In [1]:

```
import pandas as pd
import numpy as np
```

In [2]:

```
students = ['mark','simon','peter']
```

In [3]:

```
list_s = pd.Series(students)
```

In [4]:

```
list_s
```

Out[4]:

```
0     mark
1    simon
2    peter
dtype: object
```

In [5]:

```
dict_s = pd.Series({0:'mark', 1:'simon',2:'peter'})
dict_s
```

Out[5]:

```
0     mark
1    simon
2    peter
dtype: object
```

In [6]:

```
list_s == dict_s
```

Out[6]:

```
0    True
1    True
2    True
dtype: bool
```

In [7]:

```
dict_s = pd.Series({'a':'sam','b':'john','c':'thomas'})
dict_s
```

Out[7]:

```
a       sam
b      john
c    thomas
dtype: object
```

In [8]:

```
age = [43, 76, 35]
age_s = pd.Series(age)
```

In [9]:

```
age_s
```

Out[9]:

```
0    43
1    76
2    35
dtype: int64
```

In [10]:

```
age_s.dtype
```

Out[10]:

```
dtype('int64')
```

In [11]:

```
heights = [123.5,178.5,198.3]
```

In [12]:

```
pd.Series(heights)
```

Out[12]:

```
0    123.5
1    178.5
2    198.3
dtype: float64
```

In [13]:

```
mixed = [True,'say', {'mood':100}]
pd.Series(mixed)
```

Out[13]:

```
0             True
1              say
2    {'mood': 100}
dtype: object
```

In [14]:

```
type(mixed)
```

Out[14]:

```
list
```

Parameters Vs. Arguments

In [15]:

```
pd.Series(students)
```

Out[15]:

```
0     mark
1    simon
2    peter
dtype: object
```

In [16]:

```
pd.Series(data = students) #data here is the parameter while students is the argument(a
```

Out[16]:

```
0     mark
1    simon
2    peter
dtype: object
```

In [17]:

```
def greeting(something):
    print(something)
```

In [18]:

```
greeting("good morning to you")
```

```
good morning to you
```

What's in the data

In [19]:

```
books_list = ['Fooled by Randomness', 'Sapiens', 'Lenin on the Train']
```

In [20]:

```
list_s = pd.Series(books_list)
```

In [21]:

```
books_dict = {0:'Fooled by Randomness', 1:'Sapiens', 2:'Lenin on the Train'}
```

In [22]:

```
dict_s = pd.Series(books_dict)
```

In [23]:

```
list_s.equals(dict_s)
```

Out[23]:

```
True
```

In [24]:

```
pd.Series(714)
```

Out[24]:

```
0    714
dtype: int64
```

In [25]:

```
pd.Series('Andrew')
```

Out[25]:

```
0    Andrew
dtype: object
```

The .dtype Attribute

In [26]:

```
ages = [27, 49, 37]
```

In [27]:

```
ages
```

Out[27]:

```
[27, 49, 37]
```

In [28]:

```
pd.Series(ages)
```

Out[28]:

```
0    27
1    49
2    37
dtype: int64
```

In [29]:

```
pd.Series(ages, dtype = 'float')#dtype is a parameter for series and .dtype is an attri
```

Out[29]:

```
0    27.0
1    49.0
2    37.0
dtype: float64
```

In [30]:

```
pd.Series(ages).dtype # the dtype attribute
```

Out[30]:

```
dtype('int64')
```

In [31]:

```
students
```

Out[31]:

```
['mark', 'simon', 'peter']
```

In [32]:

```
name_series = pd.Series(students)
```

In [33]:

```
name_series
```

Out[33]:

```
0     mark
1    simon
2    peter
dtype: object
```

In [34]:

```
name_series.dtype
```

Out[34]:

```
dtype('O')
```

What is dtype, really?
numpy expects homogenous("same type") data

In [35]:

```
heights
```

Out[35]:

```
[123.5, 178.5, 198.3]
```

In [36]:

```
pd.Series(heights)
```

Out[36]:

```
0    123.5
1    178.5
2    198.3
dtype: float64
```

In [37]:

```
1 heights2 = [167.4, '173.2', 190.2]
```

In [38]:

```
1 pd.Series(heights2)
```

Out[38]:

```
0    167.4
1    173.2
2    190.2
dtype: object
```

Index and RangeIndex

In [39]:

```
1 books_list
```

Out[39]:

```
['Fooled by Randomness', 'Sapiens', 'Lenin on the Train']
```

In [40]:

```
1 list_s
```

Out[40]:

```
0    Fooled by Randomness
1                 Sapiens
2      Lenin on the Train
dtype: object
```

In [41]:

```
1 # Index must be a collection of some kind, could be lists or tuples
2 pd.Series(books_list, index=('funny','serious and amusing', 'kinda interesting'))
```

Out[41]:

```
funny                  Fooled by Randomness
serious and amusing                 Sapiens
kinda interesting        Lenin on the Train
dtype: object
```

In [42]:

```
1 pd.Series(data=books_list, index=['funny','serious and amusing', 'kinda interesting'],
```

Out[42]:

```
funny                  Fooled by Randomness
serious and amusing                 Sapiens
kinda interesting        Lenin on the Train
dtype: object
```

In [43]:

```
1 pd.__version__
```

Out[43]:

```
'1.0.1'
```

In [44]:

```
1 list_s.index
```

Out[44]:

```
RangeIndex(start=0, stop=3, step=1)
```

In [45]:

```
1 list_s.index.dtype
```

Out[45]:

```
dtype('int64')
```

In [46]:

```
1 type(list_s.index)
```

Out[46]:

```
pandas.core.indexes.range.RangeIndex
```

In [47]:

```
1 pd.RangeIndex(start=4, stop=7, step=1)
```

Out[47]:

```
RangeIndex(start=4, stop=7, step=1)
```

In [48]:

```
1 list(pd.RangeIndex(start=4, stop=7, step=1))
```

Out[48]:

```
[4, 5, 6]
```

In [49]:

```
1 list(pd.RangeIndex(start=10, stop=-11, step=-2))
```

Out[49]:

```
[10, 8, 6, 4, 2, 0, -2, -4, -6, -8, -10]
```

Series and Index Names

In [50]:

```
1 list_s
```

Out[50]:

```
0    Fooled by Randomness
1                 Sapiens
2      Lenin on the Train
dtype: object
```

In [51]:

```
1 book_series = list_s
```

intelligible: capable of being understood

In [52]:

```
1 book_series
```

Out[52]:

```
0    Fooled by Randomness
1                 Sapiens
2      Lenin on the Train
dtype: object
```

Methods vs Attributes
Method is a function bound to the object
Attribute is a variable bound to the object

In [53]:

```
1 book_series.size # This is an attribute
```

Out[53]:

```
3
```

In [54]:

```
1 list_s.equals(dict_s)# This is a method
```

Out[54]:

```
True
```

In [55]:

```
1 list_s.dtype
```

Out[55]:

```
dtype('O')
```

In [56]:

```
# The name attribute
book_series.name
```

In [57]:

```
book_series.name == None
```

Out[57]:

```
True
```

In [58]:

```
book_series.name = 'my favorite books'
```

In [59]:

```
book_series
```

Out[59]:

```
0    Fooled by Randomness
1                 Sapiens
2      Lenin on the Train
Name: my favorite books, dtype: object
```

In [60]:

```
book_series.index.name
```

In [61]:

```
book_series.index.name == None
```

Out[61]:

```
True
```

In [62]:

```
book_series.index.name = 'My Books'
```

In [63]:

```
book_series
```

Out[63]:

```
My Books
0    Fooled by Randomness
1                 Sapiens
2      Lenin on the Train
Name: my favorite books, dtype: object
```

Skill Challenge

In [64]:

```
actor_names = ['Chris Hemsworth', 'Hrithik Roshan', 'Vin Diesel', 'Idris Elba']
len(actor_names)
```

Out[64]:

```
4
```

In [65]:

```
actor_ages = [35, 44, 38, 47]
len(actor_ages)
```

Out[65]:

```
4
```

In [66]:

```
actor_s = pd.Series(data=actor_ages, index = actor_names)
```

In [67]:

```
actor_s
```

Out[67]:

```
Chris Hemsworth    35
Hrithik Roshan     44
Vin Diesel         38
Idris Elba         47
dtype: int64
```

In [68]:

```
actor_s.name = 'Actors'
actor_s.index.name = 'Actor name'
```

In [69]:

```
actor_s
```

Out[69]:

```
Actor name
Chris Hemsworth    35
Hrithik Roshan     44
Vin Diesel         38
Idris Elba         47
Name: Actors, dtype: int64
```

In [70]:

```
list(zip(actor_names, actor_ages)) #this creates a list of tuples
```

Out[70]:

```
[('Chris Hemsworth', 35),
 ('Hrithik Roshan', 44),
 ('Vin Diesel', 38),
 ('Idris Elba', 47)]
```

In [71]:

```
dict(zip(actor_names, actor_ages))
```

Out[71]:

```
{'Chris Hemsworth': 35,
 'Hrithik Roshan': 44,
 'Vin Diesel': 38,
 'Idris Elba': 47}
```

In [72]:

```
pd.Series(dict(zip(actor_names, actor_ages)), name = 'Actors')
```

Out[72]:

```
Chris Hemsworth    35
Hrithik Roshan     44
Vin Diesel         38
Idris Elba         47
Name: Actors, dtype: int64
```

In [73]:

```
# OR
{name:age for name, age in zip(actor_names,actor_ages)}# dictionary comprehension
```

Out[73]:

```
{'Chris Hemsworth': 35,
 'Hrithik Roshan': 44,
 'Vin Diesel': 38,
 'Idris Elba': 47}
```

In [74]:

```
pd.Series({name:age for name, age in zip(actor_names,actor_ages)}, name='Actors')
```

Out[74]:

```
Chris Hemsworth    35
Hrithik Roshan     44
Vin Diesel         38
Idris Elba         47
Name: Actors, dtype: int64
```

