

**BEDROCK  
DATA SCIENCE**

*If you don't enjoy, you won't excel!*

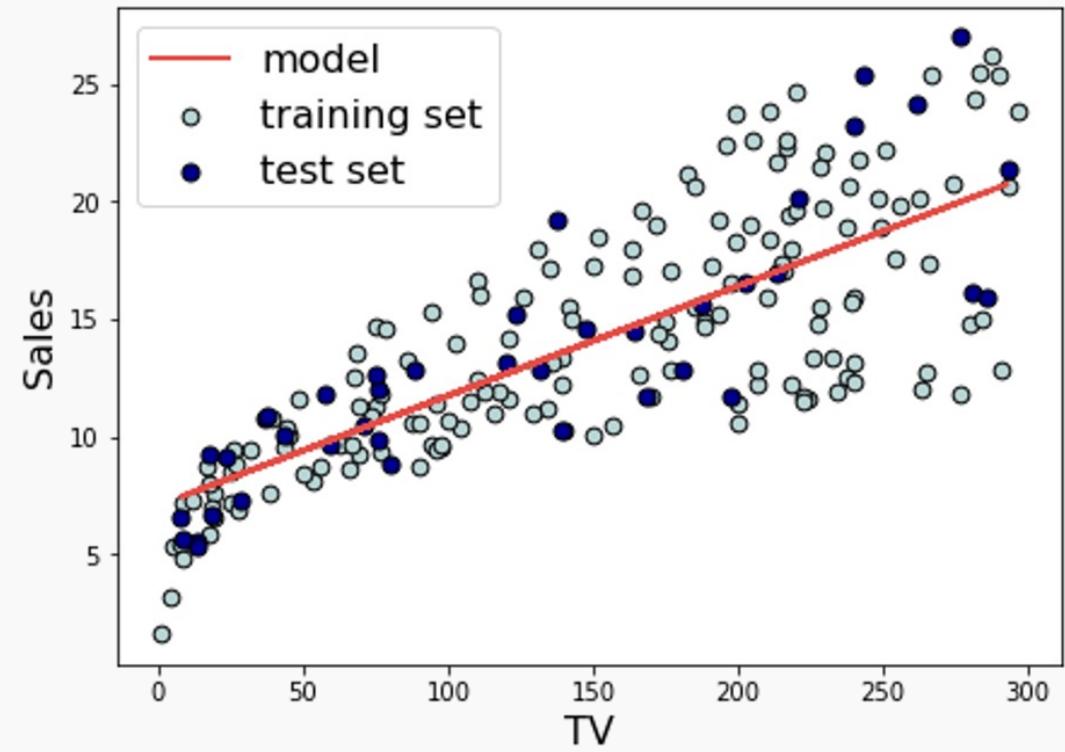


Our Goal

# CODE

```
1 # import required libraries
2 import numpy as np
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 import csv
6 from sklearn.linear_model import LinearRegression
7
8 # Read the 'Advertising.csv' dataset
9 with open('Advertising.csv', mode='r') as infile:
10     reader = csv.reader(infile)
11     values = [rows[1], rows[4]] for rows in reader]
12 data_dictionary = {i[0]:i[1] for i in list(zip(*values))}
13 # Assign TV advertising as predictor variable 'x' and sales as response variable 'y'
14 tv,sales = data_dictionary['TV'], data_dictionary['sales']
15 x,y = np.array(tv,dtype='float32').reshape(-1,1), np.array(sales,dtype='float32')
16
17 # Split the data into training and test sets
18 number_of_points = len(x)
19 train_size = 0.8
20 num_train_points = int(train_size*number_of_points)
21
22 # Create indices to split the dataset
23 train_index = np.random.choice(range(len(x)),size=num_train_points,replace=False)
24 test_index = [i for i in range(len(x)) if i not in train_index]
25 test_index = np.array(test_index)
26
27 # Create boolean masks for training and test sets
28 mask = np.zeros(len(x), dtype = 'int')
29 mask[train_index] = 1
30 mask = mask==1
31 # Use the masks to create train and test data
32 x_train,y_train = x[mask],y[mask]
33 x_test, y_test = x[~mask],y[~mask]
34
35 # Write a function to compute the mean squared error of the predictions
36 def mse(y_true, y_prediction):
37     error = y_true - y_prediction
38     squared_error = error**2
39     mean_squared_error = 1/len(y_true)*sum(squared_error).item(0)
40     return mean_squared_error
41
42 # Use the sklearn function 'LinearRegression' to fit on the training set
43 model = LinearRegression()
44 model.fit(x_train, y_train)
45 # Now predict on the test set
46 y_pred_test = model.predict(x_test)
47
48 # Now compute the MSE with the predicted values and print it
49 test_mse = mse(y_test, y_pred_test)
50 print(f'The test MSE is {test_mse}')
```

# OUTPUT



# What's going on here?

```
1 # import required libraries
2 import numpy as np
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 import csv
6 from sklearn.linear_model import LinearRegression
7
8 # Read the 'Advertising.csv' dataset
9 with open('Advertising.csv', mode='r') as infile:
10     reader = csv.reader(infile)
11     values = [(rows[1],rows[4]) for rows in reader]
12     data_dictionary = {i[0]:i[1] for i in list(zip(*values))}
13 # Assign TV advertising as predictor variable 'x' and sales as response variable 'y'
14 tv,sales = data_dictionary['TV'], data_dictionary['sales']
15 x,y = np.array(tv,dtype='float32').reshape(-1,1), np.array(sales,dtype='float32')
16
17 # Split the data into training and test sets
18 number_of_points = len(x)
19 train_size = 0.8
20 num_train_points = int(train_size*number_of_points)
21
22 # Create indices to split the dataset
23 train_index = np.random.choice(range(len(x)),size=num_train_points,replace=False)
24 test_index = [i for i in range(len(x)) if i not in train_index]
25 test_index = np.array(test_index)
26
27 # Create boolean masks for training and test sets
28 mask = np.zeros(len(x), dtype = 'int')
29 mask[train_index] = 1
30 mask = mask==1
31 # Use the masks to create train and test data
32 x_train,y_train = x[mask],y[mask]
33 x_test, y_test = x[~mask],y[~mask]
34
35 # Write a function to compute the mean squared error of the predictions
36 def mse(y_true, y_prediction):
37     error = y_true - y_prediction
38     squared_error = error**2
39     mean_squared_error = 1/len(y_true)*sum(squared_error).item(0)
40     return mean_squared_error
41
42 # Use the sklearn function 'LinearRegression' to fit on the training set
43 model = LinearRegression()
44 model.fit(x_train, y_train)
45 # Now predict on the test set
46 y_pred_test = model.predict(x_test)
47
48 # Now compute the MSE with the predicted values and print it
49 test_mse = mse(y_test, y_pred_test)
50 print(f'The test MSE is {test_mse}')
```

## What are libraries ?



```
2 import numpy as np
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 import csv
6 from sklearn.linear_model import LinearRegression

8 # Read the 'Advertising.csv' dataset
9 with open('Advertising.csv', mode='r') as infile:
10     reader = csv.reader(infile)
11     values = [(rows[1],rows[4]) for rows in reader]
12     data_dictionary = {i[0]:i[1] for i in list(zip(*values))}
13 # Assign TV advertising as predictor variable 'x' and sales as response variable 'y'
14 tv,sales = data_dictionary['TV'], data_dictionary['sales']
15 x,y = np.array(tv,dtype='float32').reshape(-1,1), np.array(sales,dtype='float32')
16
17 # Split the data into training and test sets
18 number_of_points = len(x)
19 train_size = 0.8
20 num_train_points = int(train_size*number_of_points)
21
22 # Create indices to split the dataset
23 train_index = np.random.choice(range(len(x)),size=num_train_points,replace=False)
24 test_index = [i for i in range(len(x)) if i not in train_index]
25 test_index = np.array(test_index)
26
27 # Create boolean masks for training and test sets
28 mask = np.zeros(len(x), dtype = 'int')
29 mask[train_index] = 1
30 mask = mask==1
31 # Use the masks to create train and test data
32 x_train,y_train = x[mask],y[mask]
33 x_test, y_test = x[~mask],y[~mask]
34
35 # Write a function to compute the mean squared error of the predictions
36 def mse(y_true, y_prediction):
37     error = y_true - y_prediction
38     squared_error = error**2
39     mean_squared_error = 1/len(y_true)*sum(squared_error).item(0)
40     return mean_squared_error
41
42 # Use the sklearn function 'LinearRegression' to fit on the training set
43 model = LinearRegression()
44 model.fit(x_train, y_train)
45 # Now predict on the test set
46 y_pred_test = model.predict(x_test)
47
48 # Now compute the MSE with the predicted values and print it
49 test_mse = mse(y_test, y_pred_test)
50 print(f'The test MSE is {test_mse}')
```

## What is a *dictionary* ?



```
1 # import required libraries
2 import numpy as np
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 import csv
6 from sklearn.linear_model import LinearRegression
7
9 with open('Advertising.csv', mode='r') as infile:
10     reader = csv.reader(infile)
11     values = [(rows[1],rows[4]) for rows in reader]
12     data_dictionary = {i[0]:i[1] for i in list(zip(*values))}
13 # Assign TV advertising as predictor variable 'x' and sales as response variable 'y'
15 x,y = np.array(tv,dtype='float32').reshape(-1,1), np.array(sales,dtype='float32')
16
17 # Split the data into training and test sets
18 number_of_points = len(x)
19 train_size = 0.8
20 num_train_points = int(train_size*number_of_points)
21
22 # Create indices to split the dataset
23 train_index = np.random.choice(range(len(x)),size=num_train_points,replace=False)
24 test_index = [i for i in range(len(x)) if i not in train_index]
25 test_index = np.array(test_index)
26
27 # Create boolean masks for training and test sets
28 mask = np.zeros(len(x), dtype = 'int')
29 mask[train_index] = 1
30 mask = mask==1
31 # Use the masks to create train and test data
32 x_train,y_train = x[mask],y[mask]
33 x_test, y_test = x[~mask],y[~mask]
34
35 # Write a function to compute the mean squared error of the predictions
36 def mse(y_true, y_prediction):
37     error = y_true - y_prediction
38     squared_error = error**2
39     mean_squared_error = 1/len(y_true)*sum(squared_error).item(0)
40     return mean_squared_error
41
42 # Use the sklearn function 'LinearRegression' to fit on the training set
43 model = LinearRegression()
44 model.fit(x_train, y_train)
45 # Now predict on the test set
46 y_pred_test = model.predict(x_test)
47
48 # Now compute the MSE with the predicted values and print it
49 test_mse = mse(y_test, y_pred_test)
50 print(f'The test MSE is {test_mse}')
```

