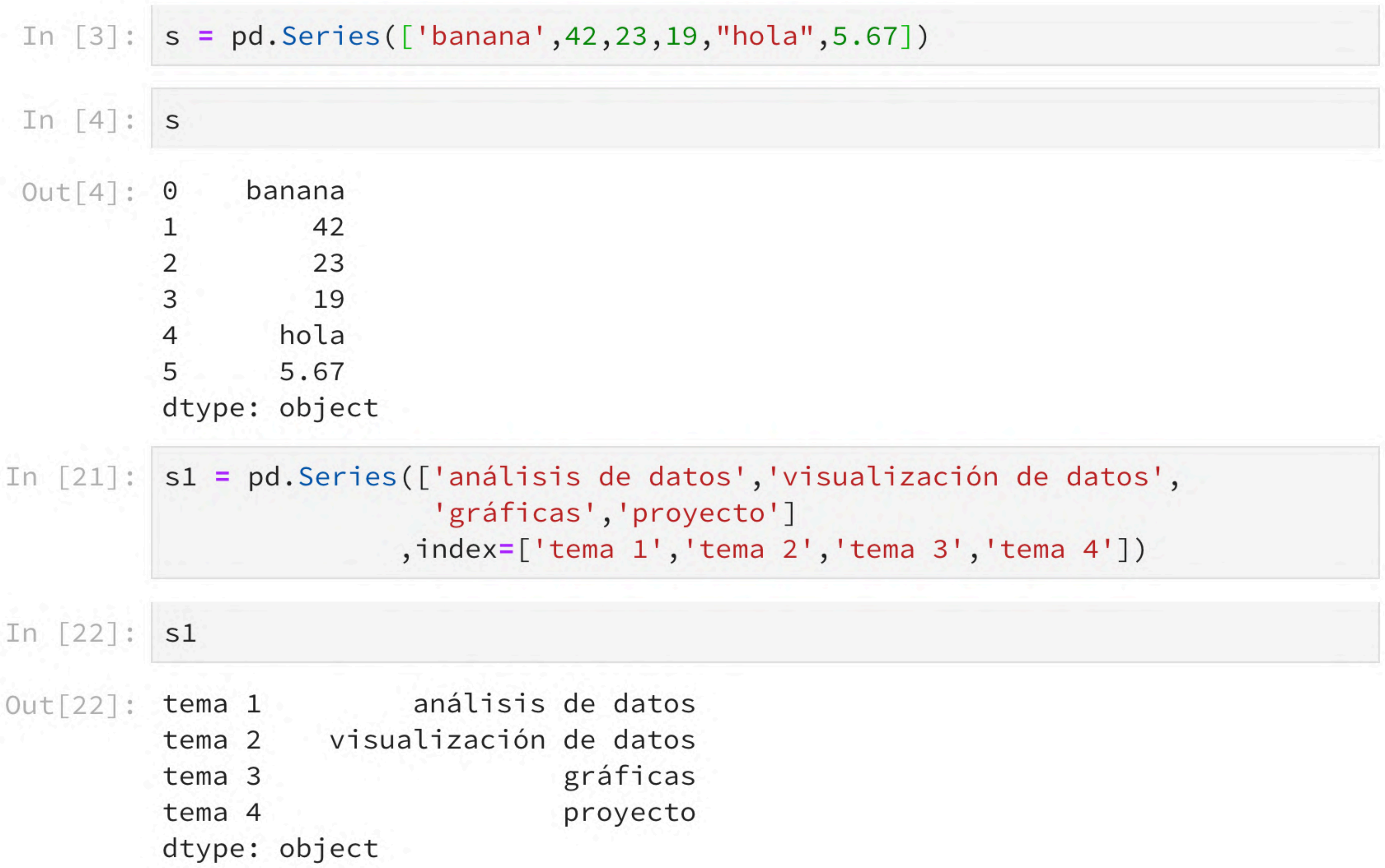
¿Cómo crear Series?



```
In [3]:  s = pd.Series(['banana',42,23,19,"hola",5.67])
```

```
In [4]:  s
```

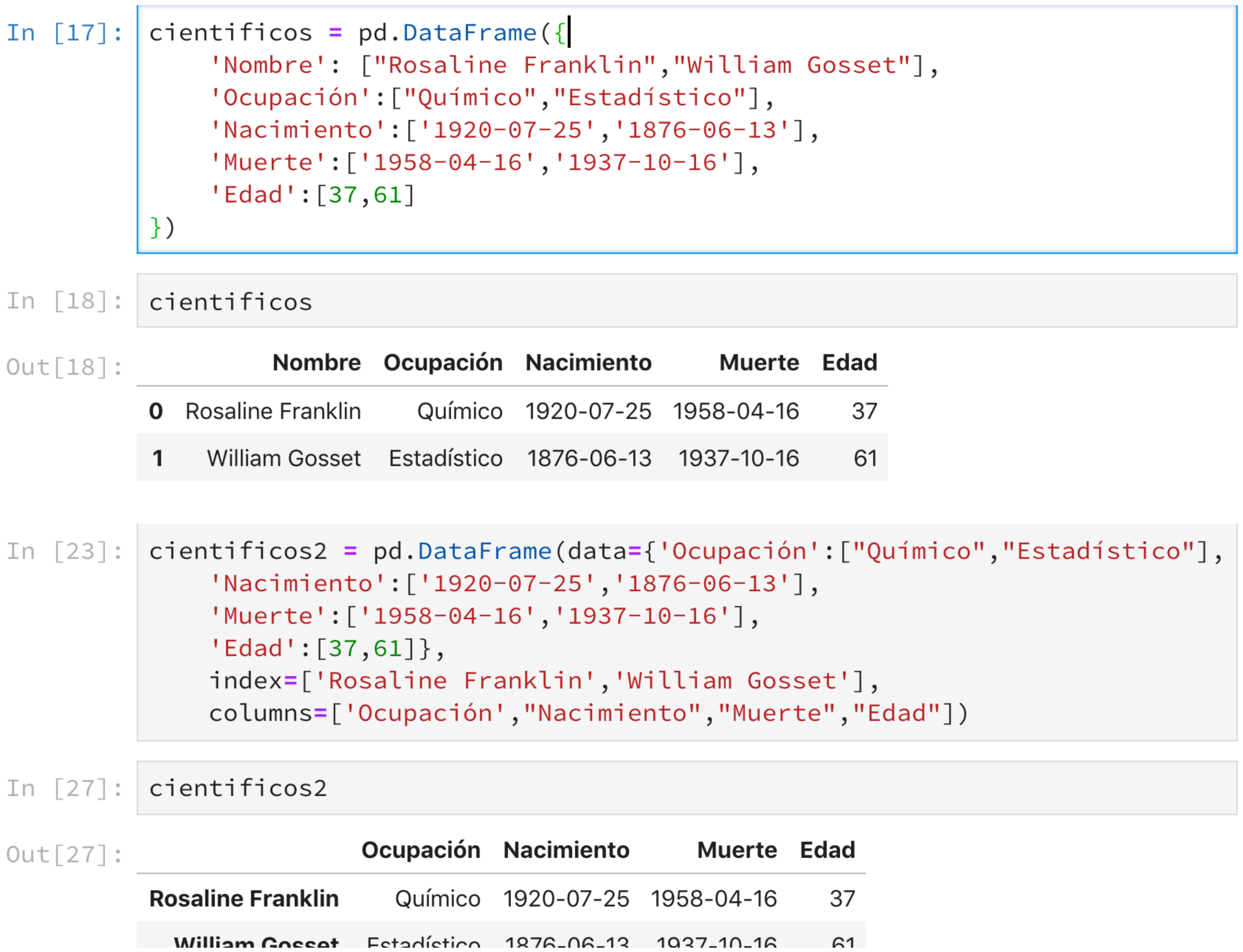
```
Out[4]: 0    banana
        1        42
        2        23
        3        19
        4      hola
        5      5.67
        dtype: object
```

```
In [21]: s1 = pd.Series(['análisis de datos','visualización de datos',
                         'gráficas','proyecto']
                       ,index=['tema 1','tema 2','tema 3','tema 4'])
```

```
In [22]: s1
```

```
Out[22]: tema 1          análisis de datos
         tema 2     visualización de datos
         tema 3                   gráficas
         tema 4                   proyecto
         dtype: object
```