The head() and tail() methods

In [75]:

```
1 int_series = pd.Series([i for i in range(60)])
```

In [76]:

```
1 int_series
```

Out[76]:

```
0      0
1      1
2      2
3      3
4      4
5      5
6      6
7      7
8      8
9      9
10    10
11    11
12    12
13    13
14    14
15    15
16    16
17    17
18    18
19    19
20    20
21    21
22    22
23    23
24    24
25    25
26    26
27    27
28    28
29    29
30    30
31    31
32    32
33    33
34    34
35    35
36    36
37    37
38    38
39    39
40    40
41    41
42    42
43    43
44    44
45    45
46    46
47    47
48    48
49    49
50    50
51    51
52    52
53    53
54    54
```

```
55    55
56    56
57    57
58    58
59    59
dtype: int64
```

In [77]:

```
1 int_series.size
```

Out[77]:

```
60
```

In [78]:

```
1 len(int_series)
```

Out[78]:

```
60
```

In [79]:

```
1 int_series.head()
```

Out[79]:

```
0    0
1    1
2    2
3    3
4    4
dtype: int64
```

In [80]:

```
1 int_series.tail()
```

Out[80]:

```
55    55
56    56
57    57
58    58
59    59
dtype: int64
```

In [81]:

```
1 int_series.head(3)
```

Out[81]:

```
0    0
1    1
2    2
dtype: int64
```

In [82]:

```
1 int_series.head(n = 3)
```

Out[82]:

```
0    0
1    1
2    2
dtype: int64
```

In [83]:

```
1 int_series.tail(n = 7)
```

Out[83]:

```
53    53
54    54
55    55
56    56
57    57
58    58
59    59
dtype: int64
```

In [84]:

```
1 pd.Series(range(100000))
```

Out[84]:

```
0            0
1            1
2            2
3            3
4            4
         ...
99995    99995
99996    99996
99997    99997
99998    99998
99999    99999
Length: 100000, dtype: int64
```

In [85]:

```
1 pd.options.display.max_rows = 10
```

In [86]:

```
1 names = ['adura', 'segun', 'ojukotimi', 'abodunde']
2 ages = [22, 24, 26, 21]
3 heights = [176.32, 143.42, 156.9, 132.67]
```

In [87]:

```
for name,age,height in zip(names,ages,heights):
    print(name+" : "+str(height)+" : "+str(age))
```

```
adura : 176.32 : 22
segun : 143.42 : 24
ojukotimi : 156.9 : 26
abodunde : 132.67 : 21
```

In [88]:

```
pd.Series({name:height for name,height in zip(names,heights)})
```

Out[88]:

```
adura        176.32
segun        143.42
ojukotimi    156.90
abodunde     132.67
dtype: float64
```

Extracting by Index Position

In [89]:

```
from string import ascii_lowercase
```

In [90]:

```
ascii_lowercase
```

Out[90]:

```
'abcdefghijklmnopqrstuvwxyz'
```

In [91]:

```
pd.Series('Andy')
```

Out[91]:

```
0    Andy
dtype: object
```

In [92]:

```
pd.Series(ascii_lowercase)
```

Out[92]:

```
0    abcdefghijklmnopqrstuvwxyz
dtype: object
```

In [93]:

```
1 list(ascii_lowercase)
```

Out[93]:

```
['a',
 'b',
 'c',
 'd',
 'e',
 'f',
 'g',
 'h',
 'i',
 'j',
 'k',
 'l',
 'm',
 'n',
 'o',
 'p',
 'q',
 'r',
 's',
 't',
 'u',
 'v',
 'w',
 'x',
 'y',
 'z']
```

In [94]:

```
1 len(list(ascii_lowercase))
```

Out[94]:

```
26
```

In [95]:

```
1 letters = list(ascii_lowercase)
```

In [96]:

```
1 alphabet = pd.Series(letters)
```

In [97]:

```
1 alphabet.head(6)
```

Out[97]:

```
0    a
1    b
2    c
3    d
4    e
5    f
dtype: object
```

In [98]:

```
1 from string import ascii_uppercase
```

In [99]:

```
1 ascii_uppercase
```

Out[99]:

```
'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

In [100]:

```
1 len(ascii_uppercase)
```

Out[100]:

```
26
```

In [101]:

```
1 li = list(ascii_uppercase)
```

In [102]:

```
1 print(li,end = " ")
```

```
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O',
'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']
```

In [103]:

```
1 alphabet.head(1)
```

Out[103]:

```
0    a
dtype: object
```

In [104]:

```
1 alphabet[1]
```

Out[104]:

```
'b'
```

In [105]:

```
alphabet[0]
```

Out[105]:

```
'a'
```

1. What is the first letter?

#2. What is the 11th letter?

#3. What are the first three letters?

#4. What are the sixth through tenth letters?

#5. What are the last six letters?

In [106]:

```
#2
alphabet[10]
```

Out[106]:

```
'k'
```

In [107]:

```
#3.
alphabet[:3]
```

Out[107]:

```
0    a
1    b
2    c
dtype: object
```

In [108]:

```
#4.
alphabet[5:10]
```

Out[108]:

```
5    f
6    g
7    h
8    i
9    j
dtype: object
```

In [109]:

```
#5.
alphabet[-6:]
```

Out[109]:

```
20    u
21    v
22    w
23    x
24    y
25    z
dtype: object
```

Accessing Elements By Label

In [110]:

```
from string import ascii_uppercase
```

In [111]:

```
ascii_lowercase
```

Out[111]:

```
'abcdefghijklmnopqrstuvwxyz'
```

In [112]:

```
ascii_uppercase
```

Out[112]:

```
'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

In [113]:

```
labelled_alphabet = pd.Series(data = list(ascii_lowercase), index = map(lambda x: 'labe
```

In [114]:

```
labelled_alphabet.index
```

Out[114]:

```
Index(['label_A', 'label_B', 'label_C', 'label_D', 'label_E', 'label_F',
       'label_G', 'label_H', 'label_I', 'label_J', 'label_K', 'label_L',
       'label_M', 'label_N', 'label_O', 'label_P', 'label_Q', 'label_R',
       'label_S', 'label_T', 'label_U', 'label_V', 'label_W', 'label_X',
       'label_Y', 'label_Z'],
      dtype='object')
```

In [115]:

```
1 labelled_alphabet
```

Out[115]:

```
label_A    a
label_B    b
label_C    c
label_D    d
label_E    e
          ..
label_V    v
label_W    w
label_X    x
label_Y    y
label_Z    z
Length: 26, dtype: object
```

In [116]:

```
1 labelled_alphabet.head()
```

Out[116]:

```
label_A    a
label_B    b
label_C    c
label_D    d
label_E    e
dtype: object
```

1. What is the first letter?

#2. What is the 11th letter?

#3. What are the first three letters?

#4. What are the sixth through tenth letters?

#5. What are the last six letters?

In [117]:

```
1 #1.
2 labelled_alphabet[0]
```

Out[117]:

```
'a'
```

In [118]:

```
1 labelled_alphabet['label_A']
```

Out[118]:

```
'a'
```

In [119]:

```
#2.
labelled_alphabet['label_K']
```

Out[119]:

```
'k'
```

In [120]:

```
#3.
labelled_alphabet[:'label_C']
```

Out[120]:

```
label_A    a
label_B    b
label_C    c
dtype: object
```

In [121]:

```
#4.
labelled_alphabet['label_F':'label_J']
```

Out[121]:

```
label_F    f
label_G    g
label_H    h
label_I    i
label_J    j
dtype: object
```

In [122]:

```
#6.
labelled_alphabet['label_U':]
```

Out[122]:

```
label_U    u
label_V    v
label_W    w
label_X    x
label_Y    y
label_Z    z
dtype: object
```

The add_prefix() and add_suffix() methods

In [123]:

```
1 alphabet.head()
```

Out[123]:

```
0    a
1    b
2    c
3    d
4    e
dtype: object
```

In [124]:

```
1 alphabet.add_prefix('label_')
```

Out[124]:

```
label_0     a
label_1     b
label_2     c
label_3     d
label_4     e
           ..
label_21    v
label_22    w
label_23    x
label_24    y
label_25    z
Length: 26, dtype: object
```

In [125]:

```
1 labelled_alphabet
```

Out[125]:

```
label_A    a
label_B    b
label_C    c
label_D    d
label_E    e
          ..
label_V    v
label_W    w
label_X    x
label_Y    y
label_Z    z
Length: 26, dtype: object
```

In [126]:

```
1 alphabet = alphabet.add_suffix('_some_cool_ending')
```

In [127]:

```
1 alphabet
```

Out[127]:

```
0_some_cool_ending     a
1_some_cool_ending     b
2_some_cool_ending     c
3_some_cool_ending     d
4_some_cool_ending     e
                      ..
21_some_cool_ending    v
22_some_cool_ending    w
23_some_cool_ending    x
24_some_cool_ending    y
25_some_cool_ending    z
Length: 26, dtype: object
```

Using Dot Notation

In [128]:

```
1 labelled_alphabet.label_Y
```

Out[128]:

```
'y'
```

Boolean Masks And The .loc Indexer

In [129]:

```
1 labelled_alphabet['label_F':'label_J']# boolean masking
```

Out[129]:

```
label_F    f
label_G    g
label_H    h
label_I    i
label_J    j
dtype: object
```

In [130]:

```
1 # loc : used for label extraction
2 labelled_alphabet.loc['label_F':'label_J'] # the loc indexer
```

Out[130]:

```
label_F    f
label_G    g
label_H    h
label_I    i
label_J    j
dtype: object
```

boolean

boolean

In [131]:

```
1 book_series
```

Out[131]:

```
My Books
0    Fooled by Randomness
1                 Sapiens
2      Lenin on the Train
Name: my favorite books, dtype: object
```

In [132]:

```
1 book_series.loc[[True,True,True]]
```

Out[132]:

```
My Books
0    Fooled by Randomness
1                 Sapiens
2      Lenin on the Train
Name: my favorite books, dtype: object
```

In [133]:

```
1 book_series.loc[[True,False,True]]
```

Out[133]:

```
My Books
0    Fooled by Randomness
2      Lenin on the Train
Name: my favorite books, dtype: object
```

In [134]:

```
1 book_series.loc[[False,True,False]]
```

Out[134]:

```
My Books
1    Sapiens
Name: my favorite books, dtype: object
```

In [135]:

```
labelled_alphabet.loc[[True for i in range(len(labelled_alphabet))]]#we could use range
```

Out[135]:

```
label_A    a
label_B    b
label_C    c
label_D    d
label_E    e
          ..
label_V    v
label_W    w
label_X    x
label_Y    y
label_Z    z
Length: 26, dtype: object
```

In [136]:

```
labelled_alphabet.loc[[True if i%2==0 else False for i in range(26)]]
```

Out[136]:

```
label_A    a
label_C    c
label_E    e
label_G    g
label_I    i
          ..
label_Q    q
label_S    s
label_U    u
label_W    w
label_Y    y
Length: 13, dtype: object
```

In [137]:

```
pd.Series(['A','B','C'])[[True,False,False]]
```

Out[137]:

```
0    A
dtype: object
```

Extraction By Position with .iloc

In [138]:

```
# iloc => integer loc => indexing by position
# loc => location => indexing by label
```

In [139]:

```
labelled_alphabet.iloc[0]
```

Out[139]:

```
'a'
```

In [140]:

```
labelled_alphabet.iloc[1]
```

Out[140]:

```
'b'
```

In [141]:

```
labelled_alphabet.iloc[1:3]
```

Out[141]:

```
label_B    b
label_C    c
dtype: object
```

In [142]:

```
labelled_alphabet.iloc[[1,4,9]]
```

Out[142]:

```
label_B    b
label_E    e
label_J    j
dtype: object
```

Using callables with .loc and .iloc

used for highly cutomizable indexing
work with [], .loc and .iloc
a single-argument function that returns indexing output
a list of labels
list of booleans
a slice,etc.