What is an array ?



```
1 # import required libraries
2 import numpy as np
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 import csv
6 from sklearn.linear_model import LinearRegression
7
8 # Read the 'Advertising.csv' dataset
9 with open('Advertising.csv', mode='r') as infile:
10     reader = csv.reader(infile)
11
12     data_dictionary = {i[0]:i[1:] for i in list(zip(*values))}
13     # Assign TV advertising as predictor variable 'x' and sales as response variable 'y'
14     tv,sales = data_dictionary['TV'], data_dictionary['sales']
15     x,y = np.array(tv,dtype='float32').reshape(-1,1), np.array(sales,dtype='float32')
16
17
18     number_of_points = len(x)
19     train_size = 0.8
20     num_train_points = int(train_size*number_of_points)
21
22     # Create indices to split the dataset
23     train_index = np.random.choice(range(len(x)),size=num_train_points,replace=False)
24     test_index = [i for i in range(len(x)) if i not in train_index]
25     test_index = np.array(test_index)
26
27     # Create boolean masks for training and test sets
28     mask = np.zeros(len(x), dtype = 'int')
29     mask[train_index] = 1
30     mask = mask==1
31     # Use the masks to create train and test data
32     x_train,y_train = x[mask],y[mask]
33     x_test, y_test = x[~mask],y[~mask]
34
35     # Write a function to compute the mean squared error of the predictions
36     def mse(y_true, y_prediction):
37         error = y_true - y_prediction
38         squared_error = error**2
39         mean_squared_error = 1/len(y_true)*sum(squared_error).item(0)
40         return mean_squared_error
41
42     # Use the sklearn function 'LinearRegression' to fit on the training set
43     model = LinearRegression()
44     model.fit(x_train, y_train)
45     # Now predict on the test set
46     y_pred_test = model.predict(x_test)
47
48     # Now compute the MSE with the predicted values and print it
49     test_mse = mse(y_test, y_pred_test)
50     print(f'The test MSE is {test_mse}')
```

What is `len(x)` ?



```
1 # import required libraries
2 import numpy as np
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 import csv
6 from sklearn.linear_model import LinearRegression
7
8 # Read the 'Advertising.csv' dataset
9 with open('Advertising.csv', mode='r') as infile:
10     reader = csv.reader(infile)
11     values = [(rows[1],rows[4]) for rows in reader]
12 data_dictionary = {i[0]:i[1:] for i in list(zip(*values))}
13 # Assign TV advertising as predictor variable 'x' and sales as response variable 'y'
14 tv_sales = data_dictionary['TV'], data_dictionary['sales']
15
16 # Split the data into training and test sets
17 number_of_points = len(x)
18 train_size = 0.8
19 num_train_points = int(train_size*number_of_points)
20
21 # Create indices to split the dataset
22 train_index = np.random.choice(range(len(x)), size=num_train_points, replace=False)
23 test_index = [i for i in range(len(x)) if i not in train_index]
24 test_index = np.array(test_index)
25
26 # Create boolean masks for training and test sets
27 mask = np.zeros(len(x), dtype = 'int')
28 mask[train_index] = 1
29 mask = mask==1
30 # Use the masks to create train and test data
31 x_train,y_train = x[mask],y[mask]
32 x_test, y_test = x[~mask],y[~mask]
33
34 # Write a function to compute the mean squared error of the predictions
35 def mse(y_true, y_prediction):
36     error = y_true - y_prediction
37     squared_error = error**2
38     mean_squared_error = 1/len(y_true)*sum(squared_error).item(0)
39     return mean_squared_error
40
41 # Use the sklearn function 'LinearRegression' to fit on the training set
42 model = LinearRegression()
43 model.fit(x_train, y_train)
44 # Now predict on the test set
45 y_pred_test = model.predict(x_test)
46
47 # Now compute the MSE with the predicted values and print it
48 test_mse = mse(y_test, y_pred_test)
49 print(f'The test MSE is {test_mse}')
```

What is a *list comprehension* ?



```
1 # import required libraries
2 import numpy as np
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 import csv
6 from sklearn.linear_model import LinearRegression
7
8 # Read the 'Advertising.csv' dataset
9 with open('Advertising.csv', mode='r') as infile:
10     reader = csv.reader(infile)
11     values = [(rows[1],rows[4]) for rows in reader]
12 data_dictionary = {i[0]:i[1:] for i in list(zip(*values))}
13 # Assign TV advertising as predictor variable 'x' and sales as response variable 'y'
14 tv,sales = data_dictionary['TV'], data_dictionary['sales']
15 x,y = np.array(tv,dtype='float32').reshape(-1,1), np.array(sales,dtype='float32')
16
17 # Split the data into training and test sets
18 number_of_points = len(x)
19 train_size = 0.8
20
21 # Create indices to split the dataset
22 train_index = np.random.choice(range(len(x)),size=num_train_points,replace=False)
23 test_index = [i for i in range(len(x)) if i not in train_index]
24 test_index = np.array(test_index)
25
26 # Create boolean masks for training and test sets
27 mask = np.zeros(len(x), dtype = 'int')
28 mask[train_index] = 1
29 mask = mask==1
30 # Use the masks to create train and test data
31 x_train,y_train = x[mask],y[mask]
32 x_test, y_test = x[~mask],y[~mask]
33
34 # Write a function to compute the mean squared error of the predictions
35 def mse(y_true, y_prediction):
36     error = y_true - y_prediction
37     squared_error = error**2
38     mean_squared_error = 1/len(y_true)*sum(squared_error).item(0)
39     return mean_squared_error
40
41 # Use the sklearn function 'LinearRegression' to fit on the training set
42 model = LinearRegression()
43 model.fit(x_train, y_train)
44 # Now predict on the test set
45 y_pred_test = model.predict(x_test)
46
47 # Now compute the MSE with the predicted values and print it
48 test_mse = mse(y_test, y_pred_test)
49 print(f'The test MSE is {test_mse}')
```

What is a Boolean mask ?



```
1 # import required libraries
2 import numpy as np
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 import csv
6 from sklearn.linear_model import LinearRegression
7
8 # Read the 'Advertising.csv' dataset
9 with open('Advertising.csv', mode='r') as infile:
10     reader = csv.reader(infile)
11     values = [(rows[1],rows[4]) for rows in reader]
12     data_dictionary = {i[0]:i[1] for i in list(zip(*values))}
13 # Assign TV advertising as predictor variable 'x' and sales as response variable 'y'
14 tv,sales = data_dictionary['TV'], data_dictionary['sales']
15 x,y = np.array(tv,dtype='float32').reshape(-1,1), np.array(sales,dtype='float32')
16
17 # Split the data into training and test sets
18 number_of_points = len(x)
19 train_size = 0.8
20 num_train_points = int(train_size*number_of_points)
21
22 # Create indices to split the dataset
23 train_index = np.random.choice(range(len(x)),size=num_train_points,replace=False)
24 test_index = [i for i in range(len(x)) if i not in train_index]
25 test_index = np.array(test_index)
26
27 # Create boolean masks for training and test sets
28 mask = np.zeros(len(x), dtype = 'int')
29 mask[train_index] = 1
30 mask = mask==1
31 # Use the masks to create train and test data
32 x_train,y_train = x[mask],y[mask]
33 x_test, y_test = x[~mask],y[~mask]
34
35 # Write a function to compute the mean squared error of the predictions
36 def mse(y_true, y_prediction):
37     error = y_true - y_prediction
38     squared_error = error**2
39     mean_squared_error = 1/len(y_true)*sum(squared_error).item(0)
40     return mean_squared_error
41
42 # Use the sklearn function 'LinearRegression' to fit on the training set
43 model = LinearRegression()
44 model.fit(x_train, y_train)
45 # Now predict on the test set
46 y_pred_test = model.predict(x_test)
47
48 # Now compute the MSE with the predicted values and print it
49 test_mse = mse(y_test, y_pred_test)
50 print(f'The test MSE is {test_mse}')
```

How to write a python function ?



```
1 # import required libraries
2 import numpy as np
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 import csv
6 from sklearn.linear_model import LinearRegression
7
8 # Read the 'Advertising.csv' dataset
9 with open('Advertising.csv', mode='r') as infile:
10     reader = csv.reader(infile)
11     values = [(rows[1],rows[4]) for rows in reader]
12     data_dictionary = {i[0]:i[1] for i in list(zip(*values))}
13 # Assign TV advertising as predictor variable 'x' and sales as response variable 'y'
14 tv,sales = data_dictionary['TV'], data_dictionary['sales']
15 x,y = np.array(tv,dtype='float32').reshape(-1,1), np.array(sales,dtype='float32')
16
17 # Split the data into training and test sets
18 number_of_points = len(x)
19 train_size = 0.8
20 num_train_points = int(train_size*number_of_points)
21
22 # Create indices to split the dataset
23 train_index = np.random.choice(range(len(x)),size=num_train_points,replace=False)
24 test_index = [i for i in range(len(x)) if i not in train_index]
25 test_index = np.array(test_index)
26
27 # Create boolean masks for training and test sets
28 mask = np.zeros(len(x), dtype = 'int')
29 mask[train_index] = 1
30 mask = mask==1
31 # Use the masks to create train and test data
32 x_train,y_train = x[mask],y[mask]
33 x_test,y_test = x[~mask],y[~mask]
34
35 # Write a function to compute the mean squared error of the predictions
36 def mse(y_true, y_prediction):
37     error = y_true - y_prediction
38     squared_error = error**2
39     mean_squared_error = 1/len(y_true)*sum(squared_error).item(0)
40     return mean_squared_error
41
42 # Use the sklearn function 'LinearRegression' to fit on the training set
43 model = LinearRegression()
44 model.fit(x_train, y_train)
45 # Now predict on the test set
46 y_pred_test = model.predict(x_test)
47
48 # Now compute the MSE with the predicted values and print it
49 test_mse = mse(y_test, y_pred_test)
50 print(f'The test MSE is {test_mse}')
```

What is a **class constructor** ?



```
1 # import required libraries
2 import numpy as np
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 import csv
6 from sklearn.linear_model import LinearRegression
7
8 # Read the 'Advertising.csv' dataset
9 with open('Advertising.csv', mode='r') as infile:
10     reader = csv.reader(infile)
11     values = [(rows[1],rows[4]) for rows in reader]
12 data_dictionary = {i[0]:i[1] for i in list(zip(*values))}
13 # Assign TV advertising as predictor variable 'x' and sales as response variable 'y'
14 tv,sales = data_dictionary['TV'], data_dictionary['sales']
15 x,y = np.array(tv,dtype='float32').reshape(-1,1), np.array(sales,dtype='float32')
16
17 # Split the data into training and test sets
18 number_of_points = len(x)
19 train_size = 0.8
20 num_train_points = int(train_size*number_of_points)
21
22 # Create indices to split the dataset
23 train_index = np.random.choice(range(len(x)),size=num_train_points,replace=False)
24 test_index = [i for i in range(len(x)) if i not in train_index]
25 test_index = np.array(test_index)
26
27 # Create boolean masks for training and test sets
28 mask = np.zeros(len(x), dtype = 'int')
29 mask[train_index] = 1
30 mask = mask==1
31 # Use the masks to create train and test data
32 x_train,y_train = x[mask],y[mask]
33 x_test, y_test = x[~mask],y[~mask]
34
35 # Write a function to compute the mean squared error of the predictions
36 def mse(y_true, y_prediction):
37     error = y_true - y_prediction
38     squared_error = error**2
39     mean_squared_error = 1/len(y_true)*sum(squared_error).item(0)
40     return mean_squared_error
41
42 # Use the sklearn function 'LinearRegression' to fit on the training set
43 model = LinearRegression()
44 model.fit(x_train, y_train)
45 # Now predict on the test set
46 y_pred_test = model.predict(x_test)
47
48 # Now compute the MSE with the predicted values and print it
49 test_mse = mse(y_test, y_pred_test)
50 print(f'The test MSE is {test_mse}')
```

