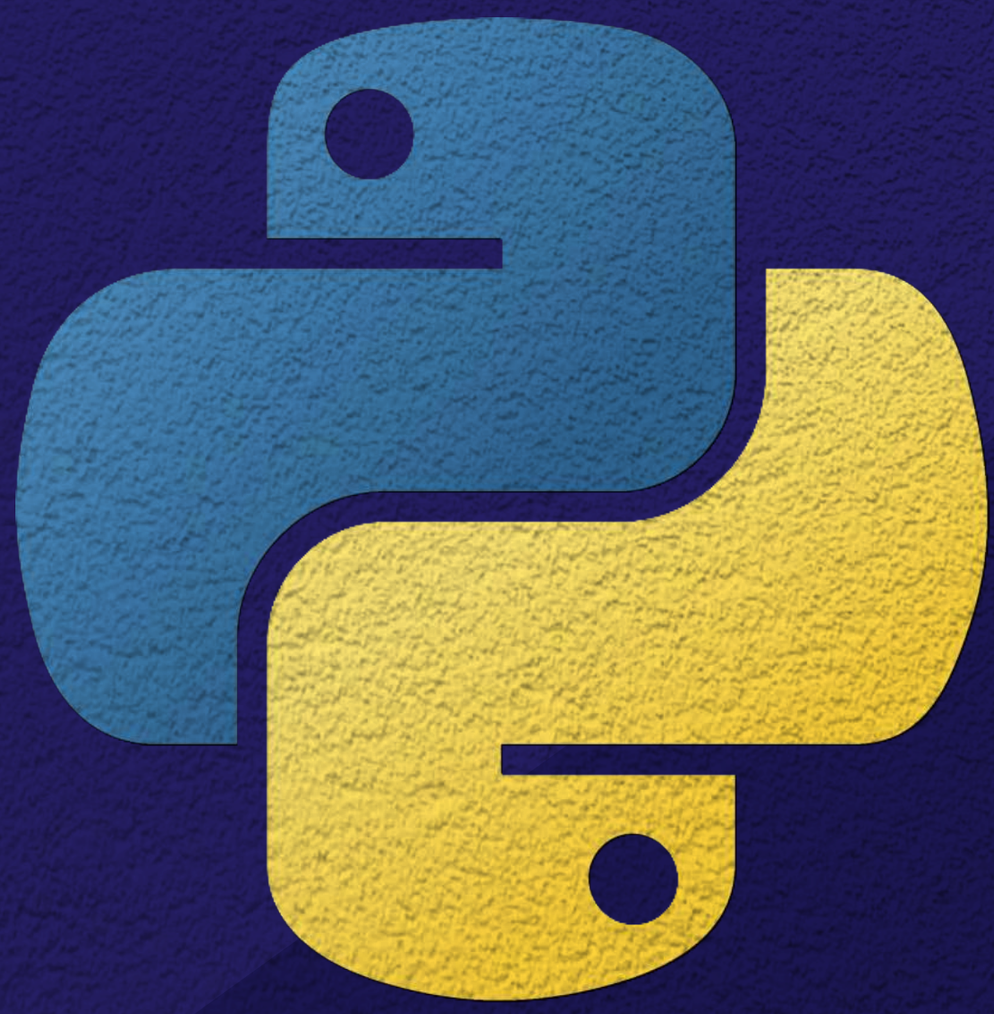
Must Learn

PYTHON FUNCTIONS

For Data Professionals



Disclaimer

Everyone learns uniquely
What matters is your ability to understand
and write Python Functions efficiently.

This Doc will help you with the same.

1. NumPy - `numpy.array()`

About:

Creates a NumPy array, which is a powerful data structure for efficient computation on large arrays and matrices.

Syntax:

```
numpy.array(object, dtype=None, copy=True, order='K', subok=False, ndmin=0)
```

Example:

```
import numpy as np

# Create a NumPy array from a Python list
my_list = [1, 2, 3, 4, 5]
my_array = np.array(my_list)

print(my_array)  # Output: [1 2 3 4 5]
```

In this example, we have a Python list **`my_list`** containing some numbers. We use the **`numpy.array()`** function to convert this list into a NumPy array called my_array. The resulting array is then printed, showing the output **`[1 2 3 4 5]`**.

Practice Questions:

1. Create a NumPy array from a list of even numbers from 2 to 10.

2. Convert the following Python list to a NumPy array: `[10, 20, 30, 40, 50]`.

3. Create a 2D NumPy array from a nested list: `[[1, 2], [3, 4], [5, 6]]`.

2. Pandas - `pandas.DataFrame()`

About:

Creates a pandas DataFrame, a tabular data structure that allows efficient data manipulation and analysis.

Syntax:

```
pandas.DataFrame(data=None, index=None, columns=None)
```

Example:

```
import pandas as pd

# Create a DataFrame from a dictionary
data = {'Name': ['John', 'Alice', 'Bob'], 'Age': [25, 28, 32]}
df = pd.DataFrame(data)

print(df)
```

In this example, we have a dictionary **`data`** containing information about individuals, including their names and ages. We use the **`pandas.DataFrame()`** function to create a DataFrame called **`df`** from this dictionary. The resulting DataFrame is then printed, displaying the following table:

```
   Name   Age
0  John   25
1  Alice  28
2  Bob    32
```

Practice Questions:

1. Create a DataFrame from a list of dictionaries containing employee information (name, age, salary).

2. Convert a NumPy array **`my_array`** into a DataFrame with two columns: 'Column1' and 'Column2'.

3. Create an empty DataFrame with columns 'A', 'B', and 'C'.

3. Matplotlib - `matplotlib.pyplot.plot()`

About:

Plots a line or marker-based chart using Matplotlib, a popular visualization library.

Syntax:

```
matplotlib.pyplot.plot(x, y, format_string='', **kwargs)
```

Example:

```
import matplotlib.pyplot as plt

# Plot a line chart
x = [1, 2, 3, 4, 5]
y = [10, 5, 7, 8, 2]

plt.plot(x, y, marker='o')

plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Line Chart')
plt.grid(True)
plt.show()
```

In this example, we have two lists, **`x`** and **`y`**, representing the x-axis and y-axis values, respectively. We use the **`matplotlib.pyplot.plot()`** function to create a line chart by plotting the points **`(1, 10)`**, **`(2, 5)`**, **`(3, 7)`**, **`(4, 8)`**, and **`(5, 2)`**. The resulting chart is then displayed with the specified x-label, y-label, and title.

Practice Questions:

1. Create a scatter plot with randomly generated x and y values.

2. Plot multiple lines on the same chart with different colors and markers.

3. Customize the appearance of the plot by adding a legend and changing the line style.

4. Seaborn - `seaborn.heatmap()`

About:

Creates a heatmap, a graphical representation of data in a matrix form, using Seaborn, a statistical data visualization library.

Syntax:

```
seaborn.heatmap(data, vmin=None, vmax=None, cmap=None, center=None,
robust=False, annot=None, fmt='.2g', annot_kws=None)
```

Example:

```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# Create a heatmap
data = np.random.rand(3, 3)

sns.heatmap(data, annot=True, cmap='YlGnBu')

plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Heatmap')
plt.show()
```

In this example, we generate a random 3×3 matrix **`data`** using NumPy. We use the **`seaborn.heatmap()`** function to create a heatmap from this matrix. The heatmap is then displayed with annotations (numbers in each cell) and a color map ('YlGnBu').

Practice Questions:

Practice Questions:

1. Generate a correlation matrix from a DataFrame and plot it as a heatmap.

2. Create a matrix with random values and plot it as a diverging colormap heatmap.

3. Plot a categorical heatmap showing the frequency of occurrence of different categories in a dataset.

5. Scikit-learn -

`sklearn.model_selection.train_test_split()`

About:

Splits a dataset into training and testing subsets for machine learning model evaluation.

Syntax:

```
sklearn.model_selection.train_test_split(*arrays, test_size=None,
train_size=None, random_state=None)
```

Example:

```
from sklearn.model_selection import train_test_split
import numpy as np

# Split a dataset into training and testing sets
X = np.arange(10).reshape((5, 2))
y = range(5)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

print(X_train)
print(X_test)
```

In this example, we have a dataset consisting of features **X** and labels **y**. We use the **`sklearn.model_selection.train_test_split()`** function to split this dataset into training and testing sets. The resulting splits are stored in **`X_train`**, **`X_test`**, **`y_train`**, and **`y_test`**.