In [143]:

```
labelled_alphabet.loc['label_V']
```

Out[143]:

```
'v'
```

In [144]:

```
#labelled_alphabet.loc['label_V', 'label_A'] this would give an error
```

In [145]:

```
labelled_alphabet.loc[['label_V', 'label_A']]
```

Out[145]:

```
label_V    v
label_A    a
dtype: object
```

In [146]:

```
labelled_alphabet.loc[lambda x: 'label_V']
```

Out[146]:

```
'v'
```

In [147]:

```
labelled_alphabet.loc[lambda x: ['label_V', 'label_A']]
```

Out[147]:

```
label_V    v
label_A    a
dtype: object
```

In [148]:

```
labelled_alphabet.loc[lambda x: [True for i in range(x.size)]]
```

Out[148]:

```
label_A    a
label_B    b
label_C    c
label_D    d
label_E    e
           ..
label_V    v
label_W    w
label_X    x
label_Y    y
label_Z    z
Length: 26, dtype: object
```

In [149]:

```
def every_fifth(x):
    return [True if i%5==0 else False for i in range(x.size)]
```

In [150]:

```
labelled_alphabet.iloc[every_fifth]
```

Out[150]:

```
label_A    a
label_F    f
label_K    k
label_P    p
label_U    u
label_Z    z
dtype: object
```

In [151]:

```
def every_fifth(x):
    return [True if (i+1)%5==0 else False for i in range(x.size)]
```

In [152]:

```
labelled_alphabet.iloc[every_fifth]
```

Out[152]:

```
label_E    e
label_J    j
label_O    o
label_T    t
label_Y    y
dtype: object
```

Seleting with .get()

In [153]:

```
labelled_alphabet.get('label_I')
```

Out[153]:

```
'i'
```

In [154]:

```
labelled_alphabet.loc['label_I']
```

Out[154]:

```
'i'
```

In [155]:

```
labelled_alphabet.get('label_inexistent', default = 'what you are looking for does not
```

Out[155]:

```
'what you are looking for does not exist'
```

In [156]:

```
1 labelled_alphabet.get(24)
```

Out[156]:

```
'y'
```

In [157]:

```
1 labelled_alphabet.iloc[24]
```

Out[157]:

```
'y'
```

In [158]:

```
1 labelled_alphabet[24]
```

Out[158]:

```
'y'
```

Skill challenge

1. Create a series of length 100 containing the squares of integers from 0 to 99. Assign it to the variable squares.

In [159]:

```
1 squares = pd.Series([x*x for x in range(100)])
```

2. Extract the last 3 items from the squares series using square bracket indexing

In [160]:

```
1 squares[-3:]
```

Out[160]:

```
97    9409
98    9604
99    9801
dtype: int64
```

3. Repeat step 2 but using the .tail() method instead

In [161]:

```
1 squares.tail(3)
```

Out[161]:

```
97    9409
98    9604
99    9801
dtype: int64
```

4. Verify that the output of 2 steps and 3 is the same using the .equals() method

In [162]:

```
1 squares[-3:].equals(squares.tail(3))
```

Out[162]:

```
True
```

SERIES PartII

In [163]:

```
1 import os
```

In [164]:

```
1 pwd
```

Out[164]:

```
'C:\\Users\\Adura\\Documents\\Data Science and Machine Learning Projects\\ju
pyter'
```

In [166]:

```
1 alcohol = pd.read_csv("drinks.csv", usecols=['country', 'wine_servings'], index_col = '
```

In [167]:

```
1 alcohol.head()
```

Out[167]:

```
country
Afghanistan      NaN
Albania         54.0
Algeria         14.0
Andorra        312.0
Angola          45.0
Name: wine_servings, dtype: float64
```

In [168]:

```
1 type(alcohol)
```

Out[168]:

```
pandas.core.series.Series
```

SERIES SIZING WITH .size, .shape, And len ()

In [169]:

```
1 alcohol.size
```

Out[169]:

```
193
```

In [170]:

```
1 alcohol
```

Out[170]:

```
country
Afghanistan      NaN
Albania         54.0
Algeria         14.0
Andorra        312.0
Angola          45.0
               ...
Venezuela        3.0
Vietnam          1.0
Yemen            NaN
Zambia           4.0
Zimbabwe         4.0
Name: wine_servings, Length: 193, dtype: float64
```

In [171]:

```
1 alcohol.values
```

Out[171]:

```
array([ nan,  54.,  14., 312.,  45.,  45., 221.,  11., 212., 191.,   5.,
        51.,   7.,  nan,  36.,  42., 212.,   8.,  13.,  nan,   8.,   8.,
        35.,  16.,   1.,  94.,   7.,  nan,   7.,  16.,   1.,   4.,   1.,
         1.,   1., 172.,   8.,   3.,   1.,   9.,  74.,  11., 254.,   5.,
       113., 134.,  nan,   1., 278.,   3.,  26.,   9.,   3.,   1.,   2.,
       233.,  nan,  59.,  nan,   1.,  97.,  37.,  59.,   1., 149., 175.,
         1., 218.,  28.,   2.,   2.,  21.,   1.,   1.,   2., 185.,  78.,
        nan,  nan,  nan,  nan, 165.,   9., 237.,   9.,  16.,   1.,  12.,
         2.,   1.,  nan,   6., 123.,  62.,  31.,  nan,   2.,  nan,  56.,
       271.,   4.,   1.,  nan,  nan,   1.,  12.,  nan,  nan,  18.,   5.,
        18.,  nan,   8., 128.,   1.,   5.,  nan,   1.,   8.,  nan,  19.,
       175.,   1.,   1.,   2.,   7., 129.,   1.,  nan,  23.,  18.,   1.,
        74.,  21.,   1.,  56., 339.,   7.,   9.,  18., 167.,  73.,  nan,
        32.,  71.,  11.,  24.,  nan,  14.,  nan,   7., 127.,  51.,   2.,
        11., 116., 276.,   1.,  nan,  81., 112.,  nan,  nan,   7.,   2.,
       186.,  28.,  16.,  nan,   1.,  86.,   4.,  19.,   5.,   7.,   2.,
         7.,  32.,   9.,  nan,  45.,   5., 195.,   1.,  84.,  22.,   8.,
        11.,   3.,   1.,  nan,   4.,   4.])
```

In [172]:

```
1 alcohol.index
```

Out[172]:

```
Index(['Afghanistan', 'Albania', 'Algeria', 'Andorra', 'Angola',
       'Antigua & Barbuda', 'Argentina', 'Armenia', 'Australia', 'Austria',
       ...
       'Tanzania', 'USA', 'Uruguay', 'Uzbekistan', 'Vanuatu', 'Venezuela',
       'Vietnam', 'Yemen', 'Zambia', 'Zimbabwe'],
      dtype='object', name='country', length=193)
```

In [173]:

```
1 alcohol.values.size
```

Out[173]:

```
193
```

In [174]:

```
1 alcohol.values.size==alcohol.index.size
```

Out[174]:

```
True
```

In [175]:

```
1 alcohol.shape
```

Out[175]:

```
(193,)
```

In [176]:

```
alcohol.size == alcohol.shape[0] # Using alcohol.shape alone would give False
```

Out[176]:

True

In [177]:

```
len(alcohol)
```

Out[177]:

193

Unique Values and Series Monotonicity

In [178]:

```
alcohol.is_unique
```

Out[178]:

False

In [179]:

```
alcohol.head().is_unique
```

Out[179]:

True

In [180]:

```
alcohol.nunique() # number of unique values in the series excluding the Nan values
```

Out[180]:

71

In [181]:

```
alcohol.nunique(dropna = False) # number of unique values in the series including the N
```

Out[181]:

72

In [182]:

```
pd.Series([1,1,2,2,3,4]).is_monotonic # i.e The values in the series increment continuc
```

Out[182]:

True

In [183]:

```
pd.Series([1,1,2,2,3,4,1]).is_monotonic #i.e The values in the series do not increment
```

Out[183]:

```
False
```

In [184]:

```
pd.Series(reversed([1,1,2,2,3,4])).is_monotonic_increasing
```

Out[184]:

```
False
```

In [185]:

```
pd.Series(reversed([1,1,2,2,3,4])).is_monotonic_decreasing
```

Out[185]:

```
True
```

The count() Method

In [186]:

```
alcohol.count()
```

Out[186]:

```
162
```

In [187]:

```
alcohol.size
```

Out[187]:

```
193
```

In [188]:

```
alcohol.count()# This method does not count the na values
```

Out[188]:

```
162
```

In [189]:

```
alcohol.hasnans
```

Out[189]:

```
True
```

Accessing and Counting NA's

In [190]:

```
1 alcohol.size
```

Out[190]:

```
193
```

In [191]:

```
1 alcohol.count()
```

Out[191]:

```
162
```

In [192]:

```
1 alcohol.isnull()
```

Out[192]:

```
country
Afghanistan     True
Albania        False
Algeria        False
Andorra        False
Angola         False
               ...
Venezuela      False
Vietnam        False
Yemen           True
Zambia         False
Zimbabwe       False
Name: wine_servings, Length: 193, dtype: bool
```

