# Python Data Structure and Pandas (DataFrame)

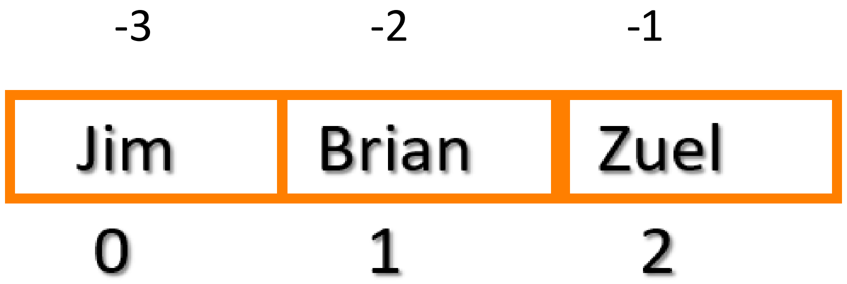
Johnny Zhang

# List (Array)

- A List is a kind of collection (data structure):
  - A collection allows us to put many values in a single “variable”.
  - A collection is nice because we can carry all multiple values around in one convenient package.
  - my_Friends=[‘Jim’, ‘Brian’, ‘Anu’, ‘Alex’]
  - carryon =[‘socks’, ‘shirt’, ‘perfume’]
  - Arrays in Python are also call lists.
  - Arrays store multiple values of the same data type

# Looking Inside Lists

- Elements of the array are referenced sequentially with an index.
- Just like strings, we can get at any single element in a list using an index specified in square brackets



| -3 | -2 | -1 |
|---|---|---|
| Jim | Brian | Zuel |
| 0 | 1 | 2 |

```
>>> my_Friends = [ Jim', ‘Brian', ‘Zuel' ]
>>> print (my_Friends[1])
>>> Brian
```

# Array Methods

- Append: List.append(elem)
- Insert: List.insert(index,elem) Note: 2 arguments
- Extend: list.extend(list2) Note: list.extend(“lucy”) vs. list.extend([“lucy”])
- Index: list.index(elem)
- Remove: list.remove(elm)
- Pop: list.pop() Default: last one or based on index
- Sort: list.sort()
- Reverse: list.reverse()

```
>>> list1.extend('lucy')
>>> list1
['Hello', 'the world', 'l', 'u', 'c', 'y']
>>> list1.extend(['lucy'])
>>> list1
['Hello', 'the world', 'l', 'u', 'c', 'y', 'lucy']
```

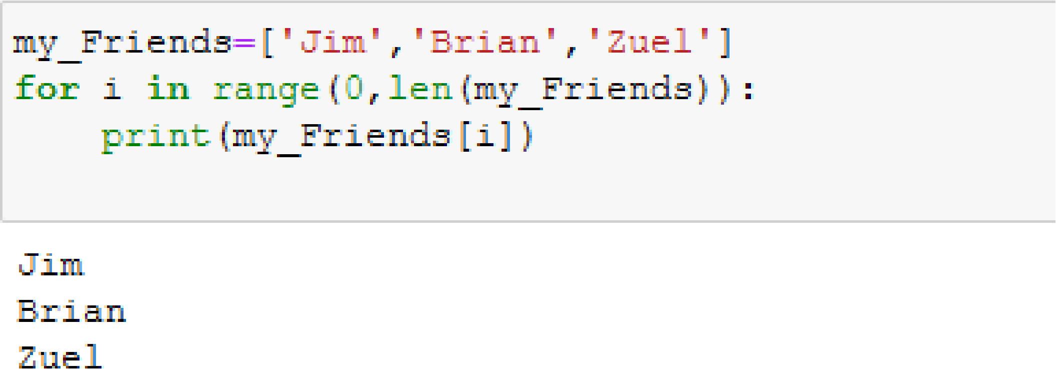
# Lists are Mutable

- Strings are "immutable" - we *cannot* change the contents of a string - we must make a new string to make any change
- Lists are "mutable" - we *can* change an element of a list using the index operator

```
>>> fruit="Apple“
>>> fruit[0]
'A'
>>> fruit[0]='B'
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    fruit[0]='B'
TypeError: 'str' object does not support item assignment
>>> new_list=[2,34,56,23,11]
>>> new_list
[2, 34, 56, 23, 11]
>>> new_list[3]=99
>>> new_list
[2, 34, 56, 99, 11]
```

# How long is a List

- The len() function takes a list as a parameter and returns the number of *elements* in the list
- Actually len() tells us the number of elements of *any* set or sequence



```
my_Friends=['Jim','Brian','Zuel']
for i in range(0,len(my_Friends)):
    print(my_Friends[i])
```

```
Jim
Brian
Zuel
```

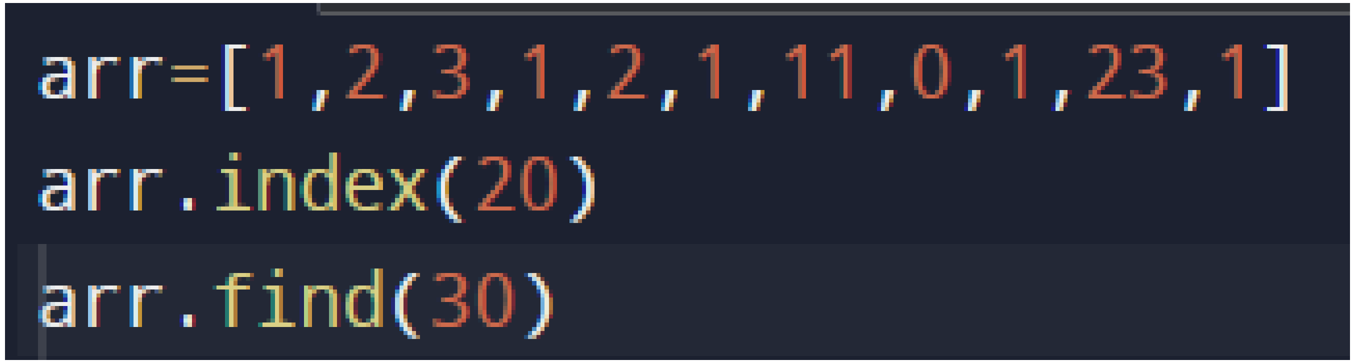
```
>>> greet = 'Hello Bob’
>>> print (len(greet))
9
>>> x = [ 1, 2, 'joe', 99]
>>> print (len(x))
4
```