Practice Questions:

1. Split a dataset consisting of features 'X' and labels 'y' into 70% training set and 30% testing set.

2. Perform a stratified train-test split on a dataset to maintain the class distribution in both sets.

3. Split a dataset into training, validation, and testing sets in a 60:20:20 ratio.

6. Statsmodels - `statsmodels.api.OLS()`

About:

Fits a linear regression model using the ordinary least squares method in Statsmodels, a statistical modeling library.

Syntax:

```
statsmodels.api.OLS(endog, exog=None, missing='none', hasconst=None)
```

Example:

```
import statsmodels.api as sm
import numpy as np

# Fit a linear regression model
X = np.array([1, 2, 3, 4, 5])
y = np.array([5, 7, 9, 11, 13])

X = sm.add_constant(X)  # Add a constant column for the intercept
model = sm.OLS(y, X)
results = model.fit()

print(results.summary())
```

In this example, we have two NumPy arrays, **`X`** and **`y`**, representing the independent and dependent variables, respectively. We use the **`statsmodels.api.OLS()`** function to fit a linear regression model to the data. The results of the model fitting are stored in the **`results`** object, and we print the summary of the fitted model using **`results.summary()`**.

Practice Questions:

1. Fit a multiple linear regression model on a dataset with three independent variables and one dependent variable.

2. Perform a logistic regression on a binary classification problem using the given dataset.

3. Fit a generalized linear model (GLM) on a dataset with a Poisson distribution.

7. TensorFlow - `tensorflow.keras.models.Sequential()`

About:

Creates a sequential model in TensorFlow, an open-source machine learning framework.

Syntax:

```
tensorflow.keras.models.Sequential(layers=None, name=None)
```

Example:

```
import tensorflow as tf

# Create a sequential model
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Dense(64, activation='relu', input_shape=(10,)))
model.add(tf.keras.layers.Dense(64, activation='relu'))
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))

print(model.summary())
```

In this example, we create a sequential model using **`tensorflow.keras.models.Sequential()`**. We add three layers to the model: two dense layers with ReLU activation functions and a final dense layer with a sigmoid activation function. The **`input_shape`** argument defines the shape of the input data. We print a summary of the model using **`model.summary()`**.

Practice Questions:

1. Build a deep neural network with three hidden layers and the following activations: 'relu', 'tanh', 'sigmoid'.

2. Create a convolutional neural network (CNN) with two convolutional layers and a max pooling layer.

3. Construct a recurrent neural network (RNN) with LSTM cells and a dense layer.

8. PyTorch - `torch.nn.Linear()`

About:

Creates a sequential model in TensorFlow, an open-source machine learning framework.

Syntax:

```
torch.nn.Linear(in_features, out_features, bias=True)
```

Example:

```
import torch

# Create a linear layer
layer = torch.nn.Linear(10, 5)

print(layer)
```

In this example, we create a linear layer using **`torch.nn.Linear()`**. The first argument **`in_features`** specifies the number of input features, and the second argument **`out_features`** specifies the number of output features. The **`bias`** parameter determines whether a bias term is included in the layer.

Practice Questions:

1. Build a neural network with two linear layers and a ReLU activation.

2. Create a convolutional layer with 16 filters, a kernel size of 3, and a stride of 1.

3. Construct a recurrent layer with LSTM cells and a hidden size of 128.

9. Datetime -

`datetime.datetime.now()`

About:

Retrieves the current date and time using the datetime module, which provides classes for manipulating dates and times.

Syntax:

```
datetime.datetime.now(tz=None)
```

Example:

```
import datetime

# Get the current date and time
now = datetime.datetime.now()

print(now)
```

In this example, we use **`datetime.datetime.now()`** to retrieve the current date and time. The resulting datetime object is stored in the variable **`now`** and then printed.

Practice Questions:

1. Get the current date and time in a specific timezone (e.g., 'America/New_York').

2. Calculate the difference in days between two given dates.

3. Convert a timestamp to a specific date and time format.

10. Math - `math.sqrt()`

About:

Calculates the square root of a number using the math module, which provides mathematical functions and constants.

Syntax:

```
math.sqrt(x)
```

Example:

```
import math

# Calculate the square root of a number
result = math.sqrt(16)

print(result)
```

In this example, we use **`math.sqrt()`** to calculate the square root of the number 16. The resulting square root is stored in the variable **`result`** and then printed.