¿Cómo crear DataFrames?



```
In [17]: cientificos = pd.DataFrame({
             'Nombre': ["Rosaline Franklin","William Gosset"],
             'Ocupación':["Químico","Estadístico"],
             'Nacimiento':['1920-07-25','1876-06-13'],
             'Muerte':['1958-04-16','1937-10-16'],
             'Edad':[37,61]
         })

In [18]: cientificos

Out[18]:
                Nombre    Ocupación   Nacimiento      Muerte   Edad
   0  Rosaline Franklin      Químico  1920-07-25  1958-04-16     37
   1     William Gosset  Estadístico  1876-06-13  1937-10-16     61

In [23]: cientificos2 = pd.DataFrame(data={'Ocupación':["Químico","Estadístico"],
             'Nacimiento':['1920-07-25','1876-06-13'],
             'Muerte':['1958-04-16','1937-10-16'],
             'Edad':[37,61]},
             index=['Rosaline Franklin','William Gosset'],
             columns=['Ocupación',"Nacimiento","Muerte","Edad"])

In [27]: cientificos2

Out[27]:
                       Ocupación   Nacimiento      Muerte   Edad
   Rosaline Franklin     Químico  1920-07-25  1958-04-16     37
      William Gosset Estadístico  1876-06-13  1937-10-16     61
```

In [30]:
```
cientificos3 = pd.read_csv("scientists.csv",index_col="Name")
```

In [31]:
```
cientificos3
```

Out[31]:

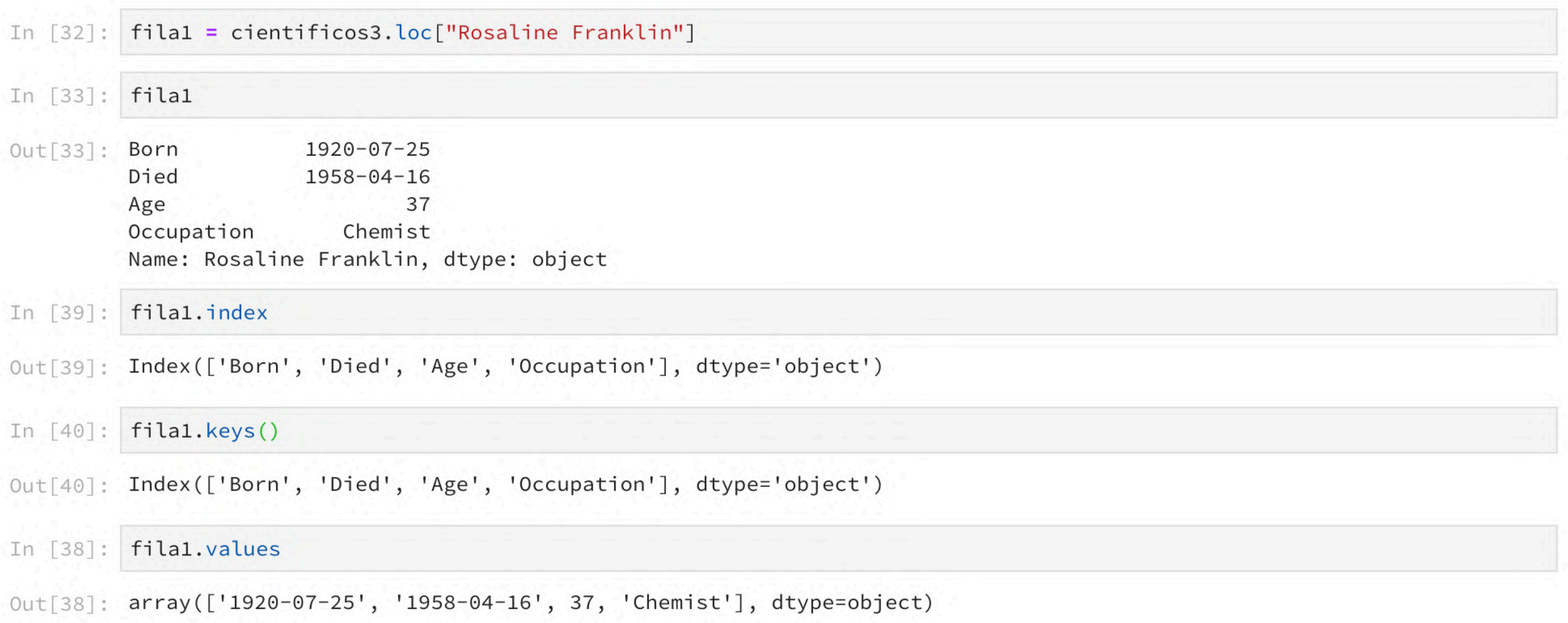
Name	Born	Died	Age	Occupation
Rosaline Franklin	1920-07-25	1958-04-16	37	Chemist
William Gosset	1876-06-13	1937-10-16	61	Statistician
Florence Nightingale	1820-05-12	1910-08-13	90	Nurse
Marie Curie	1867-11-07	1934-07-04	66	Chemist
Rachel Carson	1907-05-27	1964-04-14	56	Biologist
John Snow	1813-03-15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