# Adding elements (append() vs. extend())

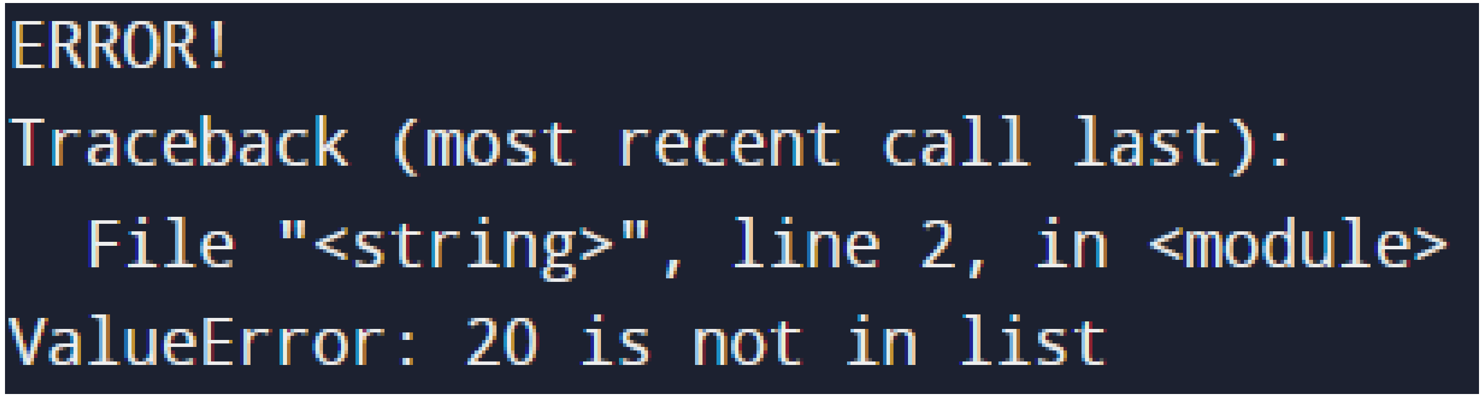
- We can create an empty list and then add elements using the append method
- The list stays in order and new elements are added at the end of the list

```
>>> stuff = list()
>>> stuff.append('book')
>>> stuff.append(99)
>>> print （stuff）
['book', 99]
>>> print (stuff)
['book', 99]
>>> stuff.append(['cookie','22'])
>>> print(stuff)
['book', 99, ['cookie', '22’]]
>>> stuff.extend(['cookie','22'])
>>> stuff
['book', 99, ['cookie', '22'], 'cookie', '22']
```

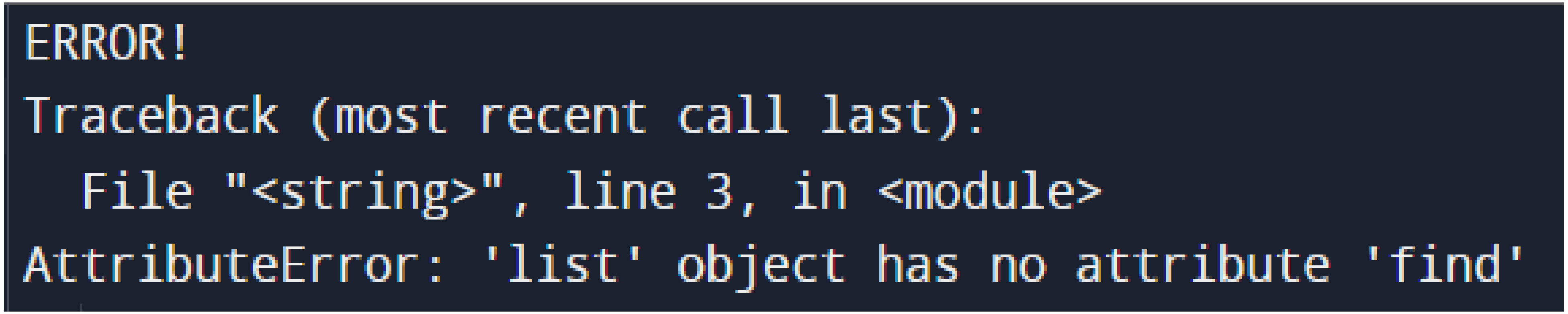
## index() vs. find()



```
arr=[1,2,3,1,2,1,11,0,1,23,1]
arr.index(20)
arr.find(30)
```



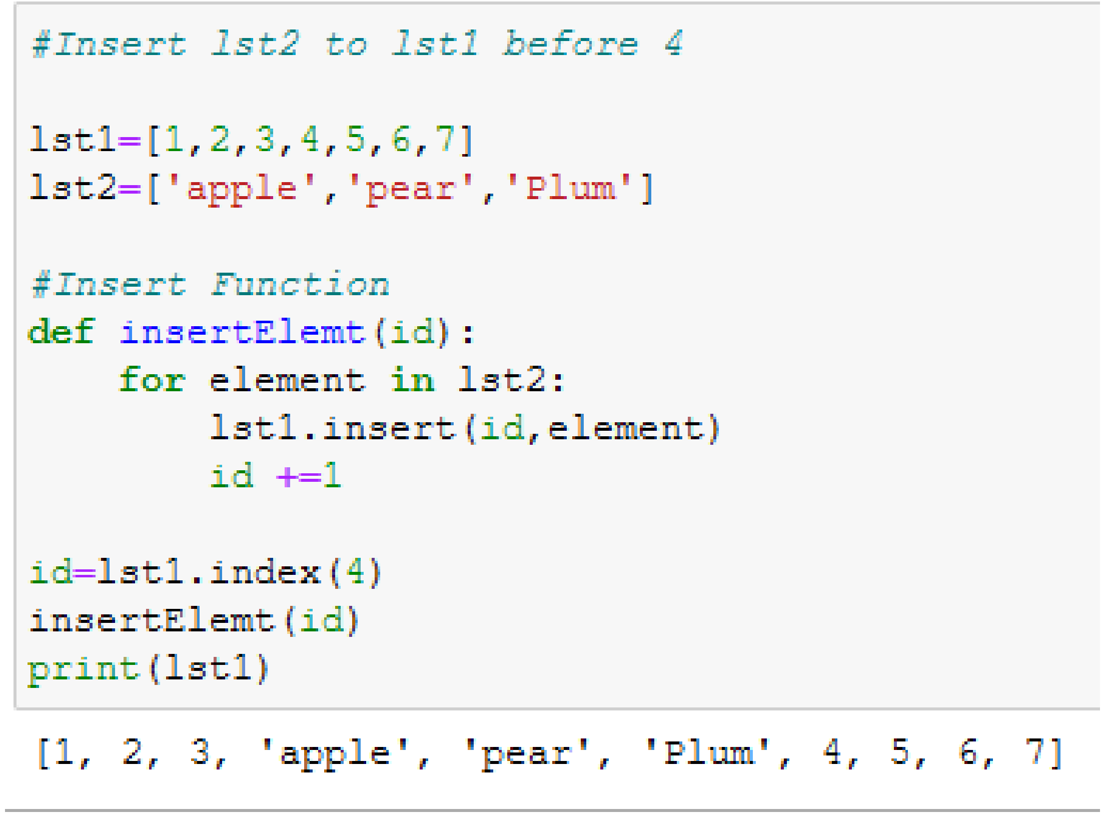
```
ERROR!
Traceback (most recent call last):
  File "<string>", line 2, in <module>
ValueError: 20 is not in list
```



```
ERROR!
Traceback (most recent call last):
  File "<string>", line 3, in <module>
AttributeError: 'list' object has no attribute 'find'
```

