404 days

- 여러 개의 열 삭제

```
scientists.drop(['Age', 'Occupation'], axis=1).head(2)
```

	Name	Born	Died	born_dt	died_dt	age_days_dt
0	Rosaline Franklin	1920-07-25	1958-04-16	1920-07-25	1958-04-16	13779 days
1	William Gosset	1876-06-13	1937-10-16	1876-06-13	1937-10-16	22404 days

■ 데이터 저장하고 불러오기

● pickle

- 데이터를 바이너리 형태로 직렬화 하여 저장
- 적은 용량으로 저장 가능

```
names.to_pickle('scientists_names_series.pickle')
```

scientists_names_series.pickle

```
1  ...pandas.
2  _new_Index....pandas.cor
3  RangeIndex...}.(.name.
4  Rachel Carson..  John Snow.
5  h...}.(h.Nh..K.h.K.h.K.u..R.
6  .-blocks.].}.(.-values.
```

시리즈 저장

```
scientists.to_pickle('scientists_df.pickle')
```

scientists_df.pickle

```
1  ...pandas.
2  _new_Index...h..|Index..
3  Occupation..•born_dt..•di
4  .pandas.core.indexes.range
5  RangeIndex...}.(h.N.|sta
6  Rachel Carson..  John Snow.
```

데이터프레임 저장

■ 데이터 저장하고 불러오기

● pickle

- 바이너리 형태로 저장된 데이터 불러오기

```
pd.read_pickle('scientists_names_series.pickle')
```

```
0       Rosaline Franklin
1          William Gosset
2    Florence Nightingale
3            Marie Curie
4          Rachel Carson
5              John Snow
6            Alan Turing
7           Johann Gauss
Name: Name, dtype: object
```

```
pd.read_pickle('scientists_df.pickle')
```

	Name	Born	Died	Age	Occupation	born_dt	died_dt	age_days_dt
0	Rosaline Franklin	1920-07-25	1958-04-16	45	Chemist	1920-07-25	1958-04-16	13779 days
1	William Gosset	1876-06-13	1937-10-16	37	Statistician	1876-06-13	1937-10-16	22404 days
2	Florence Nightingale	1820-05-12	1910-08-13	61	Nurse	1820-05-12	1910-08-13	32964 days
3	Marie Curie	1867-11-07	1934-07-04	77	Chemist	1867-11-07	1934-07-04	24345 days
4	Rachel Carson	1907-05-27	1964-04-14	41	Biologist	1907-05-27	1964-04-14	20777 days
5	John Snow	1813-03-15	1858-06-16	90	Physician	1813-03-15	1858-06-16	16529 days
6	Alan Turing	1912-06-23	1954-06-07	56	Computer Scientist	1912-06-23	1954-06-07	15324 days
7	Johann Gauss	1777-04-30	1855-02-23	66	Mathematician	1777-04-30	1855-02-23	28422 days

■ 데이터 저장하고 불러오기

● CSV (Comma Seperated Variables), TSV (Tab Seperated Variables)

```
scientists['Name'].to_csv('scientist_names_series.csv')
```

scientist_names_series.csv

```
,Name
0,Rosaline Franklin
1,William Gosset
2,Florence Nightingale
3,Marie Curie
4,Rachel Carson
5,John Snow
6,Alan Turing
7,Johann Gauss
```

```
scientists.to_csv('scientists_df.tsv', sep='\t')
```

scientists_df.tsv

	Name	Born	Died	Age	Occupation	born_dt	died_dt	age_days_dt
0	Rosaline Franklin	1920-07-25	1958-04-16	45	Chemist	1920-07-25	195	
1	William Gosset	1876-06-13	1937-10-16	37	Statistician	1876-06-13		
2	Florence Nightingale	1820-05-12	1910-08-13	61	Nurse	1820-05-12		
3	Marie Curie	1867-11-07	1934-07-04	77	Chemist	1867-11-07	1934-07-04	
4	Rachel Carson	1907-05-27	1964-04-14	41	Biologist	1907-05-27	196	
5	John Snow	1813-03-15	1858-06-16	90	Physician	1813-03-15	1858-06	
6	Alan Turing	1912-06-23	1954-06-07	56	Computer Scientist	1912-06-23		
7	Johann Gauss	1777-04-30	1855-02-23	66	Mathematician	1777-04-30		

■ 데이터 저장하고 불러오기

● CSV (Comma Seperated Variables), TSV (Tab Seperated Variables)

```
pd.read_csv('scientist_names_series.csv', index_col=0)
```

0번 컬럼을 인덱스로 지정

	Name
0	Rosaline Franklin
1	William Gosset
2	Florence Nightingale
3	Marie Curie
4	Rachel Carson
5	John Snow
6	Alan Turing
7	Johann Gauss

```
pd.read_csv('scientists_df.tsv', delimiter='\t', index_col=0)
```

