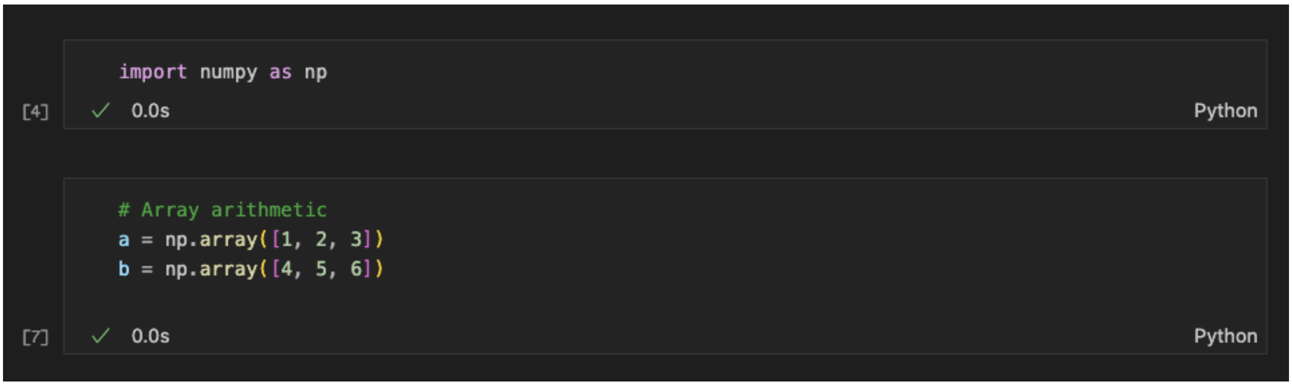
Assignment 1

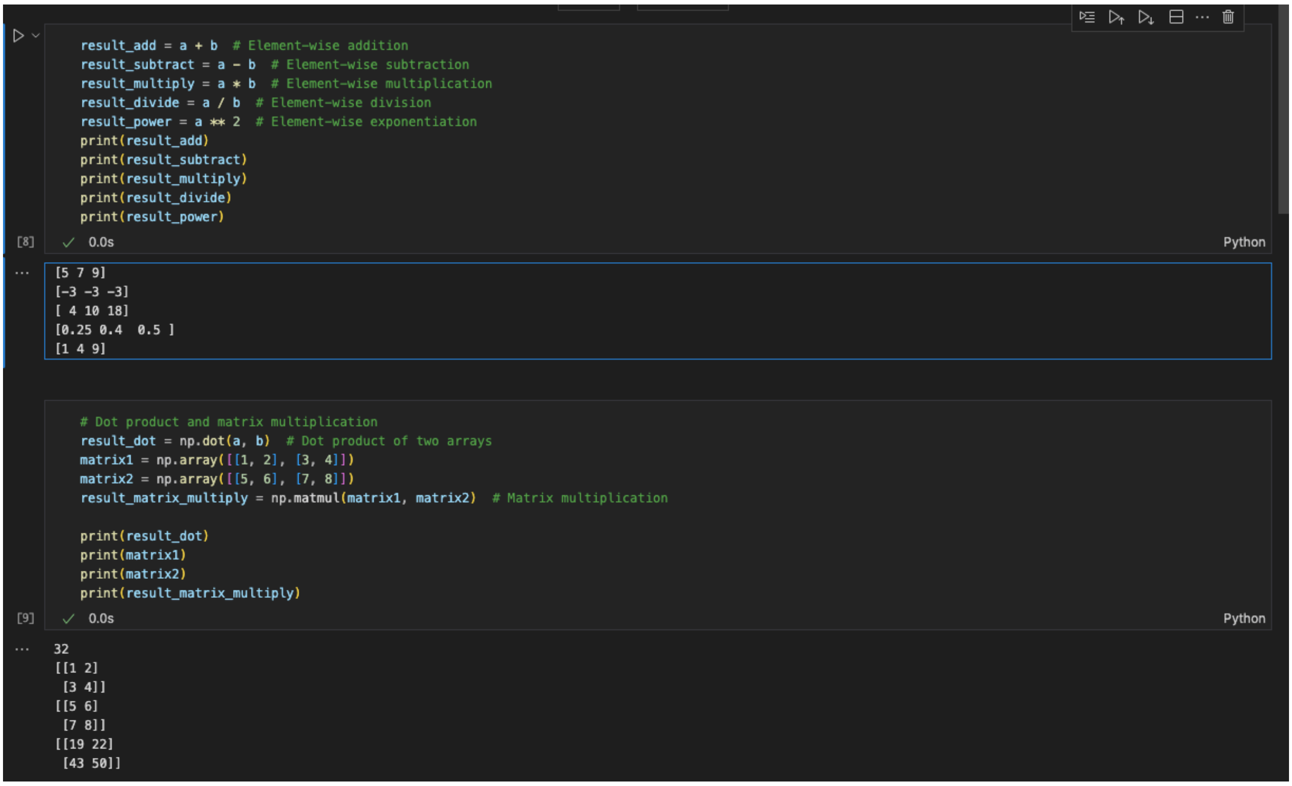
Garvit Shah
U21CS089

NumPy



```python
import numpy as np
```

```python
# Array arithmetic
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
```

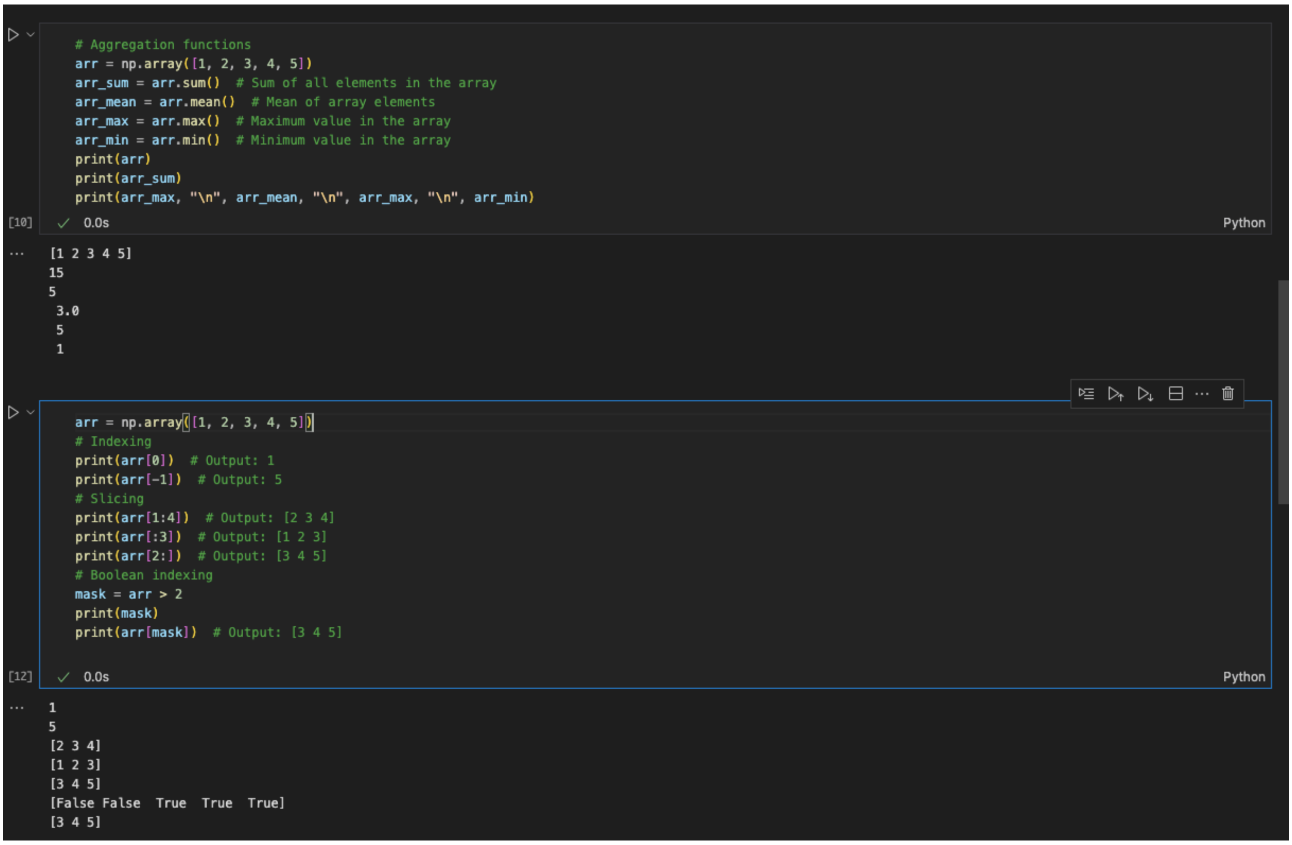


```python
result_add = a + b  # Element-wise addition
result_subtract = a - b  # Element-wise subtraction
result_multiply = a * b  # Element-wise multiplication
result_divide = a / b  # Element-wise division
result_power = a ** 2  # Element-wise exponentiation
print(result_add)
print(result_subtract)
print(result_multiply)
print(result_divide)
print(result_power)
```

```
[5 7 9]
[-3 -3 -3]
[ 4 10 18]
[0.25 0.4  0.5 ]
[1 4 9]
```

```python
# Dot product and matrix multiplication
result_dot = np.dot(a, b)  # Dot product of two arrays
matrix1 = np.array([[1, 2], [3, 4]])
matrix2 = np.array([[5, 6], [7, 8]])
result_matrix_multiply = np.matmul(matrix1, matrix2)  # Matrix multiplication

print(result_dot)
print(matrix1)
print(matrix2)
print(result_matrix_multiply)
```

```
32
[[1 2]
 [3 4]]
[[5 6]
 [7 8]]
[[19 22]
 [43 50]]
```