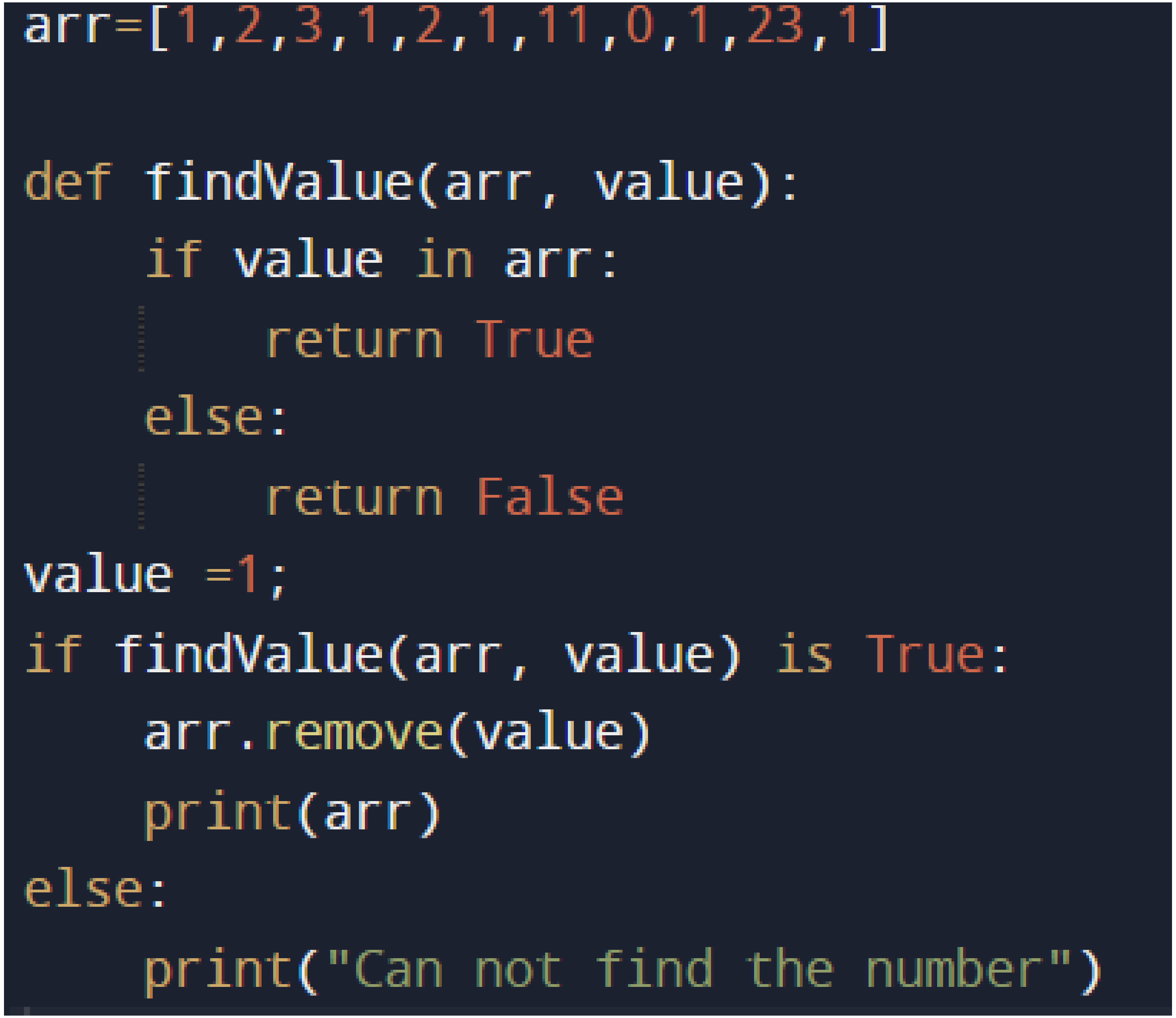
# Insert a whole List into another list

```
#Insert lst2 to lst1 before 4
lst1=[1,2,3,4,5,6,7]
lst2=['apple','pear','Plum']

#Insert Function
def insertElemt(id):
    for element in lst2:
        lst1.insert(id,element)
        id +=1
id=lst1.index(4)
insertElemt(id)
print(lst1)
```



```
#Insert lst2 to lst1 before 4

lst1=[1,2,3,4,5,6,7]
lst2=['apple','pear','Plum']

#Insert Function
def insertElemt(id):
    for element in lst2:
        lst1.insert(id,element)
        id +=1

id=lst1.index(4)
insertElemt(id)
print(lst1)
```

```
[1, 2, 3, 'apple', 'pear', 'Plum', 4, 5, 6, 7]
```



```
arr=[1,2,3,1,2,1,11,0,1,23,1]

def findValue(arr, value):
    if value in arr:
        return True
    else:
        return False
value =1;
if findValue(arr, value) is True:
    arr.remove(value)
    print(arr)
else:
    print("Can not find the number")
```

# pop() vs. remove() vs. del

```
>>> stuff
['book', 99, ['cookie', '22'], 'cookie', '22']
>>> stuff.pop()
'22'
>>> stuff
['book', 99, ['cookie', '22'], 'cookie']
>>> stuff.pop(1)
99
>>> stuff
['book', ['cookie', '22'], 'cookie']
```

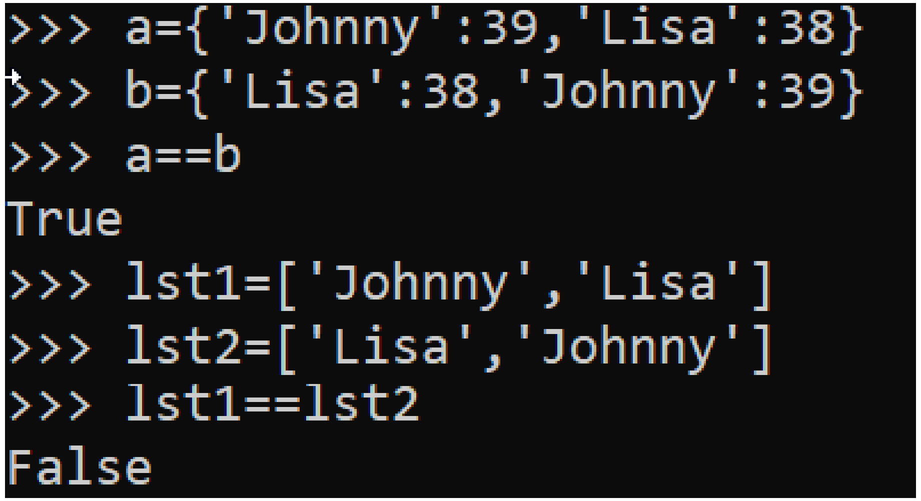
```
>>> stuff
['book', 99, ['cookie', '22'], 'cookie', '22']
>>> stuff.remove('22')
>>> stuff
['book', 99, ['cookie', '22'], 'cookie']
>>> stuff.remove(99)
>>> stuff
['book', ['cookie', '22'], 'cookie']
```

```
>>> stuff=['book', 99, ['cookie', '22'], 'cookie', '22']
>>> del stuff[4]
>>> stuff
['book', 99, ['cookie', '22'], 'cookie']
>>> del stuff[1]
>>> stuff
['book', ['cookie', '22'], 'cookie']
```

# Dictionary

- Hash map or associative array
- Dictionary is an unordered collection of key-value pairs (Based on the Key, you can get the value.)
- It is generally used when we have a huge amount of data.
- Operations:
  - Length
  - del d[k]
  - Membership Testing

```
>>> students={"Alice": 24, "Bob":27, "Dan": 21,
"Emma": 23}
>>> students["Fred"]=25
>>> students
{'Alice': 24, 'Bob': 27, 'Dan': 21, 'Emma': 23, 'Fred':
25}
>>> del students["Fred"]
>>> students
{'Alice': 24, 'Bob': 27, 'Dan': 21, 'Emma': 23}
>>> students.keys()
dict_keys(['Alice', 'Bob', 'Dan', 'Emma'])
```



```
>>> a={'Johnny':39,'Lisa':38}
>>> b={'Lisa':38,'Johnny':39}
>>> a==b
True
>>> lst1=['Johnny','Lisa']
>>> lst2=['Lisa','Johnny']
>>> lst1==lst2
False
```

# How to access each key?

```
>>> students.keys()
dict_keys(['Alice', 'Bob', 'Dan', 'Emma'])
>>> new_list=list(students.keys())
>>> print(new_list)
['Alice', 'Bob', 'Dan', 'Emma', 'Lucy']
>>> new_list[2]
'Dan'
```

```
>>> students.values()
dict_values([24, 27, 21, 23, 32])
>>> value_list=list(students.values())
>>> value_list
[24, 27, 21, 23, 32]
>>> value_list[0]
24
>>>
```

Exercise 1-3

# DataFrame

- A DataFrame organizes data into a table of rows and columns (Excel sheet)
- How to Create a DataFrame
  - Creation with Dict of equal-length lists
  - Creation with Dict of Dicts

**Ex. 3**

```
import pandas as pd

# Create data set.
dataSet = {'First Name': ['Jonny','Holly','Nira'],
         'Grade': [85,95,91] }
# Create dataframe with data set and named columns.
# Column names must match the dataSet properties.
df = pd.DataFrame(dataSet, columns= ['First Name', 'Grade'])
# Show DataFrame
print(df)
```

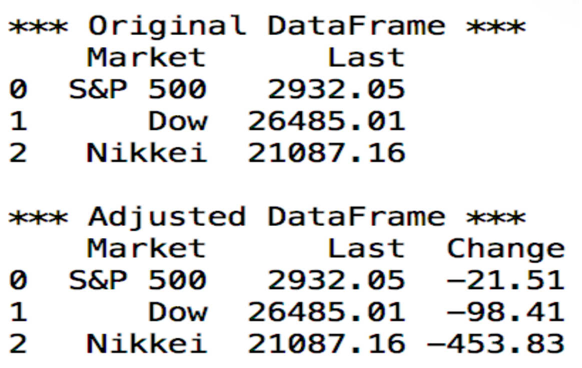
Exercise 4
4 Mins

```
>>> from pandas import DataFrame
>>> data={'state':['Ohio', 'ohio', 'Ohio', 'Nevada', 'Nevada',
'Nevada'],'year': [2000,2001,2002,2000,2001,2002], 'pop':
[1.5,1.7,3.6,2.4,2.9,3.2]}
>>> df=DataFrame(data)
>>> df
   pop   state  year
0  1.5    Ohio  2000
1  1.7    ohio  2001
2  3.6    Ohio  2002
3  2.4  Nevada  2000
4  2.9  Nevada  2001
5  3.2  Nevada  2002
```

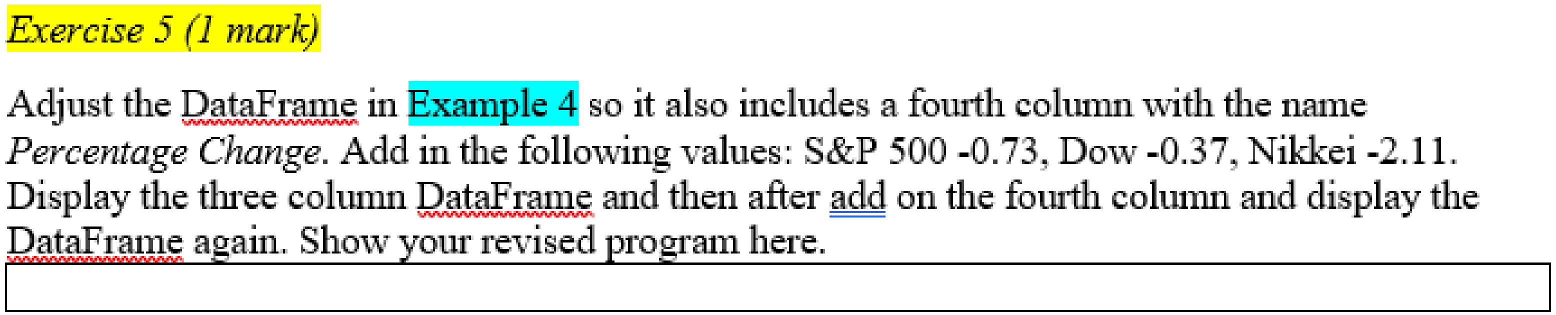
# Adding Columns to DataFrame

Syntax: df['New Column Name'] = columnDataArray

```
import pandas as pd

# Create data set.
dataSet = {'Market': ['S&P 500', 'Dow', 'Nikkei'],
          'Last': [2932.05, 26485.01, 21087.16] }

# Create dataframe with data set and named columns.
df = pd.DataFrame(dataSet, columns= ['Market', 'Last'])

# Show original DataFrame.
print("\n*** Original DataFrame ***")
print(df)

# Create change column.
change = [-21.51, -98.41, -453.83]
# Append change column.
df['Change'] = change

# Show revised DataFrame.
print("\n*** Adjusted DataFrame ***")
print (df)
```



```
*** Original DataFrame ***
    Market      Last
0  S&P 500   2932.05
1      Dow  26485.01
2   Nikkei  21087.16

*** Adjusted DataFrame ***
    Market      Last  Change
0  S&P 500   2932.05  -21.51
1      Dow  26485.01  -98.41
2   Nikkei  21087.16 -453.83
```



*Exercise 5 (1 mark)*

Adjust the DataFrame in Example 4 so it also includes a fourth column with the name *Percentage Change*. Add in the following values: S&P 500 -0.73, Dow -0.37, Nikkei -2.11. Display the three column DataFrame and then after add on the fourth column and display the DataFrame again. Show your revised program here.

# Looping Through DataFrames

- df.loc[rowNumber][‘Column Name’]

```
import pandas as pd

# Create data set.
dataSet = {'Market': ['S&P 500', 'Dow', 'Nikkei'],
        'Last': [2932.05, 26485.01, 21087.16] }

# Create dataframe with data set and named columns.
df = pd.DataFrame(dataSet, columns= ['Market', 'Last'])

# Show original DataFrame.
print("\n*** Original DataFrame ***")
print(df)

# Add new line.
print("\n")

# Show names only
for i in range(len(df)):
    print(df.loc[i]['Market'])
```

```
*** Original DataFrame ***
   Market      Last
0  S&P 500   2932.05
1      Dow  26485.01
2   Nikkei  21087.16


S&P 500
Dow
Nikkei
```

Exercise 6-8

# Appending a DataFrame to Another

- Several functions exist for adding rows to a DataFrame.
- One way to add a row of data involves appending one DataFrame object to another.
- It is important to note that the column names of both DataFrame objects must match for the *append()* function to work.

```
import pandas as pd
# Create data set.
dataSet = {'Market': ['S&P 500', 'Dow', 'Nikkei'],
        'Last': [2932.05, 26485.01, 21087.16] }
# Create dataframe with data set and named columns.
df1 = pd.DataFrame(dataSet, columns= ['Market', 'Last'])

# Show original DataFrame.
print("\n*** Original DataFrame ***")
print(df1)

dataSet2 = { 'Market': ['Hang Seng', 'DAX'],
          'Last': [26918.58, 11872.44]}
df2 = pd.DataFrame(dataSet2, columns= ['Market', 'Last'])

df1 = df1.append(df2)
print("\n*** Adjusted DataFrame ***")
print(df1)
```

df1

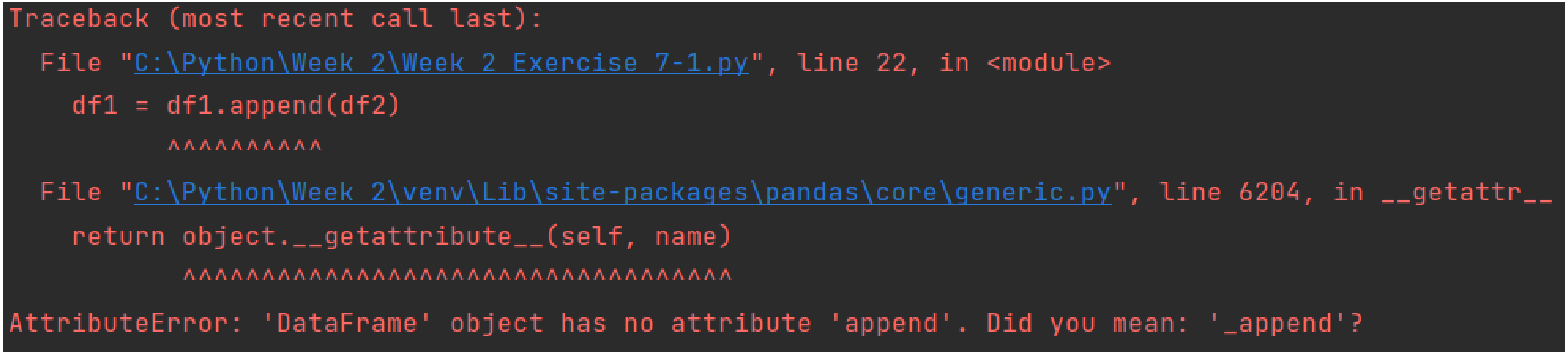
```
    Market      Last
0  S&P 500   2932.05
1      Dow  26485.01
2   Nikkei  21087.16
```

df2

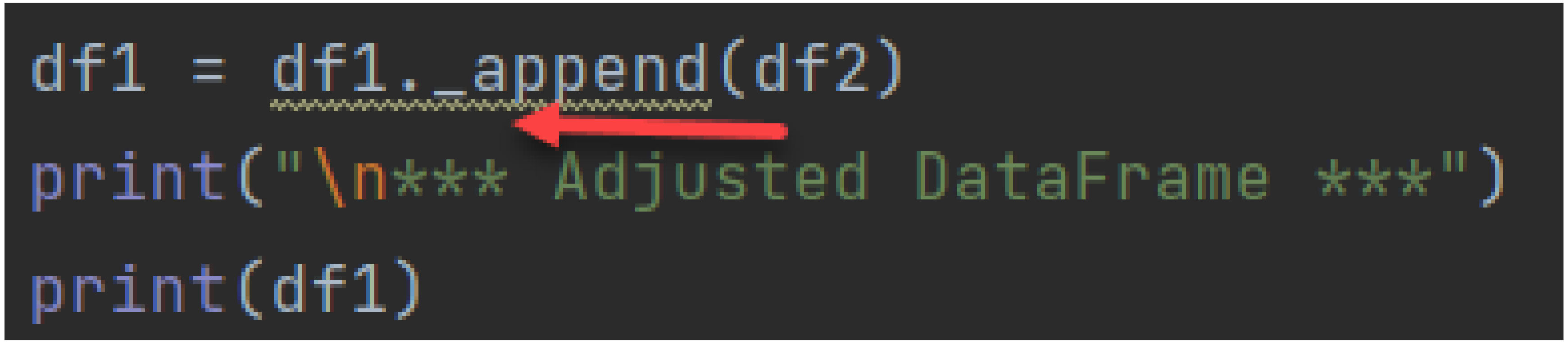
```
     Market      Last
0  Hang Seng  26918.58
1        DAX  11872.44
```

Exercise 9

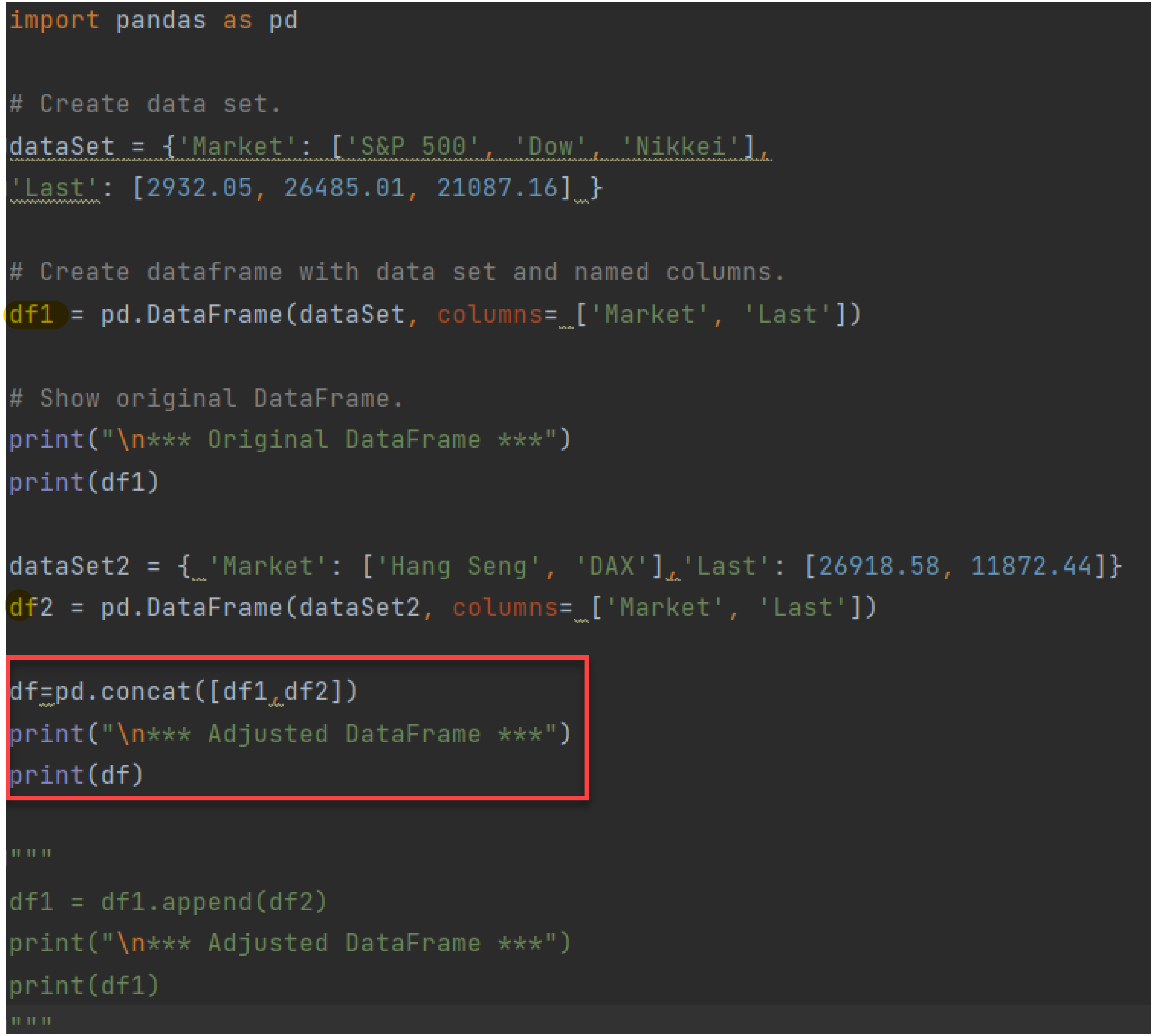
```
*** Original DataFrame ***
    Market      Last
0  S&P 500   2932.05
1      Dow  26485.01
2   Nikkei  21087.16

*** Adjusted DataFrame ***
      Market      Last
0    S&P 500   2932.05
1        Dow  26485.01
2     Nikkei  21087.16
0  Hang Seng  26918.58
1        DAX  11872.44
```



```
Traceback (most recent call last):
  File "C:\Python\Week 2\Week 2 Exercise 7-1.py", line 22, in <module>
    df1 = df1.append(df2)
          ^^^^^^^^^^
  File "C:\Python\Week 2\venv\Lib\site-packages\pandas\core\generic.py", line 6204, in __getattr__
    return object.__getattribute__(self, name)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AttributeError: 'DataFrame' object has no attribute 'append'. Did you mean: '_append'?
```



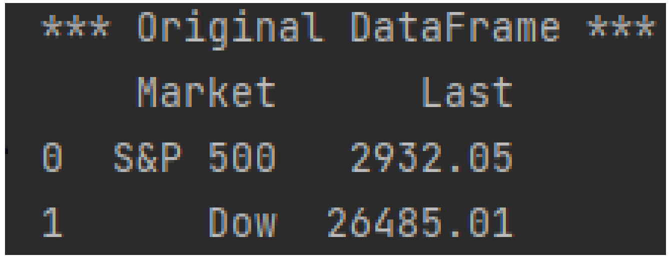
```
df1 = df1._append(df2)
print("\n*** Adjusted DataFrame ***")
print(df1)
```



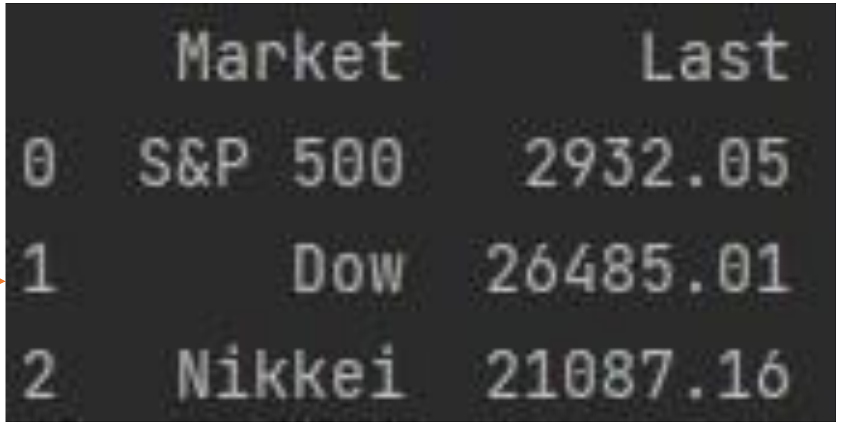
```
import pandas as pd

# Create data set.
dataSet = {'Market': ['S&P 500', 'Dow', 'Nikkei'],
'Last': [2932.05, 26485.01, 21087.16] }

# Create dataframe with data set and named columns.
df1 = pd.DataFrame(dataSet, columns= ['Market', 'Last'])

# Show original DataFrame.
print("\n*** Original DataFrame ***")
print(df1)

dataSet2 = { 'Market': ['Hang Seng', 'DAX'],'Last': [26918.58, 11872.44]}
df2 = pd.DataFrame(dataSet2, columns= ['Market', 'Last'])

df=pd.concat([df1,df2])
print("\n*** Adjusted DataFrame ***")
print(df)

"""
df1 = df1.append(df2)
print("\n*** Adjusted DataFrame ***")
print(df1)
"""
```

# Adding Rows to DataFrames

```
import pandas as pd

# Create data set.
dataSet = {'Market': ['S&P 500', 'Dow', ],
        'Last': [2932.05, 26485.01 ]}

# The dictionary is an object made of name value pairs.
stockDictionary = {'Market': 'Nikkei', 'Last': 21087.16 }

# Create dataframe with data set and named columns.
df = pd.DataFrame(dataSet, columns= ['Market', 'Last'])

# Show original DataFrame.
print("\n*** Original DataFrame ***")

df = df.append(stockDictionary, ignore_index=True)
print(df);
```

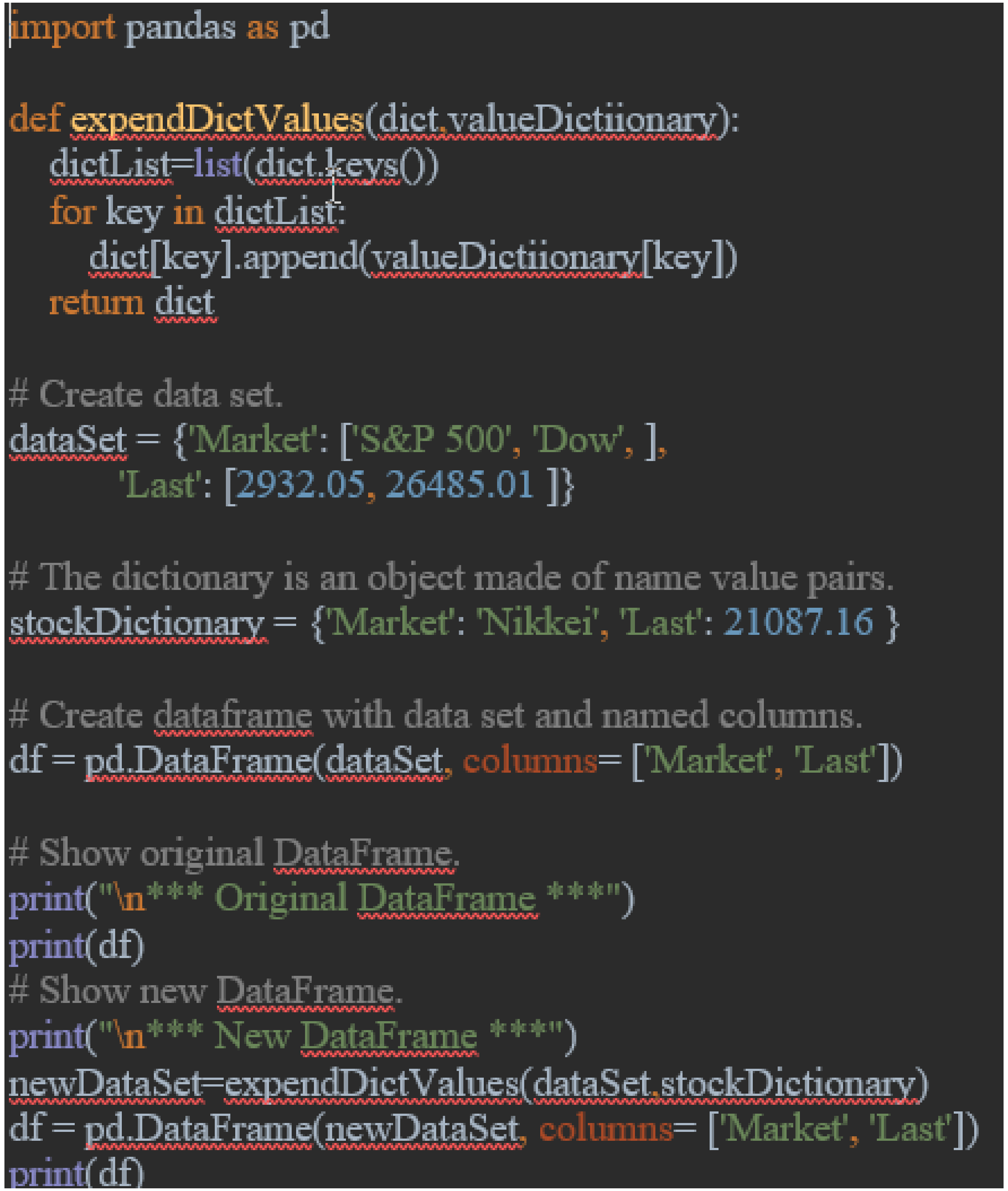


```
*** Original DataFrame ***
    Market      Last
0  S&P 500   2932.05
1      Dow  26485.01
```



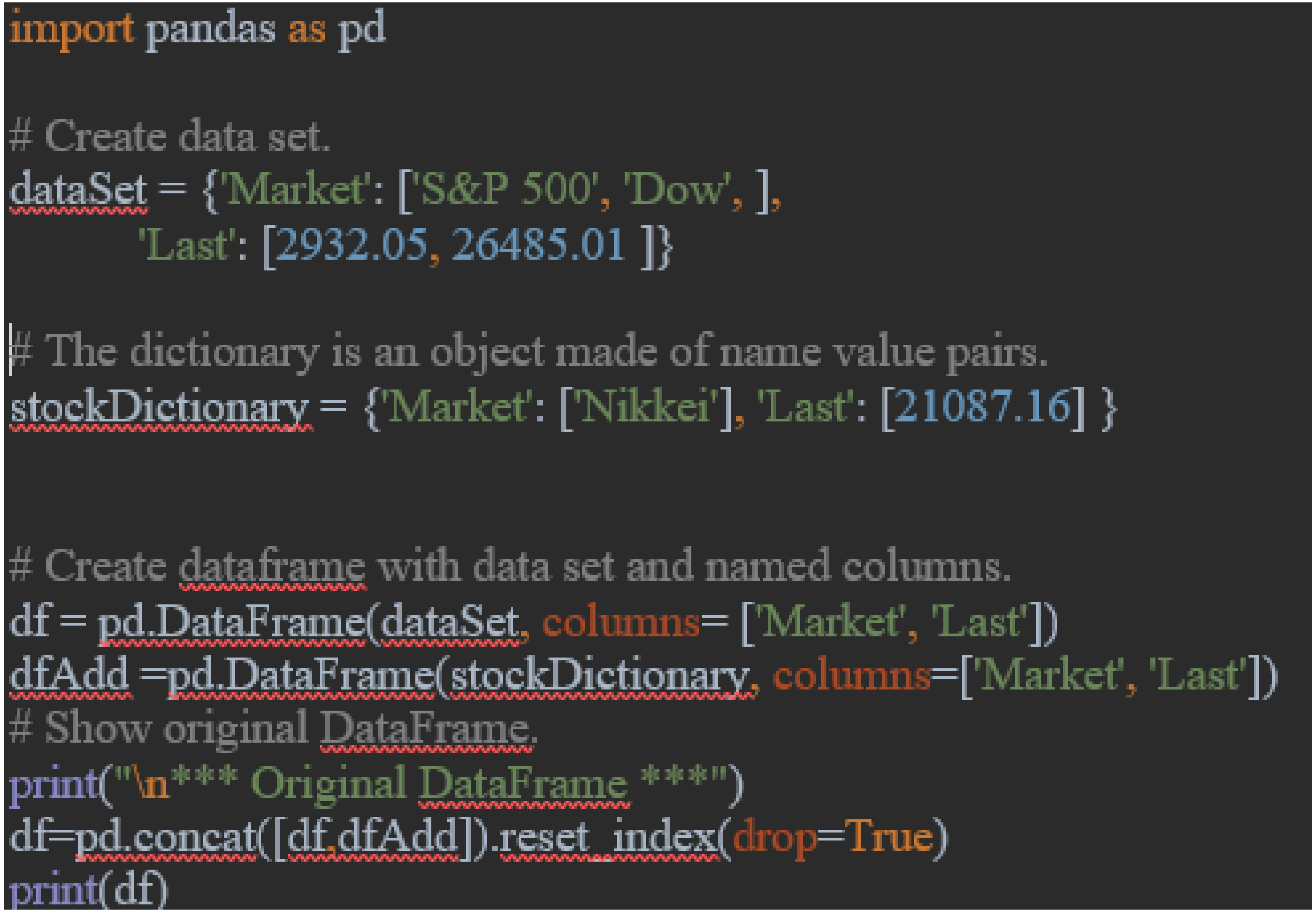
```
    Market      Last
0  S&P 500   2932.05
1      Dow  26485.01
2   Nikkei  21087.16
```

## Solution 1: (function)



```
import pandas as pd

def expendDictValues(dict,valueDictiionary):
    dictList=list(dict.keys())
    for key in dictList:
        dict[key].append(valueDictiionary[key])
    return dict

# Create data set.
dataSet = {'Market': ['S&P 500', 'Dow', ],
          'Last': [2932.05, 26485.01 ]}

# The dictionary is an object made of name value pairs.
stockDictionary = {'Market': 'Nikkei', 'Last': 21087.16 }

# Create dataframe with data set and named columns.
df = pd.DataFrame(dataSet, columns= ['Market', 'Last'])

# Show original DataFrame.
print("\n*** Original DataFrame ***")
print(df)
# Show new DataFrame.
print("\n*** New DataFrame ***")
newDataSet=expendDictValues(dataSet,stockDictionary)
df = pd.DataFrame(newDataSet, columns= ['Market', 'Last'])
print(df)
```

