basics_python_4_data

April 21, 2020

1 Basics of python

1.1 Datatypes

[3]:
```
a = "a" # declared variable, string a denoted with "" ''
```

[4]:
```
print(a)
```

```
a
```

[8]:
```
a = 1 + 2 * 4
```

[9]:
```
a
```

[9]: 9

[10]:
```
1.2 / 1.2
```

[10]: 1.0

[11]:
```
1 ** 2
```

[11]: 1

[12]:
```
2 ** 2
```

[12]: 4

1.2 Iterables

list

[22]:
```
mylist = ['a', 'b', 1, 1.2, "blub"] # list with square brackets and commas,␣
↪mixed datatypes allowed
```

[16]:
```
mylist
```

[16]: ['a', 'b', 1, 1.2, None]

[17]:
```
mylist[1]
```

[17]:
```
'b'
```

Python is zero indexed!

[18]:
```
mylist[0]
```

[18]:
```
'a'
```

[19]:
```
len(mylist)
```

[19]:
```
5
```

[24]:
```
mylist[-1]
```

[24]:
```
'blub'
```

[25]:
```
mylist.append('a')
```

[26]:
```
mylist
```

[26]:
```
['a', 'b', 1, 1.2, 'blub', 'a']
```

[34]:
```
counter = 0
for i in mylist:
    counter +=1 # counter = counter +1
    print(counter, counter**2, i, sep='|')
```

```
1|1|a
2|4|b
3|9|1
4|16|1.2
5|25|blub
6|36|a
```

1.3 Dictionary

key -> value

first_car -> green second_car -> black

[35]:
```
d = {
    'first_car': 'green',
    'second_car': 'black'
    }
```

[37]:
```
d['first_car']
```

[37]:
```
'green'
```

[38]:
```
car_color_dict = {
    'first_car': 'green',
    'second_car': 'black'
    }
```

[40]:
```
car_color_dict['second_car']
```

[40]:
```
'black'
```

1.4 Datascience with python

[41]:
```
import numpy as np  # matrix math
import pandas as pd # "excel like data"
```

[45]:
```
a = np.array([1,2, 3]) # one datatype
```

[46]:
```
b = np.array([2, 2, 2]) # one datatype
```

[47]:
```
a + b
```

[47]:
```
array([3, 4, 5])
```

[48]:
```
a * b
```

[48]:
```
array([2, 4, 6])
```

[49]:
```
a / b
```

[49]:
```
array([0.5, 1. , 1.5])
```

[50]:
```
np.sin(a)
```

[50]:
```
array([0.84147098, 0.90929743, 0.14112001])
```

[54]:
```
c = np.array([[1,2,3], [1,2,3]])
```

[55]:
```
c.shape
```

[55]:
```
(2, 3)
```

[56]:
```
c
```

[56]:
```
array([[1, 2, 3],
       [1, 2, 3]])
```

[57]:
```
a.shape
```

[57]:
```
(3,)
```

[58]:
```
a + c
```

[58]:
```
array([[2, 4, 6],
       [2, 4, 6]])
```

1.4.1 pandas gives dataframes

[60]:
```
df = pd.DataFrame(
        [
         {'name': 'Kevin',
        'email': 'kevin.jablonka@epfl.ch'
         } ,
        {'name': 'Berend',
        'email': 'berend.smit@epfl.ch'
         }
        ]
)
```

[67]:
```
df
```

[67]:

	name	email
0	Kevin	kevin.jablonka@epfl.ch
1	Berend	berend.smit@epfl.ch

[70]:
```
len(df)
```

[70]:
```
2
```

[71]:
```
df.describe()
```

[71]:

	name	email
count	2	2
unique	2	2
top	Kevin	kevin.jablonka@epfl.ch
freq	1	1

[72]:
```
df = pd.read_csv('/Users/kevinmaikjablonka/Dropbox (LSMO)/proj66_18e_rule/
↪20200417-235858_featurized_merged_df.csv')
```

