

# Python\_Pandas2

January 22, 2026



## 0.1 PAO 25-25- Pandas 2



Nombre: *Gabriel Salguero*

```
[1]: import pandas as pd  
import numpy as np
```

```
[2]: pd.__version__
```

```
[2]: '2.3.3'
```

## 1 Operaciones en pandas

### Búsqueda

```
[3]: rand_matrix = np.random.randint(6,size=(2,3))  
frame = pd.DataFrame(rand_matrix , columns=list('ABC'))  
display(frame)
```

```
      A   B   C  
0    2   3   2  
1    1   1   2
```

```
[4]: # buscando columnas (DataFrame como dic, busca en claves)  
'A' in frame
```

```
[4]: True
```

```
[5]: # buscando valores  
display(frame.isin([3,2])) # --> mask de respuesta (valores que son 3 o 2)
```

```
      A      B      C  
0  True   True   True  
1 False  False  True
```

```
[6]: # Contar el número de ocurrencias  
print(frame.isin([4]).values.sum())  
display(frame.isin([4])) #devuelve una mask con valores true si el elemento es 4  
print(frame.isin([4]).values)  
type(frame.isin([4]).values) # pandas es una capa alrededor de numpy
```

```
0
```

```
      A      B      C  
0 False  False  False  
1 False  False  False
```

```
[[False False False]  
 [False False False]]
```

```
[6]: numpy.ndarray
```

```
[7]: # Cuántos valores son >= 2  
mask = frame >= 2  
print(mask.values.sum())
```

```
display(mask)
```

4

	A	B	C
0	True	True	True
1	False	False	True

## Ordenación

```
[8]: from random import shuffle
```

```
rand_matrix = np.random.randint(20, size=(5,4))
```

```
indices = list(range(5))
shuffle(indices) # mezcla indices
```

```
frame = pd.DataFrame(rand_matrix, columns=list('DACP'), index=indices)
display(frame)
```

	D	A	C	B
2	7	11	15	2
0	2	8	12	14
4	10	5	1	15
3	18	3	10	5
1	2	11	16	14

```
[9]: # ordenar por índice
```

```
display(frame.sort_index(ascending=False))
display(frame)
```

	D	A	C	B
4	10	5	1	15
3	18	3	10	5
2	7	11	15	2
1	2	11	16	14
0	2	8	12	14

  

	D	A	C	B
2	7	11	15	2
0	2	8	12	14
4	10	5	1	15
3	18	3	10	5
1	2	11	16	14

```
[10]: # ordenar por columna
```

```
display(frame.sort_index(axis=1, ascending=False))
```

	D	C	B	A
2	7	15	2	11
0	2	12	14	8

```
4 10 1 15 5  
3 18 10 5 3  
1 2 16 14 11
```

```
[11]: # ordenar filas por valor en columna  
display(frame.sort_values(by='A', ascending=True))
```

```
D A C B  
3 18 3 10 5  
4 10 5 1 15  
0 2 8 12 14  
2 7 11 15 2  
1 2 11 16 14
```

```
[12]: # ordenar columnas por valor en fila  
display(frame.sort_values(by=1, axis=1, ascending=True))
```

```
D A B C  
2 7 11 2 15  
0 2 8 14 12  
4 10 5 15 1  
3 18 3 5 10  
1 2 11 14 16
```

```
[13]: # ordenar por valor en columna y guardar cambios  
frame.sort_values(by='A', ascending=False, inplace=True)  
display(frame)
```

```
D A C B  
2 7 11 15 2  
1 2 11 16 14  
0 2 8 12 14  
4 10 5 1 15  
3 18 3 10 5
```

## Ranking

- Construir un ranking de valores

```
[14]: display(frame)
```

```
D A C B  
2 7 11 15 2  
1 2 11 16 14  
0 2 8 12 14  
4 10 5 1 15  
3 18 3 10 5
```

```
[15]: display(frame.rank(method='max', axis=1))
```

	D	A	C	B
2	2.0	3.0	4.0	1.0
1	1.0	2.0	4.0	3.0
0	1.0	2.0	3.0	4.0
4	3.0	2.0	1.0	4.0
3	4.0	1.0	3.0	2.0

```
[16]: # Imprimir, uno a uno, los valores de la columna 'C' de mayor a menor
for x in frame.sort_values(by='C', ascending=False)['C'].values:
    print(x)
```

16  
15  
12  
10  
1

## 2 Operaciones

Operaciones matemáticas entre objetos

```
[17]: matrixA = np.random.randint(100,size=(4,4))
matrixB = np.random.randint(100,size=(4,4))
frameA = pd.DataFrame(matrixA)
frameB = pd.DataFrame(matrixB)
display(frameA)
display(frameB)
```

	0	1	2	3
0	50	83	47	11
1	90	77	28	77
2	0	94	96	58
3	26	89	71	97

  

	0	1	2	3
0	84	43	66	89
1	75	67	2	6
2	77	12	54	54
3	9	35	45	85

```
[18]: display(frameB - frameA == frameB.sub(frameA))
display(frameB - frameA)
```