What does `.fit()` do ?



```
1 # import required libraries
2 import numpy as np
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 import csv
6 from sklearn.linear_model import LinearRegression
7
8 # Read the 'Advertising.csv' dataset
9 with open('Advertising.csv', mode='r') as infile:
10     reader = csv.reader(infile)
11     values = [(rows[1],rows[4]) for rows in reader]
12     data_dictionary = {i[0]:i[1] for i in list(zip(*values))}
13 # Assign TV advertising as predictor variable 'x' and sales as response variable 'y'
14 tv,sales = data_dictionary['TV'], data_dictionary['sales']
15 x,y = np.array(tv,dtype='float32').reshape(-1,1), np.array(sales,dtype='float32')
16
17 # Split the data into training and test sets
18 number_of_points = len(x)
19 train_size = 0.8
20 num_train_points = int(train_size*number_of_points)
21
22 # Create indices to split the dataset
23 train_index = np.random.choice(range(len(x)),size=num_train_points,replace=False)
24 test_index = [i for i in range(len(x)) if i not in train_index]
25 test_index = np.array(test_index)
26
27 # Create boolean masks for training and test sets
28 mask = np.zeros(len(x), dtype = 'int')
29 mask[train_index] = 1
30 mask = mask==1
31 # Use the masks to create train and test data
32 x_train,y_train = x[mask],y[mask]
33 x_test, y_test = x[~mask],y[~mask]
34
35 # Write a function to compute the mean squared error of the predictions
36 def mse(y_true, y_prediction):
37     error = y_true - y_prediction
38     squared_error = error**2
39     mean_squared_error = 1/len(y_true)*sum(squared_error).item(0)
40     return mean_squared_error
41
42
43 model = LinearRegression()
44 model.fit(x_train, y_train)
45 # Now predict on the test set
46 y_pred_test = model.predict(x_test)
47
48 # Now compute the MSE with the predicted values and print it
49 test_mse = mse(y_test, y_pred_test)
50 print(f'The test MSE is {test_mse}')
```

What does `.predict()` do ?



```
1 # import required libraries
2 import numpy as np
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 import csv
6 from sklearn.linear_model import LinearRegression
7
8 # Read the 'Advertising.csv' dataset
9 with open('Advertising.csv', mode='r') as infile:
10     reader = csv.reader(infile)
11     values = [(rows[1],rows[4]) for rows in reader]
12     data_dictionary = {i[0]:i[1] for i in list(zip(*values))}
13 # Assign TV advertising as predictor variable 'x' and sales as response variable 'y'
14 tv,sales = data_dictionary['TV'], data_dictionary['sales']
15 x,y = np.array(tv,dtype='float32').reshape(-1,1), np.array(sales,dtype='float32')
16
17 # Split the data into training and test sets
18 number_of_points = len(x)
19 train_size = 0.8
20 num_train_points = int(train_size*number_of_points)
21
22 # Create indices to split the dataset
23 train_index = np.random.choice(range(len(x)),size=num_train_points,replace=False)
24 test_index = [i for i in range(len(x)) if i not in train_index]
25 test_index = np.array(test_index)
26
27 # Create boolean masks for training and test sets
28 mask = np.zeros(len(x), dtype = 'int')
29 mask[train_index] = 1
30 mask = mask==1
31 # Use the masks to create train and test data
32 x_train,y_train = x[mask],y[mask]
33 x_test, y_test = x[~mask],y[~mask]
34
35 # Write a function to compute the mean squared error of the predictions
36 def mse(y_true, y_prediction):
37     error = y_true - y_prediction
38     squared_error = error**2
39     mean_squared_error = 1/len(y_true)*sum(squared_error).item(0)
40     return mean_squared_error
41
42 # Use the sklearn function 'LinearRegression' to fit on the training set
43 model = LinearRegression()
44 model.fit(x_train, y_train)
45 # Now predict on the test set
46 y_pred_test = model.predict(x_test)
47
48 # Now compute the MSE with the predicted values and print it
49 test_mse = mse(y_test, y_pred_test)
50 print(f'The test MSE is {test mse}')
```

```
1 # import required libraries
2 import numpy as np
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 import csv
6 from sklearn.linear_model import LinearRegression
7
8 # Read the 'Advertising.csv' dataset
9 with open('Advertising.csv', mode='r') as infile:
10     reader = csv.reader(infile)
11     values = [(rows[1],rows[4]) for rows in reader]
12     data_dictionary = {i[0]:i[1] for i in list(zip(*values))} #zip(*values) is used to transpose the list
13 # Assign TV advertising as predictor variable 'x' and sales as response variable 'y'
14 tv,sales = data_dictionary['TV'], data_dictionary['sales']
15 x,y = np.array(tv,dtype='float32').reshape(-1,1), np.array(sales,dtype='float32')
16
17 # Split the data into training and test sets
18 number_of_points = len(x)
19 train_size = 0.8
20 num_train_points = int(train_size*number_of_points)
21
22 # Create indices to split the dataset
23 train_index = np.random.choice(range(len(x)),size=num_train_points,replace=False)
24 test_index = [i for i in range(len(x)) if i not in train_index]
25 test_index = np.array(test_index)
26
27 # Create boolean masks for training and test sets
28 mask = np.zeros(len(x), dtype = 'int')
29 mask[train_index] = 1
30 mask = mask==1
31 # Use the masks to create train and test data
32 x_train,y_train = x[mask],y[mask]
33 x_test, y_test = x[~mask],y[~mask]
34
35 # Write a function to compute the mean squared error of the predictions
36 def mse(y_true, y_prediction):
37     error = y_true - y_prediction
38     squared_error = error**2
39     mean_squared_error = 1/len(y_true)*sum(squared_error).item(0)
40     return mean_squared_error
41
42 # Use the sklearn function 'LinearRegression' to fit on the training set
43 model = LinearRegression()
44 model.fit(x_train, y_train)
45 # Now predict on the test set
46 y_pred_test = model.predict(x_test)
47
48 # Now compute the MSE with the predicted values and print it
49 test_mse = mse(y_test, y_pred_test)
50 print(f'The test MSE is {test_mse}')
```

What is numpy ?

What is with ?

What is a variable ?

What is len() ?

What is for ?

What is train & test set ?

What is sum ?

What is import ?

What is zip ?

What is if ?

What is range() ?

What is item(0) ?

SESSION	TOPICS COVERED
1	Introduction to Python Programming
2	Python Data Structures
3	Python Functions
4	Python Classes
5	Files, Strings and Regular Expressions
6	Numerical Python (NumPy)
7	Linear Algebra
8	Introduction to Probability
9	Panel Data (Pandas)
10	Calculus & Linear Regression