In [32]:
```
fila1 = cientificos3.loc["Rosaline Franklin"]
```

In [33]:
```
fila1
```

Out[33]:
```
Born            1920-07-25
Died            1958-04-16
Age                     37
Occupation         Chemist
Name: Rosaline Franklin, dtype: object
```

Elementos de la Serie



```
In [32]: fila1 = cientificos3.loc["Rosaline Franklin"]
```

```
In [33]: fila1
```

```
Out[33]: Born          1920-07-25
         Died          1958-04-16
         Age                   37
         Occupation       Chemist
         Name: Rosaline Franklin, dtype: object
```

```
In [39]: fila1.index
```

```
Out[39]: Index(['Born', 'Died', 'Age', 'Occupation'], dtype='object')
```

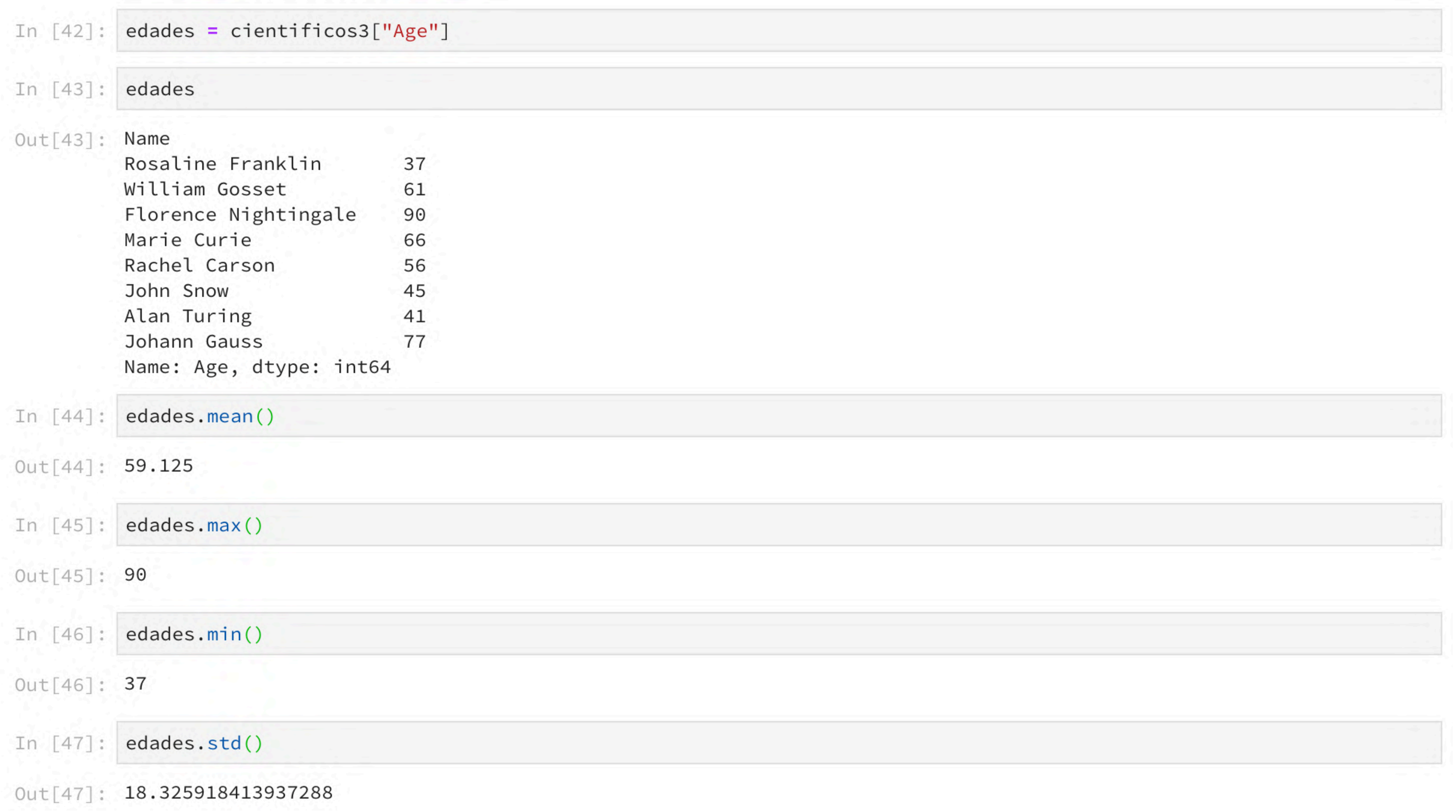
```
In [40]: fila1.keys()
```

```
Out[40]: Index(['Born', 'Died', 'Age', 'Occupation'], dtype='object')
```

```
In [38]: fila1.values
```

```
Out[38]: array(['1920-07-25', '1958-04-16', 37, 'Chemist'], dtype=object)
```

Operaciones básicas sobre una Serie



```
In [42]: edades = cientificos3["Age"]
```

```
In [43]: edades
```

```
Out[43]: Name
         Rosaline Franklin       37
         William Gosset          61
         Florence Nightingale    90
         Marie Curie             66
         Rachel Carson           56
         John Snow               45
         Alan Turing             41
         Johann Gauss            77
         Name: Age, dtype: int64
```

```
In [44]: edades.mean()
```

```
Out[44]: 59.125
```

```
In [45]: edades.max()
```

```
Out[45]: 90
```

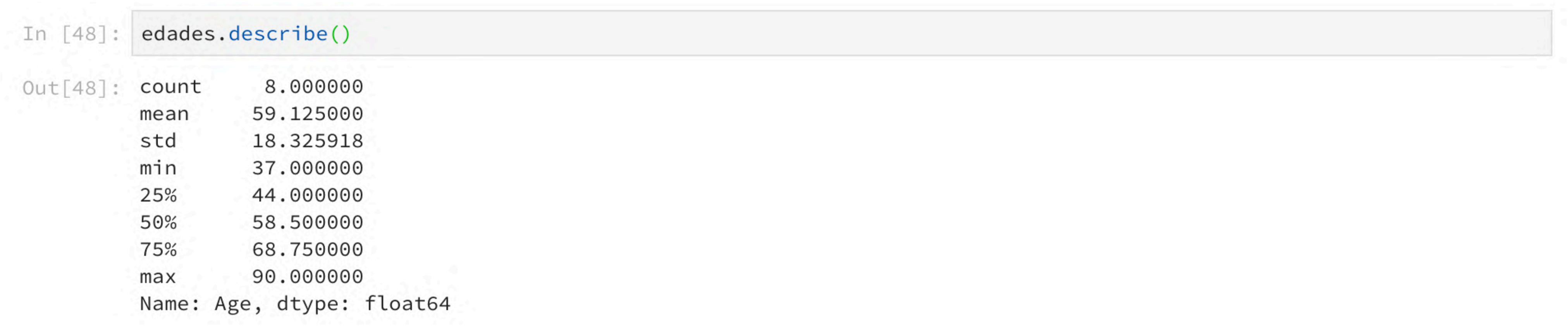
```
In [46]: edades.min()
```

```
Out[46]: 37
```

```
In [47]: edades.std()
```

```
Out[47]: 18.325918413937288
```