In [193]:

```
1 alcohol[alcohol.isnull()] #boolean masking
```

Out[193]:

```
country
Afghanistan   NaN
Bangladesh    NaN
Bhutan        NaN
Burundi       NaN
North Korea   NaN
               ..
Sri Lanka     NaN
Sudan         NaN
Tajikistan    NaN
Uganda        NaN
Yemen         NaN
Name: wine_servings, Length: 31, dtype: float64
```

In [194]:

```
1 alcohol[alcohol.isnull()].index
```

Out[194]:

```
Index(['Afghanistan', 'Bangladesh', 'Bhutan', 'Burundi', 'North Korea',
       'Eritrea', 'Ethiopia', 'India', 'Indonesia', 'Iran', 'Iraq', 'Kuwai
t',
       'Lesotho', 'Libya', 'Malaysia', 'Maldives', 'Marshall Islands',
       'Mauritania', 'Monaco', 'Myanmar', 'Nepal', 'Pakistan', 'Rwanda',
       'San Marino', 'Saudi Arabia', 'Somalia', 'Sri Lanka', 'Sudan',
       'Tajikistan', 'Uganda', 'Yemen'],
      dtype='object', name='country')
```

In [195]:

```
1 type(alcohol[alcohol.isnull()].index)
```

Out[195]:

```
pandas.core.indexes.base.Index
```

In [196]:

```
1 list(alcohol[alcohol.isnull()].index)
```

Out[196]:

```
['Afghanistan',
 'Bangladesh',
 'Bhutan',
 'Burundi',
 'North Korea',
 'Eritrea',
 'Ethiopia',
 'India',
 'Indonesia',
 'Iran',
 'Iraq',
 'Kuwait',
 'Lesotho',
 'Libya',
 'Malaysia',
 'Maldives',
 'Marshall Islands',
 'Mauritania',
 'Monaco',
 'Myanmar',
 'Nepal',
 'Pakistan',
 'Rwanda',
 'San Marino',
 'Saudi Arabia',
 'Somalia',
 'Sri Lanka',
 'Sudan',
 'Tajikistan',
 'Uganda',
 'Yemen']
```

In [197]:

```
1 len(alcohol[alcohol.isnull()].index) # This is cumbersome i.e difficult to work with
```

Out[197]:

```
31
```

In [198]:

```
1 alcohol.isnull().sum() # This is pandorable i.e involving pandas in our situation
```

Out[198]:

```
31
```

In [199]:

```
1 alcohol.isna().sum()
```

Out[199]:

```
31
```

In [200]:

```
1 sum([True,False,True]) # False here is regarded as zero
```

Out[200]:

```
2
```

In [201]:

```
1 all_ds = alcohol.size
```

In [202]:

```
1 nonnulls = alcohol.count()
```

In [203]:

```
1 nulls = alcohol.isnull().sum()
```

In [204]:

```
1 all_ds == nulls + nonnulls
```

Out[204]:

```
True
```

Bonus: Another Approach

ufunc -> universal function

In [205]:

```
1 np.isnan
```

Out[205]:

```
<ufunc 'isnan'>
```

In [206]:

```
1 ser = pd.Series(data=[True, False, None, 2], dtype=float)
```

In [207]:

```
1 np.isnan(ser)
```

Out[207]:

```
0    False
1    False
2     True
3    False
dtype: bool
```

In [211]:

```
1 ser.dtype
```

Out[211]:

```
dtype('float64')
```

In [212]:

```
1 alcohol[np.isnan]
```

Out[212]:

```
country
Afghanistan   NaN
Bangladesh    NaN
Bhutan        NaN
Burundi       NaN
North Korea   NaN
               ..
Sri Lanka     NaN
Sudan         NaN
Tajikistan    NaN
Uganda        NaN
Yemen         NaN
Name: wine_servings, Length: 31, dtype: float64
```

In [213]:

```
1 alcohol[np.isnan].size
```

Out[213]:

```
31
```

notnull() And notna()

In [214]:

```
# not null
alcohol.notnull()
```

Out[214]:

```
country
Afghanistan    False
Albania         True
Algeria         True
Andorra         True
Angola          True
               ...
Venezuela       True
Vietnam         True
Yemen          False
Zambia          True
Zimbabwe        True
Name: wine_servings, Length: 193, dtype: bool
```

In [215]:

```
alcohol.loc[alcohol.isna()]
```

Out[215]:

```
country
Afghanistan   NaN
Bangladesh    NaN
Bhutan        NaN
Burundi       NaN
North Korea   NaN
               ..
Sri Lanka     NaN
Sudan         NaN
Tajikistan    NaN
Uganda        NaN
Yemen         NaN
Name: wine_servings, Length: 31, dtype: float64
```

In [216]:

```
alcohol.loc[alcohol.notnull()]
```

Out[216]:

```
country
Albania               54.0
Algeria               14.0
Andorra              312.0
Angola                45.0
Antigua & Barbuda     45.0
                     ...
Vanuatu               11.0
Venezuela              3.0
Vietnam                1.0
Zambia                 4.0
Zimbabwe               4.0
Name: wine_servings, Length: 162, dtype: float64
```

In [217]:

```
1 alcohol.notnull().sum()
```

Out[217]:

```
162
```

In [218]:

```
1 alcohol.count()
```

Out[218]:

```
162
```

In [219]:

```
1 alcohol.notnull().sum() + alcohol.isnull().sum() == alcohol.size
```

Out[219]:

```
True
```

Booleans are literally numbers in python

In [220]:

```
1 True + 19
```

Out[220]:

```
20
```

In [221]:

```
1 True + True - False + True * 3
```

Out[221]:

```
5
```

In [222]:

```
1 5/True
```

Out[222]:

```
5.0
```

5/False This gives a ZeroDivision Error because False here is seen as zero in python

In [223]:

```
1 bool.__mro__
```

Out[223]:

```
(bool, int, object)
```

In [224]:

```
1 int.__mro__
```

Out[224]:

```
(int, object)
```

Skill Challenge

1. Isolate the nonnulls in the alcohol series and assign them to the variable wine_servings

In [225]:

```
1 wine_servings = alcohol.loc[alcohol.notnull()]
```

In [226]:

```
1 wine_servings
```

Out[226]:

```
country
Albania               54.0
Algeria               14.0
Andorra              312.0
Angola                45.0
Antigua & Barbuda     45.0
                     ...
Vanuatu               11.0
Venezuela              3.0
Vietnam                1.0
Zambia                 4.0
Zimbabwe               4.0
Name: wine_servings, Length: 162, dtype: float64
```

2. What is the total wine consumed in wine_servings

In [227]:

```
1 wine_servings_total = wine_servings.sum()
```

In [228]:

```
1 wine_servings_total
```

Out[228]:

```
8221.0
```

3. In the wine_servings dataset, what was the total wine consumed by countries less than 100 servings?

In [229]:

```
1 wine_servings[wine_servings.values<100].sum()
```

Out[229]:

```
2416.0
```

In [230]:

```
1 alcohol
```

Out[230]:

```
country
Afghanistan      NaN
Albania         54.0
Algeria         14.0
Andorra        312.0
Angola          45.0
               ...
Venezuela        3.0
Vietnam          1.0
Yemen            NaN
Zambia           4.0
Zimbabwe         4.0
Name: wine_servings, Length: 193, dtype: float64
```

Dropping And Filling Nas

In [231]:

```
1 alcohol.dropna()
```

Out[231]:

```
country
Albania               54.0
Algeria               14.0
Andorra              312.0
Angola                45.0
Antigua & Barbuda     45.0
                     ...
Vanuatu               11.0
Venezuela              3.0
Vietnam                1.0
Zambia                 4.0
Zimbabwe               4.0
Name: wine_servings, Length: 162, dtype: float64
```

In [232]:

```
1 alcohol
```

Out[232]:

```
country
Afghanistan      NaN
Albania         54.0
Algeria         14.0
Andorra        312.0
Angola          45.0
               ...
Venezuela        3.0
Vietnam          1.0
Yemen            NaN
Zambia           4.0
Zimbabwe         4.0
Name: wine_servings, Length: 193, dtype: float64
```

In [233]:

```
1 new_alcohol = alcohol.dropna()
```

In [234]:

```
1 new_alcohol
```

Out[234]:

```
country
Albania               54.0
Algeria               14.0
Andorra              312.0
Angola                45.0
Antigua & Barbuda     45.0
                     ...
Vanuatu               11.0
Venezuela              3.0
Vietnam                1.0
Zambia                 4.0
Zimbabwe               4.0
Name: wine_servings, Length: 162, dtype: float64
```

In [235]:

```
1 alcohol
```

Out[235]:

```
country
Afghanistan      NaN
Albania         54.0
Algeria         14.0
Andorra        312.0
Angola          45.0
               ...
Venezuela        3.0
Vietnam          1.0
Yemen            NaN
Zambia           4.0
Zimbabwe         4.0
Name: wine_servings, Length: 193, dtype: float64
```

In [236]:

```
1 #alcohol.dropna(inplace=True). modifies the original alcohol series
```

In [237]:

```
1 alcohol.fillna(100, inplace=False)
```

Out[237]:

```
country
Afghanistan    100.0
Albania         54.0
Algeria         14.0
Andorra        312.0
Angola          45.0
               ...
Venezuela        3.0
Vietnam          1.0
Yemen          100.0
Zambia           4.0
Zimbabwe         4.0
Name: wine_servings, Length: 193, dtype: float64
```

Descriptiive Statistics

In [238]:

```
1 alcohol.sum()# Gives the sum of all values excluding the Nas
```

Out[238]:

```
8221.0
```

In [239]:

```
1 # average
```

In [240]:

```
alcohol.count() # hol.sumgives number of values in the series excluding the Nas
```

Out[240]:

```
162
```

In [241]:

```
avg = alcohol.sum()/alcohol.count()
```

In [242]:

```
avg
```

Out[242]:

```
50.74691358024691
```

In [243]:

```
# OR
#The mean method can be used as well
alcohol.mean()
```

Out[243]:

```
50.74691358024691
```

In [244]:

```
#The median-> The middlemost element in a sorted list of element
```

In [245]:

```
alcohol.median()
```

Out[245]:

```
11.5
```

In [246]:

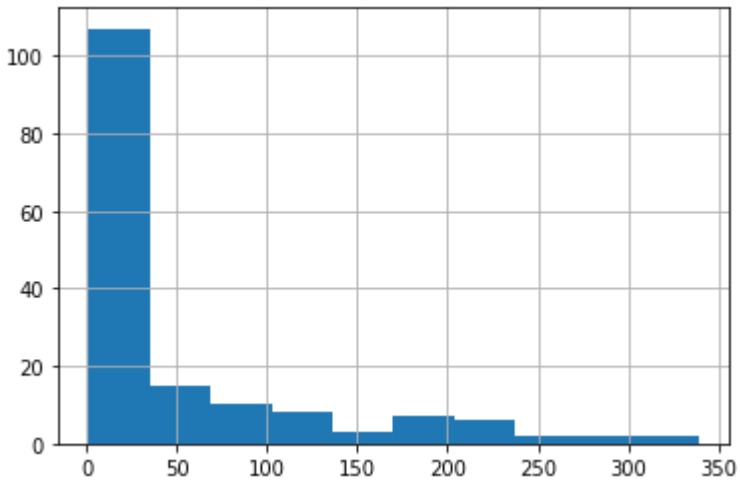
```
alcohol.quantile(q=.5)
```

Out[246]:

```
11.5
```

In [247]:

```
1 alcohol.hist();
```



In [248]:

```
1 #IQR
```

In [249]:

```
1 iqr = alcohol.quantile(q=.75) - alcohol.quantile(q=.25)
```

In [250]:

```
1 iqr
```

Out[250]:

```
58.25
```

In [251]:

```
1 #OR
2 iqr = alcohol.quantile(.75) - alcohol.quantile(.25)
```

In [252]:

```
1 iqr
```

Out[252]:

```
58.25
```

In [253]:

```
1 #min and max
```

In [254]:

```
1 alcohol.min() # gives the minimum wine_servings in our alcohol series
```

Out[254]:

```
1.0
```

In [255]:

```
1 alcohol.max()# gives the maximum wine_servings in our alcohol series
```

Out[255]:

```
339.0
```

In [256]:

```
1 # standard deviation
```

In [257]:

```
1 alcohol.std()
```

Out[257]:

```
76.13491716376173
```

In [258]:

```
1 alcohol.std()**2
```

Out[258]:

```
5796.52561153286
```

In [259]:

```
1 alcohol.var()
```

Out[259]:

```
5796.52561153286
```

In [260]:

```
1 alcohol.std()**2 == alcohol.var()
```

Out[260]:

```
True
```

In [261]:

```
1 #The describe() Method
```

In [262]:

```
1 alcohol.describe()
```

Out[262]:

```
count    162.000000
mean      50.746914
std       76.134917
min        1.000000
25%        3.000000
50%       11.500000
75%       61.250000
max      339.000000
Name: wine_servings, dtype: float64
```

In [263]:

```
1 alcohol.describe(percentiles=[.79, .19])
```

Out[263]:

```
count    162.000000
mean      50.746914
std       76.134917
min        1.000000
19%        2.000000
50%       11.500000
79%       81.570000
max      339.000000
Name: wine_servings, dtype: float64
```

In [264]:

```
1 alcohol.describe(percentiles=[.79, .19], include=float, exclude=object)
```

Out[264]:

```
count    162.000000
mean      50.746914
std       76.134917
min        1.000000
19%        2.000000
50%       11.500000
79%       81.570000
max      339.000000
Name: wine_servings, dtype: float64
```

Mode and Value_counts()

In [265]:

```
1 #mode-> one that occurs most frequently in a collection values
```

In [266]:

```
1 alcohol.mode()
```

Out[266]:

```
0    1.0
dtype: float64
```

In [267]:

```
1 alcohol == 1
```

Out[267]:

```
country
Afghanistan    False
Albania        False
Algeria        False
Andorra        False
Angola         False
               ...
Venezuela      False
Vietnam         True
Yemen          False
Zambia         False
Zimbabwe       False
Name: wine_servings, Length: 193, dtype: bool
```

In [268]:

```
1 alcohol[alcohol == 1].size
```

Out[268]:

```
28
```

value counts

In [269]:

```
1 alcohol.value_counts()
```

Out[269]:

```
1.0      28
2.0      10
7.0       9
8.0       7
5.0       6
         ..
185.0     1
218.0     1
84.0      1
149.0     1
54.0      1
Name: wine_servings, Length: 71, dtype: int64
```

In [270]:

```
alcohol.value_counts().loc[1]
```

Out[270]:

```
28
```

In [271]:

```
alcohol.value_counts().iloc[0]
```

Out[271]:

```
28
```

In [272]:

```
alcohol.value_counts(normalize=True) # This gives the percentage of each value in the s
```

Out[272]:

```
1.0      0.172840
2.0      0.061728
7.0      0.055556
8.0      0.043210
5.0      0.037037
           ...
185.0    0.006173
218.0    0.006173
84.0     0.006173
149.0    0.006173
54.0     0.006173
Name: wine_servings, Length: 71, dtype: float64
```

In [273]:

```
28/alcohol.count()# relative frequency of 1
```

Out[273]:

```
0.1728395061728395
```

idxmax() And idxmin()

In [274]:

```
alcohol.size
```

Out[274]:

```
193
```

In [275]:

```
alcohol.max()
```

Out[275]:

```
339.0
```

In [276]:

```
1 alcohol == alcohol.max()
```

Out[276]:

```
country
Afghanistan    False
Albania        False
Algeria        False
Andorra        False
Angola         False
               ...
Venezuela      False
Vietnam        False
Yemen          False
Zambia         False
Zimbabwe       False
Name: wine_servings, Length: 193, dtype: bool
```

In [277]:

```
1 alcohol[alcohol == alcohol.max()]
```

Out[277]:

```
country
Portugal    339.0
Name: wine_servings, dtype: float64
```

In [278]:

```
1 alcohol.index
```

Out[278]:

```
Index(['Afghanistan', 'Albania', 'Algeria', 'Andorra', 'Angola',
       'Antigua & Barbuda', 'Argentina', 'Armenia', 'Australia', 'Austria',
       ...
       'Tanzania', 'USA', 'Uruguay', 'Uzbekistan', 'Vanuatu', 'Venezuela',
       'Vietnam', 'Yemen', 'Zambia', 'Zimbabwe'],
      dtype='object', name='country', length=193)
```

In [279]:

```
1 alcohol.index[1]
```

Out[279]:

```
'Albania'
```

In [280]:

```
1 alcohol[alcohol == alcohol.max()].index
```

Out[280]:

```
Index(['Portugal'], dtype='object', name='country')
```

In [281]:

```
1 type(alcohol[alcohol == alcohol.max()].index)
```

Out[281]:

```
pandas.core.indexes.base.Index
```

In [282]:

```
1 alcohol[alcohol == alcohol.max()].index[0] # This is too cumbersome
```

Out[282]:

```
'Portugal'
```

In [283]:

```
1 alcohol[alcohol == alcohol.max()].index
```

Out[283]:

```
Index(['Portugal'], dtype='object', name='country')
```

In [284]:

```
1 alcohol.idxmax()
```

Out[284]:

```
'Portugal'
```

In [285]:

```
1 alcohol.idxmin()
```

Out[285]:

```
'Brunei'
```

In [286]:

```
1 alcohol.min()
```

Out[286]:

```
1.0
```

In [287]:

```
1 alcohol.value_counts().head(1)
```

Out[287]:

```
1.0    28
Name: wine_servings, dtype: int64
```

In [288]:

```
alcohol[alcohol == alcohol.min()]
```

Out[288]:

```
country
Brunei                      1.0
Cambodia                    1.0
Canada                      1.0
Central African Republic    1.0
Chad                        1.0
                           ...
Philippines                 1.0
Solomon Islands             1.0
Thailand                    1.0
Tanzania                    1.0
Vietnam                     1.0
Name: wine_servings, Length: 28, dtype: float64
```

In [289]:

```
alcohol[alcohol.idxmax()]
```

Out[289]:

```
339.0
```

Sorting with sort_values()

In [290]:

```
# We sort by value or index
# We'll be sorting by values
```

In [291]:

```
alcohol.sort_values()
```

Out[291]:

```
country
Thailand           1.0
Solomon Islands    1.0
Brunei             1.0
Haiti              1.0
Cambodia           1.0
                  ...
Sri Lanka          NaN
Sudan              NaN
Tajikistan         NaN
Uganda             NaN
Yemen              NaN
Name: wine_servings, Length: 193, dtype: float64
```

In [292]:

```
1 alcohol.sort_values(ascending=False)
```

Out[292]:

```
country
Portugal      339.0
Andorra       312.0
Denmark       278.0
Slovenia      276.0
Luxembourg    271.0
              ...
Sri Lanka       NaN
Sudan           NaN
Tajikistan      NaN
Uganda          NaN
Yemen           NaN
Name: wine_servings, Length: 193, dtype: float64
```

In [293]:

```
1 alcohol.sort_values(ascending=False, na_position='last', kind='quicksort', inplace=True
```

In [294]:

```
1 alcohol.head()
```

Out[294]:

```
country
Portugal      339.0
Andorra       312.0
Denmark       278.0
Slovenia      276.0
Luxembourg    271.0
Name: wine_servings, dtype: float64
```

nlargest() and nsmallest()

In [295]:

```
1 alcohol.min()
```

Out[295]:

```
1.0
```

In [296]:

```
1 alcohol.max()
```

Out[296]:

```
339.0
```

In [297]:

```
1 alcohol.sort_values(ascending=False).head(10)
```

Out[297]:

```
country
Portugal             339.0
Andorra              312.0
Denmark              278.0
Slovenia             276.0
Luxembourg           271.0
Croatia              254.0
Italy                237.0
Equatorial Guinea    233.0
Argentina            221.0
Greece               218.0
Name: wine_servings, dtype: float64
```

In [298]:

```
1 alcohol.sort_values(ascending=False)[:10]
```

Out[298]:

```
country
Portugal             339.0
Andorra              312.0
Denmark              278.0
Slovenia             276.0
Luxembourg           271.0
Croatia              254.0
Italy                237.0
Equatorial Guinea    233.0
Argentina            221.0
Greece               218.0
Name: wine_servings, dtype: float64
```