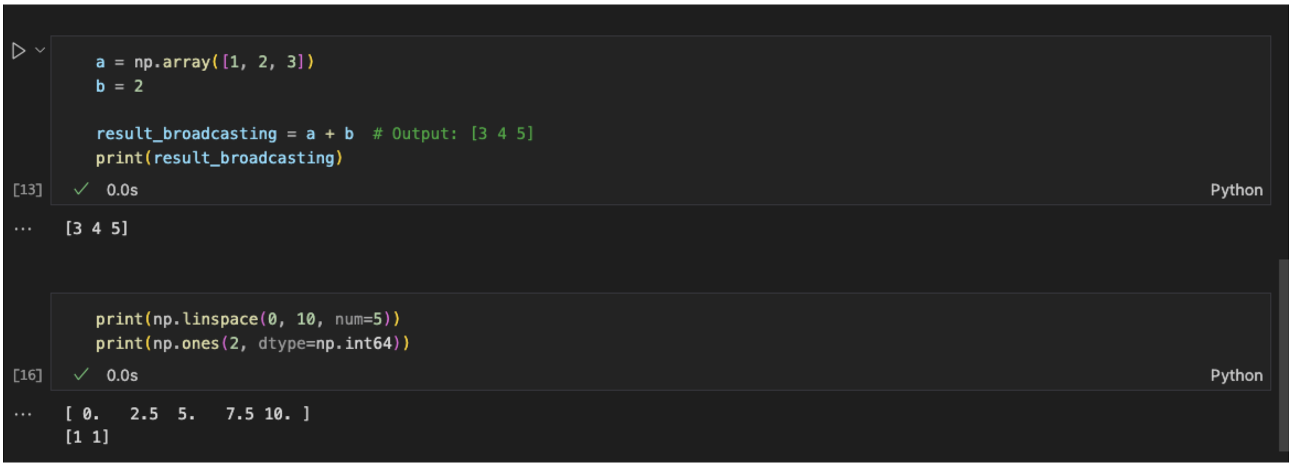


```python
# Aggregation functions
arr = np.array([1, 2, 3, 4, 5])
arr_sum = arr.sum()  # Sum of all elements in the array
arr_mean = arr.mean()  # Mean of array elements
arr_max = arr.max()  # Maximum value in the array
arr_min = arr.min()  # Minimum value in the array
print(arr)
print(arr_sum)
print(arr_max, "\n", arr_mean, "\n", arr_max, "\n", arr_min)
```

```
[1 2 3 4 5]
15
5
 3.0
 5
 1
```

```python
arr = np.array([1, 2, 3, 4, 5])
# Indexing
print(arr[0])  # Output: 1
print(arr[-1])  # Output: 5
# Slicing
print(arr[1:4])  # Output: [2 3 4]
print(arr[:3])  # Output: [1 2 3]
print(arr[2:])  # Output: [3 4 5]
# Boolean indexing
mask = arr > 2
print(mask)
print(arr[mask])  # Output: [3 4 5]
```

```
1
5
[2 3 4]
[1 2 3]
[3 4 5]
[False False  True  True  True]
[3 4 5]
```

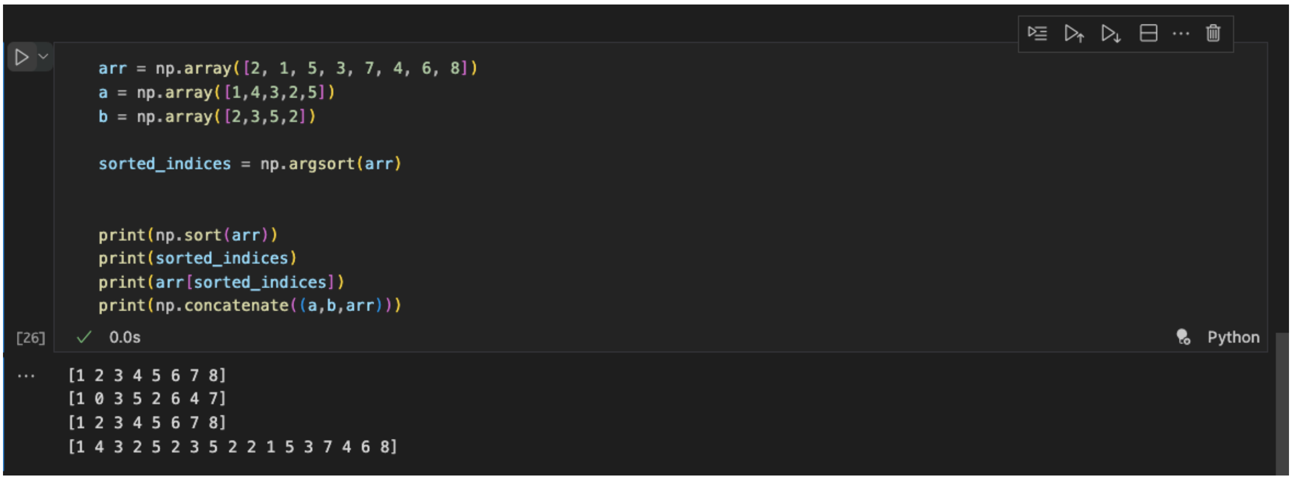


```python
a = np.array([1, 2, 3])
b = 2

result_broadcasting = a + b  # Output: [3 4 5]
print(result_broadcasting)
```

```
[3 4 5]
```

```python
print(np.linspace(0, 10, num=5))
print(np.ones(2, dtype=np.int64))
```

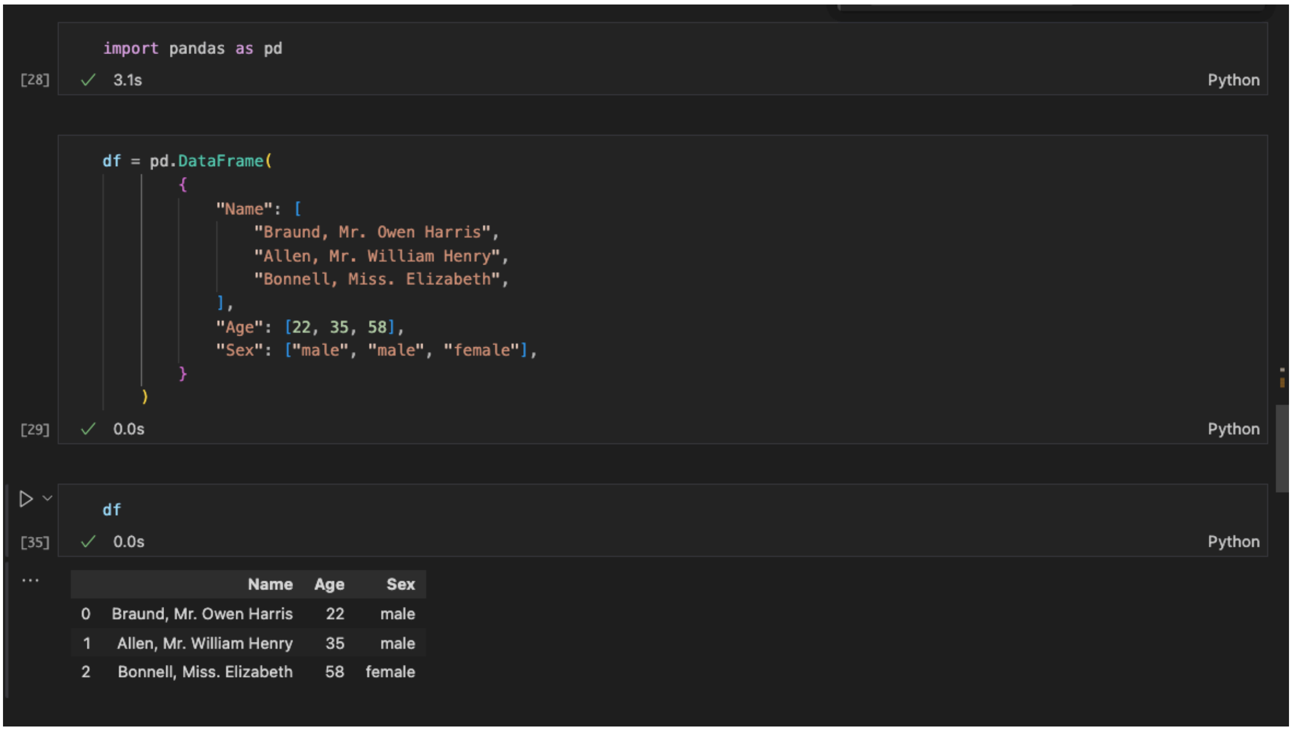
```
[ 0.   2.5  5.   7.5 10. ]
[1 1]
```



```python
arr = np.array([2, 1, 5, 3, 7, 4, 6, 8])
a = np.array([1,4,3,2,5])
b = np.array([2,3,5,2])

sorted_indices = np.argsort(arr)


print(np.sort(arr))
print(sorted_indices)
print(arr[sorted_indices])
print(np.concatenate((a,b,arr)))
```

```
[1 2 3 4 5 6 7 8]
[1 0 3 5 2 6 4 7]
[1 2 3 4 5 6 7 8]
[1 4 3 2 5 2 3 5 2 2 1 5 3 7 4 6 8]
```

