In [1]:
```
#series
```

In [4]:
```
#pip install pandas
```

In [2]:
```
import pandas as pd
import numpy as np
```

In [6]:
```
#series(object(data),index,dtype)
```

In [7]:
```
pd.Series()#create an empty series
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_928\660432477.py:1: FutureWarning: The default dtype for empty Seri
es will be 'object' instead of 'float64' in a future version. Specify a dtype explicitly to silence this warni
ng.
  pd.Series()#create an empty series
```

Out[7]:
```
Series([], dtype: float64)
```

In [8]:
```
x=pd.Series()
type(x)
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_928\1780875130.py:1: FutureWarning: The default dtype for empty Ser
ies will be 'object' instead of 'float64' in a future version. Specify a dtype explicitly to silence this warn
ing.
  x=pd.Series()
```

Out[8]:
```
pandas.core.series.Series
```

In [9]:
```
#series with values
```

In [11]:
```
pd.Series([1,2,3,4])
```

Out[11]:
```
0    1
1    2
2    3
3    4
dtype: int64
```

In [13]:
```
pd.Series([1,2,3,4,3.2])
```

Out[13]:
```
0    1.0
1    2.0
2    3.0
3    4.0
4    3.2
dtype: float64
```

In [4]:
```
pd.Series([1,'2',3,4,3,2])
```

Out[4]:
```
0    1
1    2
2    3
3    4
4    3
5    2
dtype: object
```

In [3]:
```
pd.Series([1,3,4,3+2j])
```

Out[3]:
```
0    1.0+0.0j
1    3.0+0.0j
2    4.0+0.0j
3    3.0+2.0j
dtype: complex128
```

In [16]:
```
y=pd.Series([1,2,3,2],dtype=object)#list of values
y
```

Out[16]:
```
0    1
1    2
2    3
3    2
dtype: object
```

In [10]:
```
y=pd.Series([1,2,3,2],dtype=str)#list of values
y
```

Out[10]:
```
0    1
1    2
2    3
3    2
dtype: object
```

In [12]:
```
y=pd.Series([1,2,3,2,6],dtype=complex)#list of values
y
```

Out[12]:
```
0    1.0+0.0j
1    2.0+0.0j
2    3.0+0.0j
3    2.0+0.0j
4    6.0+0.0j
dtype: complex128
```

In [6]:
```
y=pd.Series([1,2,3,5,6,7,4,8,2,9])
y
```

Out[6]:
```
0    1
1    2
2    3
3    5
4    6
5    7
6    4
7    8
8    2
9    9
dtype: int64
```

In [85]:
```
pd.Series([1,2,3,4],index=["a","b","c","d"],dtype=float)
```

Out[85]:
```
a    1.0
b    2.0
c    3.0
d    4.0
dtype: float64
```

In [84]:
```
pd.Series({1,2,3,4})
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_928\803095600.py in <module>
----> 1 pd.Series({1,2,3,4})

~\anaconda3\lib\site-packages\pandas\core\series.py in __init__(self, data, index, dtype, name, copy, fastpath)
    449                     data = data.copy()
    450             else:
--> 451                 data = sanitize_array(data, index, dtype, copy)
    452
    453                 manager = get_option("mode.data_manager")

~\anaconda3\lib\site-packages\pandas\core\construction.py in sanitize_array(data, index, dtype, copy, raise_cast_failure, allow_2d)
    582         if isinstance(data, (set, frozenset)):
    583             # Raise only for unordered sets, e.g., not for dict_keys
--> 584             raise TypeError(f"'{type(data).__name__}' type is unordered")
    585
    586         # materialize e.g. generators, convert e.g. tuples, abc.ValueView

TypeError: 'set' type is unordered
```

In [19]:
```
pd.Series((1,2,3,4))#by using sets we cannot create a series(sets are unorderd)
```

Out[19]:
```
0    1
1    2
2    3
3    4
dtype: int64
```

In [13]:
```
pd.Series([1,2,3,4])
```

Out[13]:
```
0    1
1    2
2    3
3    4
dtype: int64
```

In [16]:
```
pd.Series([1,2,3,2],index=np.arange(len([1,2,3,2])))
```

Out[16]:
```
0    1
1    2
2    3
3    2
dtype: int64
```

In [ ]:
```
#whenever  your passing your own axis should matches the  length of both axis
```

In [17]:
```
pd.Series([1,2,3,2],index=np.arange(len([1,2,3,2])))
```

Out[17]:
```
0    1
1    2
2    3
3    2
dtype: int64
```

In [23]:
```
pd.Series({'A':1,'B':2,'C':3})
```

Out[23]:
```
A    1
B    2
C    3
dtype: int64
```

In [26]:
```
pd.Series({'A':1,'B':2,'C':3},index=['a','b','c'])
```

Out[26]:
```
a   NaN
b   NaN
c   NaN
dtype: float64
```

In [25]:
```
pd.Series({'A':1,'B':2,'C':3},index=["a","b","c","A","B"])
```

Out[25]:
```
a    NaN
b    NaN
c    NaN
A    1.0
B    2.0
dtype: float64
```

In [27]:
```
y=pd.Series([1,2,3,2],dtype=object)#list of values
y
```

Out[27]:
```
0    1
1    2
2    3
3    2
dtype: object
```

In [28]:
```
y=pd.Series([1,2,3,2])#list of values
y
```

Out[28]:
```
0    1
1    2
2    3
3    2
dtype: int64
```

In [29]:
```
y[0]
```

Out[29]:
```
1
```

In [32]:
```
y[3]
```

Out[32]:
```
2
```

In [34]:
```
y[:1]
```

Out[34]:
```
0    1
dtype: int64
```

In [35]:
```
y[:-1]
```

Out[35]:
```
0    1
1    2
2    3
dtype: int64
```

In [37]:
```
y=pd.Series([1,2,3],index=["a","b","c"])
y
```

Out[37]:
```
a    1
b    2
c    3
dtype: int64
```

In [38]:
```
y[0]
```

Out[38]:
```
1
```

In [39]:
```
y[1]
```

Out[39]:
```
2
```

In [40]:
```
y[2]
```

Out[40]:
```
3
```

In [41]:
```
y["a"]
```

Out[41]:
```
1
```

In [42]:
```
y['b']
```

Out[42]:
```
2
```

In [43]:
```
y['c']
```

Out[43]:
```
3
```

In [45]:
```
y[:]
```

Out[45]:
```
a    1
b    2
c    3
dtype: int64
```

In [46]:
```
y[0:3]
```

Out[46]:
```
a    1
b    2
c    3
dtype: int64
```

In [47]:
```
y['a':'b']
```

Out[47]:
```
a    1
b    2
dtype: int64
```

In [49]:
```
y[[0,1,2]]#integer based indexing
```

Out[49]:
```
a    1
b    2
c    3
dtype: int64
```

In [ ]:
```
#when ever we using integers as our index values then in slicing default indexing will be used
```

In [70]:
```
y=pd.Series([1,2,3],index=[11,12,13])
y
```

Out[70]:
```
11    1
12    2
13    3
dtype: int64
```

In [71]:
```
y[11]
```

Out[71]:
```
1
```

In [72]:
```
y[12]
```

Out[72]:
```
2
```

In [73]:
```
y[:]
```

Out[73]:
```
11    1
12    2
13    3
dtype: int64
```

In [74]:
```
y[:2]
```

Out[74]:
```
11    1
12    2
dtype: int64
```

In [75]:
```
y[:3]
```

Out[75]:
```
11    1
12    2
13    3
dtype: int64
```

In [76]:
```
y[11]='a'
y      #we are able to addd elements with replacement
```

Out[76]:
```
11    a
12    2
13    3
dtype: object
```

In [77]:
```
y[14]='d'
y
```

Out[77]:
```
11    a
12    2
13    3
14    d
dtype: object
```

In [79]:
```
y[:4]
```

Out[79]:
```
11    a
12    2
13    3
14    d
dtype: object
```

In [80]:
```
y[16]="f"
y
```

Out[80]:
```
11    a
12    2
13    3
14    d
16    f
dtype: object
```

In [81]:
```
y[:5]
```

Out[81]:
```
11    a
12    2
13    3
14    d
16    f
dtype: object
```

In [82]:
```
y[16]
```

Out[82]:
```
'f'
```

In [86]:
```
#practice
```

In [89]:
```
z=pd.Series()
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_928\619253579.py:1: FutureWarning: The default dtype for empty Seri
es will be 'object' instead of 'float64' in a future version. Specify a dtype explicitly to silence this warni
ng.
  z=pd.Series()
```

In [90]:
```
type(z)
```

Out[90]:
```
pandas.core.series.Series
```

In [96]:
```
pd.Series([1,2,3,4,4,2],dtype=float,)
```

Out[96]:
```
0    1.0
1    2.0
2    3.0
3    4.0
4    4.0
5    2.0
dtype: float64
```

In [19]:
```
pd.Series({1:"a",2:"b"},index=[1,2,3])
```

Out[19]:
```
1      a
2      b
3    NaN
dtype: object
```

In [24]:
```
pd.Series({1:"a",2:"b"})
```

Out[24]:
```
1    a
2    b
dtype: object
```

In [22]:
```
pd.Series({1:"a",2:"b"},index=["a","b","c"])
```

Out[22]:
```
a    NaN
b    NaN
c    NaN
dtype: object
```

In [26]:
```
pd.Series({"a":1,"b":2},index=["A","B","a","b"])
```

Out[26]:
```
A    NaN
B    NaN
a    1.0
b    2.0
dtype: float64
```

In [27]:
```
pd.Series({"a":1,"b":2},index=["A","a","B","b"])
```

Out[27]:
```
A    NaN
a    1.0
B    NaN
b    2.0
dtype: float64
```

In [ ]:
```

```