In [299]:

```
1 alcohol.nsmallest(10)
```

Out[299]:

```
country
Niger                1.0
Nicaragua            1.0
Namibia              1.0
Morocco              1.0
Mali                 1.0
Malawi               1.0
Oman                 1.0
Papua New Guinea     1.0
Vietnam              1.0
Philippines          1.0
Name: wine_servings, dtype: float64
```

In [300]:

```
1 alcohol.nlargest(10)
```

Out[300]:

```
country
Portugal             339.0
Andorra              312.0
Denmark              278.0
Slovenia             276.0
Luxembourg           271.0
Croatia              254.0
Italy                237.0
Equatorial Guinea    233.0
Argentina            221.0
Greece               218.0
Name: wine_servings, dtype: float64
```

Sorting with sort_index()

In [301]:

```
1 alcohol.head()
```

Out[301]:

```
country
Portugal      339.0
Andorra       312.0
Denmark       278.0
Slovenia      276.0
Luxembourg    271.0
Name: wine_servings, dtype: float64
```

In [302]:

```
1 alcohol.sort_index(ascending=False, na_position='first', inplace=True)
```

In [303]:

```
1 alcohol.index.isnull().sum()
```

Out[303]:

```
0
```

In [304]:

```
alcohol.index.isnull()
```

Out[304]:

```
array([False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False])
```

In [305]:

```
alcohol.values
```

Out[305]:

```
array([  4.,   4.,  nan,   1.,   3.,  11.,   8.,  22., 195.,   5.,  45.,
        nan,  84.,   9.,  32.,   7.,   2.,   7.,   5.,  19.,   4.,   1.,
         1.,  nan,  16.,  28., 186.,   2.,   7.,  nan,  11.,  71.,  32.,
        nan, 112.,   9.,  81.,  nan,   1., 276., 116.,  11.,   2.,  51.,
       127.,   7.,  nan,  14.,  nan,  24.,  nan,  73., 167.,   7., 339.,
        56.,   1.,  21.,  74.,   1.,  18.,  23.,  nan,   1., 129.,  nan,
         7.,   2.,   1.,   1., 175.,  19.,  nan,   8.,   1.,  nan,   5.,
         1., 128.,   8.,  nan,  18.,  18.,   5.,  18.,  nan,  nan,  12.,
         1.,  nan,  nan,   1.,   4.,  86., 271.,  56.,  nan,   2.,  nan,
        31.,  62., 123.,   6.,  nan,   1.,   2.,  12.,   1.,  16.,   9.,
       237.,   9., 165.,  nan,  nan,  nan,  nan,  78., 185.,   2.,   1.,
         1.,  21.,   2.,   2.,  28., 218.,   1., 175., 149.,   1.,  59.,
        37.,  97.,   1.,  nan,  59.,  nan, 233.,   2.,   1.,   3.,   9.,
        26.,   3., 278.,   1., 134., 113.,   5., 254.,   7.,  11.,  74.,
         9.,   1.,   3.,   8., 172.,   1.,   1.,   1.,   4.,   1.,  16.,
        nan,   7.,  94.,   1.,  16.,  35.,   8.,   8.,  nan,  13.,   8.,
       212.,  42.,  36.,  nan,   7.,  51.,   5., 191., 212.,  11., 221.,
        45.,  45., 312.,  14.,  54.,  nan])
```

In [306]:

```
1 alcohol.head()
```

Out[306]:

```
country
Zimbabwe     4.0
Zambia       4.0
Yemen        NaN
Vietnam      1.0
Venezuela    3.0
Name: wine_servings, dtype: float64
```

Skill Challenge

In [307]:

```
1 # 1. Select the countries from alcohol that has more than 50 wine_servings and save the
```

In [308]:

```
1 fifty_plus = alcohol[alcohol>50]
```

In [309]:

```
1 fifty_plus.head(5)
```

Out[309]:

```
country
United Kingdom    195.0
USA                84.0
Sweden            186.0
St. Lucia          71.0
Spain             112.0
Name: wine_servings, dtype: float64
```

In [310]:

```
1 fifty_plus.count()
```

Out[310]:

```
48
```

In [311]:

```
1 #2. From fifty_plus, choose the countries with the smallest 20 wine serving values
```

In [312]:

```
1 fifty_plus.nsmallest(20)
```

Out[312]:

```
country
Seychelles     51.0
Bahamas        51.0
Albania        54.0
Poland         56.0
Lithuania      56.0
               ...
Macedonia      86.0
Bulgaria       94.0
Finland        97.0
Spain         112.0
Cyprus        113.0
Name: wine_servings, Length: 20, dtype: float64
```

In [313]:

```
1 #3. What is the mean, median and standard deviation for the sample from step 2?
```

In [314]:

```
1 # Mean
2 fifty_plus.nsmallest(20).mean()
```

Out[314]:

```
74.25
```

In [315]:

```
1 # Median
2 fifty_plus.nsmallest(20).median()
```

Out[315]:

```
73.5
```

In [316]:

```
1 fifty_plus.nsmallest(20).quantile(q=.5)
```

Out[316]:

```
73.5
```

In [317]:

```
1 # Standard deviation
2 fifty_plus.nsmallest(20).std()
```

Out[317]:

```
19.07292100831631
```

In [318]:

```
1 fifty_plus.nsmallest(20).describe()
```

Out[318]:

```
count     20.000000
mean      74.250000
std       19.072921
min       51.000000
25%       58.250000
50%       73.500000
75%       84.500000
max      113.000000
Name: wine_servings, dtype: float64
```

Series Arithmetics And fill_value()

In [319]:

```
1 alcohol + 2
```

Out[319]:

```
country
Zimbabwe         6.0
Zambia           6.0
Yemen            NaN
Vietnam          3.0
Venezuela        5.0
                ...
Angola          47.0
Andorra        314.0
Algeria         16.0
Albania         56.0
Afghanistan      NaN
Name: wine_servings, Length: 193, dtype: float64
```

In [320]:

```
1 (alcohol-10) * 2
```

Out[320]:

```
country
Zimbabwe       -12.0
Zambia         -12.0
Yemen            NaN
Vietnam        -18.0
Venezuela      -14.0
                ...
Angola          70.0
Andorra        604.0
Algeria          8.0
Albania         88.0
Afghanistan      NaN
Name: wine_servings, Length: 193, dtype: float64
```

In [321]:

```
1 alcohol.head()
```

Out[321]:

```
country
Zimbabwe     4.0
Zambia       4.0
Yemen        NaN
Vietnam      1.0
Venezuela    3.0
Name: wine_servings, dtype: float64
```

In [322]:

```
1 alcohol.sort_index(inplace=True)
```

In [323]:

```
1 alcohol.head()
```

Out[323]:

```
country
Afghanistan      NaN
Albania         54.0
Algeria         14.0
Andorra        312.0
Angola          45.0
Name: wine_servings, dtype: float64
```

In [324]:

```
1 more_drinks = pd.Series({'Albania':6, 'Alberia':19,'Algeria':10, 'Afghanistan':100, 'Ye
```

In [325]:

```
1 alcohol + more_drinks # wrong
```

Out[325]:

```
Afghanistan     NaN
Albania        60.0
Alberia         NaN
Algeria        24.0
Andorra         NaN
               ...
Venezuela       NaN
Vietnam         NaN
Yemen           NaN
Zambia          NaN
Zimbabwe        NaN
Length: 194, dtype: float64
```

In [326]:

```
1 alcohol.add(more_drinks, fill_value=0)
```

Out[326]:

```
Afghanistan    100.0
Albania         60.0
Alberia         19.0
Algeria         24.0
Andorra        312.0
               ...
Venezuela        3.0
Vietnam          1.0
Yemen          101.0
Zambia           4.0
Zimbabwe         4.0
Length: 194, dtype: float64
```

In [327]:

```
1 alcohol-more_drinks
```

Out[327]:

```
Afghanistan     NaN
Albania        48.0
Alberia         NaN
Algeria         4.0
Andorra         NaN
               ...
Venezuela       NaN
Vietnam         NaN
Yemen           NaN
Zambia          NaN
Zimbabwe        NaN
Length: 194, dtype: float64
```

In [328]:

```
1 alcohol.subtract(more_drinks, fill_value=0)
```

Out[328]:

```
Afghanistan   -100.0
Albania         48.0
Alberia        -19.0
Algeria          4.0
Andorra        312.0
               ...
Venezuela        3.0
Vietnam          1.0
Yemen         -101.0
Zambia           4.0
Zimbabwe         4.0
Length: 194, dtype: float64
```

In [329]:

```
1 alcohol.divide(more_drinks, fill_value=1)
```

Out[329]:

```
Afghanistan      0.010000
Albania          9.000000
Alberia          0.052632
Algeria          1.400000
Andorra        312.000000
                   ...
Venezuela        3.000000
Vietnam          1.000000
Yemen            0.009901
Zambia           4.000000
Zimbabwe         4.000000
Length: 194, dtype: float64
```

In [330]:

```
1 alcohol.multiply(more_drinks, fill_value=1)
```

Out[330]:

```
Afghanistan    100.0
Albania        324.0
Alberia         19.0
Algeria        140.0
Andorra        312.0
               ...
Venezuela        3.0
Vietnam          1.0
Yemen          101.0
Zambia           4.0
Zimbabwe         4.0
Length: 194, dtype: float64
```

In [331]:

```
1 alcohol.head()
```

Out[331]:

```
country
Afghanistan      NaN
Albania         54.0
Algeria         14.0
Andorra        312.0
Angola          45.0
Name: wine_servings, dtype: float64
```

In [332]:

```
1 more_drinks.head()
```

Out[332]:

```
Albania          6
Alberia         19
Algeria         10
Afghanistan    100
Yemen          101
dtype: int64
```

Calculating Variance And Standard Deviation

In [333]:

```
1 (alcohol.subtract(alcohol.mean())**2).sum()/(alcohol.count()-1)
```

Out[333]:

```
5796.5256115328575
```

In [334]:

```
1 alcohol.var()
```

Out[334]:

```
5796.52561153286
```

In [335]:

```
1 alcohol.std()
```

Out[335]:

```
76.13491716376173
```

In [336]:

```
1 ((alcohol.subtract(alcohol.mean())**2).sum()/(alcohol.count()-1))**(1/2)
```