[76]:
```
df['new_feature'] = df['sgl_bd'] + df['tri_bipyr']
```

[80]:
```
int(1.2)
```

[80]:
```
1
```

[81]:
```
float(1.2)
```

[81]:
```
1.2
```

[85]:
```
1 == 1
```

[85]:
```
True
```

[89]:
```
df
```

[89]:

	sgl_bd	bent	T	see_saw_rect	tet	oct	\
0	4.022136e-03	0.092255	0.345547	0.229039	0.019030	0.074364	
1	2.136424e-01	0.096205	0.368450	0.235727	0.018147	0.072852	
2	1.132352e-02	0.094367	0.364063	0.248527	0.019591	0.077272	
3	3.403701e-03	0.094124	0.358381	0.264267	0.019050	0.077548	
4	2.421093e-03	0.093739	0.358767	0.255406	0.019325	0.081393	
…	…	…	…	…	…	…	
67745	1.554312e-15	0.200379	0.347704	0.185002	0.101527	0.506645	
67746	2.220446e-16	0.200120	0.493788	0.771903	0.036373	0.854225	
67747	1.110223e-16	0.200164	0.369578	0.473603	0.031159	0.753196	
67748	2.553513e-15	0.200135	0.504980	0.767596	0.037106	0.829385	
67749	3.330669e-16	0.200151	0.416668	0.532722	0.031306	0.791856	

	tri_plan	sq_plan	pent_plan	hex_bipyr	…	based_on_smiles	CN	\
0	0.034016	0.092733	0.047836	0.316051	…	False	10	
1	0.035168	0.090308	0.045841	0.327994	…	True	11	
2	0.035736	0.095921	0.044025	0.376013	…	True	11	
3	0.035305	0.096320	0.045428	0.373334	…	True	11	
4	0.035736	0.100615	0.044202	0.367071	…	True	11	
…	…	…	…	…	…	…	..	
67745	0.003065	0.242988	0.011906	0.556566	…	False	6	
67746	0.001838	0.343532	0.009952	0.553505	…	False	6	
67747	0.001077	0.315353	0.005358	0.507564	…	False	6	
67748	0.001792	0.336072	0.009551	0.569624	…	False	6	
67749	0.001700	0.318466	0.008428	0.478841	…	False	6	

	number_ligands	homoleptic	ligands_clean	\
0	2	True	('Cp', 'Cp')	
1	3	False	('F', 'Cp', 'Cp')	
2	3	False	('Cl', 'Cp', 'Cp')	
3	3	False	('Br', 'Cp', 'Cp')	
4	3	False	('I', 'Cp', 'Cp')	
…	…	…	…	
67745	6	False	('NR3', 'NR3', 'SR2', 'SR2', 'SR2', 'SR2')	
67746	6	True	('SR2', 'SR2', 'SR2', 'SR2', 'SR2', 'SR2')	

```
67747               6        True  ('SR2', 'SR2', 'SR2', 'SR2', 'SR2', 'SR2')
67748               6        True  ('SR2', 'SR2', 'SR2', 'SR2', 'SR2', 'SR2')
67749               6        True  ('SR2', 'SR2', 'SR2', 'SR2', 'SR2', 'SR2')

       electroncount_ionic  electroncount_neutral  \
0                     12.0                     12
1                      NaN                     14
2                      NaN                     14
3                      NaN                     14
4                      NaN                     14
…                      …                       …
67745                 22.0                     22
67746                 22.0                     22
67747                 22.0                     22
67748                 22.0                     22
67749                 22.0                     22

       formal_charges_metal_centers  electron_counts_ionic_from_formal_charge  \
0                                 3                                        12
1                                 3                                        14
2                                 3                                        14
3                                 3                                        14
4                                 3                                        14
…                                 …                                        …
67745                             0                                        24
67746                             2                                        22
67747                             2                                        22
67748                             2                                        22
67749                             2                                        22

       new_feature
0         0.231137
1         0.423926
2         0.266155
3         0.259847
4         0.268151
…            …
67745     0.285254
67746     0.447939
67747     0.386436
67748     0.439859
67749     0.399146

[67750 rows x 66 columns]
```