Practice Questions:

1. Compute the square root of a given list of numbers.

2. Calculate the hypotenuse of a right-angled triangle using the Pythagorean theorem.

3. Determine the distance between two points in a Cartesian coordinate system.

11. Random - `random.choice()`

About:

Selects a random element from a sequence using the random module, which implements pseudo-random number generators.

Syntax:

```
random.choice(seq)
```

Example:

```
import random

# Choose a random element from a list
fruits = ['apple', 'banana', 'orange', 'grape']
random_fruit = random.choice(fruits)

print(random_fruit)
```

In this example, we have a list of fruits. We use **`random.choice()`** to select a random fruit from the list. The chosen fruit is then stored in the variable **`random_fruit`** and printed.

Practice Questions:

1. Select a random sample of 'k' elements from a given list.

2. Simulate a coin toss and print either 'Heads' or 'Tails'.

3. Shuffle the elements of a list in a random order.

12. CSV - `csv.reader()`

About:

Reads a CSV (Comma-Separated Values) file using the csv module, which provides functionality for working with CSV files.

Syntax:

```
csv.reader(csvfile, dialect='excel', **fmtparams)
```

Example:

```
import csv

# Read a CSV file
with open('data.csv', 'r') as file:
    reader = csv.reader(file)

    for row in reader:
        print(row)
```

In this example, we read a CSV file named 'data.csv' using **`csv.reader()`**. We iterate over the rows of the CSV file and print each row.

Practice Questions:

1. Read a CSV file containing student information (name, age, grade) and print the rows.

2. Extract specific columns from a CSV file and store them in a list.

3. Calculate the average value of a specific column in a CSV file.

13. OS - `os.listdir()`

About:

Retrieves a list of files and directories in a specified directory using the os module, which provides a way to interact with the operating system.

Syntax:

```
os.listdir(path='.')
```

Example:

```
import os

# List files in a directory
files = os.listdir('path/to/directory')

print(files)
```

In this example, we use os.listdir() to retrieve a list of files and directories in a specified directory. The resulting list of files is stored in the variable files and then printed.

Practice Questions:

1. Count the number of files in a specific directory (excluding directories).

2. Find all files with a specific extension (e.g., '.txt') in a directory.

3. Print the size of each file in a directory.

Here are some sample datasets that you can use for the practice questions:

1. Practice Question

Generate a scatter plot with randomly generated x and y values.

Sample Dataset:

```
import numpy as np
import matplotlib.pyplot as plt

# Generate random x and y values
np.random.seed(0)
x = np.random.rand(100)
y = np.random.rand(100)

# Scatter plot
plt.scatter(x, y)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Scatter Plot')
plt.show()
```

2. Practice Question

Plot multiple lines on the same chart with different colors and markers.

Sample Dataset:

```
import numpy as np
import matplotlib.pyplot as plt

# Generate x values
x = np.linspace(0, 10, 100)

# Generate y values for three lines
y1 = np.sin(x)
y2 = np.cos(x)
y3 = np.tan(x)

# Plot multiple lines
plt.plot(x, y1, color='red', marker='o', label='sin(x)')
plt.plot(x, y2, color='blue', marker='s', label='cos(x)')
plt.plot(x, y3, color='green', marker='^', label='tan(x)')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Multiple Lines Plot')
plt.legend()
plt.show()
```

3. Practice Question

Generate a correlation matrix from a DataFrame and plot it as a heatmap.

Sample Dataset:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Create a DataFrame
data = {'A': np.random.rand(100),
        'B': np.random.rand(100),
        'C': np.random.rand(100)}
df = pd.DataFrame(data)

# Generate the correlation matrix
corr_matrix = df.corr()

# Plot the correlation matrix as a heatmap
sns.heatmap(corr_matrix, annot=True)
plt.xlabel('Columns')
plt.ylabel('Columns')
plt.title('Correlation Heatmap')
plt.show()
```

4. Practice Question

Split a dataset consisting of features 'X' and labels 'y' into 70% training set and 30% testing set.

Sample Dataset:

```
import numpy as np
from sklearn.model_selection import train_test_split

# Generate sample data
X = np.random.rand(100, 3)  # Features
y = np.random.randint(0, 2, 100)  # Labels (binary)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)
```