Pandas

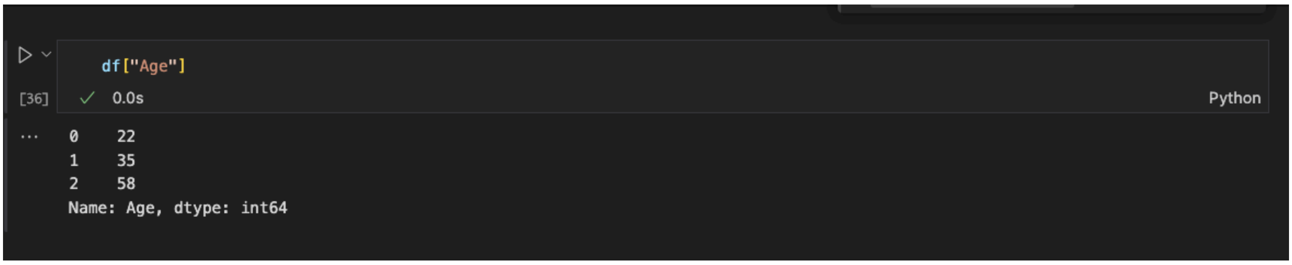


```python
import pandas as pd
```

```python
df = pd.DataFrame(
    {
        "Name": [
            "Braund, Mr. Owen Harris",
            "Allen, Mr. William Henry",
            "Bonnell, Miss. Elizabeth",
        ],
        "Age": [22, 35, 58],
        "Sex": ["male", "male", "female"],
    }
)
```

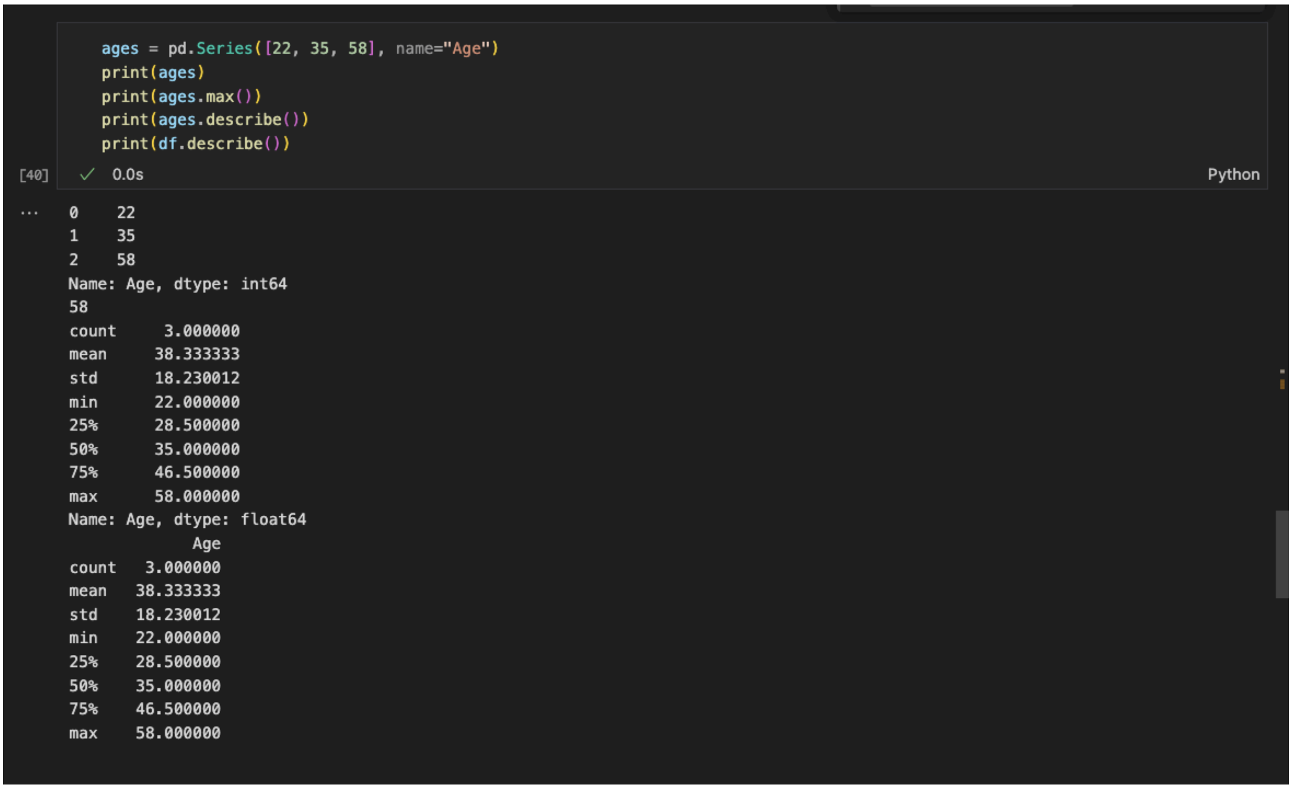
```python
df
```

	Name	Age	Sex
0	Braund, Mr. Owen Harris	22	male
1	Allen, Mr. William Henry	35	male
2	Bonnell, Miss. Elizabeth	58	female



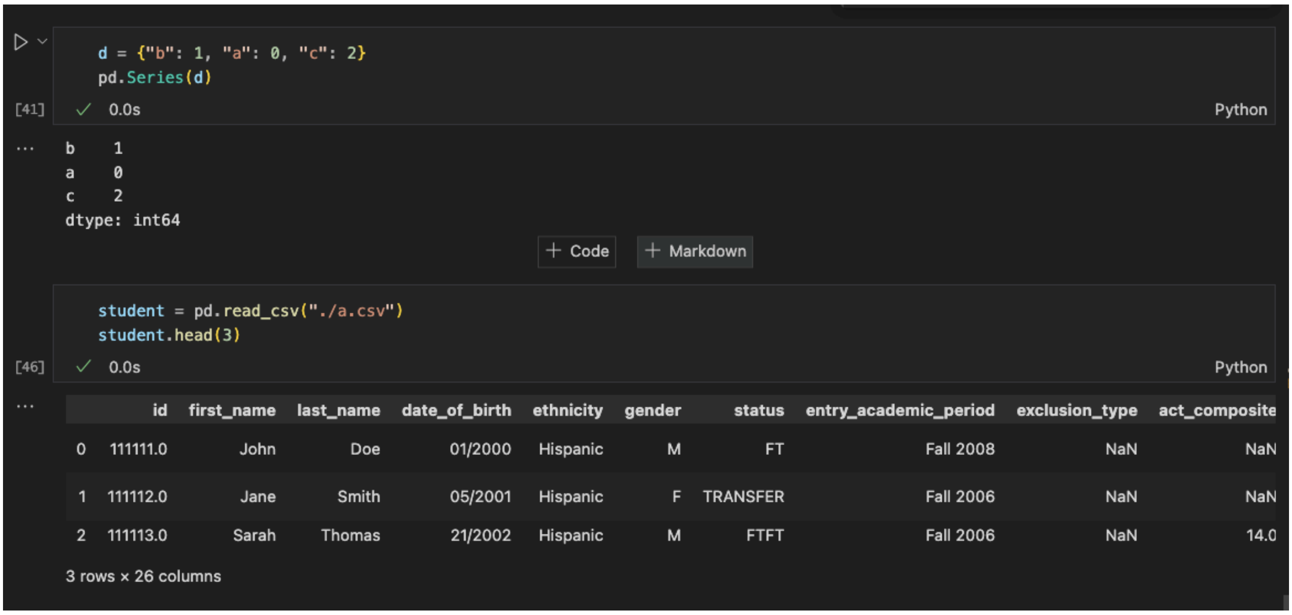
```python
df["Age"]
```

```
0    22
1    35
2    58
Name: Age, dtype: int64
```



```python
ages = pd.Series([22, 35, 58], name="Age")
print(ages)
print(ages.max())
print(ages.describe())
print(df.describe())
```

```
0    22
1    35
2    58
Name: Age, dtype: int64
58
count     3.000000
mean     38.333333
std      18.230012
min      22.000000
25%      28.500000
50%      35.000000
75%      46.500000
max      58.000000
Name: Age, dtype: float64
             Age
count   3.000000
mean   38.333333
std    18.230012
min    22.000000
25%    28.500000
50%    35.000000
75%    46.500000
max    58.000000
```



```python
d = {"b": 1, "a": 0, "c": 2}
pd.Series(d)
```

[41] ✓ 0.0s Python

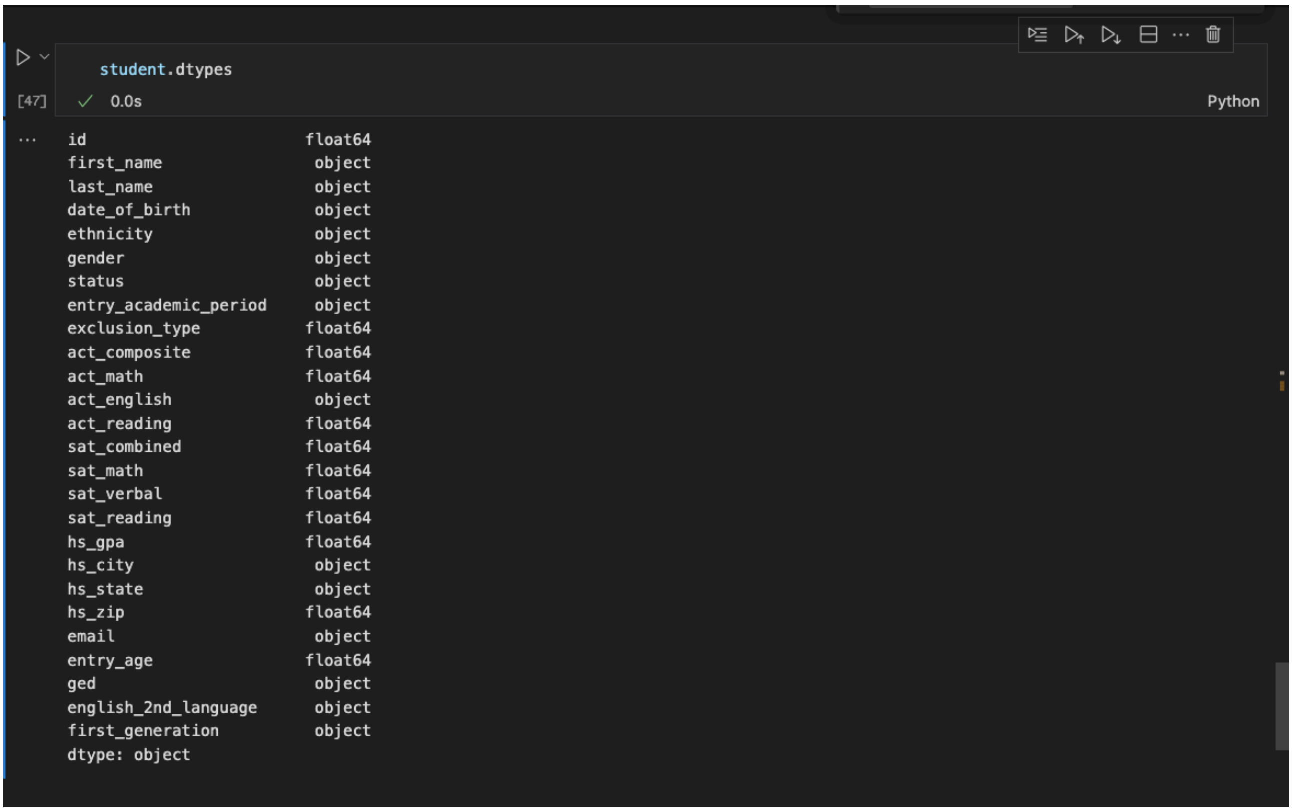
```
b    1
a    0
c    2
dtype: int64
```

```python
student = pd.read_csv("./a.csv")
student.head(3)
```

[46] ✓ 0.0s Python

	id	first_name	last_name	date_of_birth	ethnicity	gender	status	entry_academic_period	exclusion_type	act_composite
0	111111.0	John	Doe	01/2000	Hispanic	M	FT	Fall 2008	NaN	NaN
1	111112.0	Jane	Smith	05/2001	Hispanic	F	TRANSFER	Fall 2006	NaN	NaN
2	111113.0	Sarah	Thomas	21/2002	Hispanic	M	FTFT	Fall 2006	NaN	14.0

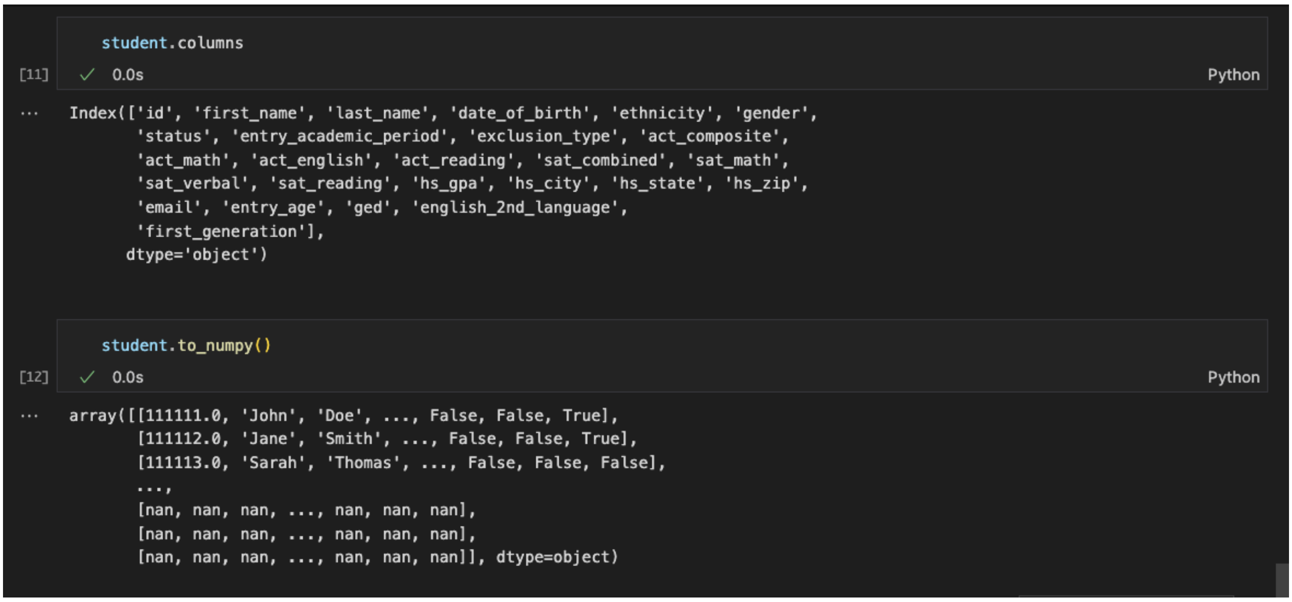
3 rows × 26 columns



```python
student.dtypes
```

[47] ✓ 0.0s Python

```
id                       float64
first_name                object
last_name                 object
date_of_birth             object
ethnicity                 object
gender                    object
status                    object
entry_academic_period     object
exclusion_type           float64
act_composite            float64
act_math                 float64
act_english               object
act_reading              float64
sat_combined             float64
sat_math                 float64
sat_verbal               float64
sat_reading              float64
hs_gpa                   float64
hs_city                   object
hs_state                  object
hs_zip                   float64
email                     object
entry_age                float64
ged                       object
english_2nd_language      object
first_generation          object
dtype: object
```



```
student.columns
```

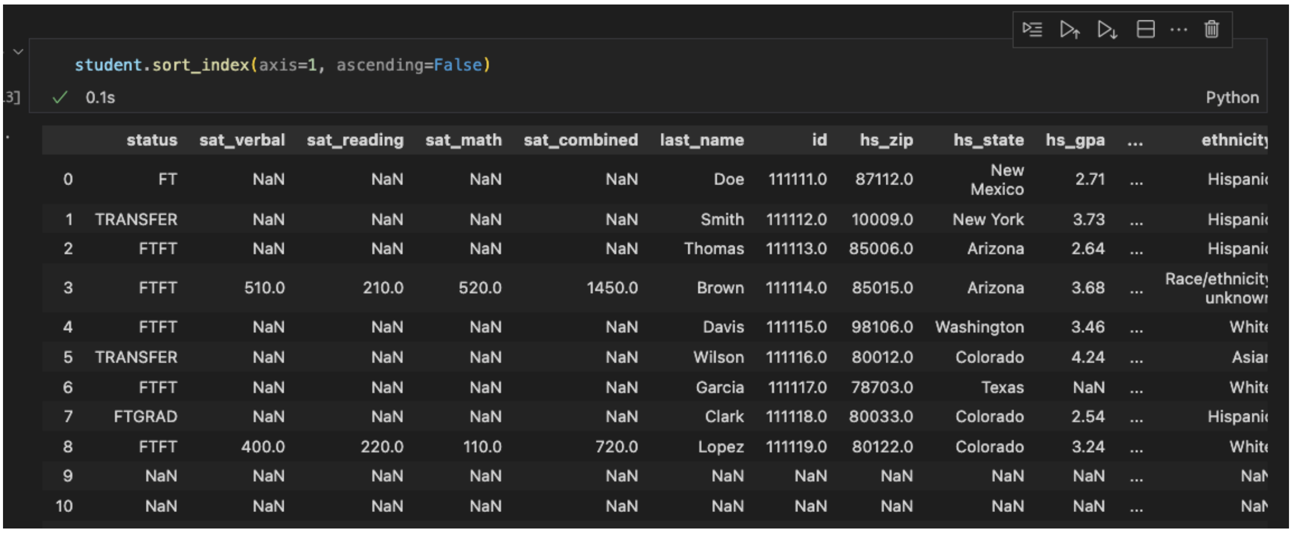
[11] ✓ 0.0s Python

```
Index(['id', 'first_name', 'last_name', 'date_of_birth', 'ethnicity', 'gender',
       'status', 'entry_academic_period', 'exclusion_type', 'act_composite',
       'act_math', 'act_english', 'act_reading', 'sat_combined', 'sat_math',
       'sat_verbal', 'sat_reading', 'hs_gpa', 'hs_city', 'hs_state', 'hs_zip',
       'email', 'entry_age', 'ged', 'english_2nd_language',
       'first_generation'],
      dtype='object')
```

```
student.to_numpy()
```

[12] ✓ 0.0s Python

```
array([[111111.0, 'John', 'Doe', ..., False, False, True],
       [111112.0, 'Jane', 'Smith', ..., False, False, True],
       [111113.0, 'Sarah', 'Thomas', ..., False, False, False],
       ...,
       [nan, nan, nan, ..., nan, nan, nan],
       [nan, nan, nan, ..., nan, nan, nan],
       [nan, nan, nan, ..., nan, nan, nan]], dtype=object)
```

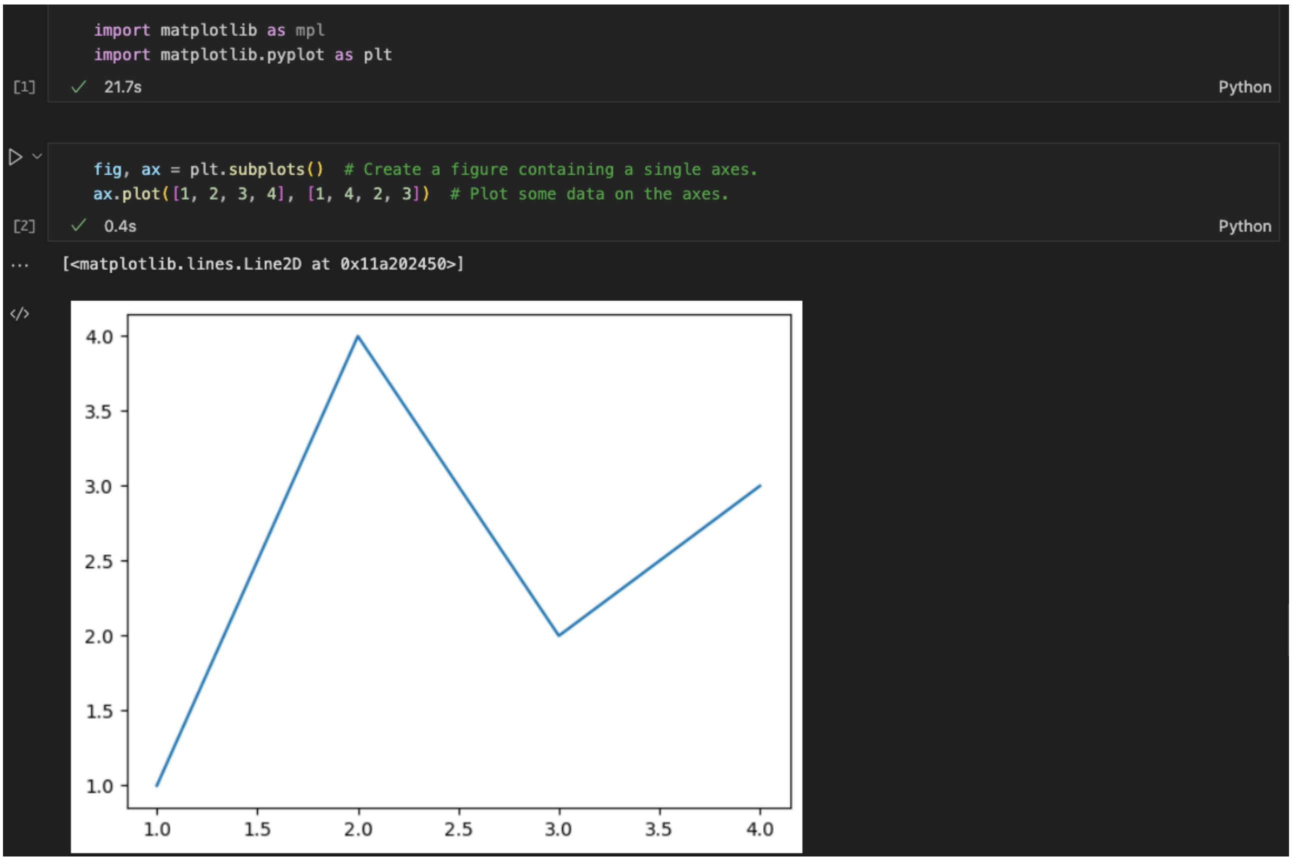


```
student.sort_index(axis=1, ascending=False)
```

3] ✓ 0.1s Python

	status	sat_verbal	sat_reading	sat_math	sat_combined	last_name	id	hs_zip	hs_state	hs_gpa	...	ethnicit
0	FT	NaN	NaN	NaN	NaN	Doe	111111.0	87112.0	New Mexico	2.71	...	Hispani
1	TRANSFER	NaN	NaN	NaN	NaN	Smith	111112.0	10009.0	New York	3.73	...	Hispani
2	FTFT	NaN	NaN	NaN	NaN	Thomas	111113.0	85006.0	Arizona	2.64	...	Hispani
3	FTFT	510.0	210.0	520.0	1450.0	Brown	111114.0	85015.0	Arizona	3.68	...	Race/ethnicit unknow
4	FTFT	NaN	NaN	NaN	NaN	Davis	111115.0	98106.0	Washington	3.46	...	Whit
5	TRANSFER	NaN	NaN	NaN	NaN	Wilson	111116.0	80012.0	Colorado	4.24	...	Asia
6	FTFT	NaN	NaN	NaN	NaN	Garcia	111117.0	78703.0	Texas	NaN	...	Whit
7	FTGRAD	NaN	NaN	NaN	NaN	Clark	111118.0	80033.0	Colorado	2.54	...	Hispani
8	FTFT	400.0	220.0	110.0	720.0	Lopez	111119.0	80122.0	Colorado	3.24	...	Whit
9	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	Na
10	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	Na

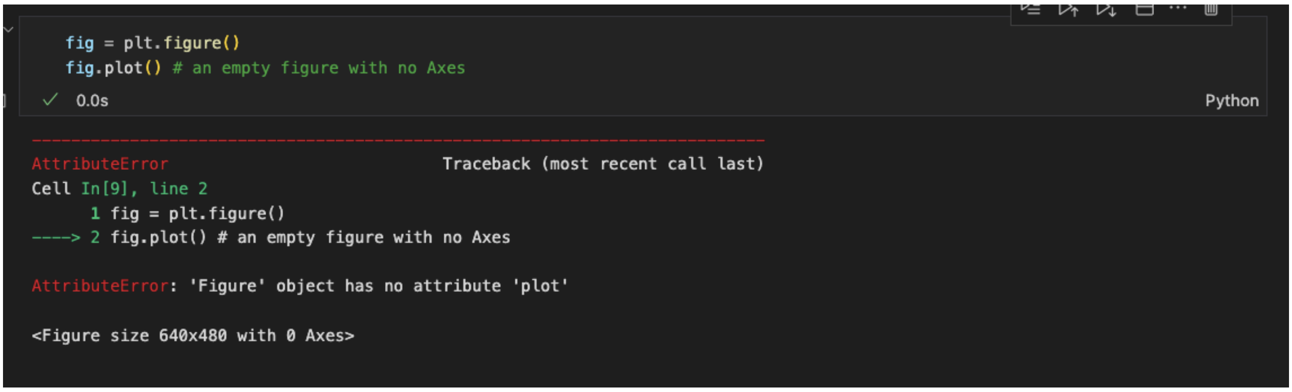
Matplotlib



```
import matplotlib as mpl
import matplotlib.pyplot as plt
```

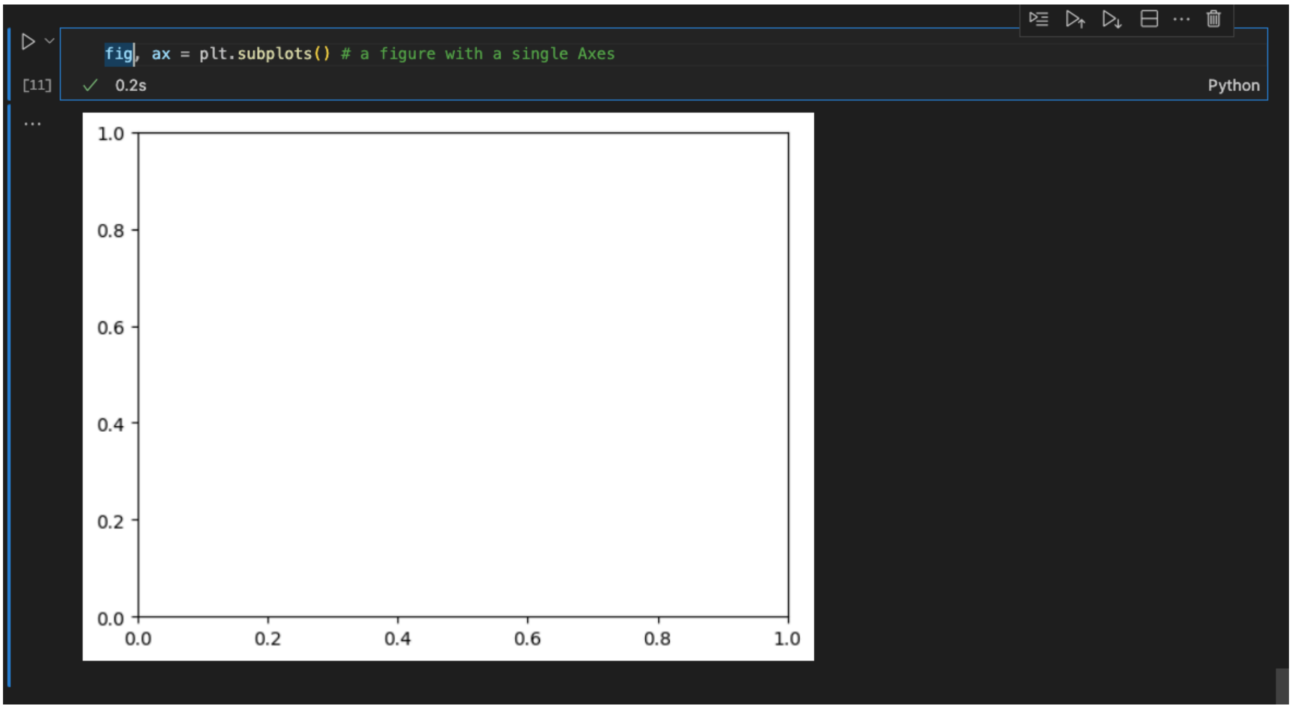
```
fig, ax = plt.subplots()  # Create a figure containing a single axes.
ax.plot([1, 2, 3, 4], [1, 4, 2, 3])  # Plot some data on the axes.
```

```
[<matplotlib.lines.Line2D at 0x11a202450>]
```