[116]:
```
df.describe()
```

[116]:

	sgl_bd	bent	T	see_saw_rect	tet	\
count	6.775000e+04	67750.000000	6.775000e+04	67750.000000	6.663800e+04	
mean	8.230312e-03	0.187261	3.991934e-01	0.709934	6.176000e-02	
std	2.840706e-02	0.123226	1.840283e-01	0.389851	1.813803e-01	
min	0.000000e+00	0.000095	9.069719e-14	0.000000	2.519396e-47	
25%	6.661338e-16	0.140345	3.354978e-01	0.374318	1.453297e-02	
50%	3.931390e-04	0.198908	4.190666e-01	0.955060	2.396363e-02	
75%	5.062573e-03	0.200074	5.002765e-01	0.983490	2.555994e-02	
max	5.207001e-01	1.000000	1.000000e+00	1.000000	1.000000e+00	

	oct	tri_plan	sq_plan	pent_plan	hex_bipyr	\
count	67743.000000	6.663800e+04	67743.000000	66638.000000	67750.000000	
mean	0.682264	1.774031e-02	0.327322	0.018350	0.478373	
std	0.393073	5.687895e-02	0.231265	0.019986	0.216983	
min	0.000188	2.983762e-14	0.000157	0.000001	0.000000	
25%	0.150791	1.995151e-03	0.124281	0.009718	0.439798	
50%	0.938318	2.092845e-03	0.369719	0.012636	0.507540	
75%	0.979954	1.907063e-02	0.380385	0.025201	0.527425	
max	1.000000	1.000000e+00	1.000000	0.921674	1.000000	

	…	molecule_formal_charge	volume_y	oxidationstate_clean	\
count	…	67750.000000	63418.000000	52066.000000	
mean	…	1.019793	224.205760	1.974398	
std	…	1.267220	173.472214	0.635589	
min	…	-8.000000	51.157789	-4.000000	
25%	…	0.000000	112.656762	2.000000	
50%	…	2.000000	118.415462	2.000000	
75%	…	2.000000	310.502123	2.000000	
max	…	6.000000	1897.442320	6.000000	

	CN	number_ligands	electroncount_ionic	\
count	67750.000000	67750.000000	52066.000000	
mean	6.352369	4.915557	18.743652	
std	2.009347	1.520042	2.188327	
min	2.000000	2.000000	6.000000	
25%	6.000000	4.000000	17.000000	
50%	6.000000	6.000000	19.000000	
75%	6.000000	6.000000	20.000000	
max	20.000000	6.000000	28.000000	

	electroncount_neutral	formal_charges_metal_centers	\
count	67750.000000	67750.000000	
mean	18.384546	1.726273	
std	2.128086	1.202532	
min	6.000000	-8.000000	
25%	17.000000	2.000000	
50%	18.000000	2.000000	

```
75%                 20.000000                       2.000000
max                 28.000000                       8.000000

       electron_counts_ionic_from_formal_charge   new_feature
count                              67750.000000  67750.000000
mean                                  18.698111      0.435522
std                                    2.412398      0.188874
min                                    6.000000      0.000000
25%                                   17.000000      0.441341
50%                                   18.000000      0.497549
75%                                   20.000000      0.518747
max                                   28.000000      1.288540

[8 rows x 42 columns]
```

1.4.2 boolean indexing

& and operator | or operator

[117]:
```
df[(df['bent'] > 0.5) | (df['oct'] < 0.5)]['refcode']
```

[117]:
```
0        SAXTOB
1        OGANIW
2        OGANOC
3        OGANES
4        OGANAO
          ...
67740    XEKZIX
67741    WONKOZ
67742    PODWEK
67743    TORNUK
67744    YIVJES
Name: refcode, Length: 21283, dtype: object
```