	0	1	2	3
0	True	True	True	True
1	True	True	True	True
2	True	True	True	True
3	True	True	True	True

```
      0   1   2   3  
0  34 -40  19  78  
1 -15 -10 -26 -71  
2  77 -82 -42  -4  
3 -17 -54 -26 -12
```

[19]: *# a través de métodos u operadores*

```
display(frameA + frameB == frameA.add(frameB))  
display(frameA + frameB)
```

```
      0   1   2   3  
0  True  True  True  True  
1  True  True  True  True  
2  True  True  True  True  
3  True  True  True  True  
  
      0   1   2   3  
0  134  126  113  100  
1  165  144   30   83  
2   77  106  150  112  
3   35  124  116  182
```

[20]: *# si los frames no son iguales, valor por defecto NaN*

```
frameC = pd.DataFrame(np.random.randint(100,size=(3,3)))  
display(frameA)  
display(frameC)  
display(frameC + frameA)
```

```
      0   1   2   3  
0  50  83  47  11  
1  90  77  28  77  
2   0  94  96  58  
3  26  89  71  97  
  
      0   1   2  
0  92  96  88  
1  96  51  84  
2  38   4  98  
  
      0       1       2   3  
0  142.0  179.0  135.0  NaN  
1  186.0  128.0  112.0  NaN  
2   38.0   98.0  194.0  NaN  
3    NaN     NaN     NaN  NaN
```

[21]: *# se puede especificar el valor por defecto con el argumento fill\_value*

```
display(frameA.add(frameC, fill_value=0))
```

```
      0       1       2   3  
0  142.0  179.0  135.0  11.0  
1  186.0  128.0  112.0  77.0
```

```
2 38.0 98.0 194.0 58.0
3 26.0 89.0 71.0 97.0
```

Operadores aritméticos solo válidos en elementos aceptables

```
[22]: frameD = pd.DataFrame({0: ['a','b'],1:['d','f']})
display(frameD)
frameA = frameD
```

```
0 1
0 a d
1 b f
```

```
-----
TypeError                                 Traceback (most recent call last)
File E:\anaconda3\envs\machines2026\Lib\site-packages\pandas\core\ops\array_ops.py:218, in _na_arithmetic_op(left, right, op, is_cmp)
    217     try:
--> 218         result = func(left, right)
    219     except TypeError:

File E:
    \anaconda3\envs\machines2026\Lib\site-packages\pandas\core\computation\expressions.py:242, in evaluate(op, a, b, use_numexpr)
    240     if use_numexpr:
    241         # error: "None" not callable
--> 242         return _evaluate(op, op_str, a, b) # type: ignore[misc]
    243     return _evaluate_standard(op, op_str, a, b)

File E:
    \anaconda3\envs\machines2026\Lib\site-packages\pandas\core\computation\expressions.py:73, in _evaluate_standard(op, op_str, a, b)
    72     _store_test_result(False)
--> 73     return op(a, b)

TypeError: unsupported operand type(s) for -: 'int' and 'str'
```

During handling of the above exception, another exception occurred:

```
TypeError                                 Traceback (most recent call last)
Cell In[22], line 3
      1 frameD = pd.DataFrame({0: ['a','b'],1:['d','f']})
      2 display(frameD)
----> 3 frameA = frameD
```

```
File E:\anaconda3\envs\machines2026\Lib\site-packages\pandas\core\ops\common.py
    76, in _unpack_zerodim_and_defer.<locals>.new_method(self, other)
    72             return NotImplemented
    74     other = item_from_zerodim(other)
```

```

---> 76     return method(self, other)

File E:\anaconda3\envs\machines2026\Lib\site-packages\pandas\core\arraylike.py:
  ↵194, in OpsMixin.__sub__(self, other)
  192     @unpack_zerodim_and_defer("__sub__")
  193     def __sub__(self, other):
--> 194         return self._arith_method(other, operator.sub)

File E:\anaconda3\envs\machines2026\Lib\site-packages\pandas\core\frame.py:7927
  ↵in DataFrame._arith_method(self, other, op)
  7925     def _arith_method(self, other, op):
  7926         if self._should_reindex_frame_op(other, op, 1, None, None):
-> 7927             return self._arith_method_with_reindex(other, op)
  7929         axis: Literal[1] = 1 # only relevant for Series other case
  7930         other = ops.maybe_prepare_scalar_for_op(other, (self.shape[axis],))

File E:\anaconda3\envs\machines2026\Lib\site-packages\pandas\core\frame.py:8059
  ↵in DataFrame._arith_method_with_reindex(self, right, op)
  8057     new_left = left.iloc[:, lcols]
  8058     new_right = right.iloc[:, rcols]
-> 8059     result = op(new_left, new_right)
  8061 # Do the join on the columns instead of using left._align_for_op
  8062 # to avoid constructing two potentially large/sparse DataFrames
  8063     join_columns, _, _ = left.columns.join(
  8064         right.columns, how="outer", level=None, return_indexers=True
  8065     )

File E:\anaconda3\envs\machines2026\Lib\site-packages\pandas\core\ops\common.py
  ↵76, in _unpack_zerodim_and_defer.<locals>.new_method(self, other)
    72             return NotImplemented
    74     other = item_from_zerodim(other)
---> 76     return method(self, other)

File E:\anaconda3\envs\machines2026\Lib\site-packages\pandas\core\arraylike.py:
  ↵194, in OpsMixin.__sub__(self, other)
  192     @unpack_zerodim_and_defer("__sub__")
  193     def __sub__(self, other):
--> 194         return self._arith_method(other, operator.sub)

File E:\anaconda3\envs\machines2026\Lib\site-packages\pandas\core\frame.py:7935
  ↵in DataFrame._arith_method(self, other, op)
  7932     self, other = self._align_for_op(other, axis, flex=True, level=None)
  7934     with np.errstate(all="ignore"):
-> 7935         new_data = self._dispatch_frame_op(other, op, axis=axis)
  7936     return self._construct_result(new_data)

File E:\anaconda3\envs\machines2026\Lib\site-packages\pandas\core\frame.py:7978
  ↵in DataFrame._dispatch_frame_op(self, right, func, axis)

