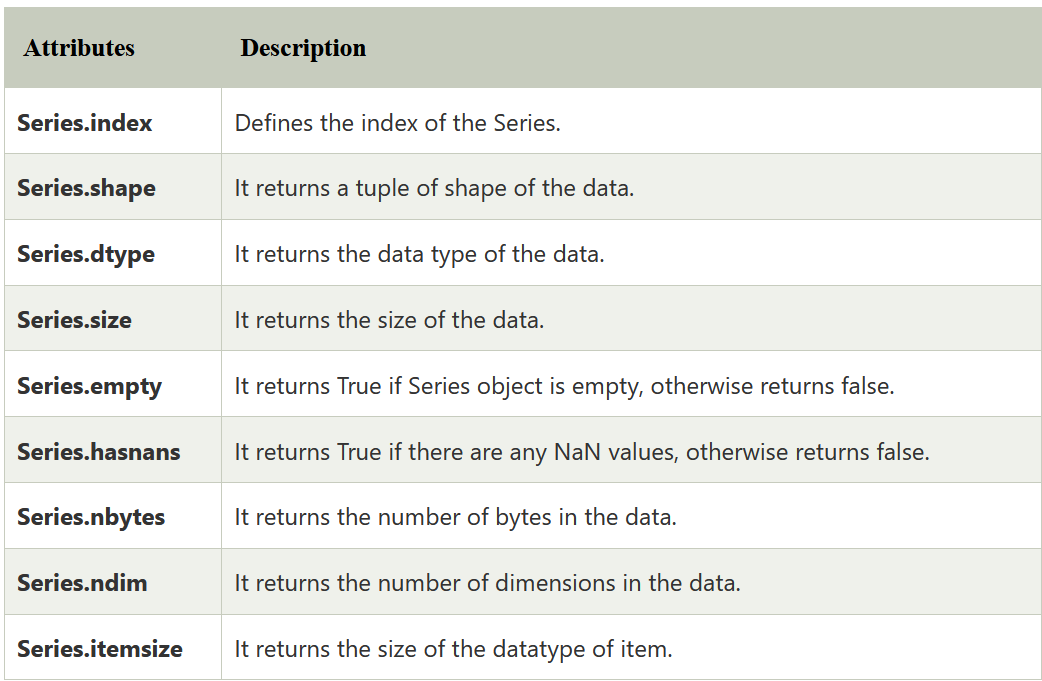
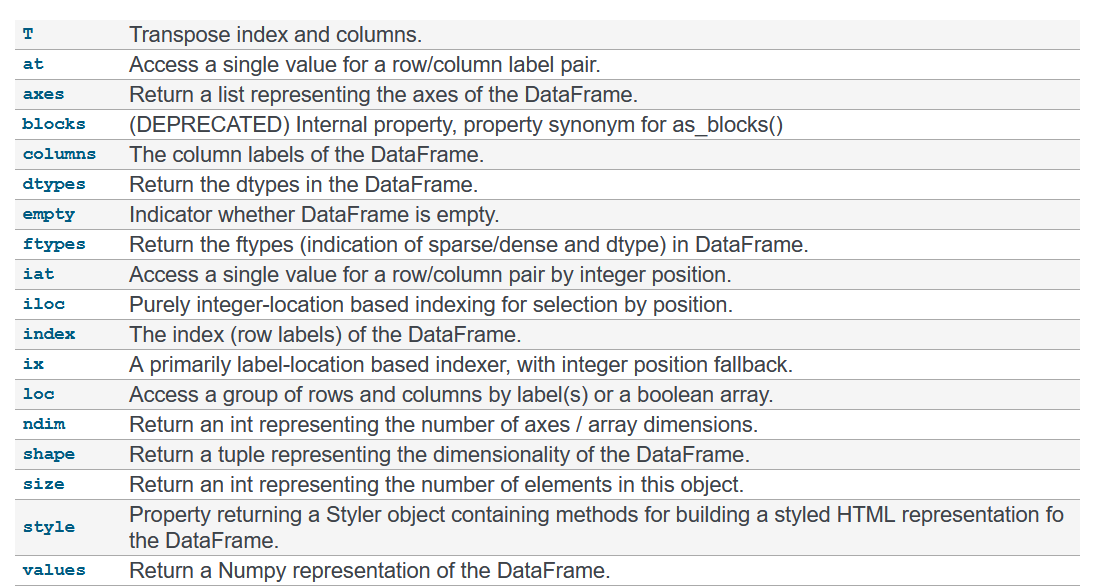
Pandas Series attributes:

Pandas series have below attributes-



**Pandas Dataframe attributes**

Pandas dataframe have below attributes-



**Pandas Index**

Pandas Index is an immutable ndarray implementing an ordered, sliceable set. It is the basic object which stores the axis labels for all pandas objects.

pandas.Index(data=None, dtype=None, copy=False, name=None, tupleize\_cols=True, \*\*kwargs)

**data**: array-like (1-dimensional)

**dtype**: NumPy dtype (default: object), If dtype is None, we find the dtype that best fits the data. If an actual dtype is provided, we coerce to that dtype if it’s safe. Otherwise, an error will be raised.

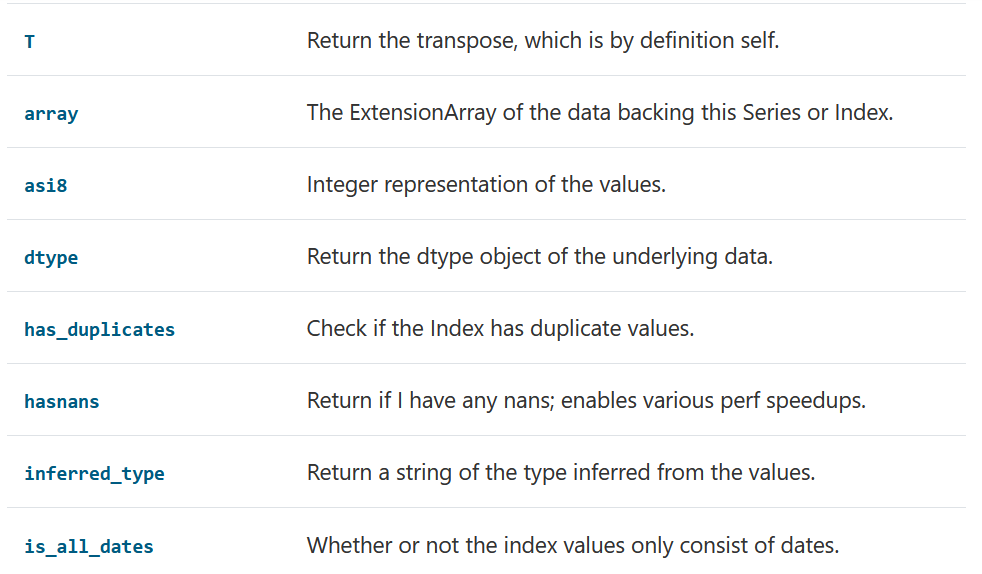
**copy**: bool, Make a copy of input ndarray.

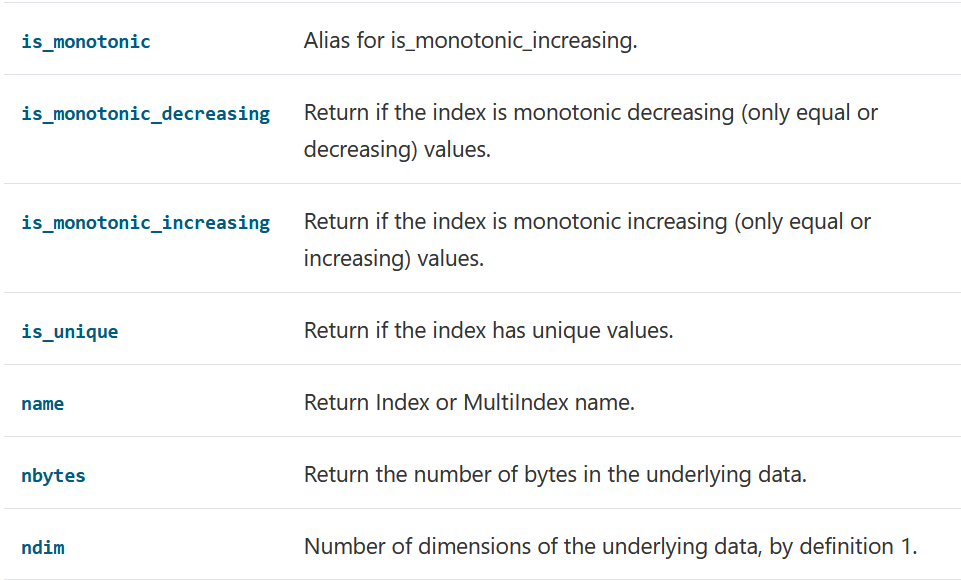
idx = pd.Index(['Jan', 'Feb', 'Mar', 'Apr', 'May'])