	Name	Born	Died	Age	Occupation	born_dt	died_dt	age_days_dt
0	Rosaline Franklin	1920-07-25	1958-04-16	45	Chemist	1920-07-25	1958-04-16	13779 days
1	William Gosset	1876-06-13	1937-10-16	37	Statistician	1876-06-13	1937-10-16	22404 days
2	Florence Nightingale	1820-05-12	1910-08-13	61	Nurse	1820-05-12	1910-08-13	32964 days
3	Marie Curie	1867-11-07	1934-07-04	77	Chemist	1867-11-07	1934-07-04	24345 days
4	Rachel Carson	1907-05-27	1964-04-14	41	Biologist	1907-05-27	1964-04-14	20777 days
5	John Snow	1813-03-15	1858-06-16	90	Physician	1813-03-15	1858-06-16	16529 days
6	Alan Turing	1912-06-23	1954-06-07	56	Computer Scientist	1912-06-23	1954-06-07	15324 days
7	Johann Gauss	1777-04-30	1855-02-23	66	Mathematician	1777-04-30	1855-02-23	28422 days

■ 데이터 저장하고 불러오기

● Excel (저장하기)

- conda install xlwt

- conda install openpyxl

```
(base) C:₩Users₩GGoReb>conda install xlwt
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:₩Users₩GGoReb₩miniconda3

  added / updated specs:
    - xlwt


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    ca-certificates-2021.1.19  |       haa95532_0         122 KB
    jedi-0.17.2                |   py38haa95532_1         921 KB
    xlwt-1.3.0                 |           py38_0         160 KB
    ------------------------------------------------------------
                                           Total:         1.2 MB
```

■ 데이터 저장하고 불러오기

● Excel (저장하기)

```
names_df = scientists['Name'].to_frame()

import xlwt
names_df.to_excel('scientists_names_series_df.xls')

import openpyxl
names_df.to_excel('scientists_names_series_df.xlsx')
```

	A	B
1		**Name**
2	**0**	Rosaline Franklin
3	**1**	William Gosset
4	**2**	Florence Nightingale
5	**3**	Marie Curie
6	**4**	Rachel Carson
7	**5**	John Snow
8	**6**	Alan Turing
9	**7**	Johann Gauss

■ 데이터 저장하고 불러오기

● Excel (저장하기)

```
import xlwt
scientists.to_excel('scientists_df.xls')

import openpyxl
scientists.to_excel('scientists_df.xlsx')
```

	A	B	C	D	E	F	G	H	I
1		**Name**	**Born**	**Died**	**Age**	**Occupation**	**born_dt**	**died_dt**	**ge_days_dt**
2	**0**	Rosaline F	1920-07-2	1958-04-1	45	Chemist	#######	#######	13779
3	**1**	William Gc	1876-06-1	1937-10-1	37	Statistician	#######	#######	22404
4	**2**	Florence N	1820-05-1	1910-08-1	61	Nurse	#######	#######	32964
5	**3**	Marie Curi	1867-11-0	1934-07-0	77	Chemist	#######	#######	24345
6	**4**	Rachel Car	1907-05-2	1964-04-1	41	Biologist	#######	#######	20777
7	**5**	John Snow	1813-03-1	1858-06-1	90	Physician	#######	#######	16529
8	**6**	Alan Turin	1912-06-2	1954-06-0	56	Computer	#######	#######	15324
9	**7**	Johann Ga	1777-04-3	1855-02-2	66	Mathemat	#######	#######	28422

■ 데이터 저장하고 불러오기

● Excel (불러오기)

- conda install xlrd

```
(base) C:₩Users₩GGoReb>conda install xlrd
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:₩Users₩GGoReb₩miniconda3

  added / updated specs:
    - xlrd


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    xlrd-2.0.1                 |     pyhd3eb1b0_0          90 KB
    ------------------------------------------------------------
                                           Total:          90 KB
```

■ 데이터 저장하고 불러오기

● Excel (불러오기)

```
pd.read_excel('scientists_names_series_df.xls', index_col=0)

pd.read_excel('scientists_df.xls', index_col=0)
```

	Name	Born	Died	Age	Occupation	born_dt	died_dt	age_days_dt
0	Rosaline Franklin	1920-07-25	1958-04-16	45	Chemist	1920-07-25	1958-04-16	13779
1	William Gosset	1876-06-13	1937-10-16	37	Statistician	1876-06-13	1937-10-16	22404
2	Florence Nightingale	1820-05-12	1910-08-13	61	Nurse	1820-05-12	1910-08-13	32964
3	Marie Curie	1867-11-07	1934-07-04	77	Chemist	1867-11-07	1934-07-04	24345
4	Rachel Carson	1907-05-27	1964-04-14	41	Biologist	1907-05-27	1964-04-14	20777
5	John Snow	1813-03-15	1858-06-16	90	Physician	1813-03-15	1858-06-16	16529
6	Alan Turing	1912-06-23	1954-06-07	56	Computer Scientist	1912-06-23	1954-06-07	15324
7	Johann Gauss	1777-04-30	1855-02-23	66	Mathematician	1777-04-30	1855-02-23	28422