Series.describe()

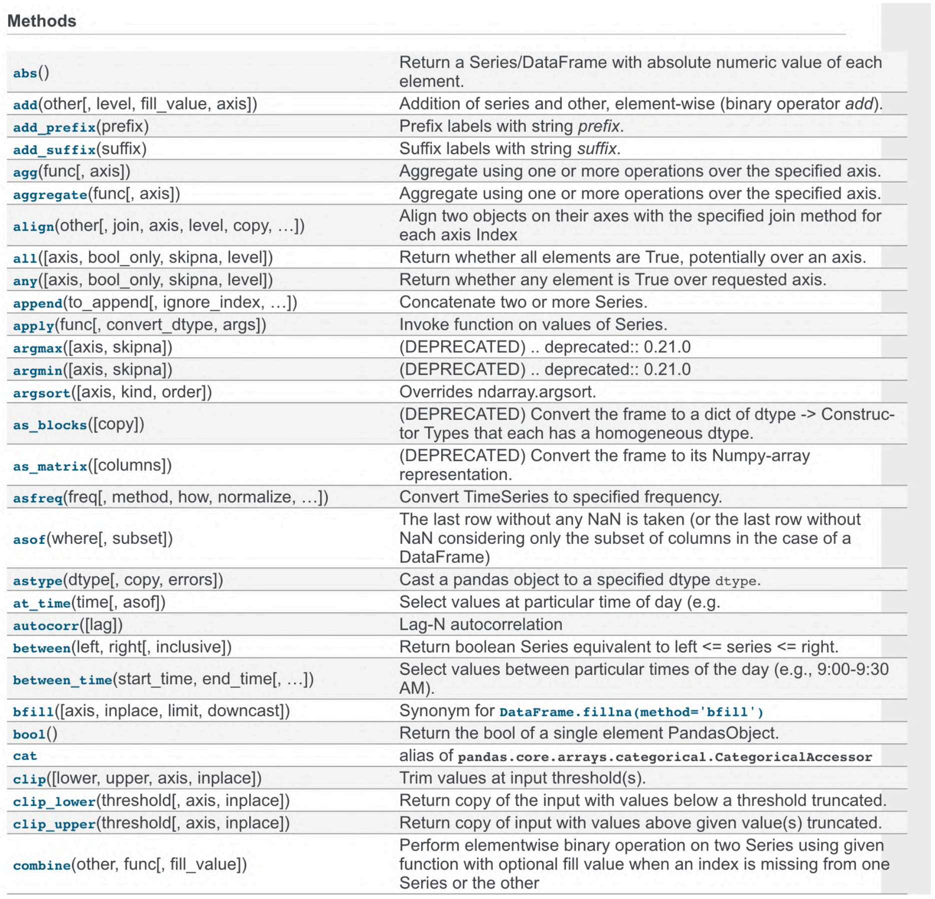


```
In [48]: edades.describe()

Out[48]: count     8.000000
         mean     59.125000
         std      18.325918
         min      37.000000
         25%      44.000000
         50%      58.500000
         75%      68.750000
         max      90.000000
         Name: Age, dtype: float64
```

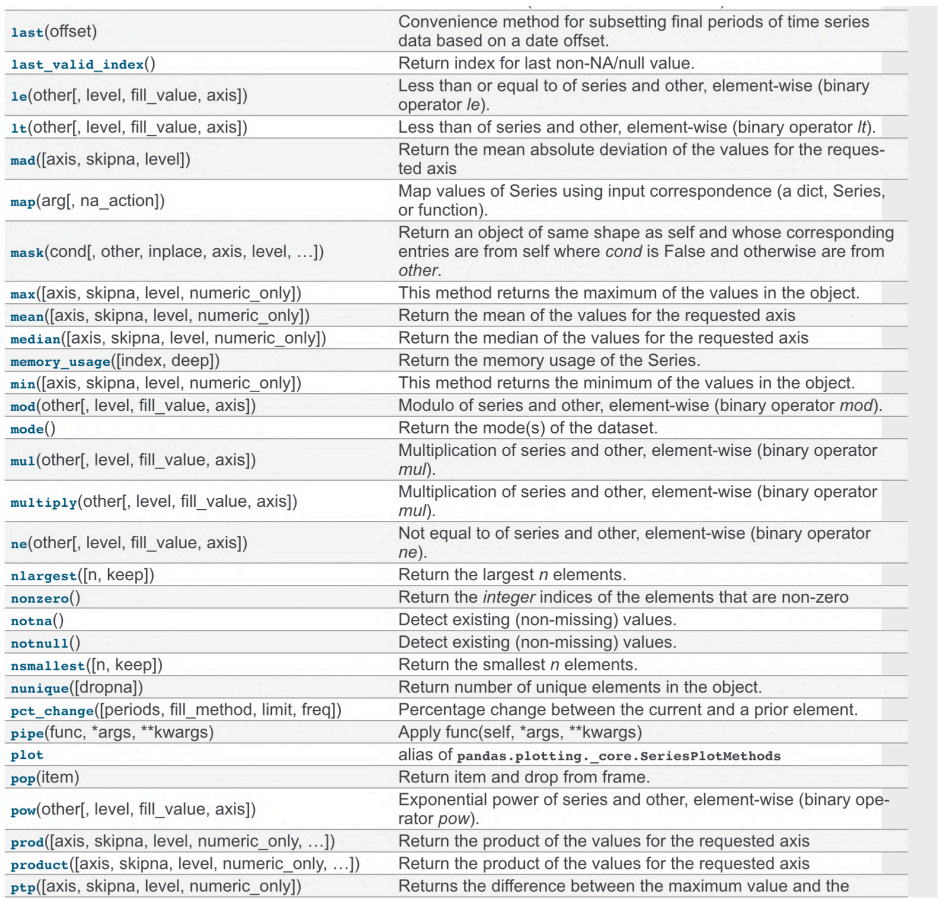
Documentación de Series

https://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.html



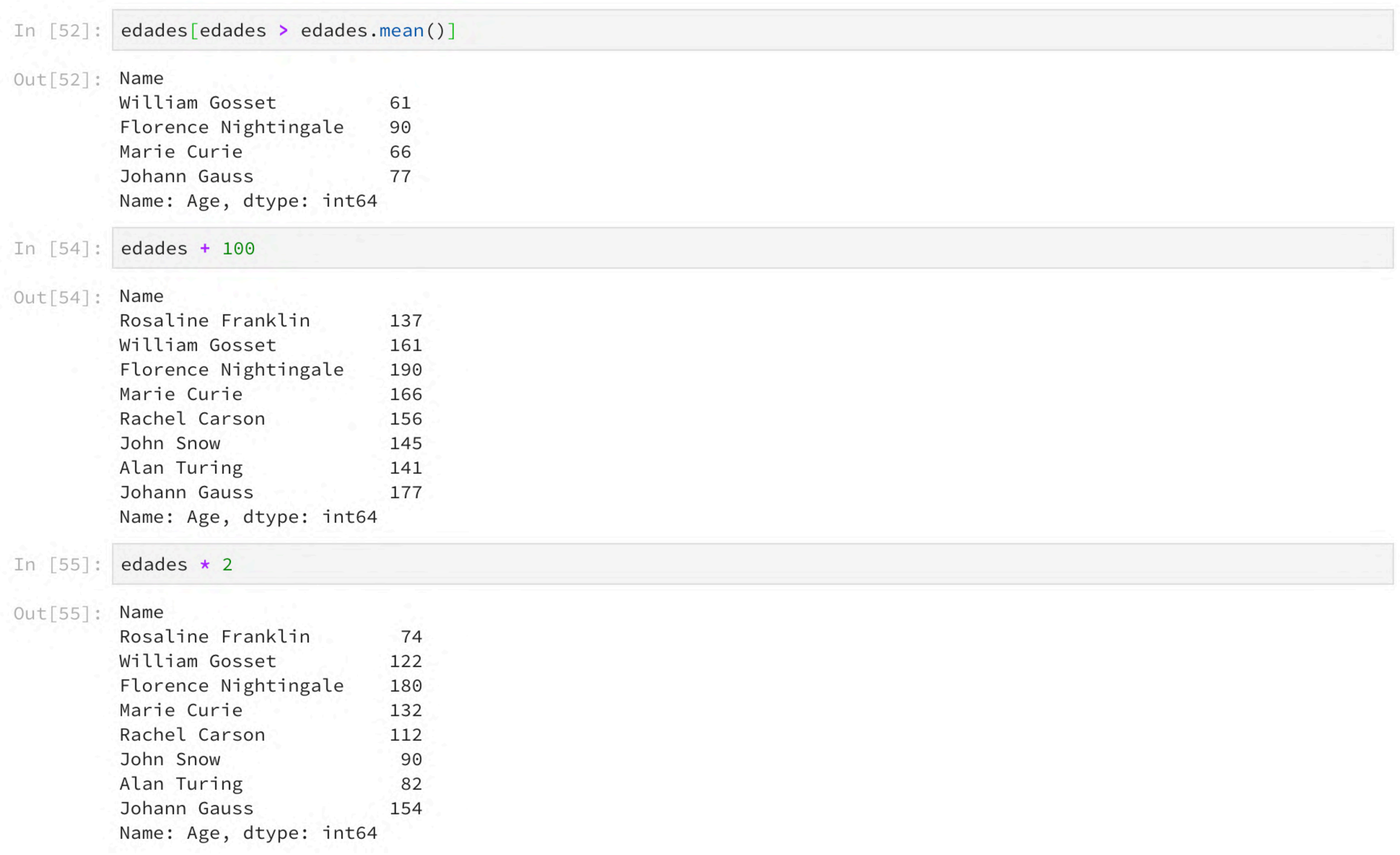
Methods

`abs`()	Return a Series/DataFrame with absolute numeric value of each element.
`add`(other[, level, fill_value, axis])	Addition of series and other, element-wise (binary operator *add*).
`add_prefix`(prefix)	Prefix labels with string *prefix*.
`add_suffix`(suffix)	Suffix labels with string *suffix*.
`agg`(func[, axis])	Aggregate using one or more operations over the specified axis.
`aggregate`(func[, axis])	Aggregate using one or more operations over the specified axis.
`align`(other[, join, axis, level, copy, …])	Align two objects on their axes with the specified join method for each axis Index
`all`([axis, bool_only, skipna, level])	Return whether all elements are True, potentially over an axis.
`any`([axis, bool_only, skipna, level])	Return whether any element is True over requested axis.
`append`(to_append[, ignore_index, …])	Concatenate two or more Series.
`apply`(func[, convert_dtype, args])	Invoke function on values of Series.
`argmax`([axis, skipna])	(DEPRECATED) .. deprecated:: 0.21.0
`argmin`([axis, skipna])	(DEPRECATED) .. deprecated:: 0.21.0
`argsort`([axis, kind, order])	Overrides ndarray.argsort.
`as_blocks`([copy])	(DEPRECATED) Convert the frame to a dict of dtype -> Constructor Types that each has a homogeneous dtype.
`as_matrix`([columns])	(DEPRECATED) Convert the frame to its Numpy-array representation.
`asfreq`(freq[, method, how, normalize, …])	Convert TimeSeries to specified frequency.
`asof`(where[, subset])	The last row without any NaN is taken (or the last row without NaN considering only the subset of columns in the case of a DataFrame)
`astype`(dtype[, copy, errors])	Cast a pandas object to a specified dtype `dtype`.
`at_time`(time[, asof])	Select values at particular time of day (e.g.
`autocorr`([lag])	Lag-N autocorrelation
`between`(left, right[, inclusive])	Return boolean Series equivalent to left <= series <= right.
`between_time`(start_time, end_time[, …])	Select values between particular times of the day (e.g., 9:00-9:30 AM).
`bfill`([axis, inplace, limit, downcast])	Synonym for `DataFrame.fillna(method='bfill')`
`bool`()	Return the bool of a single element PandasObject.
`cat`	alias of `pandas.core.arrays.categorical.CategoricalAccessor`
`clip`([lower, upper, axis, inplace])	Trim values at input threshold(s).
`clip_lower`(threshold[, axis, inplace])	Return copy of the input with values below a threshold truncated.
`clip_upper`(threshold[, axis, inplace])	Return copy of input with values above given value(s) truncated.
`combine`(other, func[, fill_value])	Perform elementwise binary operation on two Series using given function with optional fill value when an index is missing from one Series or the other



`last`(offset)	Convenience method for subsetting final periods of time series data based on a date offset.
`last_valid_index`()	Return index for last non-NA/null value.
`le`(other[, level, fill_value, axis])	Less than or equal to of series and other, element-wise (binary operator *le*).
`lt`(other[, level, fill_value, axis])	Less than of series and other, element-wise (binary operator *lt*).
`mad`([axis, skipna, level])	Return the mean absolute deviation of the values for the requested axis
`map`(arg[, na_action])	Map values of Series using input correspondence (a dict, Series, or function).
`mask`(cond[, other, inplace, axis, level, …])	Return an object of same shape as self and whose corresponding entries are from self where *cond* is False and otherwise are from *other*.
`max`([axis, skipna, level, numeric_only])	This method returns the maximum of the values in the object.
`mean`([axis, skipna, level, numeric_only])	Return the mean of the values for the requested axis
`median`([axis, skipna, level, numeric_only])	Return the median of the values for the requested axis
`memory_usage`([index, deep])	Return the memory usage of the Series.
`min`([axis, skipna, level, numeric_only])	This method returns the minimum of the values in the object.
`mod`(other[, level, fill_value, axis])	Modulo of series and other, element-wise (binary operator *mod*).
`mode`()	Return the mode(s) of the dataset.
`mul`(other[, level, fill_value, axis])	Multiplication of series and other, element-wise (binary operator *mul*).
`multiply`(other[, level, fill_value, axis])	Multiplication of series and other, element-wise (binary operator *mul*).
`ne`(other[, level, fill_value, axis])	Not equal to of series and other, element-wise (binary operator *ne*).
`nlargest`([n, keep])	Return the largest *n* elements.
`nonzero`()	Return the *integer* indices of the elements that are non-zero
`notna`()	Detect existing (non-missing) values.
`notnull`()	Detect existing (non-missing) values.
`nsmallest`([n, keep])	Return the smallest *n* elements.
`nunique`([dropna])	Return number of unique elements in the object.
`pct_change`([periods, fill_method, limit, freq])	Percentage change between the current and a prior element.
`pipe`(func, *args, **kwargs)	Apply func(self, *args, **kwargs)
`plot`	alias of `pandas.plotting._core.SeriesPlotMethods`
`pop`(item)	Return item and drop from frame.
`pow`(other[, level, fill_value, axis])	Exponential power of series and other, element-wise (binary operator *pow*).
`prod`([axis, skipna, level, numeric_only, …])	Return the product of the values for the requested axis
`product`([axis, skipna, level, numeric_only, …])	Return the product of the values for the requested axis
`ptp`([axis, skipna, level, numeric_only])	Returns the difference between the maximum value and the

Operaciones con Series



```
In [52]: edades[edades > edades.mean()]

Out[52]: Name
         William Gosset           61
         Florence Nightingale     90
         Marie Curie              66
         Johann Gauss             77
         Name: Age, dtype: int64

In [54]: edades + 100

Out[54]: Name
         Rosaline Franklin       137
         William Gosset          161
         Florence Nightingale    190
         Marie Curie             166
         Rachel Carson           156
         John Snow               145
         Alan Turing             141
         Johann Gauss            177
         Name: Age, dtype: int64

In [55]: edades * 2

Out[55]: Name
         Rosaline Franklin        74
         William Gosset          122
         Florence Nightingale    180
         Marie Curie             132
         Rachel Carson           112
         John Snow                90
         Alan Turing              82
         Johann Gauss            154
         Name: Age, dtype: int64
```

```
In [50]: edades + edades
```

```
Out[50]: Name
         Rosaline Franklin       74
         William Gosset         122
         Florence Nightingale   180
         Marie Curie            132
         Rachel Carson          112
         John Snow               90
         Alan Turing             82
         Johann Gauss           154
         Name: Age, dtype: int64
```

```
In [51]: edades * edades
```

```
Out[51]: Name
         Rosaline Franklin       1369
         William Gosset          3721
         Florence Nightingale    8100
         Marie Curie             4356
         Rachel Carson           3136
         John Snow               2025
         Alan Turing             1681
         Johann Gauss            5929
         Name: Age, dtype: int64
```

```
In [57]: edades + pd.Series([1,99],index=["Johann Gauss","John Snow"])
```

```
Out[57]: Alan Turing               NaN
         Florence Nightingale      NaN
         Johann Gauss             78.0
         John Snow               144.0
         Marie Curie               NaN
         Rachel Carson             NaN
         Rosaline Franklin         NaN
         William Gosset            NaN
         dtype: float64
```

```
In [58]: edades
```

```
Out[58]: Name
         Rosaline Franklin      37
         William Gosset         61
         Florence Nightingale   90
         Marie Curie            66
         Rachel Carson          56
         John Snow              45
         Alan Turing            41
         Johann Gauss           77
         Name: Age, dtype: int64
```

```
In [68]: edades_ordenado = edades.sort_index(ascending = False)
```

```
In [69]: edades_ordenado
```

```
Out[69]: Name
         William Gosset         61
         Rosaline Franklin      37
         Rachel Carson          56
         Marie Curie            66
         John Snow              45
         Johann Gauss           77
         Florence Nightingale   90
         Alan Turing            41
         Name: Age, dtype: int64
```

```
In [81]: edades + edades_ordenado
```

```
Out[81]: Name
         Alan Turing             82
         Florence Nightingale   180
         Johann Gauss           154
         John Snow               90
         Marie Curie            132
         Rachel Carson          112
         Rosaline Franklin       74
         William Gosset         122
```

```
In [83]: ocupaciones = cientificos3["Occupation"]
```

```
In [85]: ocupaciones
```

```
Out[85]: Name
Rosaline Franklin                    Chemist
William Gosset                  Statistician
Florence Nightingale                   Nurse
Marie Curie                          Chemist
Rachel Carson                      Biologist
John Snow                          Physician
Alan Turing               Computer Scientist
Johann Gauss                   Mathematician
Name: Occupation, dtype: object
```

```
In [89]: ocupaciones + " Hola"
```

```
Out[89]: Name
Rosaline Franklin                    Chemist Hola
William Gosset                  Statistician Hola
Florence Nightingale                   Nurse Hola
Marie Curie                          Chemist Hola
Rachel Carson                      Biologist Hola
John Snow                          Physician Hola
Alan Turing               Computer Scientist Hola
Johann Gauss                   Mathematician Hola
Name: Occupation, dtype: object
```

```
In [99]: ocupaciones * 2
```

```
Out[99]: Name
Rosaline Franklin                                   ChemistChemist
William Gosset                            StatisticianStatistician
Florence Nightingale                                    NurseNurse
Marie Curie                                         ChemistChemist
Rachel Carson                                   BiologistBiologist
John Snow                                       PhysicianPhysician
Alan Turing               Computer ScientistComputer Scientist
Johann Gauss                              MathematicianMathematician
```

```
In [148]: ocupaciones + ocupaciones
```

```
Out[148]: Name
Rosaline Franklin                              ChemistChemist
William Gosset                       StatisticianStatistician
Florence Nightingale                               NurseNurse
Marie Curie                                    ChemistChemist
Rachel Carson                              BiologistBiologist
John Snow                                  PhysicianPhysician
Alan Turing                Computer ScientistComputer Scientist
Johann Gauss                     MathematicianMathematician
Name: Occupation, dtype: object
```

```
In [149]: ocupaciones * ocupaciones
```

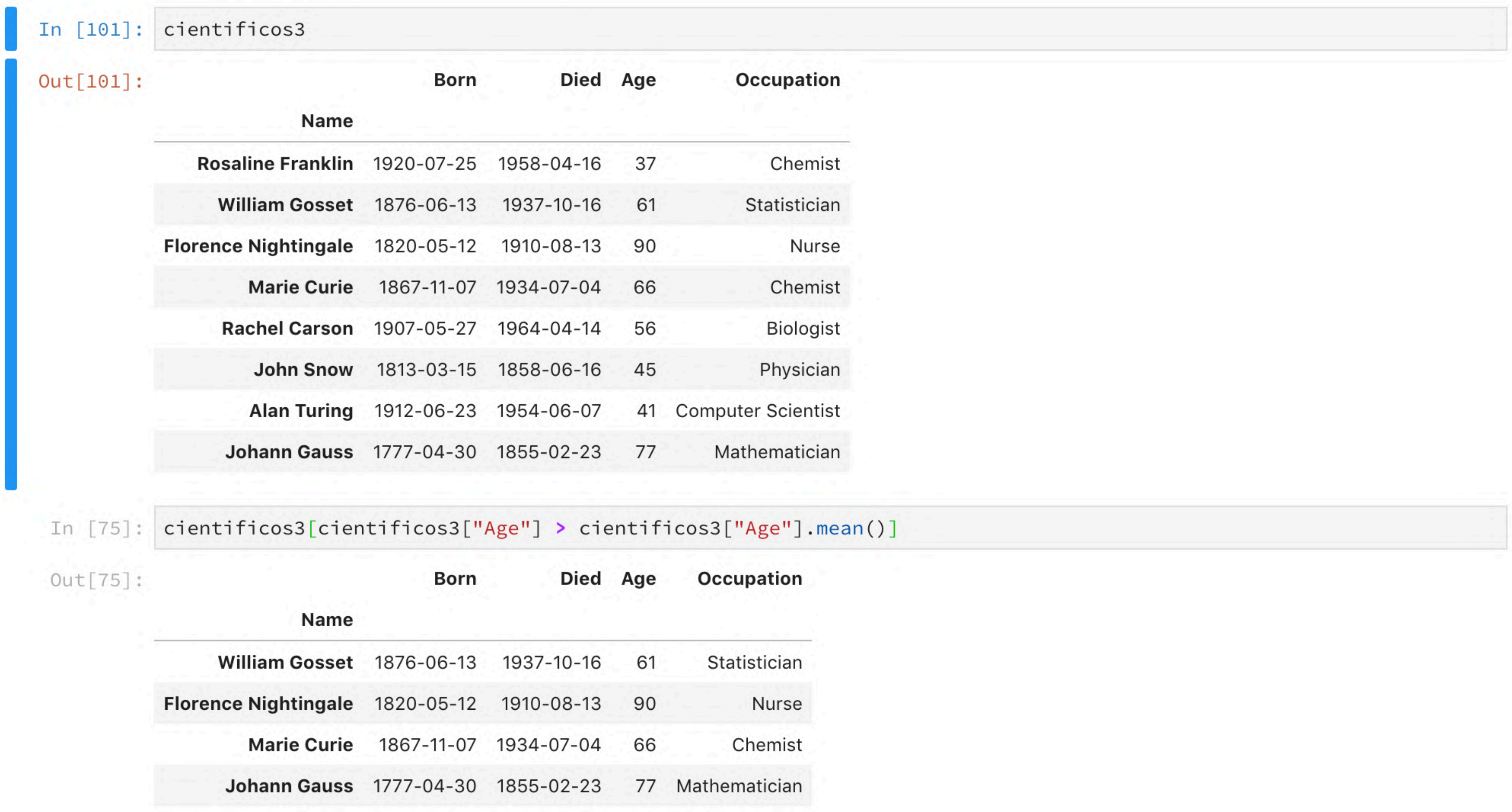
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/anaconda3/lib/python3.6/site-packages/pandas/core/ops.py in na_op(x, y)
   1008         try:
-> 1009             result = expressions.evaluate(op, str_rep, x, y, **eval_kwargs)
   1010         except TypeError:

/anaconda3/lib/python3.6/site-packages/pandas/core/computation/expressions.py in evaluate(op, op_str, a, b, use_n
umexpr, **eval_kwargs)
    204     if use_numexpr:
--> 205         return _evaluate(op, op_str, a, b, **eval_kwargs)
    206     return _evaluate_standard(op, op_str, a, b)

/anaconda3/lib/python3.6/site-packages/pandas/core/computation/expressions.py in _evaluate_numexpr(op, op_str, a,
 b, truediv, reversed, **eval_kwargs)
    119     if result is None:
--> 120         result = _evaluate_standard(op, op_str, a, b)
    121 

/anaconda3/lib/python3.6/site-packages/pandas/core/computation/expressions.py in _evaluate_standard(op, op_str,
 a, b, **eval_kwargs)
     64     with np.errstate(all='ignore'):
---> 65         return op(a, b)
```

Operaciones con DataFrames



In [101]:
```
cientificos3
```

Out[101]:

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

In [75]:
```
cientificos3[cientificos3["Age"] > cientificos3["Age"].mean()]
```

Out[75]:

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

In [102]:
```
cientificos3 + 2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/anaconda3/lib/python3.6/site-packages/pandas/core/ops.py in na_op(x, y)
   1466         try:
-> 1467             result = expressions.evaluate(op, str_rep, x, y, **eval_kwargs)
   1468         except TypeError:

/anaconda3/lib/python3.6/site-packages/pandas/core/computation/expressions.py in evaluate(op, op_str, a, b, use_n
umexpr, **eval_kwargs)
    204     if use_numexpr:
--> 205         return _evaluate(op, op_str, a, b, **eval_kwargs)
    206     return _evaluate_standard(op, op_str, a, b)
```

In [104]:
```
cientificos3 * 2
```

Out[104]:

	Born	Died	Age	Occupation
Name				
Rosaline Franklin	1920-07-251920-07-25	1958-04-161958-04-16	74	ChemistChemist
William Gosset	1876-06-131876-06-13	1937-10-161937-10-16	122	StatisticianStatistician
Florence Nightingale	1820-05-121820-05-12	1910-08-131910-08-13	180	NurseNurse
Marie Curie	1867-11-071867-11-07	1934-07-041934-07-04	132	ChemistChemist
Rachel Carson	1907-05-271907-05-27	1964-04-141964-04-14	112	BiologistBiologist
John Snow	1813-03-151813-03-15	1858-06-161858-06-16	90	PhysicianPhysician
Alan Turing	1912-06-231912-06-23	1954-06-071954-06-07	82	Computer ScientistComputer Scientist
Johann Gauss	1777-04-301777-04-30	1855-02-231855-02-23	154	MathematicianMathematician

In [71]:
```
cientificos3a = cientificos3[:4]
cientificos3b = cientificos3[4:]
```

In [72]:
```
cientificos3a
```

Out[72]:

	Born	Died	Age	Occupation
Name				
Rosaline Franklin	1920-07-25	1958-04-16	37	Chemist
William Gosset	1876-06-13	1937-10-16	61	Statistician
Florence Nightingale	1820-05-12	1910-08-13	90	Nurse
Marie Curie	1867-11-07	1934-07-04	66	Chemist

In [73]:
```
cientificos3b
```

Out[73]:

	Born	Died	Age	Occupation
Name				
Rachel Carson	1907-05-27	1964-04-14	56	Biologist
John Snow	1813-03-15	1858-06-16	45	Physician
Alan Turing	1912-06-23	1954-06-07	41	Computer Scientist
Johann Gauss	1777-04-30	1855-02-23	77	Mathematician

In [108]:
```
cientificos3a + cientificos3b
```

Out[108]:

	Born	Died	Age	Occupation
Name				
Alan Turing	NaN	NaN	NaN	NaN
Florence Nightingale	NaN	NaN	NaN	NaN
Johann Gauss	NaN	NaN	NaN	NaN

```
In [109]: nacimientos = cientificos3["Born"]
```

```
In [110]: nacimientos
```

```
Out[110]: Name
          Rosaline Franklin       1920-07-25
          William Gosset          1876-06-13
          Florence Nightingale    1820-05-12
          Marie Curie             1867-11-07
          Rachel Carson           1907-05-27
          John Snow               1813-03-15
          Alan Turing             1912-06-23
          Johann Gauss            1777-04-30
          Name: Born, dtype: object
```

```
In [113]: nac_fecha = pd.to_datetime(nacimientos, format = "%Y-%m-%d")
```

```
In [114]: nac_fecha
```

```
Out[114]: Name
          Rosaline Franklin      1920-07-25
          William Gosset         1876-06-13
          Florence Nightingale   1820-05-12
          Marie Curie            1867-11-07
          Rachel Carson          1907-05-27
          John Snow              1813-03-15
          Alan Turing            1912-06-23
          Johann Gauss           1777-04-30
          Name: Born, dtype: datetime64[ns]
```

```
In [116]: muerte_fecha = pd.to_datetime(cientificos3["Died"],format="%Y-%m-%d")
```

```
In [117]: muerte_fecha
```

```
Out[117]: Name
          Rosaline Franklin      1958-04-16
          William Gosset         1937-10-16
          Florence Nightingale   1910-08-13
          Marie Curie            1934-07-04
          Rachel Carson          1964-04-14
          John Snow              1858-06-16
          Alan Turing            1954-06-07
          Johann Gauss           1855-02-23
          Name: Died, dtype: datetime64[ns]
```

```
In [118]: cientificos3["born_date"],cientificos3["dead_date"]=(nac_fecha,muerte_fecha)
```

```
In [119]: cientificos3
```

Out[119]:

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

```
In [123]: edad_calculada = cientificos3["dead_date"] - cientificos3["born_date"]
```

```
In [125]: edad_calculada_y = edad_calculada.astype('timedelta64[Y]')
```

```
In [126]: cientificos3["edad_c"] = edad_calculada_y
```

```
In [127]: cientificos3
```

Out[127]:

	Born	Died	Age	Occupation	born_date	dead_date	edad_c
Name							
Rosaline Franklin	1920-07-25	1958-04-16	37	Chemist	1920-07-25	1958-04-16	37.0
William Gosset	1876-06-13	1937-10-16	61	Statistician	1876-06-13	1937-10-16	61.0
Florence Nightingale	1820-05-12	1910-08-13	90	Nurse	1820-05-12	1910-08-13	90.0
Marie Curie	1867-11-07	1934-07-04	66	Chemist	1867-11-07	1934-07-04	66.0
Rachel Carson	1907-05-27	1964-04-14	56	Biologist	1907-05-27	1964-04-14	56.0
John Snow	1813-03-15	1858-06-16	45	Physician	1813-03-15	1858-06-16	45.0
Alan Turing	1912-06-23	1954-06-07	41	Computer Scientist	1912-06-23	1954-06-07	41.0
Johann Gauss	1777-04-30	1855-02-23	77	Mathematician	1777-04-30	1855-02-23	77.0

In [128]:
```
cientificos3.drop(["John Snow"])
```

Out[128]:

	Born	Died	Age	Occupation	born_date	dead_date	edad_c
Name							
Rosaline Franklin	1920-07-25	1958-04-16	37	Chemist	1920-07-25	1958-04-16	37.0
William Gosset	1876-06-13	1937-10-16	61	Statistician	1876-06-13	1937-10-16	61.0
Florence Nightingale	1820-05-12	1910-08-13	90	Nurse	1820-05-12	1910-08-13	90.0
Marie Curie	1867-11-07	1934-07-04	66	Chemist	1867-11-07	1934-07-04	66.0
Rachel Carson	1907-05-27	1964-04-14	56	Biologist	1907-05-27	1964-04-14	56.0
Alan Turing	1912-06-23	1954-06-07	41	Computer Scientist	1912-06-23	1954-06-07	41.0
Johann Gauss	1777-04-30	1855-02-23	77	Mathematician	1777-04-30	1855-02-23	77.0

In [134]:
```
cientificos3.drop(["Born","Died"],axis=1)
```

Out[134]:

	Age	Occupation	born_date	dead_date	edad_c
Name					
Rosaline Franklin	37	Chemist	1920-07-25	1958-04-16	37.0
William Gosset	61	Statistician	1876-06-13	1937-10-16	61.0
Florence Nightingale	90	Nurse	1820-05-12	1910-08-13	90.0
Marie Curie	66	Chemist	1867-11-07	1934-07-04	66.0
Rachel Carson	56	Biologist	1907-05-27	1964-04-14	56.0
John Snow	45	Physician	1813-03-15	1858-06-16	45.0
Alan Turing	41	Computer Scientist	1912-06-23	1954-06-07	41.0
Johann Gauss	77	Mathematician	1777-04-30	1855-02-23	77.0