Notebook_Practice

April 14, 2021

1 Markdown

This is a markdown cell where we write instructions and math formulas. **Double click** to edit and **press Run** to see the formatted texts.

2 Load and Save Notebook

To load a note book, just go to “File” and navigate to “Open…”. Use keyboard shortcut **Cmd/Ctrl+s** to save the file. You will be saving a checkpoint as well if you use this keyboard shortcut. You can click on “File” on the menu bar and select an option to revert to a certain checkpoint. Jupyter Notebook will also autosave your file every 120s, but it’s still a good practice to press **Cmd/Ctrl+s** often when you write programs.

3 Code Blocks

Below will be an example of a Python code block. Hit **Run** to run the block.

[1]:
```python
print("Hello world!")
```

```
Hello world!
```

You can also hit **Run All** in “Cell” menu. To run every blocks.

[2]:
```python
print("Hello world, again!")
```

```
Hello world, again!
```

4 Python Programming

We have finished Jupyter Notebook introduction. Below we will start Python Programming.

4.1 Importing Libraries

For the lab, we will be using different libraries. This is how you do it in Python.

[3]:
```python
import pandas as pd
import numpy as np
```

By importing pandas as pd, you can use the shorthand `pd` when you need to use pandas. The same method applies to `np`.

4.2 NumPy Tutorial

Below we show you a few ways to create arrays.

[4]:
```
# Create a basic array
x = np.array([0, 1, 2, 3])
print("x:\n", x)

# Create a 2D array
y = np.array([[0, 1, 2, 3],
              [4, 5, 6, 7]])
print("y:\n", y)

# Create a 3 by 4 empty array
z = np.zeros((3, 4))
print("z:\n", z)

# Create a 5 by 5 array filled with 1s
a = np.ones((5, 5))
print("a:\n", a)

# Create a sequence of numbers, note the array does not include 20
b = np.arange(0, 20, 2)
print("b:\n", b)

# Use reshape to change the dimention of the array
c = np.arange(0, 20, 2, dtype=float).reshape(2, 5)
print("c:\n", c)

# Create a random 6 by 3 array with value between [0, 1).
d = np.random.rand(6, 3)
print("d:\n", d)

    # This creates random values ranging between [1, 4).
d = d*3 + 1
print("New d:\n", d)

# Operations
e = d + 10
print("e:\n", e)
```

```
x:
 [0 1 2 3]
y:
 [[0 1 2 3]
 [4 5 6 7]]
z:
 [[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
a:
 [[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]]
b:
 [ 0  2  4  6  8 10 12 14 16 18]
c:
 [[ 0.  2.  4.  6.  8.]
 [10. 12. 14. 16. 18.]]
d:
 [[0.01043392 0.27136147 0.8516948 ]
 [0.61517377 0.09017255 0.87724232]
 [0.71287234 0.66720175 0.78703106]
 [0.38180917 0.88074736 0.25719089]
 [0.29094915 0.45519366 0.94813764]
 [0.13283727 0.82504084 0.61680693]]
New d:
 [[1.03130177 1.81408441 3.55508439]
 [2.84552131 1.27051766 3.63172697]
 [3.13861702 3.00160526 3.36109319]
 [2.14542751 3.64224208 1.77157268]
 [1.87284744 2.36558099 3.84441291]
 [1.39851182 3.47512253 2.85042078]]
e:
 [[11.03130177 11.81408441 13.55508439]
 [12.84552131 11.27051766 13.63172697]
 [13.13861702 13.00160526 13.36109319]
 [12.14542751 13.64224208 11.77157268]
 [11.87284744 12.36558099 13.84441291]
 [11.39851182 13.47512253 12.85042078]]
```

Now it's your turn.

To-do: 1. Create and print out an empty array with dimensions of 15 by 29. 2. From the 0 array you created, make it to an ones array without using np.ones. Print out the array. 3. Create an 8 by 9 array with values between $[5, 11)$. Note that 11 is excluded. Print out the array.

[5]:
```
### Insert code below ###
foo = np.zeros((15, 29))
bar = foo + 1
baz = np.arange(5, 11, 1).repeat(12).reshape((8, 9))
```

[6]:
```
print(bar)
```

```
[[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
  1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
  1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
  1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
  1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
  1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
  1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
  1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
  1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
  1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
  1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
  1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
  1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
  1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
  1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
  1. 1. 1. 1. 1.]]
```

[7]:
```
baz
```

[7]:
```
array([[ 5,  5,  5,  5,  5,  5,  5,  5,  5],
       [ 5,  5,  5,  6,  6,  6,  6,  6,  6],
       [ 6,  6,  6,  6,  6,  6,  7,  7,  7],
       [ 7,  7,  7,  7,  7,  7,  7,  7,  7],
       [ 8,  8,  8,  8,  8,  8,  8,  8,  8],
       [ 8,  8,  8,  9,  9,  9,  9,  9,  9],
       [ 9,  9,  9,  9,  9,  9, 10, 10, 10],
       [10, 10, 10, 10, 10, 10, 10, 10, 10]])
```

4.3 Pandas Tutorial

4.3.1 Create Data Frame

Pandas is a data frame that will be used to handel our data.

[8]:
```
# Create a data frame from numpy array
df_1 = pd.DataFrame(y, columns=['col 1', 'col 2', 'col 3', 'col 4'])
print("df_1:\n", df_1, '\n')
print("------------------------------------") # for readability only

# You can also use display() to make the dataframe looks nicer at the output
print("df_1 using display:")
display(df_1)
print("------------------------------------")

# Create a data frame with strings
data_2 = {'Aminal': ['Dog', 'Cat'], 'Color': ['Yellow', 'Pink'], 'Age': [1, 3]}
df_2 = pd.DataFrame(data=data_2)
print("df_2:\n", df_2, '\n')
print("------------------------------------")

# Print the type of the data. Notice that Age is int
df_2.dtypes
```

```
df_1:
    col 1  col 2  col 3  col 4
0      0      1      2      3
1      4      5      6      7

------------------------------------
df_1 using display:

   col 1  col 2  col 3  col 4
0      0      1      2      3
1      4      5      6      7

------------------------------------
df_2:
   Aminal   Color  Age
0    Dog  Yellow    1
1    Cat    Pink    3

------------------------------------
```

[8]:
```
Aminal    object
Color     object
Age        int64
dtype: object
```

Now it's your turn. To-do:

1. Create a data frame containing demographics of you and your 2 friends. The data frame should be 3 by 4. The columns will be 'Name', 'Age', 'Height', 'Hobby'. You can make up data if you like. Print or display the result.

```
[9]: ### Insert your code below ###
friends = pd.DataFrame( {"Name": ["Rick", "Bert", "Ernie", "Big Bird"],
                        "Age": [41, 31, 31, 12],
                        "Height": [70, 12, 2, 86],
                        "Hobby": ["IoT", "Puppets", "Puppets", "Snuffleupagus␣
 ↪Sightings"]})



print(friends)
```

```
       Name  Age  Height                  Hobby
0      Rick   41      70                    IoT
1      Bert   31      12                Puppets
2     Ernie   31       2                Puppets
3  Big Bird   12      86  Snuffleupagus Sightings
```

4.3.2 Modify Data Frame

Notice that when you use `=`, you are not copying data frame. You are just saying that `df_3` is now referring to the same data frame as `df_1`. If you change values in `df_3` you will change the values in `df_1` too, since both are referring to the same dataframe. Also notice that every time you click **Run** in this block, the values in `'col 1'` changes.

```
[10]: # Add a new column
df_2['Weight'] = [89, 60]
print("New df_2:")
display(df_2)
print("-------------------------------------")

# Special thing to take notice
df_3 = df_1
df_3['col 1'] = df_3['col 1'] - 1
print("df_3:")
display(df_3)
print("df_1:")
display(df_1)
```

```
New df_2:

  Aminal   Color  Age  Weight
0    Dog  Yellow    1      89
1    Cat    Pink    3      60
```

```
------------------------------------
df_3:

   col 1  col 2  col 3  col 4
0     -1      1      2      3
1      3      5      6      7

df_1:

   col 1  col 2  col 3  col 4
0     -1      1      2      3
1      3      5      6      7
```

4.3.3 Print Specific Data

Here we use `.loc`, `.at` to obtain the cell values by providing the **labels** (e.g. 'Age', 'Weight'). For rows, since we do not create labels for them the defaults will be 0, 1, 2, 3...etc. We use `.iloc`, `.iat` to obtain the cell values by prividing the indicies (positions) of the rows and columns.

[11]:
```python
# Few ways to view the values

# Create a data frame based on data_2; the values are copied
df_4 = pd.DataFrame(data=data_2)
print("df_4:")
display(df_4)
print("------------------------------------")

# Selection by Label
    # Getting the scalar value
dog_age = df_4.loc[0, 'Age']
print("Age of Dog:", dog_age, '\n')
print("------------------------------------")

    # Getting the whole column
aminals_age = df_4.loc[:, ['Age']]
print("Age Column:")
display(aminals_age)
print("------------------------------------")

    # Faster way to get a scalar
cat_age = df_4.at[1, 'Age']
print("Age of Cat:", cat_age, '\n')
print("------------------------------------")

# Selection by position
    # Selecting Row based on row number
dog = df_4.iloc[0]
print("Dog row:")
print(dog, '\n')
print("------------------------------------")

    # Selecting Col based on col number
animals_age_p = df_4.iloc[:, 2]
print("Animals' age:")
print(animals_age_p, '\n')
print("------------------------------------")
    # Selecting cell based on col number
cat_age_p = df_4.iat[1, 2]
print("Age of Cat: ", cat_age_p)
```

```
df_4:

  Aminal   Color  Age
0    Dog  Yellow    1
1    Cat    Pink    3

------------------------------------
Age of Dog: 1

------------------------------------
Age Column:

   Age
0    1
1    3

------------------------------------
Age of Cat: 3

------------------------------------
Dog row:
Aminal       Dog
Color     Yellow
Age            1
Name: 0, dtype: object

------------------------------------
Animals' age:
0    1
1    3
Name: Age, dtype: int64

------------------------------------
Age of Cat:  3
```

Now it's your turn. To-do:

1. Add a new **row** to your **demographics** data frame. The new row will contain information of another friend. *Note that we haven't taught you how to do so but you should be able to find*

resources online easily. Print/display the data frame.

2. Print all the information (whole row) about you using `.loc`.
3. Print the 'Name' of your second friend using `.iat`.

[12]:
```
### Insert your code below ###
new_friends = friends.append({"Name": "Jerry Seinfeld", "Age": 65, "Height":␣
↪74, "Hobby": "Comedy"}, ignore_index=True)
print(new_friends)
print("\nMyself\n=======")
print(new_friends.loc[0])
print("\n\n\nMy 2nd Friend\n=====")
print(new_friends.loc[2].iat[0])
```

```
             Name  Age  Height                  Hobby
0            Rick   41      70                    IoT
1            Bert   31      12                Puppets
2           Ernie   31       2                Puppets
3        Big Bird   12      86  Snuffleupagus Sightings
4  Jerry Seinfeld   65      74                 Comedy

Myself
=======
Name      Rick
Age         41
Height      70
Hobby      IoT
Name: 0, dtype: object



My 2nd Friend
=====
Ernie
```

Congradulation on finishing the tutorial! Now you can move on to the next step of the lab.