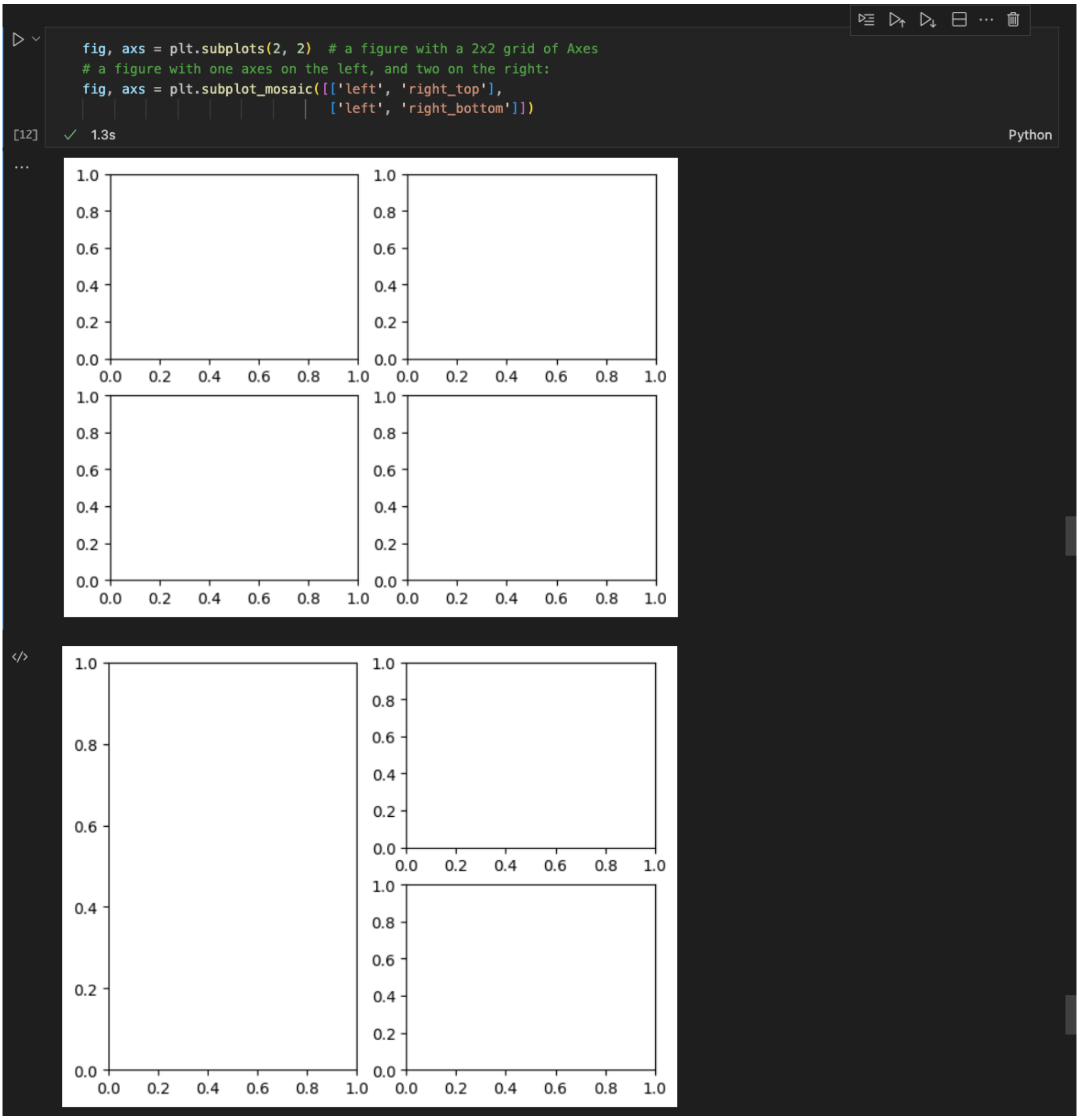


```
fig = plt.figure()
fig.plot() # an empty figure with no Axes
```

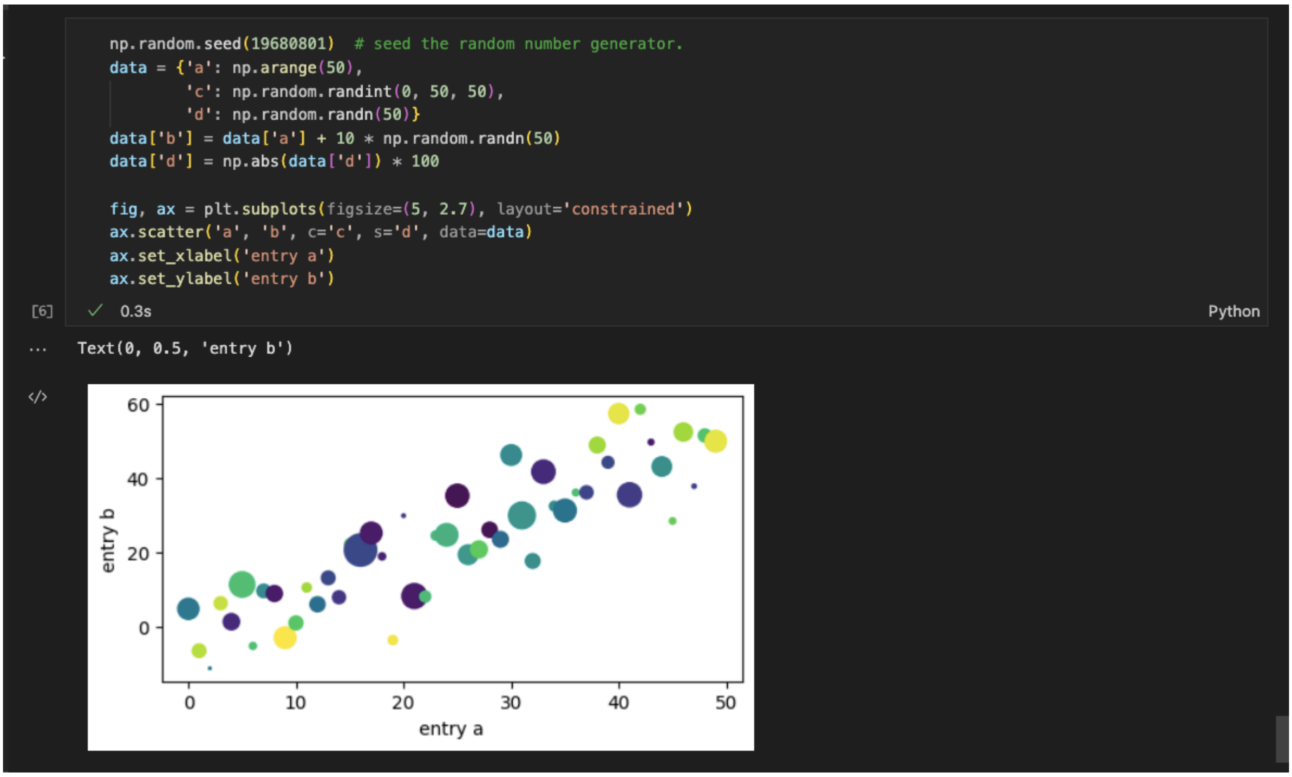
```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[9], line 2
      1 fig = plt.figure()
----> 2 fig.plot() # an empty figure with no Axes

AttributeError: 'Figure' object has no attribute 'plot'

<Figure size 640x480 with 0 Axes>
```



```
fig, ax = plt.subplots() # a figure with a single Axes
```