Out[336]:

```
76.13491716376171
```

Cumulative Operations

sum

In [337]:

```
1 alcohol.sum()
```

Out[337]:

```
8221.0
```

In [338]:

```
1 alcohol.cumsum()
```

Out[338]:

```
country
Afghanistan       NaN
Albania          54.0
Algeria          68.0
Andorra         380.0
Angola          425.0
                ...
Venezuela      8212.0
Vietnam        8213.0
Yemen             NaN
Zambia         8217.0
Zimbabwe       8221.0
Name: wine_servings, Length: 193, dtype: float64
```

In [339]:

```
1 x = 0
2 for i in range(194):
3     x = x + i
4 print(x)
```

```
18721
```

In [340]:

```
1 18528/193
```

Out[340]:

```
96.0
```

In [341]:

```
1 alcohol.tail()
```

Out[341]:

```
country
Venezuela    3.0
Vietnam      1.0
Yemen        NaN
Zambia       4.0
Zimbabwe     4.0
Name: wine_servings, dtype: float64
```

In [342]:

```
1 alcohol.head()
```

Out[342]:

```
country
Afghanistan      NaN
Albania         54.0
Algeria         14.0
Andorra        312.0
Angola          45.0
Name: wine_servings, dtype: float64
```

In [343]:

```
1 np.NaN + 133546
```

Out[343]:

```
nan
```

In [344]:

```
1 alcohol.prod()
```

Out[344]:

```
3.4276115052182805e+183
```

In [345]:

```
1 alcohol.cumprod()
```

Out[345]:

```
country
Afghanistan              NaN
Albania         5.400000e+01
Algeria         7.560000e+02
Andorra         2.358720e+05
Angola          1.061424e+07
                        ...
Venezuela      2.142257e+182
Vietnam        2.142257e+182
Yemen                    NaN
Zambia         8.569029e+182
Zimbabwe       3.427612e+183
Name: wine_servings, Length: 193, dtype: float64
```

In [346]:

```
1 alcohol.cumprod()[-1]
```

Out[346]:

```
3.4276115052182805e+183
```

In [347]:

```
1 alcohol.cumprod()[-1] == alcohol.prod()
```

Out[347]:

```
True
```

cummin cummax

In [348]:

```
1 alcohol.min()
```

Out[348]:

```
1.0
```

In [349]:

```
1 alcohol.cummin()
```

Out[349]:

```
country
Afghanistan     NaN
Albania        54.0
Algeria        14.0
Andorra        14.0
Angola         14.0
               ...
Venezuela       1.0
Vietnam         1.0
Yemen           NaN
Zambia          1.0
Zimbabwe        1.0
Name: wine_servings, Length: 193, dtype: float64
```

In [350]:

```
1 alcohol.max()
```

Out[350]:

```
339.0
```

In [351]:

```
1 alcohol.cummax()
```

Out[351]:

```
country
Afghanistan      NaN
Albania         54.0
Algeria         54.0
Andorra        312.0
Angola         312.0
               ...
Venezuela      339.0
Vietnam        339.0
Yemen            NaN
Zambia         339.0
Zimbabwe       339.0
Name: wine_servings, Length: 193, dtype: float64
```

Pairwise Differences With diff()

In [352]:

```
1 alcohol.head()
```

Out[352]:

```
country
Afghanistan      NaN
Albania         54.0
Algeria         14.0
Andorra        312.0
Angola          45.0
Name: wine_servings, dtype: float64
```

In [353]:

```
1 alcohol.diff().head()
```

Out[353]:

```
country
Afghanistan      NaN
Albania          NaN
Algeria        -40.0
Andorra        298.0
Angola        -267.0
Name: wine_servings, dtype: float64
```

In [354]:

```
1 alcohol.diff(periods=2).head()
```

Out[354]:

```
country
Afghanistan      NaN
Albania          NaN
Algeria          NaN
Andorra        258.0
Angola          31.0
Name: wine_servings, dtype: float64
```

Series Iteration

In [355]:

```
1 mini_alc = alcohol[:10]
```

In [356]:

```
1 mini_alc
```

Out[356]:

```
country
Afghanistan            NaN
Albania               54.0
Algeria               14.0
Andorra              312.0
Angola                45.0
Antigua & Barbuda     45.0
Argentina            221.0
Armenia               11.0
Australia            212.0
Austria              191.0
Name: wine_servings, dtype: float64
```

In [357]:

```
1 mini_alc.count()
```

Out[357]:

```
9
```

In [358]:

```
for i in mini_alc:
    print(i)
```

```
nan
54.0
14.0
312.0
45.0
45.0
221.0
11.0
212.0
191.0
```

In [359]:

```
for i in mini_alc.index:
    print(i)
```

```
Afghanistan
Albania
Algeria
Andorra
Angola
Antigua & Barbuda
Argentina
Armenia
Australia
Austria
```

In [360]:

```
for i in mini_alc.index:
    print(i, mini_alc[i])
```

```
Afghanistan nan
Albania 54.0
Algeria 14.0
Andorra 312.0
Angola 45.0
Antigua & Barbuda 45.0
Argentina 221.0
Armenia 11.0
Australia 212.0
Austria 191.0
```

In [361]:

```
mini_alc[i]
```

Out[361]:

```
191.0
```

In [362]:

```
for i in mini_alc.items():
    print(i)
```

```
('Afghanistan', nan)
('Albania', 54.0)
('Algeria', 14.0)
('Andorra', 312.0)
('Angola', 45.0)
('Antigua & Barbuda', 45.0)
('Argentina', 221.0)
('Armenia', 11.0)
('Australia', 212.0)
('Austria', 191.0)
```

Filtering:filter(),where(),And Mask()

In [363]:

```
alcohol.filter(regex='v')# this prints out all labels with the small "v" in them.
```

Out[363]:

```
country
Bolivia                 8.0
Bosnia-Herzegovina      8.0
Cote d'Ivoire           7.0
El Salvador             2.0
Latvia                 62.0
                      ...
Moldova                18.0
Slovakia              116.0
Slovenia              276.0
St. Kitts & Nevis      32.0
Tuvalu                  9.0
Name: wine_servings, Length: 11, dtype: float64
```

In [364]:

```
alcohol.filter(regex='V')# this prints out all labels with the capital "V"
```

Out[364]:

```
country
Cabo Verde                      16.0
St. Vincent & the Grenadines    11.0
Vanuatu                         11.0
Venezuela                        3.0
Vietnam                          1.0
Name: wine_servings, dtype: float64
```

In [365]:

```
alcohol.filter(regex='^V') # carat sign makes the v a capital one allowing the line of
```

Out[365]:

```
country
Vanuatu      11.0
Venezuela     3.0
Vietnam       1.0
Name: wine_servings, dtype: float64
```

In [366]:

```
alcohol.filter(like='stan')
```

Out[366]:

```
country
Afghanistan      NaN
Kazakhstan      12.0
Kyrgyzstan       6.0
Pakistan         NaN
Tajikistan       NaN
Turkmenistan    32.0
Uzbekistan       8.0
Name: wine_servings, dtype: float64
```

In [367]:

```
alcohol.filter(like='zue').dropna()
```

Out[367]:

```
country
Venezuela    3.0
Name: wine_servings, dtype: float64
```

In [368]:

```
alcohol['Venezuela']
```

Out[368]:

```
3.0
```

In [369]:

```
alcohol[alcohol == 3]
```

Out[369]:

```
country
Colombia     3.0
Djibouti     3.0
Ecuador      3.0
Venezuela    3.0
Name: wine_servings, dtype: float64
```

In [370]:

```
alcohol[alcohol>200].head(6)
```

Out[370]:

```
country
Andorra      312.0
Argentina    221.0
Australia    212.0
Belgium      212.0
Croatia      254.0
Denmark      278.0
Name: wine_servings, dtype: float64
```

In [371]:

```
#Using a function to get values greater than 200
def greater_than_200(x):
    return x>200
```

In [372]:

```
alcohol[greater_than_200]
```

Out[372]:

```
country
Andorra       312.0
Argentina     221.0
Australia     212.0
Belgium       212.0
Croatia       254.0
              ...
Greece        218.0
Italy         237.0
Luxembourg    271.0
Portugal      339.0
Slovenia      276.0
Name: wine_servings, Length: 12, dtype: float64
```

In [373]:

```
def less_than_200(x):
    return x<200
```

In [374]:

```
1 alcohol[less_than_200]
```

Out[374]:

```
country
Albania              54.0
Algeria              14.0
Angola               45.0
Antigua & Barbuda    45.0
Armenia              11.0
                     ...
Vanuatu              11.0
Venezuela             3.0
Vietnam               1.0
Zambia                4.0
Zimbabwe              4.0
Name: wine_servings, Length: 150, dtype: float64
```

In [375]:

```
1 alcohol[less_than_200].count()
```

Out[375]:

```
150
```

In [376]:

```
1 alcohol[alcohol == less_than_200]
```

Out[376]:

```
Series([], Name: wine_servings, dtype: float64)
```

In [377]:

```
1 #the where() method...Replaces values where condition is False
```

In [378]:

```
1 alcohol.where(lambda y: y>200, other='less than 200').head(5)
```

Out[378]:

```
country
Afghanistan    less than 200
Albania        less than 200
Algeria        less than 200
Andorra                  312
Angola         less than 200
Name: wine_servings, dtype: object
```

In [379]:

```
alcohol.where(lambda z : z<200).dropna().head()
```

Out[379]:

```
country
Albania              54.0
Algeria              14.0
Angola               45.0
Antigua & Barbuda    45.0
Armenia              11.0
Name: wine_servings, dtype: float64
```

In [380]:

```
len(alcohol.where(lambda z : z<200).dropna())
```

Out[380]:

```
150
```

In [381]:

```
alcohol.where(lambda p: p < 200).count()
```

Out[381]:

```
150
```

In [382]:

```
#the mask() method.... Replaces values where  the condition is True
alcohol.mask(lambda q : q<150).dropna()
```

Out[382]:

```
country
Andorra           312.0
Argentina         221.0
Australia         212.0
Austria           191.0
Belgium           212.0
                  ...
Portugal          339.0
Romania           167.0
Slovenia          276.0
Sweden            186.0
United Kingdom    195.0
Name: wine_servings, Length: 21, dtype: float64
```

