Python Commands

March 16, 2021

[1]:
```
# comments start with a "#"
"""
A comment block is 3 double quotes in a row and
ends with three double quotes

"""
test = ['a','b','c']
print(test[0])
print(test[1])
print(test[2])
print(len(test))

#the type of quote is not too important in python
# both single and double quotes can be used
# best practice is to pick a style and stay consistent
test = ["a",'b','c']
print(test[0])
print(test[1])
print(test[2])
print(len(test))
```

```
a
b
c
3
a
b
c
3
```

[2]:
```
# intervals
print("range(0,5)")
for i in range(0,5):
    print(i)

print("------------")
print("range(5,7)")
for i in range(5,7):
```

```
    print(i)

print("------------")
print("test[0:2]")
print(test[0:2])
```

```
range(0,5)
0
1
2
3
4
------------
range(5,7)
5
6
------------
test[0:2]
['a', 'b']
```

[3]:
```
# this code will fail because the "if" statement
# has an extra space in front of it
for i in range(5):
    print(i)
    print(i*i)
     if i == 4:
        print("Hi")
```

```
  File "<ipython-input-3-57fbe8158572>", line 6
    if i == 4:
    ^
IndentationError: unexpected indent
```

[4]:
```
# properly aligned with tabs

for i in range(5):
    print(i)
    print(i*i)
    if i == 4:
        print("Hi")
```

```
0
0
1
1
2
```

```
4
3
9
4
16
Hi
```

[5]:
```python
# properly aligned with 2 spaces
for i in range(5):
  print(i)
  print(i*i)
  if i == 4:
    print("Hi")
```

```
0
0
1
1
2
4
3
9
4
16
Hi
```

[6]:
```python
# create a list of numbers
list_item = [i for i in range(10)]

stuff = []
for i in list_item:
    if i > 5:
        stuff.append(i+1)

stuff2 = [i+1 for i in list_item if i> 5]

# check if they are equivalent
stuff == stuff2
```

[6]: True

[7]:
```python
# dictionary example
# {key:value for item in list if condition}
stuff3 = {str(i+1):i for i in list_item if i > 5}
```

[8]:
```python
import pandas as pd
```

```
ts =␣
 ↪['2011-06-20','2011-06-23','2011-06-27','2011-09-01','2011-09-05','2011-09-06']
f = [43,34,43,89,77,67]
l = [27.0, 19.0, 29.0, 50.0, 77.0, 46.0]
r = [4.0,3.0,2.0,9.0,4.0,14.0]
data = pd.DataFrame({"timestamp":ts,"full_sq":f,"life_sq":l,"floor":r})
```

[9]:
```
#display dataframe
data
```

[9]:

	timestamp	full_sq	life_sq	floor
0	2011-06-20	43	27.0	4.0
1	2011-06-23	34	19.0	3.0
2	2011-06-27	43	29.0	2.0
3	2011-09-01	89	50.0	9.0
4	2011-09-05	77	77.0	4.0
5	2011-09-06	67	46.0	14.0

[10]:
```
# indexed rows 0 to 3 and columns timestamp, full_sq
data.loc[0:3,["timestamp","full_sq"]]
```

[10]:

	timestamp	full_sq
0	2011-06-20	43
1	2011-06-23	34
2	2011-06-27	43
3	2011-09-01	89

[11]:
```
#positional location--rows 0:3 and columns 0:2
#note the open intervals at the end
data.iloc[0:3,0:2]
```

[11]:

	timestamp	full_sq
0	2011-06-20	43
1	2011-06-23	34
2	2011-06-27	43

[12]:
```
data[data['full_sq']>43]
```

[12]:

	timestamp	full_sq	life_sq	floor
3	2011-09-01	89	50.0	9.0
4	2011-09-05	77	77.0	4.0
5	2011-09-06	67	46.0	14.0

[13]:
```
data.loc[data['full_sq']>43, ['timestamp','floor']]
```

[13]:

	timestamp	floor
3	2011-09-01	9.0

```
4  2011-09-05    4.0
5  2011-09-06   14.0
```

[14]:
```
data['new'] = data['full_sq']/data['life_sq']
data
```

[14]:
```
    timestamp  full_sq  life_sq  floor       new
0  2011-06-20       43     27.0    4.0  1.592593
1  2011-06-23       34     19.0    3.0  1.789474
2  2011-06-27       43     29.0    2.0  1.482759
3  2011-09-01       89     50.0    9.0  1.780000
4  2011-09-05       77     77.0    4.0  1.000000
5  2011-09-06       67     46.0   14.0  1.456522
```

[15]:
```
data['full_sq'].sum()
```

[15]:
```
353
```

[16]:
```
data['full_sq'].isnull()
```

[16]:
```
0    False
1    False
2    False
3    False
4    False
5    False
Name: full_sq, dtype: bool
```

[17]:
```
data[data['timestamp'].str.contains('06')]
```

[17]:
```
    timestamp  full_sq  life_sq  floor       new
0  2011-06-20       43     27.0    4.0  1.592593
1  2011-06-23       34     19.0    3.0  1.789474
2  2011-06-27       43     29.0    2.0  1.482759
5  2011-09-06       67     46.0   14.0  1.456522
```

[18]:
```
data.shape
```

[18]:
```
(6, 5)
```

[19]:
```
data.describe()
```

[19]:
```
          full_sq    life_sq      floor       new
count    6.000000   6.000000   6.000000  6.000000
mean    58.833333  41.333333   6.000000  1.516891
std     22.021959  21.096603   4.604346  0.290256
min     34.000000  19.000000   2.000000  1.000000
```

```
25%    43.000000  27.500000   3.250000  1.463081
50%    55.000000  37.500000   4.000000  1.537676
75%    74.500000  49.000000   7.750000  1.733148
max    89.000000  77.000000  14.000000  1.789474
```

[20]:
```
data.columns
```

[20]:
```
Index(['timestamp', 'full_sq', 'life_sq', 'floor', 'new'], dtype='object')
```

[21]:
```
data.index
```

[21]:
```
RangeIndex(start=0, stop=6, step=1)
```

[22]:
```
data.dtypes
```

[22]:
```
timestamp     object
full_sq        int64
life_sq      float64
floor        float64
new          float64
dtype: object
```

[ ]:
```

```

[ ]:
```

```