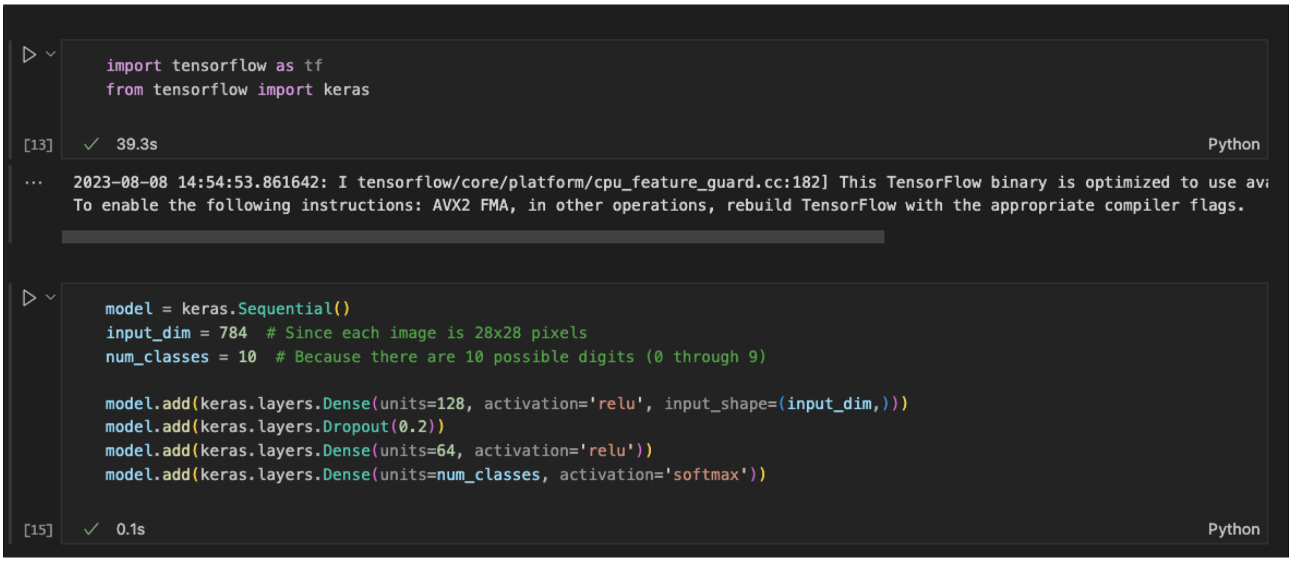
```
fig, axs = plt.subplots(2, 2)  # a figure with a 2x2 grid of Axes
# a figure with one axes on the left, and two on the right:
fig, axs = plt.subplot_mosaic([['left', 'right_top'],
                               ['left', 'right_bottom']])
```



```
np.random.seed(19680801)  # seed the random number generator.
data = {'a': np.arange(50),
        'c': np.random.randint(0, 50, 50),
        'd': np.random.randn(50)}
data['b'] = data['a'] + 10 * np.random.randn(50)
data['d'] = np.abs(data['d']) * 100

fig, ax = plt.subplots(figsize=(5, 2.7), layout='constrained')
ax.scatter('a', 'b', c='c', s='d', data=data)
ax.set_xlabel('entry a')
ax.set_ylabel('entry b')
```

```
Text(0, 0.5, 'entry b')
```

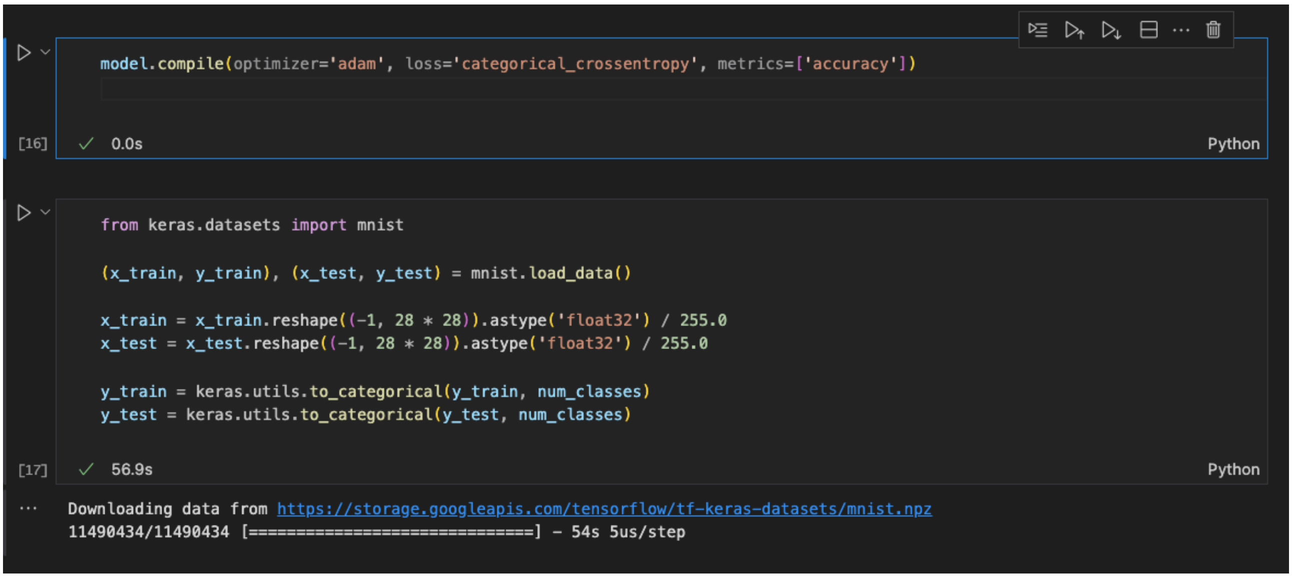
Keras



```
import tensorflow as tf
from tensorflow import keras
```

```
2023-08-08 14:54:53.861642: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use ava
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
```

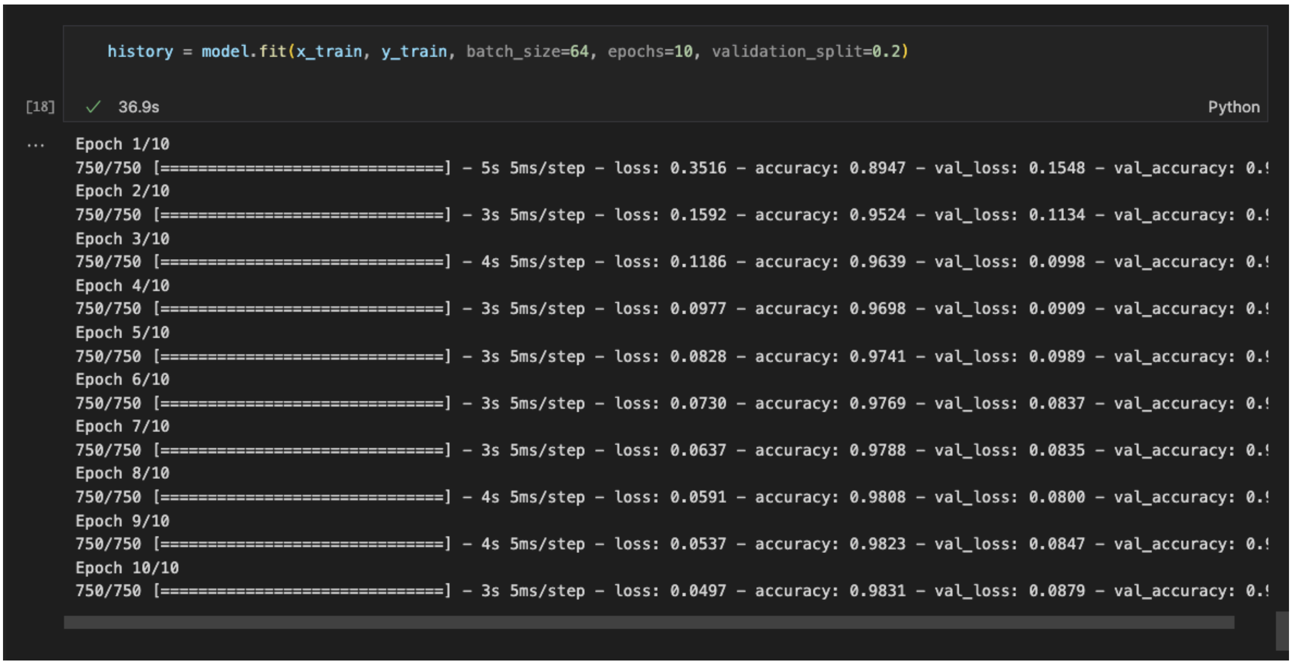
```
model = keras.Sequential()
input_dim = 784  # Since each image is 28x28 pixels
num_classes = 10  # Because there are 10 possible digits (0 through 9)

model.add(keras.layers.Dense(units=128, activation='relu', input_shape=(input_dim,)))
model.add(keras.layers.Dropout(0.2))
model.add(keras.layers.Dense(units=64, activation='relu'))
model.add(keras.layers.Dense(units=num_classes, activation='softmax'))
```



```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

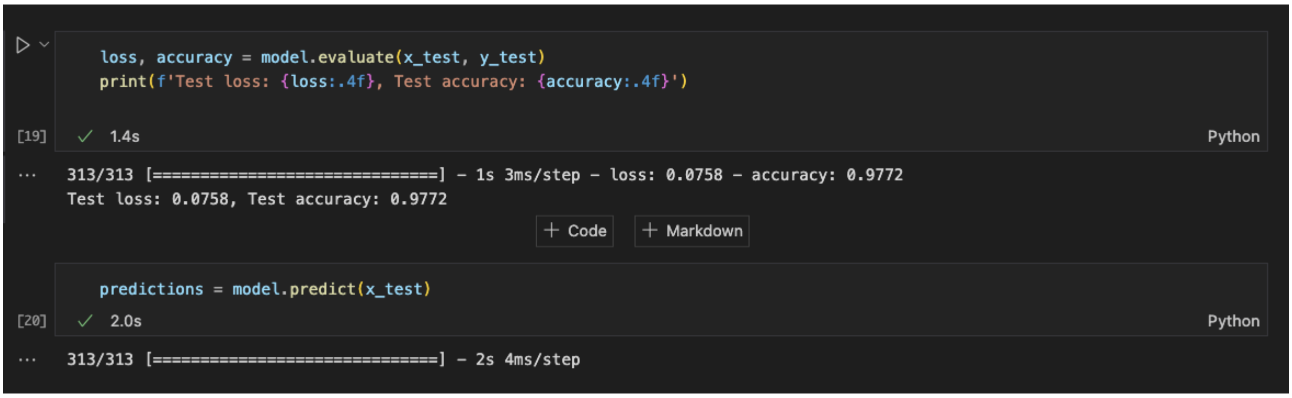
```
from keras.datasets import mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train.reshape((-1, 28 * 28)).astype('float32') / 255.0
x_test = x_test.reshape((-1, 28 * 28)).astype('float32') / 255.0

y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 54s 5us/step
```



```
history = model.fit(x_train, y_train, batch_size=64, epochs=10, validation_split=0.2)
```