In [383]:

```
alcohol.mask(lambda w : w <150).count()
```

Out[383]:

```
21
```

In [384]:

```
alcohol.where(lambda t : t < 150).count()
```

Out[384]:

141

In [385]:

```
alcohol.mask(lambda t : t > 150).count()
```

Out[385]:

141

In [386]:

```
alcohol.where(lambda t : t > 150).count()
```

Out[386]:

21

In [387]:

```
alcohol.mask(lambda t : t > 150).count() == (alcohol.where(lambda t : t < 150).count())
```

Out[387]:

True

Transforming with update(), apply(), map()

In [388]:

```
#spot vs Global transformation
```

In [389]:

```
#spot
```

In [390]:

```
alcohol['Algeria']
```

Out[390]:

14.0

In [391]:

```
alcohol.loc['Algeria'] = 18
```

In [392]:

```
alcohol.head()
```

Out[392]:

```
country
Afghanistan      NaN
Albania         54.0
Algeria         18.0
Andorra        312.0
Angola          45.0
Name: wine_servings, dtype: float64
```

In [393]:

```
#the update() method
alcohol.update(pd.Series(data=[200,50], index=['Albania', 'Algeria']))
```

In [394]:

```
alcohol.head()
```

Out[394]:

```
country
Afghanistan      NaN
Albania        200.0
Algeria         50.0
Andorra        312.0
Angola          45.0
Name: wine_servings, dtype: float64
```

In [395]:

```
#Global Transform
alcohol.apply(lambda b : b **2) # the apply method affects the entire series
```

Out[395]:

```
country
Afghanistan        NaN
Albania        40000.0
Algeria         2500.0
Andorra        97344.0
Angola          2025.0
                  ...
Venezuela          9.0
Vietnam            1.0
Yemen              NaN
Zambia           16.0
Zimbabwe         16.0
Name: wine_servings, Length: 193, dtype: float64
```

In [396]:

```
def multiply_by_self(x):
    return x*x
```

In [397]:

```
alcohol.apply(multiply_by_self)
```

Out[397]:

```
country
Afghanistan        NaN
Albania        40000.0
Algeria         2500.0
Andorra        97344.0
Angola          2025.0
                ...
Venezuela          9.0
Vietnam            1.0
Yemen              NaN
Zambia            16.0
Zimbabwe          16.0
Name: wine_servings, Length: 193, dtype: float64
```

In [398]:

```
def multiply_by_self_with_min(x, min_servings):
    if x < min_servings:
        return x ** 2
    return x
```

In [399]:

```
alcohol.apply(multiply_by_self_with_min, args=(200,))
```

Out[399]:

```
country
Afghanistan       NaN
Albania         200.0
Algeria        2500.0
Andorra         312.0
Angola         2025.0
                ...
Venezuela         9.0
Vietnam           1.0
Yemen             NaN
Zambia           16.0
Zimbabwe         16.0
Name: wine_servings, Length: 193, dtype: float64
```

In [400]:

```
alcohol.apply(multiply_by_self_with_min, min_servings=200)
```

Out[400]:

```
country
Afghanistan       NaN
Albania         200.0
Algeria        2500.0
Andorra         312.0
Angola         2025.0
                ...
Venezuela         9.0
Vietnam           1.0
Yemen             NaN
Zambia           16.0
Zimbabwe         16.0
Name: wine_servings, Length: 193, dtype: float64
```

In [401]:

```
#map() method can also be used except when used in a defined and not anonymous functior
alcohol.map(lambda c : c ** 2)
```

Out[401]:

```
country
Afghanistan        NaN
Albania        40000.0
Algeria         2500.0
Andorra        97344.0
Angola          2025.0
                ...
Venezuela          9.0
Vietnam            1.0
Yemen              NaN
Zambia            16.0
Zimbabwe          16.0
Name: wine_servings, Length: 193, dtype: float64
```

In [402]:

```
alcohol.map(multiply_by_self_with_min, min_servings=200) #this should give a TypeError
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-402-9f4372cf29cd> in <module>
----> 1 alcohol.map(multiply_by_self_with_min, min_servings=200) #this shoul
d give a TypeError

TypeError: map() got an unexpected keyword argument 'min_servings'
```

In [404]:

```
beer = pd.read_csv("drinks.csv", usecols=['country', 'beer_servings'], index_col='count
```

In [405]:

```
1 beer.head()
```

Out[405]:

```
country
Afghanistan      NaN
Albania         89.0
Algeria         25.0
Andorra        245.0
Angola         217.0
Name: beer_servings, dtype: float64
```

Calculate the mean, median and standard deviation of beer servings in beers. Is the distribution right or left skewed?.

In [406]:

```
1 beer.mean()
```

Out[406]:

```
102.87078651685393
```

In [407]:

```
1 avg = beer.sum()/beer.count()
```

In [408]:

```
1 avg
```

Out[408]:

```
102.87078651685393
```

In [409]:

```
1 beer.count()
```

Out[409]:

```
178
```

In [410]:

```
1 beer.isna().sum()
```

Out[410]:

```
15
```

In [411]:

```
1 beer.notnull().sum()
```

Out[411]:

```
178
```

In [412]:

```
1 beer.isnull()
```

Out[412]:

```
country
Afghanistan     True
Albania        False
Algeria        False
Andorra        False
Angola         False
               ...
Venezuela      False
Vietnam        False
Yemen          False
Zambia         False
Zimbabwe       False
Name: beer_servings, Length: 193, dtype: bool
```

In [413]:

```
1 beer[beer.notnull()]
```

Out[413]:

```
country
Albania               89.0
Algeria               25.0
Andorra              245.0
Angola               217.0
Antigua & Barbuda     12.0
                     ...
Venezuela            333.0
Vietnam              111.0
Yemen                  6.0
Zambia                32.0
Zimbabwe              64.0
Name: beer_servings, Length: 178, dtype: float64
```

In [414]:

```
1 beer.describe()
```

Out[414]:

```
count    178.000000
mean     102.870787
std      100.645713
min        1.000000
25%       21.000000
50%       60.000000
75%      172.500000
max      376.000000
Name: beer_servings, dtype: float64
```

In [415]:

```
1 beer.median()
```

Out[415]:

```
60.0
```

In [416]:

```
1 beer.quantile(.5)
```

Out[416]:

```
60.0
```

In [417]:

```
1 beer.std()
```

Out[417]:

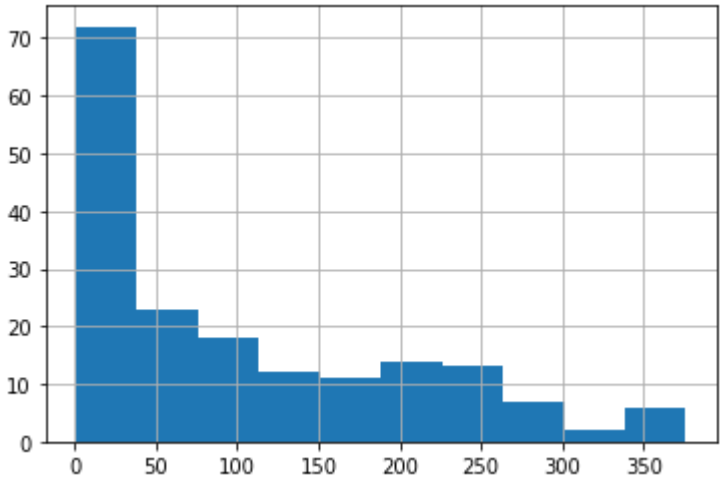
```
100.64571267934778
```

In [418]:

```
1 beer.hist() # This plot is right-skewed(positive skewness)
```

Out[418]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1ac0b6d0108>
```



Z-scores

In [419]:

```
1 beer[:10]
```

Out[419]:

```
country
Afghanistan             NaN
Albania                89.0
Algeria                25.0
Andorra               245.0
Angola                217.0
Antigua & Barbuda      12.0
Argentina             193.0
Armenia                21.0
Australia             261.0
Austria               279.0
Name: beer_servings, dtype: float64
```

In [420]:

```
1 beer[:10] - beer.mean()
```

Out[420]:

```
country
Afghanistan                NaN
Albania                -13.870787
Algeria                -77.870787
Andorra                142.129213
Angola                 114.129213
Antigua & Barbuda      -90.870787
Argentina               90.129213
Armenia                -81.870787
Australia              158.129213
Austria                176.129213
Name: beer_servings, dtype: float64
```

In [421]:

```
1 (beer[:10] - beer.mean()).apply(lambda x : 'low' if x<0 else 'high')
```

Out[421]:

```
country
Afghanistan           high
Albania                low
Algeria                low
Andorra               high
Angola                high
Antigua & Barbuda      low
Argentina             high
Armenia                low
Australia             high
Austria               high
Name: beer_servings, dtype: object
```

In [422]:

```
(beer - beer.mean()).apply(lambda x : 'low' if x < 0 else 'high').value_counts()
```

Out[422]:

```
low     112
high     81
Name: beer_servings, dtype: int64
```

In [423]:

```
z_scores = (beer - beer.mean())/beer.std()
```

In [424]:

```
z_scores
```

Out[424]:

```
country
Afghanistan         NaN
Albania       -0.137818
Algeria       -0.773712
Andorra        1.412174
Angola         1.133970
                  ...
Venezuela      2.286528
Vietnam        0.080771
Yemen         -0.962493
Zambia        -0.704161
Zimbabwe      -0.386214
Name: beer_servings, Length: 193, dtype: float64
```

In [425]:

```
z_scores.min()
```

Out[425]:

```
-1.012172141315241
```

In [426]:

```
z_scores.max()
```

Out[426]:

```
2.7137689844109123
```

In [427]:

```
z_scores.abs().max()
```

Out[427]:

```
2.7137689844109123
```

In [428]:

```
1 beer.idxmax()
```

Out[428]:

```
'Namibia'
```

In [429]:

```
1 beer['Namibia']
```

Out[429]:

```
376.0
```

In [430]:

```
1 beer[beer.idxmax()]
```

Out[430]:

```
376.0
```

In [431]:

```
1 z_scores.abs().max() * beer.std() + beer.mean()
```

Out[431]:

```
376.0
```

In [ ]:

```
1
```