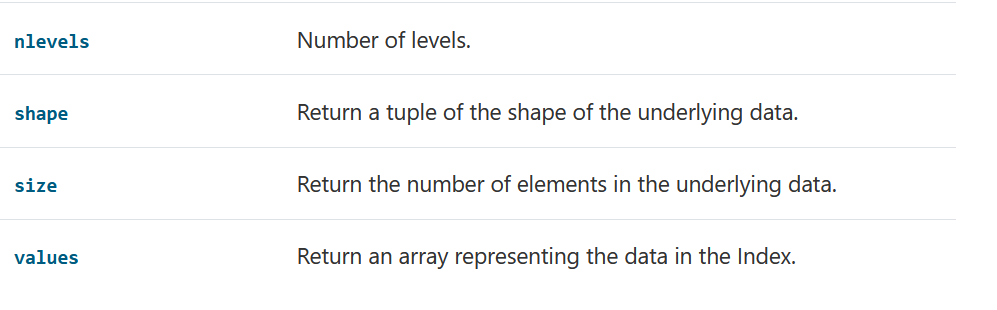
# Print the index

print(idx)

**Attributes of pandas Index**







**Selection methods for pandas Indexes**

Index.get\_loc(key, method=None, tolerance=None)

Returns loc: int if unique index, slice if monotonic index, else mask

**Key**: labels or datas

**default**: exact matches only.

**pad / ffill**: find the PREVIOUS index value if no exact match.

**backfill / bfill**: use NEXT index value if no exact match

**nearest**: use the NEAREST index value if no exact match. Tied distances are broken by preferring the larger index value.

**tolerance**: int or float, optional

**Example**

pd.Index(list('abc'))

unique\_index.get\_loc('b') #1

**Example**

monotonic\_index = pd.Index(list('abbc'))

monotonic\_index.get\_loc('b') #slice(1, 3, None)

**Example**

non\_monotonic\_index = pd.Index(list('abcb'))

non\_monotonic\_index.get\_loc('b') # array([False, True, False, True]) --- this is mask

Index.get\_value(series, key) ------ Returns series or scalar

Fast lookup of value from 1-dimensional ndarray. Only use this if you know what you’re doing.

Index.get\_slice\_bound(label, side, kind=None)

Calculate slice bound that corresponds to given label.

Returns leftmost (one-past-the-rightmost if side=='right') position of given label.

Returns: int, index od label

Index.isin(values, level=None)

Return a boolean array where the index values are in values.

Index.slice\_locs(start=None, end=None, step=None, kind=None)

Compute slice locations for input labels. Returns start,end as integer

start: label, default None

end: label, default None

step: int, defaults None

In output first values will be index value of start argument and second value will be position value of end argument.

**Note:**

This method only works if the index is monotonic or unique.

Example:

idx = pd.Index(list('abcd'))

idx.slice\_locs(start='b', end='c') #(1, 3) 1--> index of b, 3 --> position of c

In output first values will be index value of start argument and second value will be position value of end argument.

**Question ------ Good**

Write a Pandas program to get the positions of items of a given series(series2) in another given series(series1).

<https://www.w3resource.com/python-exercises/pandas/python-pandas-data-series-exercise-23.php>

series1 = pd.Series([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])

series2 = pd.Series([1, 3, 5, 7, 10])

print("Original Series:")

print(series1)

print(series2)