# Solution 2: (concat)



```
import pandas as pd

# Create data set.
dataSet = {'Market': ['S&P 500', 'Dow', ],
          'Last': [2932.05, 26485.01 ]}

# The dictionary is an object made of name value pairs.
stockDictionary = {'Market': ['Nikkei'], 'Last': [21087.16] }


# Create dataframe with data set and named columns.
df = pd.DataFrame(dataSet, columns= ['Market', 'Last'])
dfAdd =pd.DataFrame(stockDictionary, columns=['Market', 'Last'])
# Show original DataFrame.
print("\n*** Original DataFrame ***")
df=pd.concat([df,dfAdd]).reset_index(drop=True)
print(df)
```

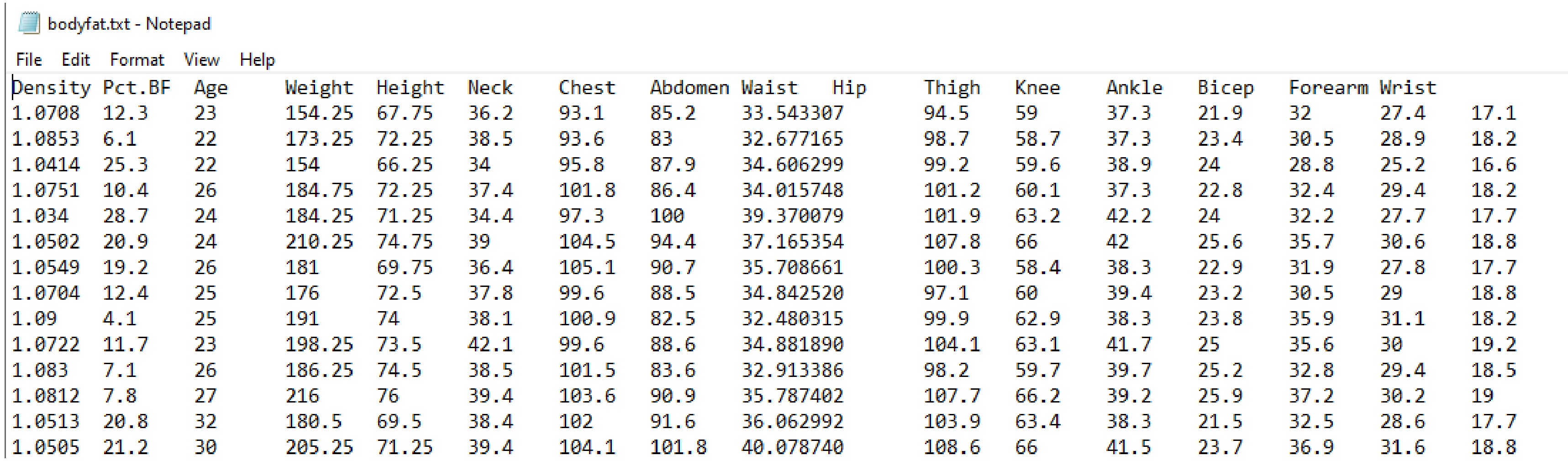
# Built-in Functions

- head() & tail(): The *head()* and *tail()* functions of the DataFrame allow you to display the first and last columns of a DataFrame. These two functions are useful for a quick scan of contents of the DataFrame.
- dtypes(): The *dtypes* property allows you to determine the data type required for cells of a specific column
- describe(): The describe() function lets us quickly generate statistical summaries for the numerical columns of a data frame. Here we can observe count, mean, std, min, max and percentile.

Exercise 11

# pd.read_csv()

```
import pandas as pd
path = "/Users/pm/Desktop/DayDocs/2019_2020/PythonForDataAnalytics/workingData/bodyfat.txt"
df = pd.read_csv(path, skiprows=1,
                 sep='\t',
                 names=('Density', 'Pct.BF', 'Age',   'Weight', 'Height',
                     'Neck', 'Chest', 'Abdomen',  'Waist', 'Hip',  'Thigh',
                    'Ankle', 'Knee', 'Bicep', 'Forearm', 'Wrist'))
# Show all columns.
pd.set_option('display.max_columns', None)

# Increase number of columns that display on one line.
pd.set_option('display.width', 1000)
```



```
Density Pct.BF  Age     Weight  Height  Neck    Chest   Abdomen Waist   Hip     Thigh   Knee    Ankle   Bicep   Forearm Wrist
1.0708  12.3    23      154.25  67.75   36.2    93.1    85.2    33.543307       94.5    59      37.3    21.9    32      27.4    17.1
1.0853  6.1     22      173.25  72.25   38.5    93.6    83      32.677165       98.7    58.7    37.3    23.4    30.5    28.9    18.2
1.0414  25.3    22      154     66.25   34      95.8    87.9    34.606299       99.2    59.6    38.9    24      28.8    25.2    16.6
1.0751  10.4    26      184.75  72.25   37.4    101.8   86.4    34.015748       101.2   60.1    37.3    22.8    32.4    29.4    18.2
1.034   28.7    24      184.25  71.25   34.4    97.3    100     39.370079       101.9   63.2    42.2    24      32.2    27.7    17.7
1.0502  20.9    24      210.25  74.75   39      104.5   94.4    37.165354       107.8   66      42      25.6    35.7    30.6    18.8
1.0549  19.2    26      181     69.75   36.4    105.1   90.7    35.708661       100.3   58.4    38.3    22.9    31.9    27.8    17.7
1.0704  12.4    25      176     72.5    37.8    99.6    88.5    34.842520       97.1    60      39.4    23.2    30.5    29      18.8
1.09    4.1     25      191     74      38.1    100.9   82.5    32.480315       99.9    62.9    38.3    23.8    35.9    31.1    18.2
1.0722  11.7    23      198.25  73.5    42.1    99.6    88.6    34.881890       104.1   63.1    41.7    25      35.6    30      19.2
1.083   7.1     26      186.25  74.5    38.5    101.5   83.6    32.913386       98.2    59.7    39.7    25.2    32.8    29.4    18.5
1.0812  7.8     27      216     76      39.4    103.6   90.9    35.787402       107.7   66.2    39.2    25.9    37.2    30.2    19
1.0513  20.8    32      180.5   69.5    38.4    102     91.6    36.062992       103.9   63.4    38.3    21.5    32.5    28.6    17.7
1.0505  21.2    30      205.25  71.25   39.4    104.1   101.8   40.078740       108.6   66      41.5    23.7    36.9    31.6    18.8
```

# Extracting Column Subsets

```
import pandas as pd

# Create data set.
dataSet = {'Market': ['S&P 500', 'Dow', 'Nikkei'],
           'Last': [2932.05, 26485.01, 21087.16] }

# Create dataframe with data set and named columns.
df = pd.DataFrame(dataSet, columns= ['Market', 'Last'])
change = [-21.51, -98.41, -453.83]

df['Change'] = change

df2 = df[["Market", "Change"]]
# Show DataFrame
print("\n*** Adjusted DataFrame ***")
print (df2)
```