```

```

7972     assert self.columns.equals(right.columns)
7973     # TODO: The previous assertion `assert right._indexed_same(self)`-
7974     # fails in cases with empty columns reached via
7975     # _frame_arith_method_with_reindex
7976
7977     # TODO operate_blockwise expects a manager of the same type
-> 7978     bm = self._mgr.operate_blockwise(
7979         # error: Argument 1 to "operate_blockwise" of "ArrayManager" has
7980         # incompatible type "Union[ArrayManager, BlockManager]"; expected
7981         # "ArrayManager"
7982         # error: Argument 1 to "operate_blockwise" of "BlockManager" has
7983         # incompatible type "Union[ArrayManager, BlockManager]"; expected
7984         # "BlockManager"
7985         right._mgr, # type: ignore[arg-type]
7986         array_op,
7987     )
7988     return self._constructor_from_mgr(bm, axes=bm.axes)
7990 elif isinstance(right, Series) and axis == 1:
7991     # axis=1 means we want to operate row-by-row

```

File E:

```

↳\anaconda3\envs\machines2026\Lib\site-packages\pandas\core\internals\managers
↪py:1530, in BlockManager.operate_blockwise(self, other, array_op)
 1526 def operate_blockwise(self, other: BlockManager, array_op) ->_
↪BlockManager:
 1527     """
 1528     Apply array_op blockwise with another (aligned) BlockManager.
 1529     """
-> 1530     return operate_blockwise(self, other, array_op)

```

```

File E:\anaconda3\envs\machines2026\Lib\site-packages\pandas\core\internals\ops
↪py:65, in operate_blockwise(left, right, array_op)
 63 res_blks: list[Block] = []
 64 for lvals, rvals, locs, left_ea, right_ea, rblk in_
↪_iter_block_pairs(left, right):
---> 65     res_values = array_op(lvals, rvals)
 66     if (
 67         left_ea
 68         and not right_ea
 69         and hasattr(res_values, "reshape")
 70         and not is_1d_only_ea_dtype(res_values.dtype)
 71     ):
 72         res_values = res_values.reshape(1, -1)

```

```

File E:\anaconda3\envs\machines2026\Lib\site-packages\pandas\core\ops\array_ops
↪py:283, in arithmetic_op(left, right, op)
 279     _bool_arith_check(op, left, right) # type: ignore[arg-type]
 281     # error: Argument 1 to "_na_arithmetic_op" has incompatible type

```

```

 282      # "Union[ExtensionArray, ndarray[Any, Any]]"; expected "ndarray[Any
↳Any]"
--> 283      res_values = _na_arithmetic_op(left, right, op)  # type:↳
↳ignore[arg-type]
 285  return res_values

File E:\anaconda3\envs\machines2026\Lib\site-packages\pandas\core\ops\array_ops
↳py:227, in _na_arithmetic_op(left, right, op, is_cmp)
 219 except TypeError:
 220     if not is_cmp and (
 221         left.dtype == object or getattr(right, "dtype", None) == object
 222     ):
(...). 225             # Don't do this for comparisons, as that will handle
↳complex numbers
 226             # incorrectly, see GH#32047
--> 227             result = _masked_arith_op(left, right, op)
 228     else:
 229         raise

File E:\anaconda3\envs\machines2026\Lib\site-packages\pandas\core\ops\array_ops
↳py:163, in _masked_arith_op(x, y, op)
 161     # See GH#5284, GH#5035, GH#19448 for historical reference
 162     if mask.any():
--> 163         result[mask] = op(xrav[mask], yrav[mask])
 165 else:
 166     if not is_scalar(y):

TypeError: unsupported operand type(s) for -: 'int' and 'str'