[121]:
```
df['bent'].mean()
```

[121]:
```
0.18726050251841003
```

1.5 if statements

[96]:
```
mylist = [1,2,3,4,5,6,7]
```

[104]:
```
counter = 0
for listitem in mylist:
    counter +=1
    if listitem > 4:
        print(counter, listitem, sep='|')
```

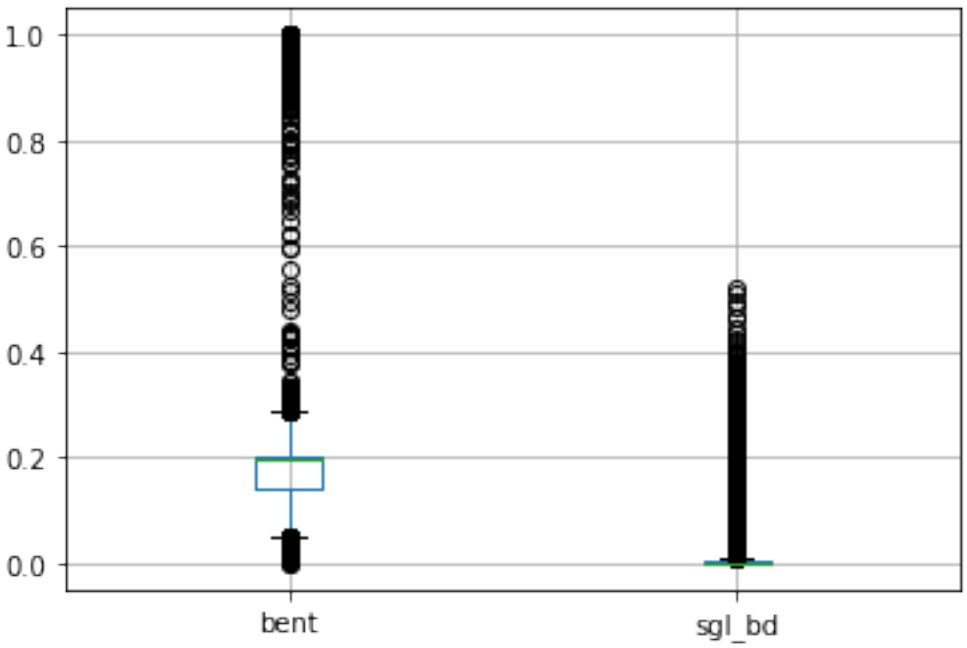
```
    elif listitem <= 2:
        print(counter, 'smaller or equal than 2')
    else:
        print(counter, 'not greater than 4')
```

```
1 smaller or equal than 2
2 smaller or equal than 2
3 not greater than 4
4 not greater than 4
5|5
6|6
7|7
```