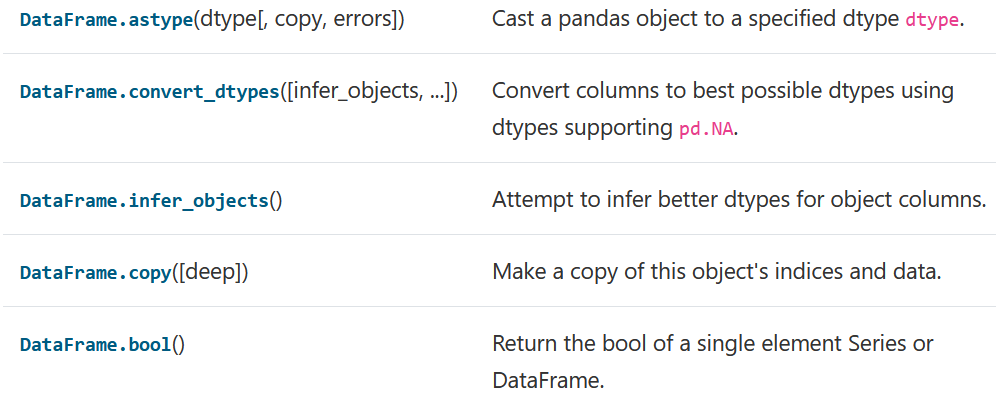
**Solution**

result = [pd.Index(series1).get\_loc(i) for i in series2]

print("Positions of items of series2 in series1:")

print(result)

**Dataframe conversion function**



DataFrame.astype(dtype, copy=True, errors='raise')

Cast a pandas object to a specified dtype dtype.

**dtype**: data type which to be casted,it's umpy.dtype or Python type.

**copy**: bool, default True, Returns a copy when copy=True

**errors**: {‘raise’, ‘ignore’}, default. Control raising of exceptions on invalid data for provided dtype.

raise : allow exceptions to be raised

ignore : suppress exceptions. On error return original object.

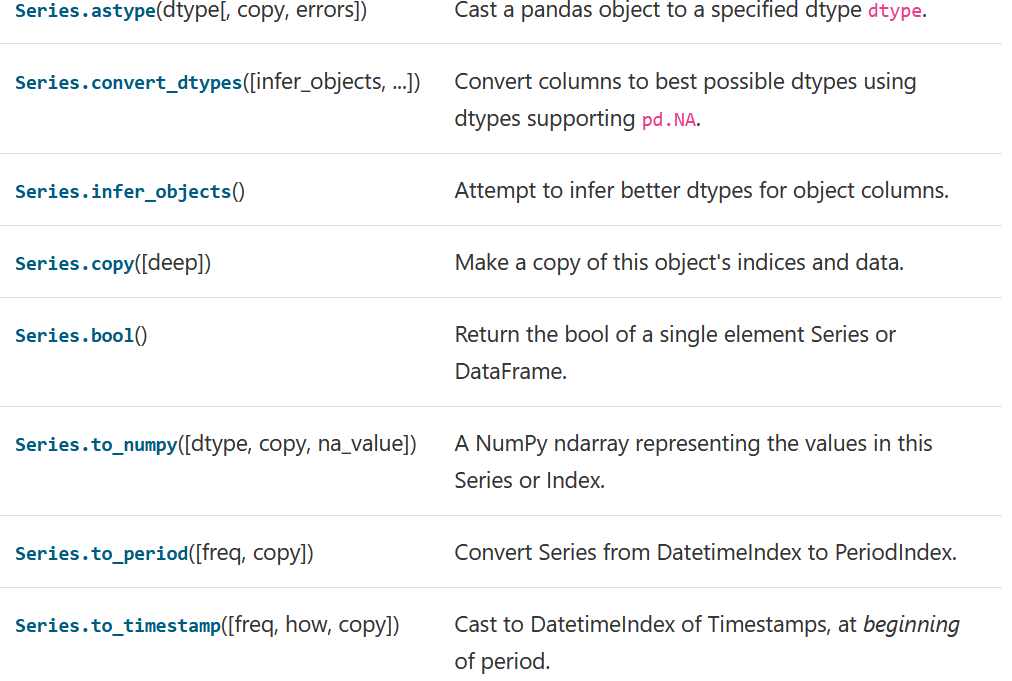
DataFrame.copy(deep=True)

Make copy (deep or shallow copy) of dataframe and return it.

**deep**: bool, default True

**Series Conversion functions**

We have below methods for conversion will see few of them.





Series.astype() --- same as df

Series.copy() ---- same as df

Series.to\_numpy(dtype=None, copy=False, na\_value=NoDefault.no\_default, \*\*kwargs)

Gives a NumPy ndarray representing the values in this Series or Index.

*dtype*: str or numpy.dtype, optional

*copy*: bool, default False

*na\_value* : Any, optional

Series.to\_timestamp(freq=None, how='start', copy=True)

Cast to DatetimeIndex of Timestamps, at beginning of period

Series.to\_list()

Convert numpy series to a list types and returns a list of values.