MT. BIG DATA



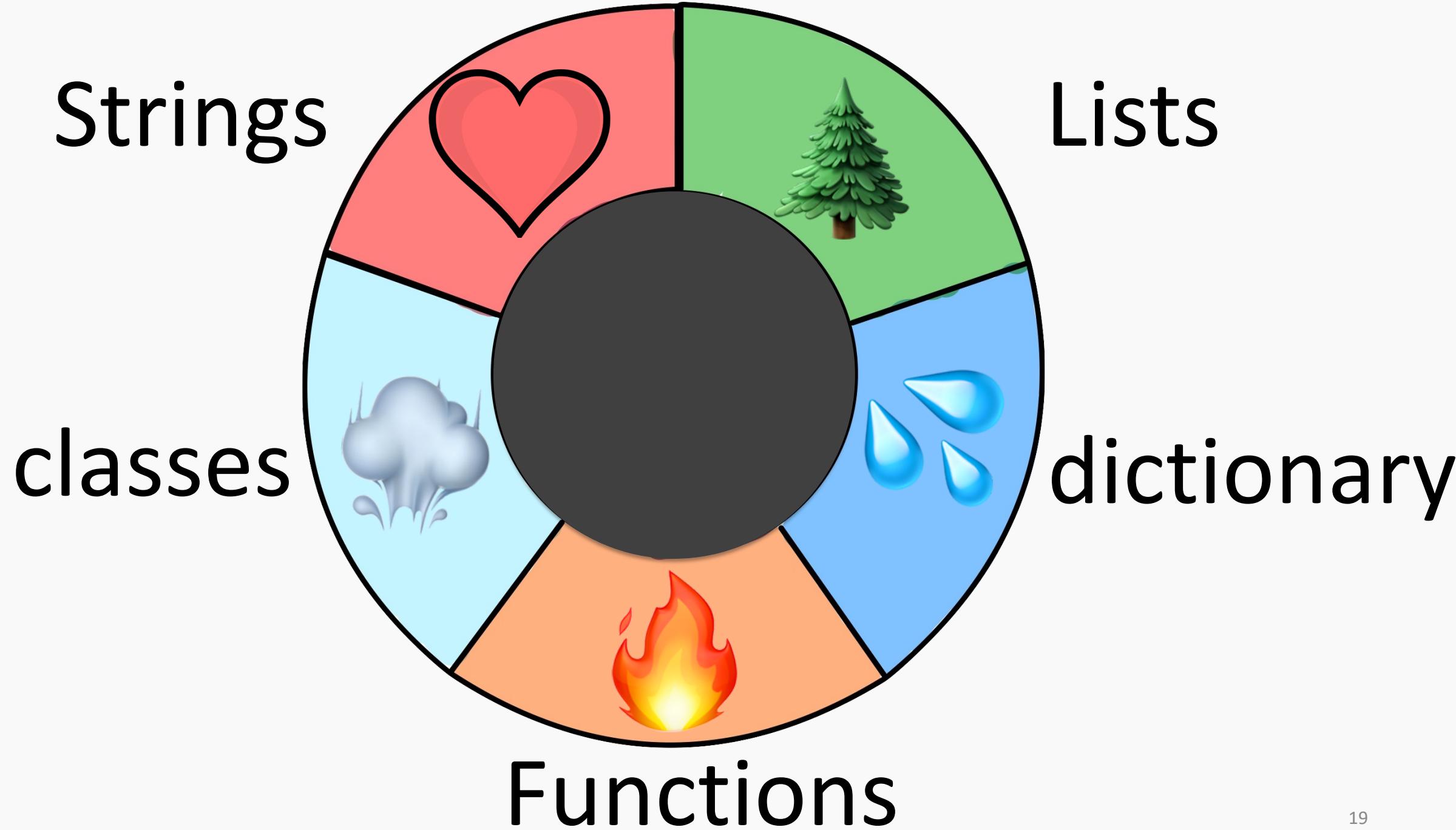
N DEEP WOODS  
OF PYTHON



OASIS  
OF CHANCE



LINEAR  
REGRESSION



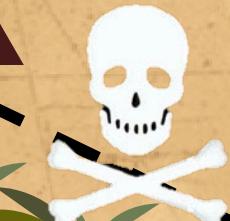
MT. BIG DATA



N DEEP WOODS  
OF PYTHON



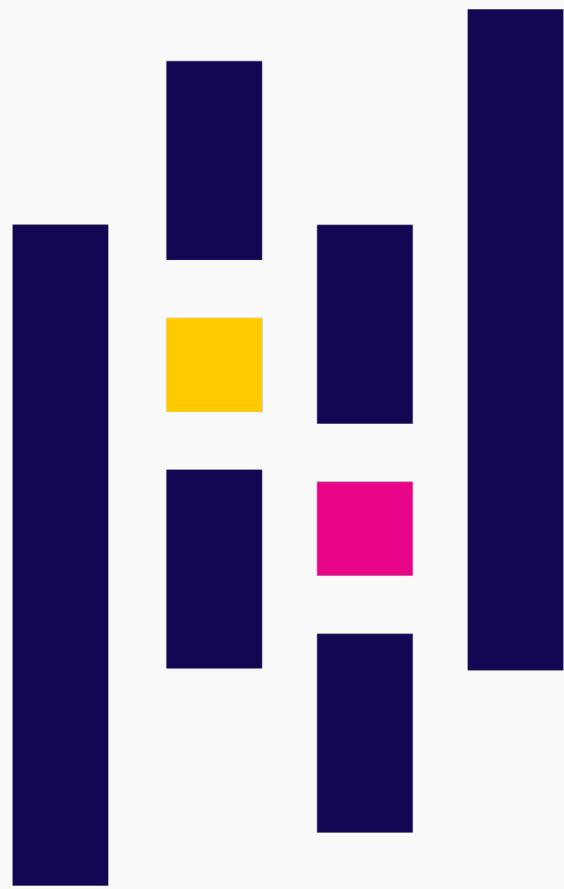
OASIS  
OF CHANCE



LINEAR  
REGRESSION



# pandas



# NumPy



MT. BIG DATA



N DEEP WOODS  
OF PYTHON



OASIS  
OF CHANCE



LINEAR  
REGRESSION





## Machine Le

Stanford University

Computer Science / Machine Learning ▾

 COLUMBIA UNIVERSITY  
IN THE CITY OF NEW YORK

CSCI E-109A



# Introduction to Data Science

Pavlos Protopapas, PhD

Scientific Program Director and Lecturer, Institute for Applied Computational Science, Harvard University

Kevin A. Rader, PhD

Senior Preceptor in Statistics, Harvard University

Christopher Tanner, PhD

Lecturer on Computational Science, Harvard University

ete

Write review

Fall Term 2020 | CRN 15178

# *Our Schedule*

*Our Expectation*



*Complete sessions*



*Complete session quiz and exercises*

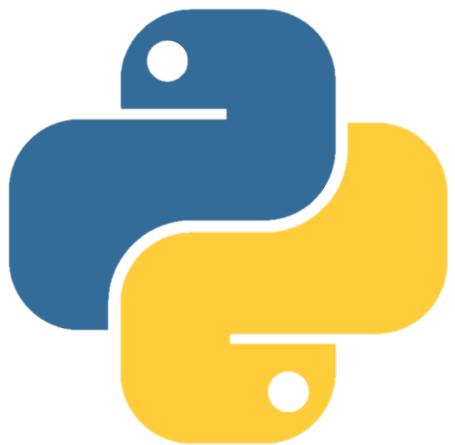


*Ask and answer questions on the forum*

# “Your Journey Begins Here!”



# Let's begin

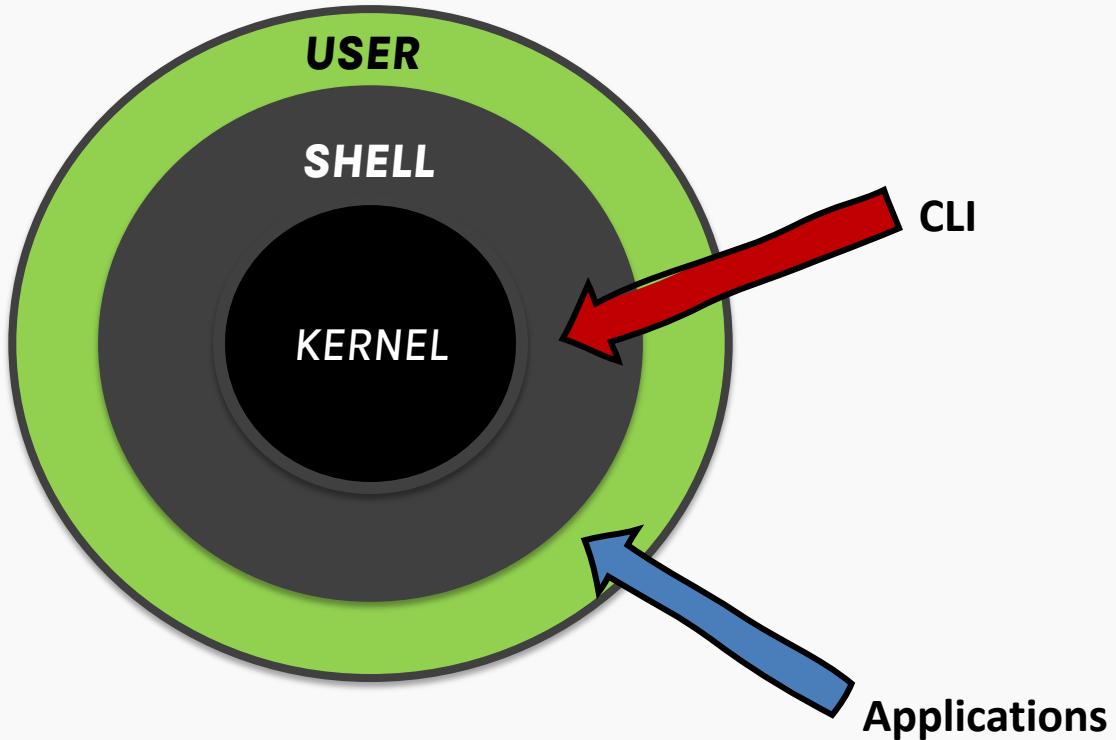


~~What is Python?~~

what is the  
*Terminal*?

# What is the terminal/Unix Shell ?

- A command line interface to a machine
- Controls the execution of the system



# *What is Python?*



## What is Python? Executive Summary

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

**Interpreted ?**

**Objected Oriented ?**

## **What is Python? Executive Summary**

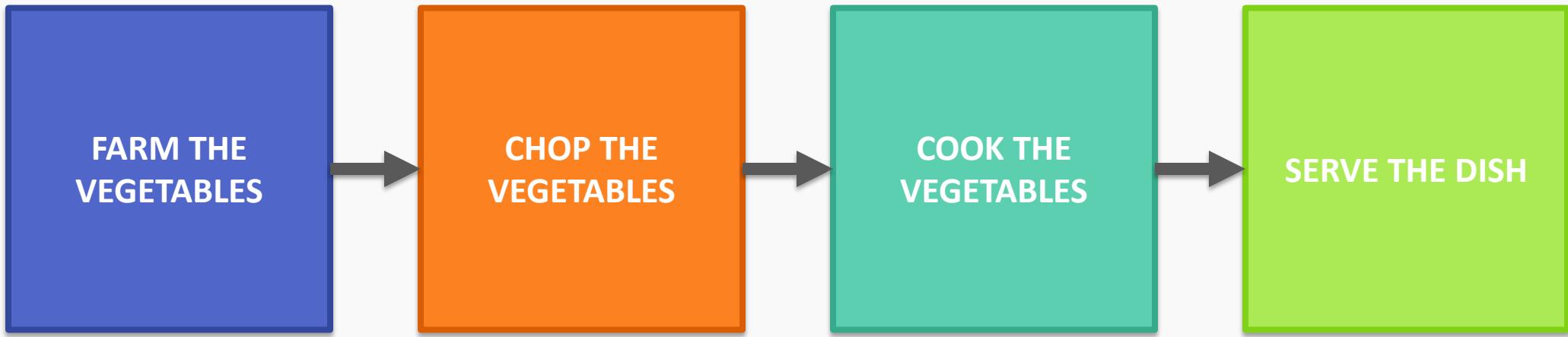
Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

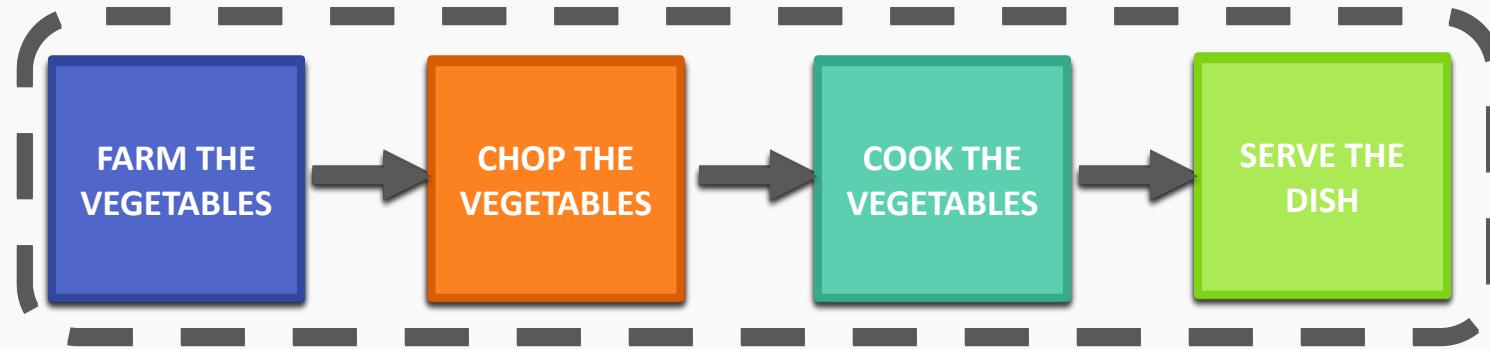
**High-Level ?**

*Let's break it down*

*Think of programming  
as preparing a meal*

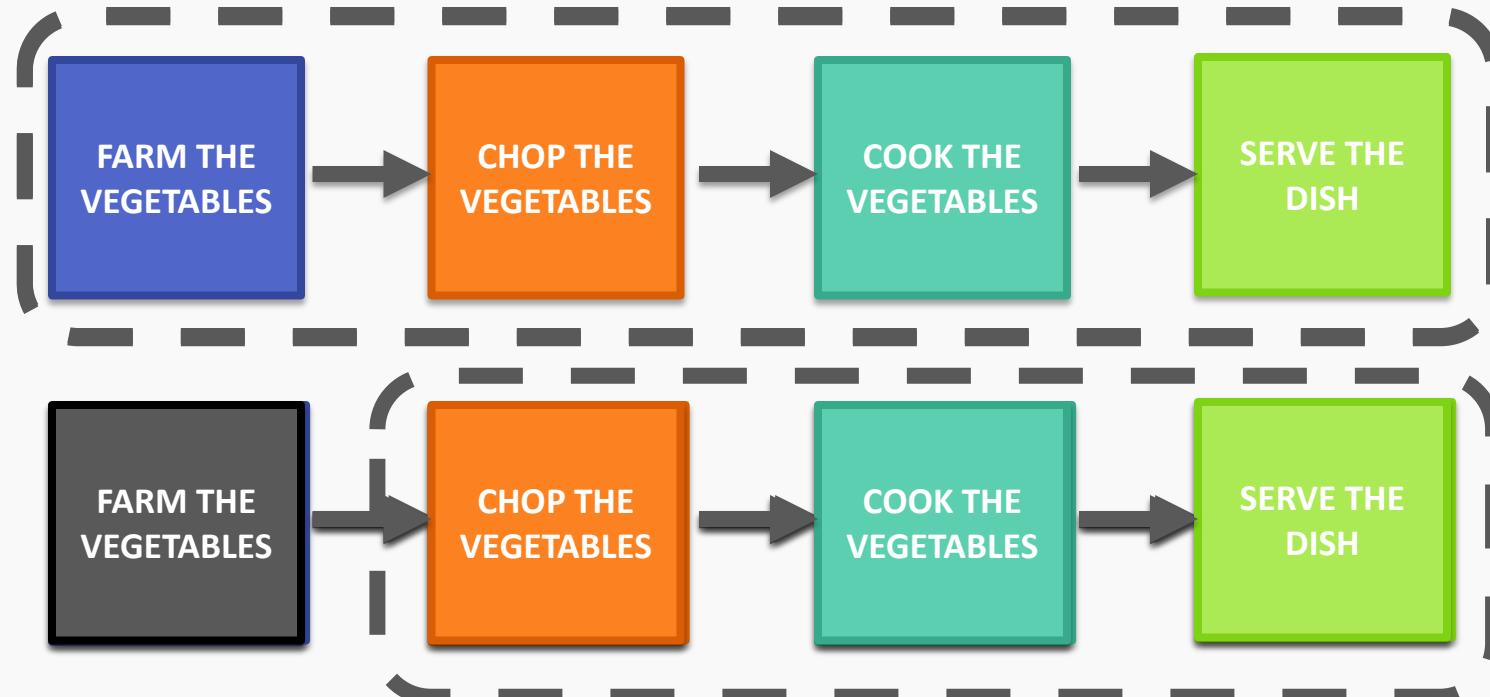






Assembly Level Code

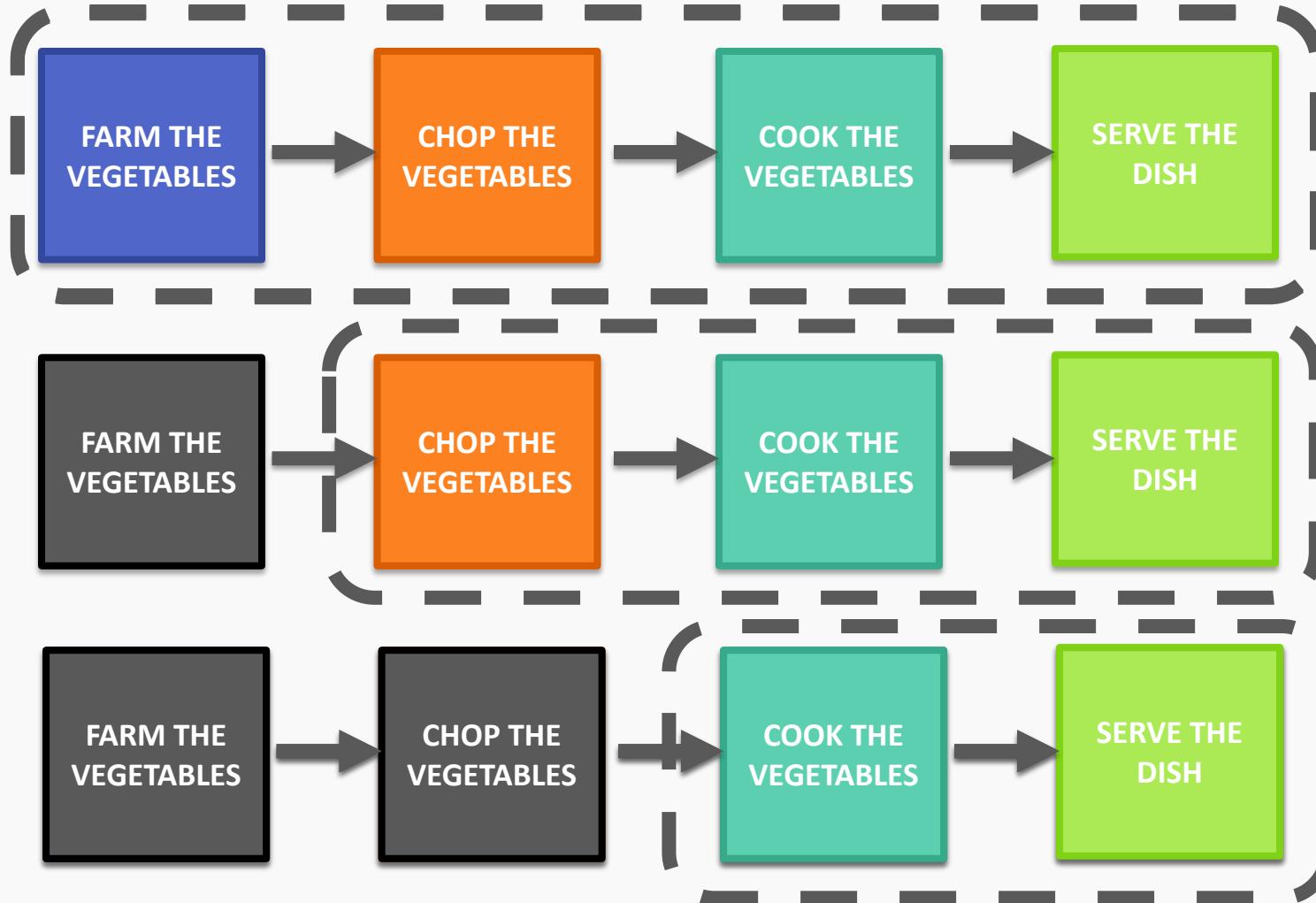




Assembly Level Code  
(Machine Level)

C programming  
(Low Level)



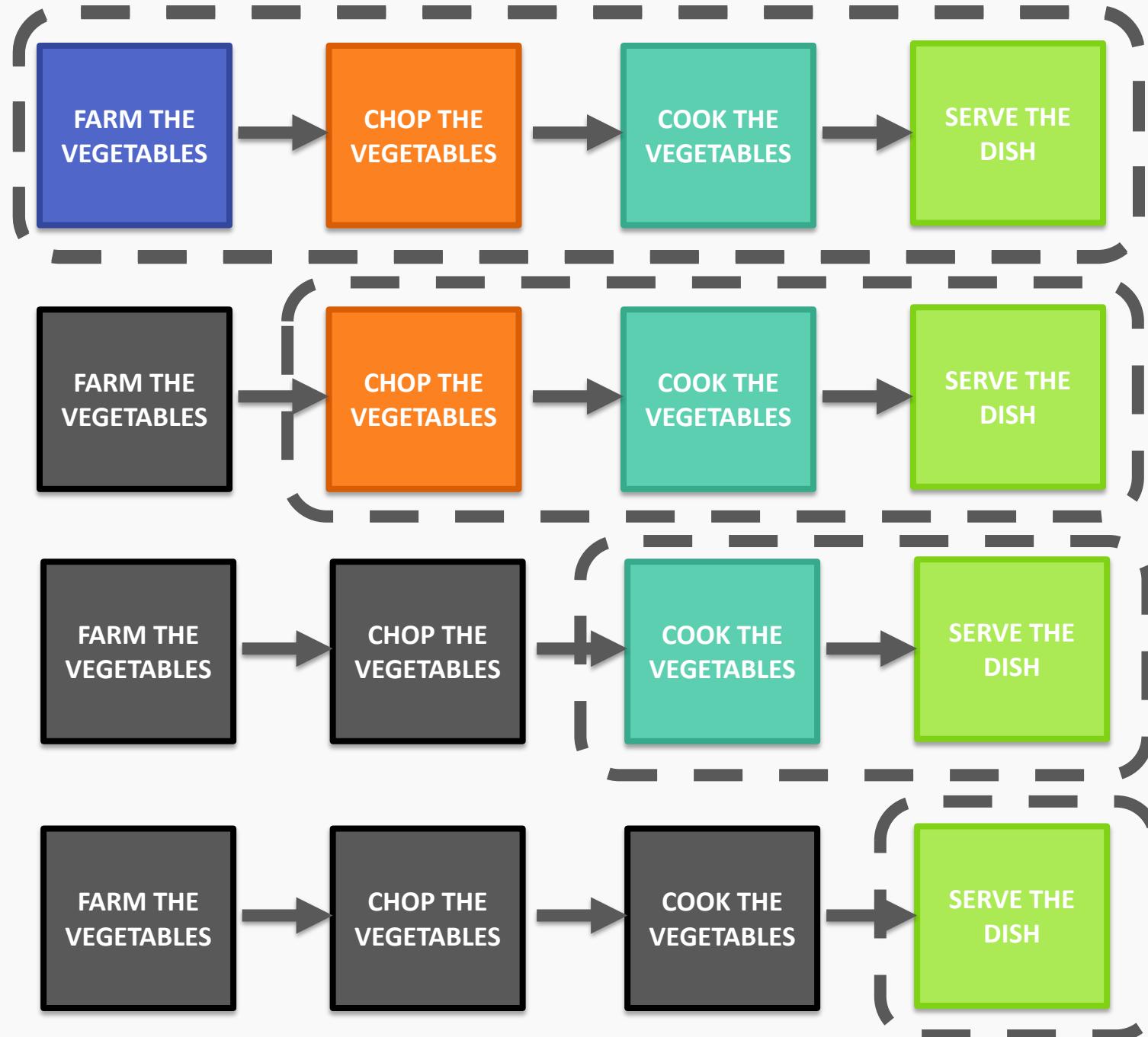


Assembly Level Code  
(Machine Level)

C programming  
(Low Level)

Python programming  
(High Level)





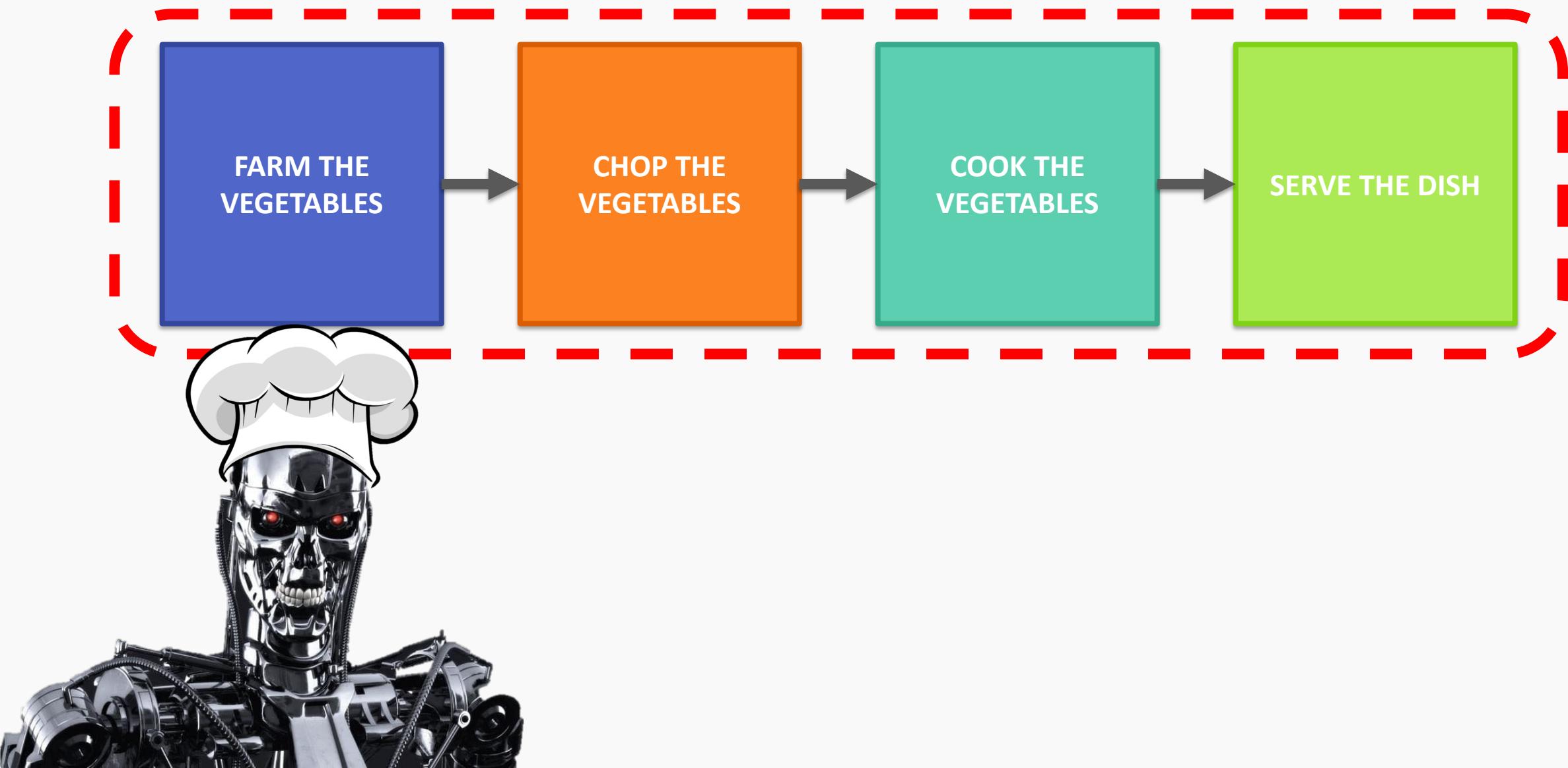
Assembly Level Code  
(Machine Level)

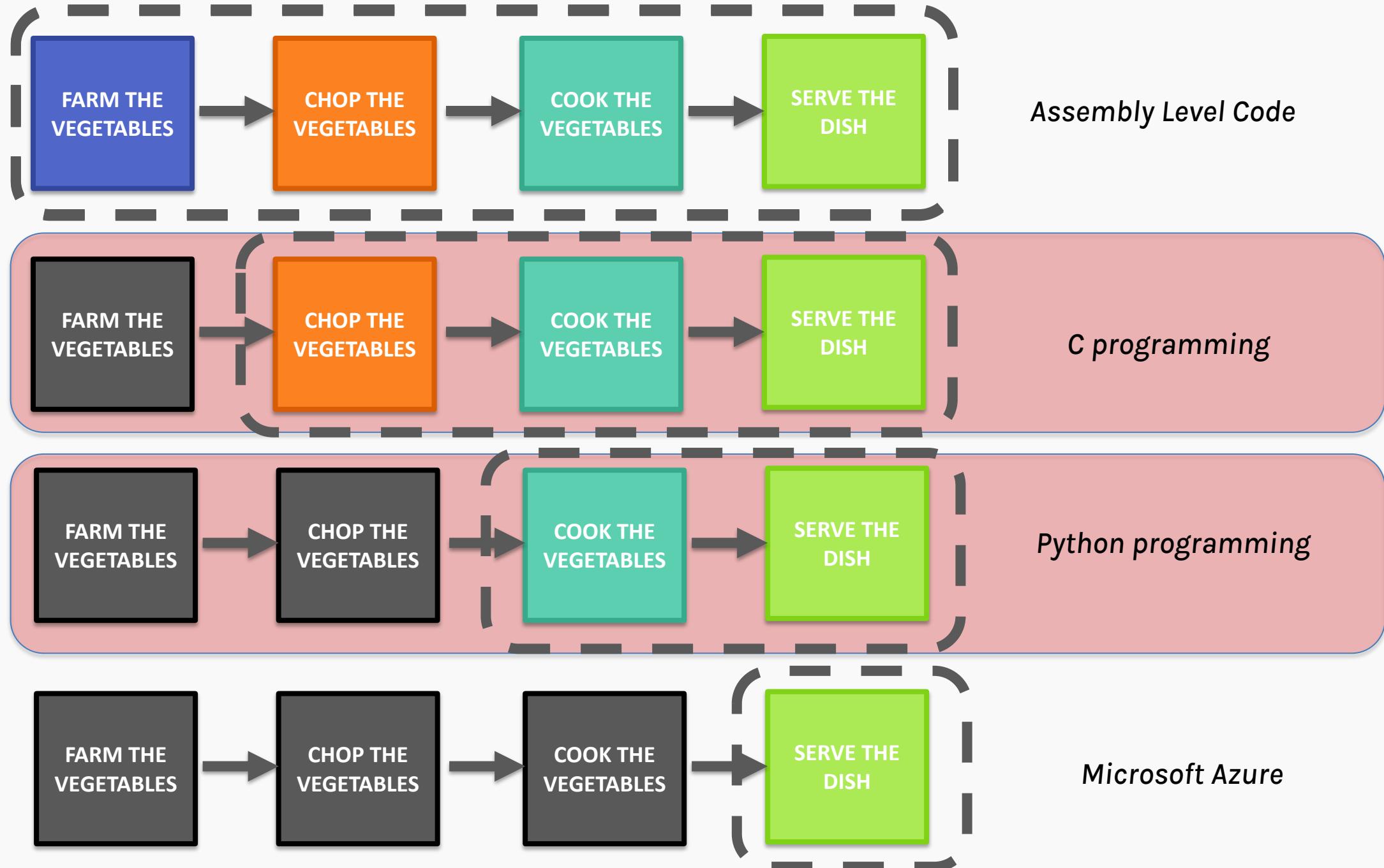
C programming  
(Low Level)

Python programming  
(High Level)

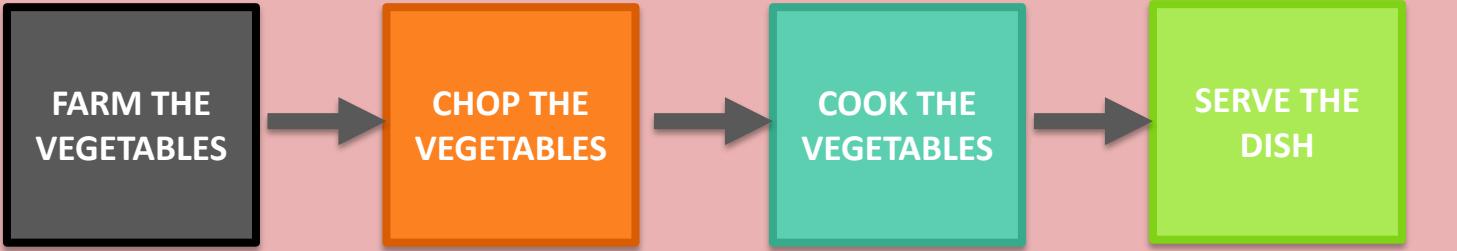
Microsoft Azure

# AI Apocalypse





# Cook your own meal



- Making a meal from scratch is not easy, but it gives you a lot of control.
- The C-programming language is as such. It is a **compiled** language (can be directly converted to machine code for execution).
- C is fast but requires low-level design choices (type casting, memory allocation)



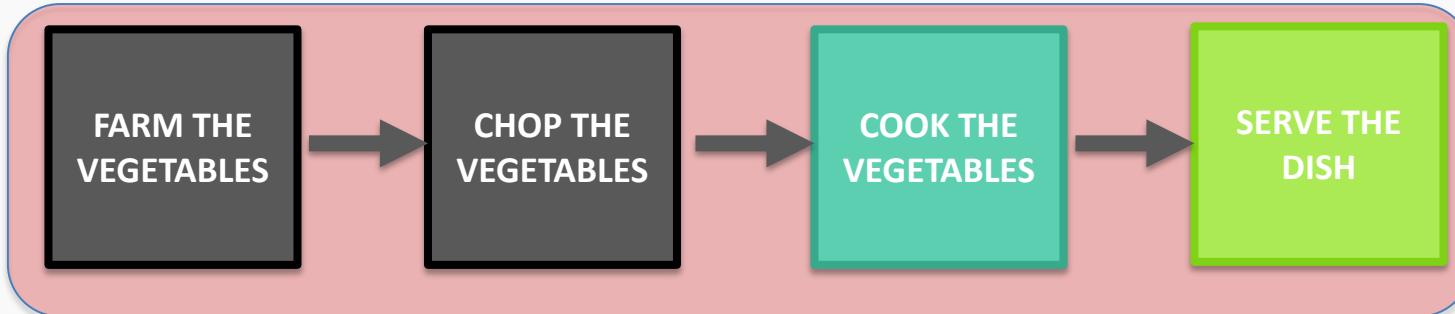
# C programming

## C ISSUES?

- C is still the language of choice for low-level applications such as embedded systems design.
- However, it is painstakingly verbose for data science applications.
- On the right is an example code to print a list of integers, needing several lines of code.

```
#include <stdio.h>
void main()
{
    int arr[10];
    int i;
    printf("\n\nRead and Print elements of an
array:\n");
    printf("-----\n");
    printf("Input 10 elements in the array :\n");
    for(i=0; i<10; i++)
    {
        printf("element - %d : ",i);
        scanf("%d", &arr[i]);
    }
    printf("\nElements in array are: ");
    for(i=0; i<10; i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");
```

# Meal-kits



- Imagine getting all ingredients pre-chopped and ready to be used.
- The Python programming language is as such. It is an **interpreted** language.
- This saves the user from taking all the technical design decisions (memory management, type-casting etc.)



# Python source code vs C source code

```
#include <stdio.h>
void main()
{
int arr[10];
int i;
printf("\n\nRead and Print elements of an
array:\n");
printf("-----\n");
printf("Input 10 elements in the array :\n");
for(i=0; i<10; i++)
{
printf("element - %d : ",i);
scanf("%d", &arr[i]);
}
printf("\nElements in array are: ");
for(i=0; i<10; i++)
{
printf("%d ", arr[i]);
}
printf("\n");
```

```
1 print("Input 10 elements in the
array")
2 num_elements = [int(input(f"Element
{i+1}: ")) for i in range(10)]
3 print(f"The elements in the array
are: {num_elements}")
```

# Python source code

## Python ISSUES?

- Python takes away the arduous steps of machine micro-management.
- However, all instructions must be provided before execution. (**ISSUE #1**)
- As Python handles the low-level decisions for you, there is a cost overhead for execution, making it slow.\* (**ISSUE #2**)

```
from pytube import YouTube  
Youtube(url).streams.first().download(location)
```

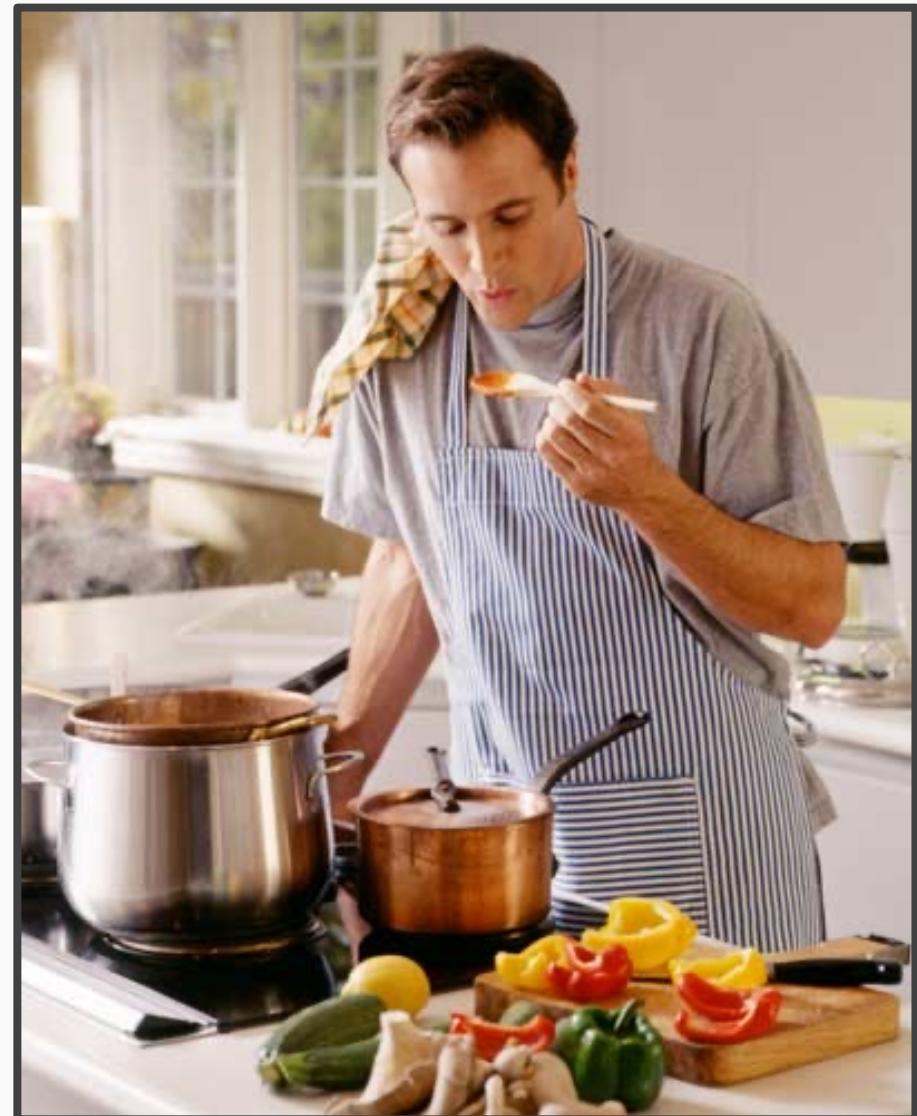


You just need two lines of code to download youtube videos using python

# IPython

---

- Any good chef knows the importance of tasting the meals while preparing a meal.
- The **Interactive Python** (IPython) project is as such. It is a **READ-EVAL-PRINT-LOOP** environment
- This is especially useful for experimentation.
- First introduced in 2001, IPython has become extremely popular among data scientists.



# IPython

## IPython ISSUES?

- IPython runs on the command line, it becomes easy to lose track of operations.
- It is also difficult to organize your code and share it with others.
- To overcome such issues, a web-app, called **Project Jupyter** was implemented to make it user-friendly & collaborative.

```
[ ]$ ipython
Python 3.8.3 (default, Jul  2 2020, 11:26:31)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.16.1 -- An enhanced Interactive Python
. Type '?' for help.

In [1]: num_elements = 5
[

In [2]: new_array = [i for i in range(num_e
[     ...: lements)]

In [3]: print(f'The elements in the array a
[     ...: re {new_array}')
The elements in the array are [0, 1, 2, 3, 4]

In [4]:
```

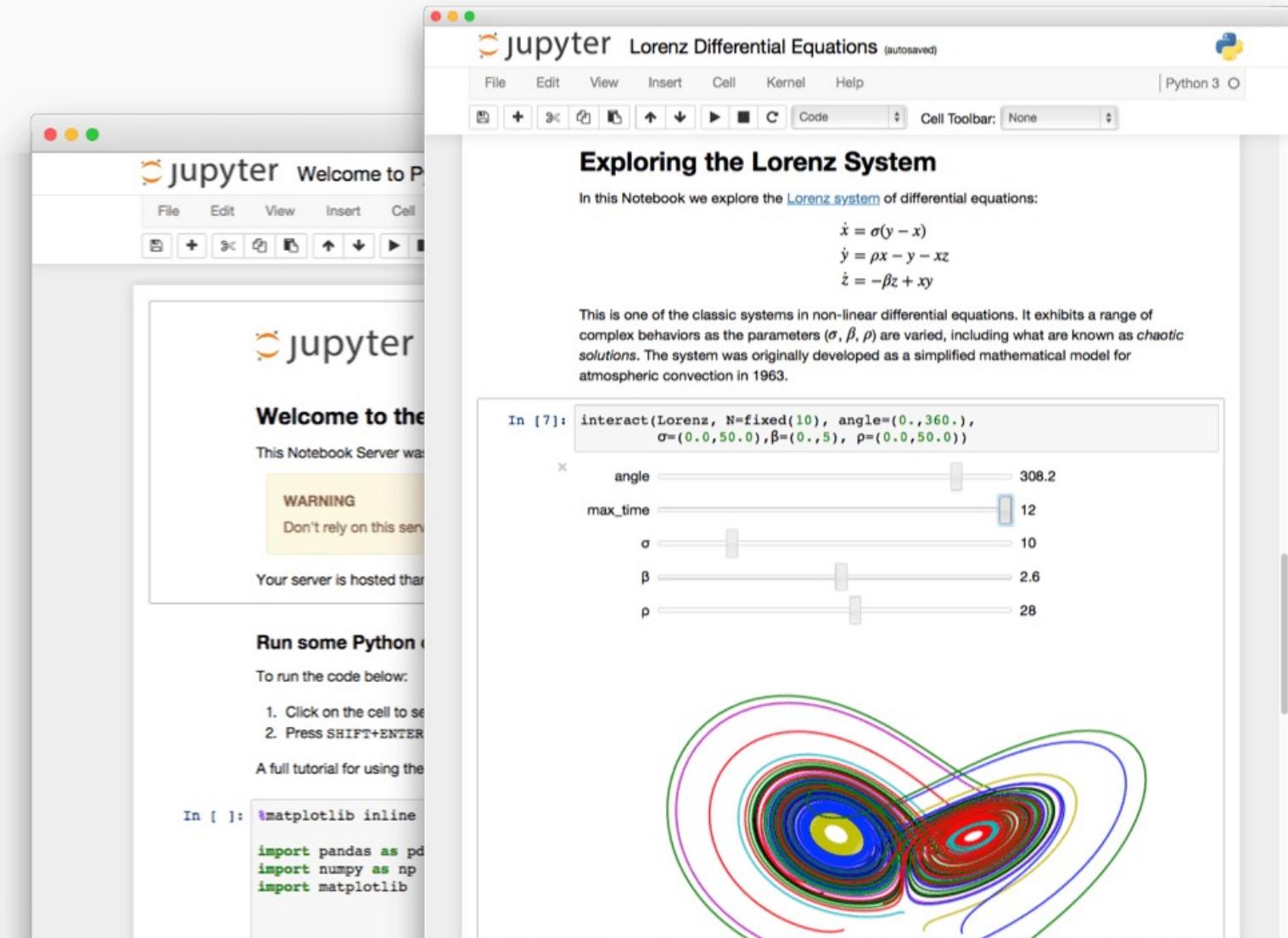
IPython will run on your terminal, unix shell, command prompt

# *Project Jupyter*



# Experimental Programming with Jupyter

- Project Jupyter offers a web-app interface where you can create notebooks that can your code **cell-by-cell**.
- This feature allows you to easily experiment with your code and make corrections as you go.
- Jupyter notebooks can be easily shared with others just like pdf files.



# Jupyter Lab interface

The screenshot displays the Jupyter Lab interface with the following components:

- File Browser:** On the left, a sidebar shows a file tree under "Session1 / linear-regression /". The files listed are:
  - Advertising.csv (3 days ago)
  - coors\_new.csv (2 days ago)
  - linear\_regression.py (4 hours ago)
  - lr\_model.png (2 days ago)
  - s1-modified.ipynb (a day ago)
  - s1-exc3-solution.ipynb (2 days ago) - This file is currently selected.
- Code Editor:** The main area contains the content of the selected notebook, s1-exc3-solution.ipynb. The code is divided into cells, each containing a numerical index and a Python code snippet. The visible cells are:
  - [2]:

```
# import required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import csv
from sklearn.linear_model import LinearRegression
%matplotlib inline
```
  - [3]:

```
# Read the 'Advertising.csv' dataset
with open('Advertising.csv', mode='r') as infile:
    reader = csv.reader(infile)
    values = [rows[1],rows[4]] for rows in reader]
data_dictionary = {i[0]:i[1:] for i in list(zip(*values))}
```
  - [243]:

```
# Assign TV advertising as predictor variable 'x' and
tv,sales = data_dictionary['TV'], data_dictionary['sal']
x,y = np.array(tv,dtype='float32').reshape(-1,1), np.array(sales,dtype='float32')
```
  - [244]:

```
# Split the data into training and test sets
number_of_points = len(x)
train_size = 0.8
num_train_points = int(train_size*number_of_points)
```
  - [245]:

```
# Create indices to split the dataset
train_index = np.random.choice(range(len(x)),size=num_train_points)
test_index = [i for i in range(len(x)) if i not in train_index]
test_index = np.array(test_index)
```
  - [246]:

```
# Create boolean masks for training and test sets
mask = np.zeros(len(x), dtype = 'int')
mask[train_index] = 1
mask = mask==1
```
- Terminal:** On the right, a terminal window titled "s1-exc3-solution.ipynb" shows the Python environment setup. It includes the Python version, copyright information, and a help message. Below this, two command-line inputs are shown:
  - [5]: data\_dictionary.keys()
  - [5]: dict\_keys(['TV', 'sales'])