```

Operaciones entre Series y DataFrames

```
[ ]: rand_matrix = np.random.randint(10, size=(3, 4))
df = pd.DataFrame(rand_matrix , columns=list('ABCD'))
display(df)

display(df.iloc[0])
display(type(df.iloc[0]))
# uso común, averiguar la diferencia entre una fila y el resto
display(df - df.iloc[0])
display(df.sub(df.iloc[0], axis=1))
# Por columnas cómo se restaría
display(df.sub(df['A'], axis=0))
```

pandas se basa en NumPy, np operadores binarios y unarios son aceptables

Tipo	Operación	Descripción
Unario	<i>abs</i>	Valor absoluto de cada elemento
	<i>sqrt</i>	Raíz cuadrada de cada elemento
	<i>exp</i>	$e^x$ , siendo x cada elemento
	<i>log, log10, log2</i>	Logaritmos en distintas bases de cada elemento
	<i>sign</i>	Retorna el signo de cada elemento (-1 para negativo, 0 o 1 para positivo)
	<i>ceil</i>	Redondea cada elemento por arriba
	<i>floor</i>	Redondea cada elemento por abajo
	<i>isnan</i>	Retorna si cada elemento es Nan
	<i>cos, sin, tan</i>	Operaciones trigonométricas en cada elemento
	<i>arccos, arcsin, arctan</i>	Inversas de operaciones trigonométricas en cada elemento
Binario	<i>add</i>	Suma de dos arrays
	<i>subtract</i>	Resta de dos arrays
	<i>multiply</i>	Multiplicación de dos arrays
	<i>divide</i>	División de dos arrays
	<i>maximum, minimum</i>	Retorna el valor máximo/mínimo de cada pareja de elementos
	<i>equal, not_equal</i>	Retorna la comparación (igual o no igual) de cada pareja de elementos
	<i>greater, greater_equal, less, less_equal</i>	Retorna la comparación ( $>$ , $\geq$ , $<$ , $\leq$ respectivamente) de cada pareja de elementos

Aplicación de funciones a medida con lambda

```
[ ]: rand_matrix = np.random.randint(10, size=(3, 4))
frame = pd.DataFrame(rand_matrix , columns=list('ABCD'))
display(frame)
```

```
print(frame.apply(lambda x : x.max() - x.min(), axis = 1)) # diferencia por columna
```

```
[ ]: def max_min(x):
      return x.max() - x.min()

print(frame.apply(max_min, axis = 0)) # diferencia por columna
```

```
[ ]: # diferencia entre min y max por fila (no columna)
rand_matrix = np.random.randint(10, size=(3, 4))
frame = pd.DataFrame(rand_matrix , columns=list('ABCD'))
display(frame)

print(frame.apply(lambda x : x.max() - x.min(), axis = 1)) # diferencia por fila
```

### 3 Estadística descriptiva

- Análisis preliminar de los datos
- Para Series y DataFrame

Operación	Descripción
count	Número de valores no NaN
describe	Conjunto de estadísticas sumarias
min, max	Valores mínimo y máximo
argmin, argmax	Índices posicionales del valor mínimo y máximo
idxmin, idxmax	Índices semánticos del valor mínimo y máximo
sum	Suma de los elementos
mean	Media de los elementos
median	Mediana de los elementos
mad	Desviación absoluta media del valor medio
var	Varianza de los elementos
std	Desviación estándar de los elementos
cumsum	Suma acumulada de los elementos
diff	Diferencia aritmética de los elementos

```
[ ]: diccionario = { "nombre" : ["Marisa", "Laura", "Manuel", "Carlos"], "edad" : [
    ↪[34,34,11, 30],
    "puntos" : [98,12,98,np.nan], "genero": ["F", "F", "M", "M"] }
frame = pd.DataFrame(diccionario)
display(frame)
display(frame.describe()) # datos generales de elementos
```

```
[23]: # operadores básicos
print(frame.sum())
```

```
display(frame)
print(frame.sum(axis=1, numeric_only=True))
```

```
D      39
A      38
C      54
B      50
dtype: int64

      D   A   C   B
2    7  11  15   2
1    2  11  16  14
0    2   8  12  14
4   10   5   1  15
3   18   3  10   5

2     35
1     43
0     36
4     31
3     36
dtype: int64
```

```
[24]: frame.mean(numeric_only=True)
```

```
[24]: D      7.8
       A      7.6
       C     10.8
       B     10.0
dtype: float64
```

```
[25]: frame.cumsum()
```

```
[25]:      D   A   C   B
2    7  11  15   2
1    9  22  31  16
0   11  30  43  30
4   21  35  44  45
3   39  38  54  50
```

```
[26]: frame.count(axis=1)
```

```
[26]: 2     4
       1     4
       0     4
       4     4
       3     4
dtype: int64
```

```
[27]: print(frame['edad'].std())
```

```
-----  
KeyError Traceback (most recent call last)  
File E:\anaconda3\envs\machines2026\Lib\site-packages\pandas\core\indexes\base.  
py:3812, in Index.get_loc(self, key)  
    3811     try:  
-> 3812         return self._engine.get_loc(casted_key)  
    3813     except KeyError as err:  
  
File pandas/_libs/index.pyx:167, in pandas._libs.index.IndexEngine.get_loc()  
  
File pandas/_libs/index.pyx:196, in pandas._libs.index.IndexEngine.get_loc()  
  
File pandas/_libs/hashtable_class_helper.pxi:7088, in pandas._libs.hashtable.  
PyObjectHashTable.get_item()  
  
File pandas/_libs/hashtable_class_helper.pxi:7096, in pandas._libs.hashtable.  
PyObjectHashTable.get_item()  
  
KeyError: 'edad'
```

The above exception was the direct cause of the following exception:

```
KeyError Traceback (most recent call last)  
Cell In[27], line 1  
----> 1 print(frame[      ].std())  
  
File E:\anaconda3\envs\machines2026\Lib\site-packages\pandas\core\frame.py:4113  
in DataFrame.__getitem__(self, key)  
    4111     if self.columns.nlevels > 1:  
    4112         return self._getitem_multilevel(key)  
-> 4113     indexer = self.columns.get_loc(key)  
    4114     if is_integer(indexer):  
    4115         indexer = [indexer]  
  
File E:\anaconda3\envs\machines2026\Lib\site-packages\pandas\core\indexes\base.  
py:3819, in Index.get_loc(self, key)  
    3814     if isinstance(casted_key, slice) or (  
    3815         isinstance(casted_key, abc.Iterable)  
    3816         and any(isinstance(x, slice) for x in casted_key)  
    3817     ):  
    3818         raise InvalidIndexError(key)  
-> 3819     raise KeyError(key) from err  
3820 except TypeError:  
    3821     # If we have a listlike key, _check_indexing_error will raise  
    3822     # InvalidIndexError. Otherwise we fall through and re-raise
```

```

3823      # the TypeError.
3824      self._check_indexing_error(key)

KeyError: 'edad'

[ ]: frame['edad'].idxmax()

[ ]: frame['puntos'].idxmin()

[ ]: # frame con las filas con los valores maximos de una columna
print(frame['puntos'].max())

display(frame[frame['puntos']] == frame['puntos'].max())

[28]: frame["ranking"] = frame["puntos"].rank(method='max')

```

```

-----
KeyError                                         Traceback (most recent call last)
File E:\anaconda3\envs\machines2026\Lib\site-packages\pandas\core\indexes\base.
  ↪py:3812, in Index.get_loc(self, key)
  3811 try:
-> 3812     return self._engine.get_loc(casted_key)
  3813 except KeyError as err:

File pandas/_libs/index.pyx:167, in pandas._libs.index.IndexEngine.get_loc()

File pandas/_libs/index.pyx:196, in pandas._libs.index.IndexEngine.get_loc()

File pandas/_libs/hashtable_class_helper.pxi:7088, in pandas._libs.hashtable.
  ↪PyObjectHashTable.get_item()

File pandas/_libs/hashtable_class_helper.pxi:7096, in pandas._libs.hashtable.
  ↪PyObjectHashTable.get_item()

KeyError: 'puntos'
```

The above exception was the direct cause of the following exception:

```

KeyError                                         Traceback (most recent call last)
Cell In[28], line 1
----> 1 frame["ranking"] = frame[          ].rank(method='max')

File E:\anaconda3\envs\machines2026\Lib\site-packages\pandas\core\frame.py:4113
  ↪in DataFrame.__getitem__(self, key)
  4111 if self.columns.nlevels > 1:
  4112     return self._getitem_multilevel(key)
-> 4113 indexer = self.columns.get_loc(key)
```

```

4114 if is_integer(indexer):
4115     indexer = [indexer]

File E:\anaconda3\envs\machines2026\Lib\site-packages\pandas\core\indexes\base.
py:3819, in Index.get_loc(self, key)
3814     if isinstance(casted_key, slice) or (
3815         isinstance(casted_key, abc.Iterable)
3816         and any(isinstance(x, slice) for x in casted_key)
3817     ):
3818         raise InvalidIndexError(key)
-> 3819     raise KeyError(key) from err
3820 except TypeError:
3821     # If we have a listlike key, _check_indexing_error will raise
3822     # InvalidIndexError. Otherwise we fall through and re-raise
3823     # the TypeError.
3824     self._check_indexing_error(key)

KeyError: 'puntos'

```

[29]: display(frame)

	D	A	C	B
2	7	11	15	2
1	2	11	16	14
0	2	8	12	14
4	10	5	1	15
3	18	3	10	5

### 3.1 Agregaciones

[30]: display(frame)  
df = frame.groupby('genero').count()  
display(df)

	D	A	C	B
2	7	11	15	2
1	2	11	16	14
0	2	8	12	14
4	10	5	1	15
3	18	3	10	5

---

<pre> KeyError Cell In[30], line 2     1 display(frame) ----&gt; 2 df = frame.groupby(      ).count()     3 display(df) </pre>	<pre> Traceback (most recent call last) </pre>
--------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------

```

File E:\anaconda3\envs\machines2026\Lib\site-packages\pandas\core\frame.py:9210
  ↵in DataFrame.groupby(self, by, axis, level, as_index, sort, group_keys, u
  ↵observed, dropna)
  9207 if level is None and by is None:
  9208     raise TypeError("You have to supply one of 'by' and 'level'")
-> 9210 return DataFrameGroupBy(
  9211     obj=self,
  9212     keys=by,
  9213     axis=axis,
  9214     level=level,
  9215     as_index=as_index,
  9216     sort=sort,
  9217     group_keys=group_keys,
  9218     observed=observed,
  9219     dropna=dropna,
  9220 )