```
Epoch 1/10
750/750 [==============================] - 5s 5ms/step - loss: 0.3516 - accuracy: 0.8947 - val_loss: 0.1548 - val_accuracy: 0.
Epoch 2/10
750/750 [==============================] - 3s 5ms/step - loss: 0.1592 - accuracy: 0.9524 - val_loss: 0.1134 - val_accuracy: 0.
Epoch 3/10
750/750 [==============================] - 4s 5ms/step - loss: 0.1186 - accuracy: 0.9639 - val_loss: 0.0998 - val_accuracy: 0.
Epoch 4/10
750/750 [==============================] - 3s 5ms/step - loss: 0.0977 - accuracy: 0.9698 - val_loss: 0.0909 - val_accuracy: 0.
Epoch 5/10
750/750 [==============================] - 3s 5ms/step - loss: 0.0828 - accuracy: 0.9741 - val_loss: 0.0989 - val_accuracy: 0.
Epoch 6/10
750/750 [==============================] - 3s 5ms/step - loss: 0.0730 - accuracy: 0.9769 - val_loss: 0.0837 - val_accuracy: 0.
Epoch 7/10
750/750 [==============================] - 3s 5ms/step - loss: 0.0637 - accuracy: 0.9788 - val_loss: 0.0835 - val_accuracy: 0.
Epoch 8/10
750/750 [==============================] - 4s 5ms/step - loss: 0.0591 - accuracy: 0.9808 - val_loss: 0.0800 - val_accuracy: 0.
Epoch 9/10
750/750 [==============================] - 4s 5ms/step - loss: 0.0537 - accuracy: 0.9823 - val_loss: 0.0847 - val_accuracy: 0.
Epoch 10/10
750/750 [==============================] - 3s 5ms/step - loss: 0.0497 - accuracy: 0.9831 - val_loss: 0.0879 - val_accuracy: 0.
```

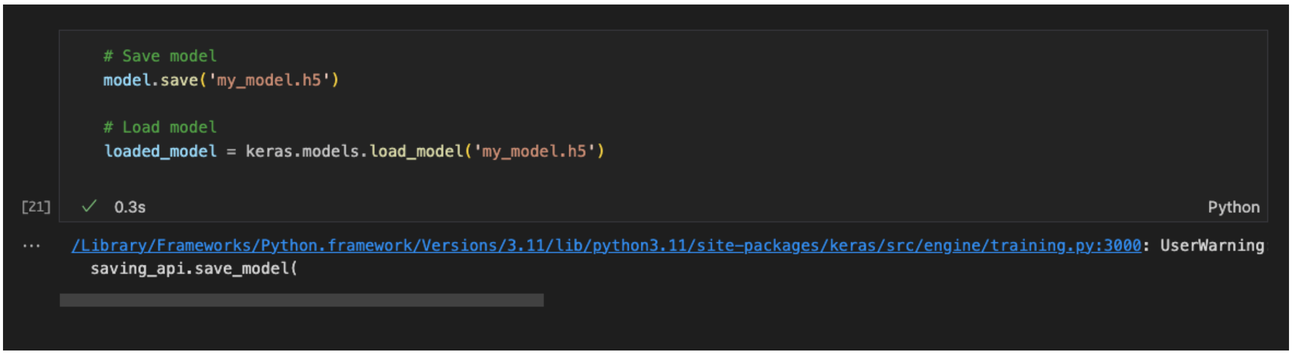


```
loss, accuracy = model.evaluate(x_test, y_test)
print(f'Test loss: {loss:.4f}, Test accuracy: {accuracy:.4f}')
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.0758 - accuracy: 0.9772
Test loss: 0.0758, Test accuracy: 0.9772
```

```
predictions = model.predict(x_test)
```

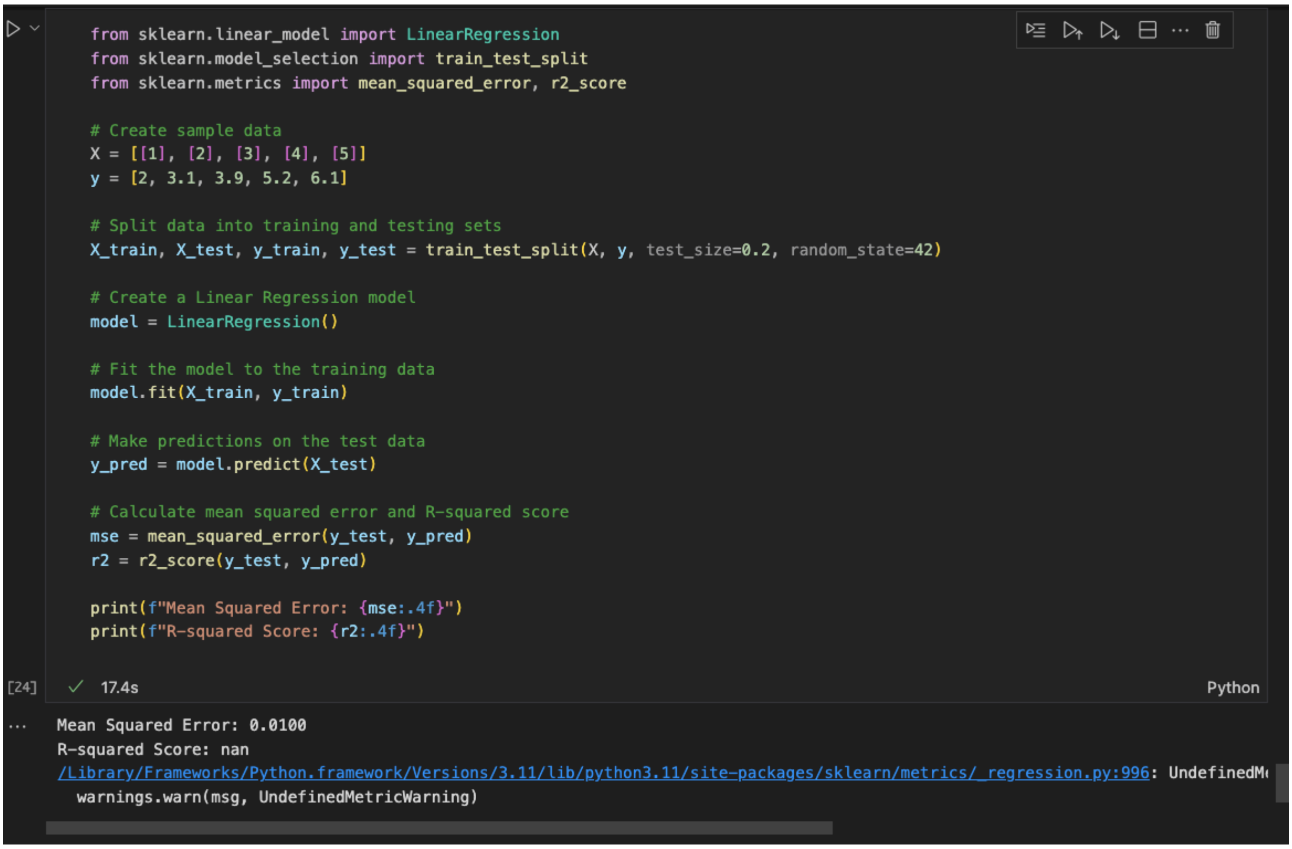
```
313/313 [==============================] - 2s 4ms/step
```



```
# Save model
model.save('my_model.h5')

# Load model
loaded_model = keras.models.load_model('my_model.h5')
```

```
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/keras/src/engine/training.py:3000: UserWarning
  saving_api.save_model(
```

Scikit-Learn



```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

# Create sample data
X = [[1], [2], [3], [4], [5]]
y = [2, 3.1, 3.9, 5.2, 6.1]

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a Linear Regression model
model = LinearRegression()

# Fit the model to the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Calculate mean squared error and R-squared score
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse:.4f}")
print(f"R-squared Score: {r2:.4f}")
```

```
Mean Squared Error: 0.0100
R-squared Score: nan
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/sklearn/metrics/_regression.py:996: UndefinedM
  warnings.warn(msg, UndefinedMetricWarning)
```