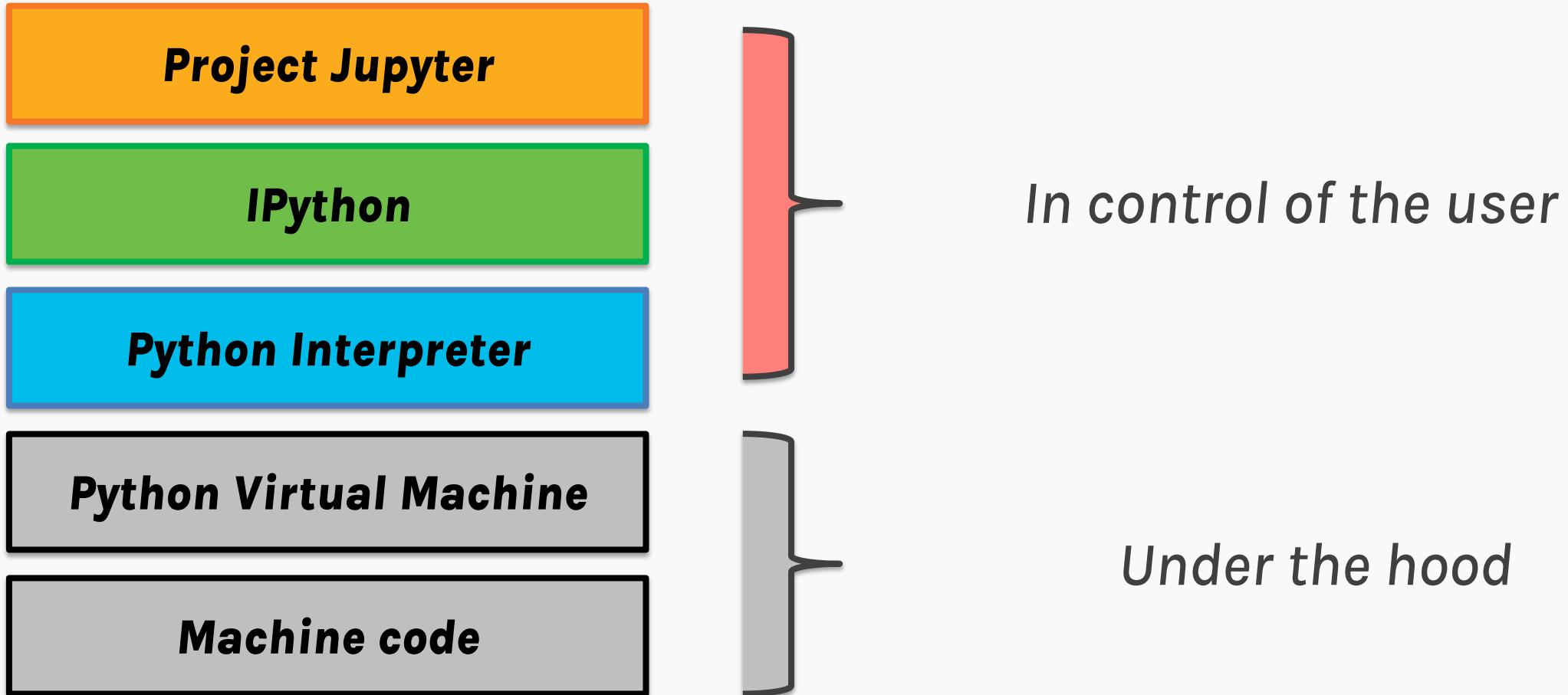
# Google Colaboratory

Google's custom implementation of Project Jupyter

The screenshot shows the Google Colaboratory interface. At the top, there's a navigation bar with the 'Welcome To Colaboratory' logo, 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help' menus, and a user profile icon. Below the navigation bar is a toolbar with 'Share', 'Settings', and a user profile icon. On the left, there's a sidebar titled 'Table of contents' with sections like 'Getting started', 'Data science', 'Machine learning', 'More Resources', 'Machine Learning Examples', and a 'Section' button. The main content area features a large heading 'What is Colaboratory?' with a sub-section 'Getting started'. It describes Colab as an environment for writing and executing Python in a browser, highlighting features like zero configuration, free access to GPUs, and easy sharing. It also encourages users to watch an introduction video. At the bottom, there's a code cell example:

```
[ ] seconds_in_a_day = 24 * 60 * 60  
seconds_in_a_day
```

# *Bringing it all together*



# My Specials



**BEDROCK**

Code first  
introduction



**CS109A CS109B**

Machine  
learning

Computer  
Vision

Natural  
Language  
Processing

Generative  
Models

**AC215**

Application  
Development  
& Deployment

The key concepts of programming, math and statistics that you learn in BRDS will be relevant throughout your journey as a Data Scientist



# Programming on your own system

- There are several ways to get project Jupyter on your system, (e.g. **Anaconda**)
- However, beginners generally hesitate to use local environments.
- We will be using **Ed-Stem** instead so that you can learn to program without the fear of breaking something.



*What is EdStem?*

# EdStem: An integrated platform

The screenshot displays the EdStem interface for a lesson titled "Session 1: Introduction to Python Programming". The left sidebar lists various resources, with "Exercise: Python Savings Calculator" highlighted. The main content area shows the exercise details and a code editor.

**Exercise: Python Savings Calculator (Hidden)**

**Description**

## Exercise: Python Savings Calculator

The goal of the exercise is to build a simple savings calculator using python.

**Instructions:**

- Follow along the notebook to fill in the gaps and build a simple savings calculator.
- Drawing upon the Tutorial in Session 0 , combine different arithmetic operators.
- Print the results using variables, operators and python conditional statements.

**Hints:**

**Formulae for Simple Interest:**

$$S.I. = P \times R \times T$$

**Code Editor:**

```
[ ] # Suppose you have $1000 leftover from your first paycheck  
# Create a variable 'savings' to reflect the same  
# (e.g. savings = 1000)  
# Your code here  
savings = 1000
```

**Question:** What is the type of the variable `savings`?

A. float  
B. str  
C. int  
D. tuple

```
[ ] ### edTest(test_chow1) ###
```

ed PyDS - Lessons

Lessons Slides Prev Next

Exercise: Hello World! (Hidden)

Challenge Submissions Solution (hidden) Edit Slide ..

Session 0: Course Objective (Not graded)

- Story Board: Welcome ✓
- Pre-Course Survey
- Course Objective
- Exercise: Linear Regression using scikit-learn
- Pre-Class Assignment ✓
- Pre-Class Quiz
- Course Overview ✓
- Exercise: Hello World! ✓
- Exercise: Hello Ed!
- Tutorial: Introduction to Python
- Post-Class Quiz

Description

## Exercise: Hello World!

The goal of this exercise is to use the python interpreter to print 'Hello, World' to the console.

> HELLO, WORLD!

### Instructions

1. Complete the source code in the `hello.py` file.
2. Activate the terminal at the bottom, and run the program with the command `python hello.py`
3. You will then type `ipython` on the terminal to launch the interactive python console, and again print `Hello, World!`

Read more about IPython [here] (<https://ipython.org/>)

hello.py

```
1 print(___)
```

/home/hello.py Spaces: 4 (Auto) All changes saved

Terminal

```
[user@sahara ~]$
```

✓ Mark Reset

This screenshot shows a user interface for a Python Data Science course. The left sidebar lists various course components: Session 0, Story Board, Pre-Course Survey, Course Objective, Exercise: Linear Regression, Pre-Class Assignment, Pre-Class Quiz, Course Overview, Exercise: Hello World! (selected), Exercise: Hello Ed!, Tutorial: Introduction to Python, and Post-Class Quiz. Most items have a green checkmark indicating completion. The main panel is titled 'Exercise: Hello World! (Hidden)' and contains a 'Description' section with instructions and a large green button labeled '> HELLO, WORLD!'. Below this is an 'Instructions' section with three numbered steps. Step 1 asks to complete the source code in 'hello.py'. Step 2 asks to run the program in a terminal. Step 3 asks to use IPython to print 'Hello, World!'. A note at the bottom suggests reading more about IPython. On the right, there's a code editor with 'hello.py' containing the line 'print(\_\_\_)', a terminal window showing a prompt, and a status bar indicating the file is saved. The top navigation bar includes links for Lessons, Slides, Prev, Next, Challenge, Submissions, Solution (hidden), Edit Slide, and other course-related icons.

# Discussion forums for students

ed Bedrock Data Science – Discussion

New Thread

COURSES +

- Bedrock Data Science
  - CS 109A 707
  - CS 109a (2020) 1008
  - ComputeFest2022
  - GEC Class 3
  - 22 more

CATEGORIES

- General
- Lectures
- Sections
- Problem Sets
- Assignments
- Social

14 others online

Search Filter

Welcome!

General Pavlos Protopapas STAFF 16m 1

Video Links

General Pavlos Protopapas STAFF 2d

This Week

Math Anxiety

General Anonymous 11h 2 1

Schedules and Lectures

General Edij 2d 4

Math Anxiety #3

Anonymous 11 hours ago in General

Hi Folks,

I am very excited about this course, but I haven't had maths since high school. Any advice on tackling my math anxiety?

Comment Edit Delete Endorse ...

PIN STAR WATCH VIEWS

1 Answer

Chris Gumb STAFF 9 hours ago

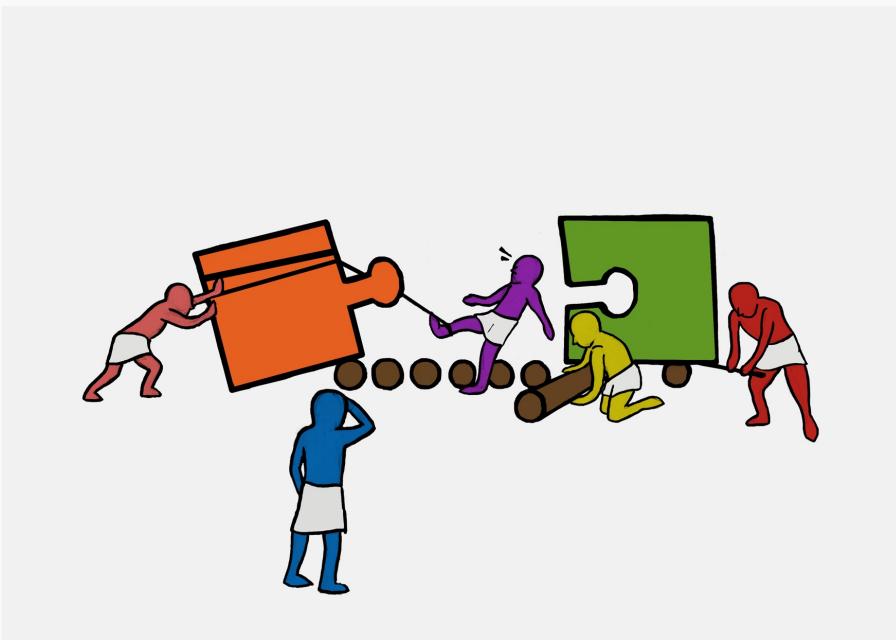
Hi there!

We've created a little section on the website with some math resources to help people prepare:  
<https://www.computefest.seas.harvard.edu/background>

We understand that math anxiety is real and want everyone to feel comfortable and confident so they can get the most out of Bedrock Data Science.

# Exercise

## > Hello, World!\_



ed PyDS - Lessons

Lessons Slides Prev Next

Exercise: Hello World! (Hidden)

Challenge

Session 0: Course Objective (Not graded)

- Story Board: Welcome ✓
- Pre-Course Survey
- Course Objective ✓
- Exercise: Linear Regression using scikit-learn
- Pre-Class Assignment ✓
- Pre-Class Quiz
- Course Overview ✓
- Exercise: Hello-World! (selected)
- Exercise: Hello-Edit
- Tutorial: Introduction to Python
- Post-Class Quiz

Description

Exercise: Hello World!

The goal of this exercise is to use the python interpreter to print 'Hello, World' to the console.

> HELLO, WORLD!\_

Instructions

1. Complete the source code in the `hello.py` file.
2. Activate the terminal at the bottom, and run the program with the command `python hello.py`
3. You will then type `ipython` on the terminal to launch the interactive python console, and again print `Hello, World!`

Read more about IPython [here] (<https://ipython.org/>)

/home/hello.py Spaces: 4 (Auto)

Terminal

```
[user@sahara ~]$
```

# ANY QUESTIONS ?