1.5.1 plotting

[106]:
```
pd.plotting.boxplot(df, ['bent', 'sgl_bd'])
```

[106]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x12246c390>
```

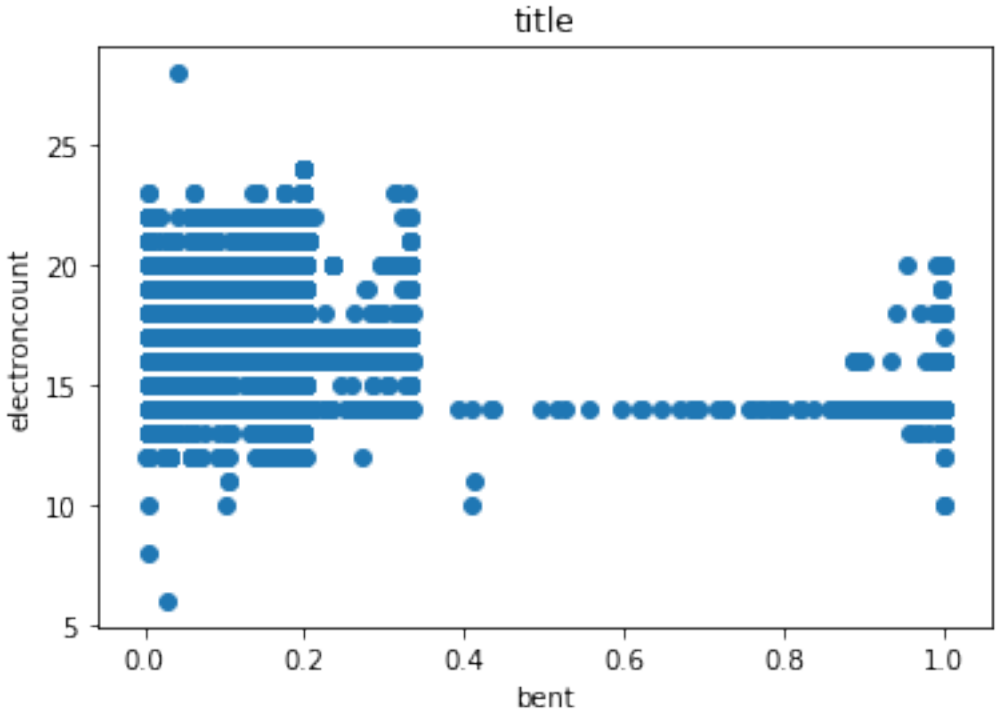


[108]:
```
import matplotlib.pyplot as plt
%matplotlib inline # magic comment
```

```
UsageError: unrecognized arguments: # magic comment
```

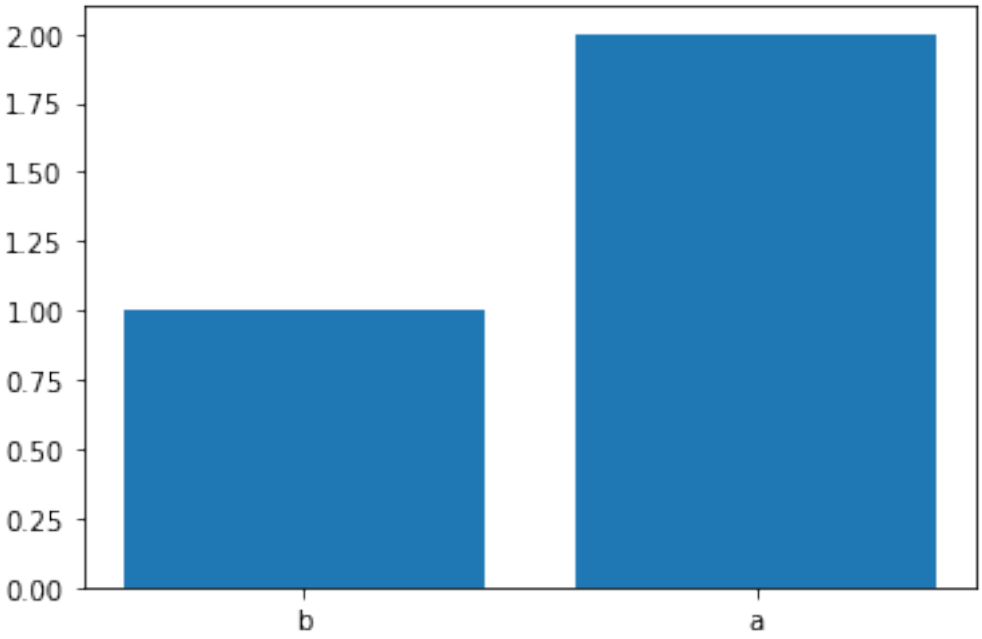
[118]:
```
plt.scatter(df['bent'], df['electroncount_ionic'])
plt.xlabel('bent')
```

```
plt.ylabel('electroncount')
plt.title('title')
plt.show()
```



[114]:
```
plt.bar(['b', 'a'], [1,2])
```

[114]: <BarContainer object of 2 artists>


2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
b
a

[ ]: