# 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.

# 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.

# Code Blocks

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

In [26]:

```
print("Hello world!")
```

```
Hello world!
```

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

In [2]:

```
print("Hello world, again!")
```

```
Hello world, again!
```

# Python Programming

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

## Importing Libraries

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

In [2]:

```
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`.

## NumPy Tutorial

Below we show you a few ways to create arrays.

In [14]:

```
# 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:
 [[0.10769134 0.90335426 0.26376816]
 [0.56614851 0.08040202 0.98168606]
 [0.60836736 0.4511608  0.21463848]
 [0.11412946 0.15256207 0.94490592]
 [0.37240798 0.30002266 0.71380106]
 [0.12298735 0.97421158 0.05734405]]
New d:
 [[1.32307402 3.71006279 1.79130449]
 [2.69844554 1.24120606 3.94505817]
 [2.82510207 2.35348241 1.64391543]
 [1.34238839 1.45768621 3.83471776]
 [2.11722393 1.90006799 3.14140318]
 [1.36896205 3.92263474 1.17203215]]
e:
 [[11.32307402 13.71006279 11.79130449]
 [12.69844554 11.24120606 13.94505817]
 [12.82510207 12.35348241 11.64391543]
 [11.34238839 11.45768621 13.83471776]
 [12.11722393 11.90006799 13.14140318]
 [11.36896205 13.92263474 11.17203215]]
```

### 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.

In [12]:

```
### Insert code below ###
arr = np.zeros((15, 29))
print("arr:\n", arr)

arr = arr + 1
print("arr:\n", arr)

arr = np.random.randint(5, 11, (8,9))
print("arr:\n", arr)
```

```
arr:
 [[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0.
  0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0.
  0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0.
  0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0.
  0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0.
  0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0.
  0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0.
  0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0.
  0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0.
  0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0.
  0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0.
  0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0.
  0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0.
  0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0.
  0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0.
  0. 0. 0. 0. 0.]]
arr:
 [[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1.
  1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1.
  1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1.
  1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1.
  1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1.
  1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1.
  1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1.
  1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1.
  1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1.
  1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1.
  1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1.
  1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1.
  1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1.
  1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1.
  1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1.
  1. 1. 1. 1. 1.]]
arr:
 [[ 7  7  6  8  6  7  5  7 10]
 [ 5  9  9  5 10  8 10  5  9]
 [ 9  9  7 10  9  8  5  5  8]
 [ 5  7  6  5 10  6  5  8  9]
 [10  5 10  8  6  7  8  9  7]
 [ 5  7  5  6  7  7  5  6  6]
 [ 7  8  9  7  5  7  9  5 10]
 [ 9  7  8  6  8 10  7 10  6]]
```

# Pandas Tutorial

## Create Data Frame

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

In [15]:
```
# 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

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

Out[15]:
```
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.

In [22]:

```
### Insert your code below ###
friends_data = np.array([['Ryan', '40', '6.0', 'Snowboarding'],
                         ['Roger', '39', '5.75', 'Writing'],
                         ['Steve', '39', '5.6', 'Reading']])

df_friends = pd.DataFrame(friends_data, columns=['Name', 'Age', 'Height'
, 'Hobby'])
print("df_friends:\n", df_friends, '\n')
print("------------------------------------") # for readability only
```

```
df_friends:
     Name Age Height         Hobby
0   Ryan  40    6.0  Snowboarding
1  Roger  39   5.75       Writing
2  Steve  39    5.6       Reading

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

## 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.

In [23]:

```
# 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** | -2 | 1 | 2 | 3 |
| **1** | 2 | 5 | 6 | 7 |

```
df_1:
```

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

## 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.

In [24]:

```
# 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` .

In [25]:

```
### Insert your code below ###
df_friends = df_friends.append({'Name' : 'Kevin', 'Age' : '37', 'Height'
: 6.1 , 'Hobby': 'Photography'}, ignore_index = True)
print("df_friends:\n", df_friends, '\n')
print("------------------------------------") # for readability only
print("Me row:\n", df_friends.iloc[0], '\n')
print("------------------------------------") # for readability only
print("Second friend Name:\n", df_friends.iat[1, 0], '\n')
print("------------------------------------") # for readability only
```

```
df_friends:
      Name Age Height         Hobby
0    Ryan  40    6.0  Snowboarding
1   Roger  39   5.75       Writing
2   Steve  39    5.6       Reading
3   Kevin  37    6.1   Photography

------------------------------------
Me row:
 Name                Ryan
Age                   40
Height               6.0
Hobby     Snowboarding
Name: 0, dtype: object

------------------------------------
Second friend Name:
 Roger

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

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