```

```

File E:
  ↵\anaconda3\envs\machines2026\Lib\site-packages\pandas\core\groupby\groupby.py
  ↵1331, in GroupBy.__init__(self, obj, keys, axis, level, grouper, exclusions, u
  ↵selection, as_index, sort, group_keys, observed, dropna)
  1328 self.dropna = dropna
  1330 if grouper is None:
-> 1331     grouper, exclusions, obj = get_grouper(
  1332         obj,
  1333         keys,
  1334         axis=axis,
  1335         level=level,
  1336         sort=sort,
  1337         observed=False if observed is lib.no_default else observed,
  1338         dropna=self.dropna,
  1339     )
  1341 if observed is lib.no_default:
  1342     if any(_.passed_categorical for ping in grouper.groupings):

```

```

File E:
  ↵\anaconda3\envs\machines2026\Lib\site-packages\pandas\core\groupby\grouper.py
  ↵1043, in get_grouper(obj, key, axis, level, sort, observed, validate, dropna)
  1041     in_axis, level, gpr = False, gpr, None
  1042 else:
-> 1043     raise KeyError(gpr)
  1044 elif isinstance(gpr, Grouper) and gpr.key is not None:
  1045     # Add key to exclusions
  1046     exclusions.add(gpr.key)

```

KeyError: 'genero'

```
[ ]: # si es Nan descarta la fila
df = frame.groupby('puntos').count()
display(df)

[ ]: display(frame.groupby('genero').mean(numeric_only=True))

[31]: display(frame.groupby('genero').max())
```

-----

**KeyError** Traceback (most recent call last)

Cell In[31], line 1  
----> 1 display(frame.groupby().max())

File E:\anaconda3\envs\machines2026\Lib\site-packages\pandas\core\frame.py:9210  
  in DataFrame.groupby(self, by, axis, level, as\_index, sort, group\_keys, u  
  r observed, dropna)  
9207 if level is None and by is None:  
9208 raise TypeError("You have to supply one of 'by' and 'level'")  
-> 9210 return DataFrameGroupBy(  
9211 obj=self,  
9212 keys=by,  
9213 axis=axis,  
9214 level=level,  
9215 as\_index=as\_index,  
9216 sort=sort,  
9217 group\_keys=group\_keys,  
9218 observed=observed,  
9219 dropna=dropna,  
9220 )

File E:  
  \anaconda3\envs\machines2026\Lib\site-packages\pandas\core\groupby\groupby.py  
  1331, in GroupBy.\_\_init\_\_(self, obj, keys, axis, level, grouper, exclusions, u  
  r selection, as\_index, sort, group\_keys, observed, dropna)  
1328 self.dropna = dropna  
1330 if grouper is None:  
-> 1331 grouper, exclusions, obj = get\_grouper(  
1332 obj,  
1333 keys,  
1334 axis=axis,  
1335 level=level,  
1336 sort=sort,  
1337 observed=False if observed is lib.no\_default else observed,  
1338 dropna=self.dropna,  
1339 )  
1341 if observed is lib.no\_default:  
1342 if any(\_.passed\_categorical for ping in grouper.groupings):

```

File E:
  ↵\anaconda3\envs\machines2026\Lib\site-packages\pandas\core\groupby\grouper.py
  ↵1043, in get_grouper(obj, key, axis, level, sort, observed, validate, dropna)
    1041         in_axis, level, gpr = False, gpr, None
    1042     else:
-> 1043         raise KeyError(gpr)
1044 elif isinstance(gpr, Grouper) and gpr.key is not None:
1045     # Add key to exclusions
1046     exclusions.add(gpr.key)

KeyError: 'genero'

```

```
[ ]: # funciones de agregación de varias columnas para obtener distintos estadísticos
display(frame.groupby('genero')[['edad', 'puntos']].aggregate(['min', 'mean', ↵'max']))
```

```
[32]: # Filtrado de los datos en el que el conjunto no supera una media determinada
def media(x):
    return x["edad"].mean()

display(frame)
frame.groupby('genero').filter(media)
```

	D	A	C	B
2	7	11	15	2
1	2	11	16	14
0	2	8	12	14
4	10	5	1	15
3	18	3	10	5

---

```

KeyError                                         Traceback (most recent call last)
Cell In[32], line 6
      3     return x["edad"].mean() > 30
      5 display(frame)
----> 6 frame.groupby().filter(media)

```

```

File E:\anaconda3\envs\machines2026\Lib\site-packages\pandas\core\frame.py:9210
  ↵in DataFrame.groupby(self, by, axis, level, as_index, sort, group_keys, ↵
  ↵observed, dropna)
    9207 if level is None and by is None:
    9208     raise TypeError("You have to supply one of 'by' and 'level'")
-> 9210 return DataFrameGroupBy(
    9211     obj=self,
    9212     keys=by,
    9213     axis=axis,
    9214     level=level,
    9215     as_index=as_index,
```

```

9216     sort=sort,
9217     group_keys=group_keys,
9218     observed=observed,
9219     dropna=dropna,
9220 )

```

File E:

```

\anaconda3\envs\machines2026\Lib\site-packages\pandas\core\groupby\groupby.py
1331, in GroupBy.__init__(self, obj, keys, axis, level, grouper, exclusions,
selection, as_index, sort, group_keys, observed, dropna)
1328 self.dropna = dropna
1330 if grouper is None:
-> 1331     grouper, exclusions, obj = get_grouper(
1332         obj,
1333         keys,
1334         axis=axis,
1335         level=level,
1336         sort=sort,
1337         observed=False if observed is lib.no_default else observed,
1338         dropna=self.dropna,
1339     )
1341 if observed is lib.no_default:
1342     if any(_.passed_categorical for ping in grouper.groupings):

```

File E:

```

\anaconda3\envs\machines2026\Lib\site-packages\pandas\core\groupby\grouper.py
1043, in get_grouper(obj, key, axis, level, sort, observed, validate, dropna)
1041     in_axis, level, gpr = False, gpr, None
1042 else:
-> 1043     raise KeyError(gpr)
1044 elif isinstance(gpr, Grouper) and gpr.key is not None:
1045     # Add key to exclusions
1046     exclusions.add(gpr.key)

```

KeyError: 'genero'

## 3.2 Correlaciones

pandas incluye métodos para analizar correlaciones - Relación matemática entre dos variables (-1 negativamente relacionadas, 1 positivamente relacionadas, 0 sin relación) - obj.corr(obj2) –> medida de correlación entre los datos de ambos objetos - <https://blogs.oracle.com/ai-and-datascience/post/introduction-to-correlation>

### 3.2.1 Ejemplo Fuel efficiency

- <https://archive.ics.uci.edu/ml/datasets/Auto+MPG>

```
[33]: import pandas as pd
path = 'http://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/
        ↴auto-mpg.data'

mpg_data = pd.read_csv(path, sep='\s+', header=None,
                      names = ['mpg', 'cilindros', 'desplazamiento', 'potencia',
                               'peso', 'aceleracion', 'año', 'origen', 'nombre'],
                      na_values='?', engine='c')
```

```
[34]: display(mpg_data.sample(5))
```

	mpg	cilindros	desplazamiento	potencia	peso	aceleracion	año	\
214	13.0	8	302.0	130.0	3870.0	15.0	76	
170	23.0	4	140.0	78.0	2592.0	18.5	75	
392	27.0	4	151.0	90.0	2950.0	17.3	82	
334	23.7	3	70.0	100.0	2420.0	12.5	80	
80	22.0	4	122.0	86.0	2395.0	16.0	72	
	origen		nombre					
214	1		ford f108					
170	1		pontiac astro					
392	1		chevrolet camaro					
334	3		mazda rx-7 gs					
80	1		ford pinto (sw)					

```
[35]: display(mpg_data.describe(include='all'))
```

	mpg	cilindros	desplazamiento	potencia	peso	\
count	398.000000	398.000000	398.000000	392.000000	398.000000	
unique	NaN	NaN	NaN	NaN	NaN	
top	NaN	NaN	NaN	NaN	NaN	
freq	NaN	NaN	NaN	NaN	NaN	
mean	23.514573	5.454774	193.425879	104.469388	2970.424623	
std	7.815984	1.701004	104.269838	38.491160	846.841774	
min	9.000000	3.000000	68.000000	46.000000	1613.000000	
25%	17.500000	4.000000	104.250000	75.000000	2223.750000	
50%	23.000000	4.000000	148.500000	93.500000	2803.500000	
75%	29.000000	8.000000	262.000000	126.000000	3608.000000	
max	46.600000	8.000000	455.000000	230.000000	5140.000000	
	aceleracion	año	origen	nombre		
count	398.000000	398.000000	398.000000	398		
unique	NaN	NaN	NaN	305		
top	NaN	NaN	NaN	ford pinto		
freq	NaN	NaN	NaN	6		
mean	15.568090	76.010050	1.572864	NaN		
std	2.757689	3.697627	0.802055	NaN		
min	8.000000	70.000000	1.000000	NaN		

```

25%      13.825000  73.000000  1.000000      NaN
50%      15.500000  76.000000  1.000000      NaN
75%      17.175000  79.000000  2.000000      NaN
max      24.800000  82.000000  3.000000      NaN

```

[36]: mpg\_data.info()

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column            Non-Null Count  Dtype  
--- 
 0   mpg               398 non-null    float64
 1   cilindros         398 non-null    int64   
 2   desplazamiento   398 non-null    float64
 3   potencia          392 non-null    float64
 4   peso              398 non-null    float64
 5   aceleracion       398 non-null    float64
 6   año               398 non-null    int64   
 7   origen            398 non-null    int64   
 8   nombre            398 non-null    object  
dtypes: float64(5), int64(3), object(1)
memory usage: 28.1+ KB

```

### 3.2.2 Correlaciones entre valores

[37]: mpg\_data['mpg'].corr(mpg\_data['peso']) # + mpg = - peso

[37]: np.float64(-0.8317409332443354)

[38]: mpg\_data['peso'].corr(mpg\_data['aceleracion']) # + peso = - aceleracion

[38]: np.float64(-0.41745731994039337)

### 3.2.3 Correlaciones entre todos los valores

[39]: mpg\_data.corr(numeric\_only=True)

```

[39]:      mpg  cilindros  desplazamiento  potencia  peso \
mpg      1.000000 -0.775396     -0.804203 -0.778427 -0.831741
cilindros -0.775396  1.000000      0.950721  0.842983  0.896017
desplazamiento -0.804203  0.950721      1.000000  0.897257  0.932824
potencia     -0.778427  0.842983      0.897257  1.000000  0.864538
peso        -0.831741  0.896017      0.932824  0.864538  1.000000
aceleracion    0.420289 -0.505419     -0.543684 -0.689196 -0.417457
a o           0.579267 -0.348746     -0.370164 -0.416361 -0.306564
origen        0.563450 -0.562543     -0.609409 -0.455171 -0.581024

```

```

          aceleracion      año      origen
mpg           0.420289  0.579267  0.563450
cilindros     -0.505419 -0.348746 -0.562543
desplazamiento -0.543684 -0.370164 -0.609409
potencia      -0.689196 -0.416361 -0.455171
peso           -0.417457 -0.306564 -0.581024
aceleracion     1.000000  0.288137  0.205873
año            0.288137  1.000000  0.180662
origen         0.205873  0.180662  1.000000

```

[40]: #año y origen no parecen correlacionables

#eliminar columnas de la correlacion

```
corr_data = mpg_data.drop(['año','origen'],axis=1).corr(numeric_only=True)
display(corr_data)
```

```

          mpg  cilindros  desplazamiento  potencia  peso \
mpg       1.000000   -0.775396    -0.804203 -0.778427 -0.831741
cilindros   -0.775396    1.000000     0.950721  0.842983  0.896017
desplazamiento -0.804203    0.950721     1.000000  0.897257  0.932824
potencia     -0.778427    0.842983     0.897257  1.000000  0.864538
peso          -0.831741    0.896017     0.932824  0.864538  1.000000
aceleracion    0.420289   -0.505419    -0.543684 -0.689196 -0.417457

          aceleracion
mpg           0.420289
cilindros     -0.505419
desplazamiento -0.543684
potencia      -0.689196
peso           -0.417457
aceleracion     1.000000

```

[41]: # representación gráfica matplotlib

```
import matplotlib.pyplot as plt
```

```

-----
ModuleNotFoundError                         Traceback (most recent call last)
Cell In[41], line 2
      1 # representación gráfica matplotlib
----> 2 import matplotlib.pyplot as plt

ModuleNotFoundError: No module named 'matplotlib'
```

[42]: # representación gráfica

```
corr_data.style.background_gradient(cmap=plt.get_cmap('RdYlGn'), axis=1)
```

```

-----
NameError                                Traceback (most recent call last)
```

```

Cell In[42], line 2
    1 # representación gráfica
----> 2 corr_data.style.background_gradient(cmap=plt.get_cmap('RdYlGn'), axis=1)

NameError: name 'plt' is not defined

```

```
[43]: # correlación más negativa
mpg_data.drop(['año','origen'],axis=1).corr(numeric_only=True).idxmin()
```

```
[43]: mpg           peso
cilindros      mpg
desplazamiento mpg
potencia       mpg
peso           mpg
aceleracion    potencia
dtype: object
```

```
[44]: # correlación más positiva
mpg_data.drop(['año','origen'],axis=1).corr(numeric_only=True).idxmax()
↳#consigo misma....
```

```
[44]: mpg           mpg
cilindros      cilindros
desplazamiento desplazamiento
potencia       potencia
peso           peso
aceleracion    aceleracion
dtype: object
```

```
[45]: # tabla similar con las correlaciones más positivas (evitar parejas del mismo
      ↳valor)
positive_corr = mpg_data.drop(['año','origen'],axis=1).corr(numeric_only=True)
np.fill_diagonal(positive_corr.values, 0)
display(positive_corr)
positive_corr.idxmax()
```

	mpg	cilindros	desplazamiento	potencia	peso	\
mpg	0.000000	-0.775396	-0.804203	-0.778427	-0.831741	
cilindros	-0.775396	0.000000	0.950721	0.842983	0.896017	
desplazamiento	-0.804203	0.950721	0.000000	0.897257	0.932824	
potencia	-0.778427	0.842983	0.897257	0.000000	0.864538	
peso	-0.831741	0.896017	0.932824	0.864538	0.000000	
aceleracion	0.420289	-0.505419	-0.543684	-0.689196	-0.417457	
		aceleracion				
mpg		0.420289				
cilindros		-0.505419				

```
desplazamiento      -0.543684
potencia            -0.689196
peso                -0.417457
aceleracion         0.000000

[45]: mpg          aceleracion
       cilindros    desplazamiento
       desplazamiento cilindros
       potencia      desplazamiento
       peso          desplazamiento
       aceleracion   mpg
dtype: object
```

```
[46]: positive_corr.style.background_gradient(cmap=plt.get_cmap('RdYlGn'), axis=1,
    ↪vmin=-1.0, vmax=1.0)
```

```
-----
NameError                                 Traceback (most recent call last)
Cell In[46], line 1
----> 1 positive_corr.style.background_gradient(cmap=plt.get_cmap('RdYlGn'), ↪
    ↪axis=1, vmin=-1.0, vmax=1.0)

NameError: name 'plt' is not defined
```

### 3.3 Ejercicios

- Ejercicios para practicar Pandas: <https://github.com/ajcr/100-pandas-puzzles/blob/master/100-pandas-puzzles.ipynb>

### 3.4 Repositorio

- GitHub: <https://github.com/reromash1/machines2026>