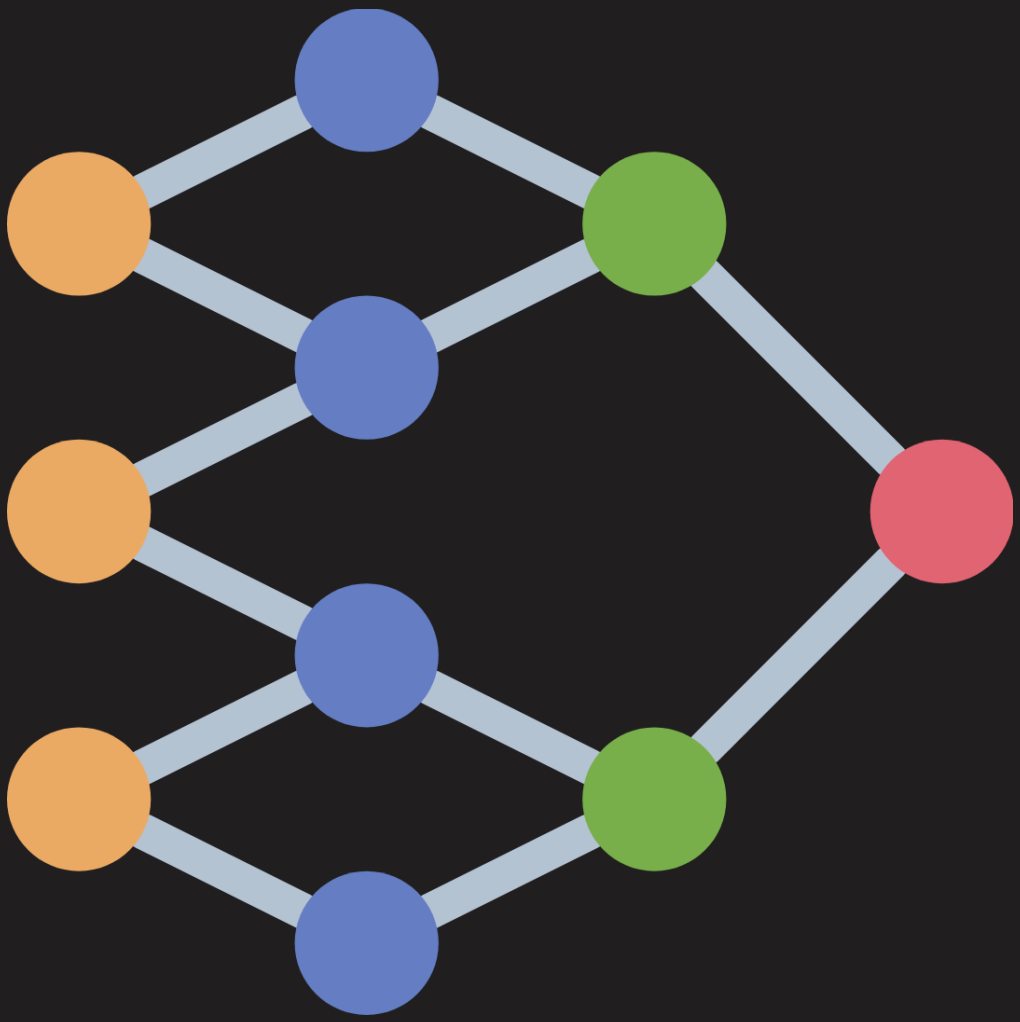
Python AI essentials



By Sutton

1. Python syntax. Python syntax is the set of rules that define how code should be written so that Python can interpret and execute it correctly.

1.01 Printing messages. The function `'print()'` It works to print messages to the console.

1.02 Comments. Comments in python start with the symbol `'#'`.

1.03 Variables. Variables in Python are created by entering the name of the variable and then what it will store. `'Name_1 = Santiago'`.

1.04 Data types. Python supports several data types `'str = "Python"', 'int = 10', 'float = 3.14' and 'bool = True'`.

1.05 Conditionals. Conditionals work like "If this is this, do this but if not, do this" for this we use `'"if", "elif" and "else"'`.

1.06 Loops. Loops work to repeat a piece of code, the most famous of which are `''for' and 'while''`.

1.07 Define functions. Functions are defined using the keyword `'def'`, which works to store a piece of code.

1.08 Lists. Lists are ordered and mutable collections, lists are created by putting data inside `'[ ]'`, an example is `'fruits = [ apple,  banana,  strawberry ]'`.

1.09 Tuples. Tuples are ordered and immutable collections, lists are created by putting data inside `'( )'`, an example is `'car = ( motor, window, tire)'`.

1.10 Dictionaries. Dictionaries are unordered key-value collections, lists are created by putting the data inside `'{ }'`, an example is `'person_1 = {"name": "Ana", "age": 25}'`.

1.11 Sets. Sets are unordered collections within single elements, an example is `'numbers = {1, 2, 3, 4, 5}'`.

1.12 Import of modules. The modules allow us to take external or our code, this is done using `'import'`.

2. Pandas. Pandas is an essential library for data analysis in Python. Allows the manipulation and analysis of structured data (tables).

```python
import pandas as pd

# Create a DataFrame
data = {'Name': ['Ana', 'Juan', 'Luis'], 'Age': [23, 35, 45]}
df = pd.DataFrame(data)
print(df)

# Read a CSV file
df = pd.read_csv('file.csv')

# Basic operations
print(df.head())  # First 5 rows
print(df.describe())  # Descriptive statistics
```

3. Numpy. NumPy is a fundamental library for numerical computation and matrix manipulation.

```python
import numpy as np

# Create an array
arr = np.array([1, 2, 3])
print(arr)

# Basic operations
arr2 = arr + 10
print(arr2)
```

4. Matplotlib. Matplotlib is the reference library for creating plots in Python.

```python
import matplotlib.pyplot as plt

# Create a simple plot
plt.plot([1, 2, 3], [4, 5, 6])
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Simple Plot')
plt.show()
```

5. Learn. Scikit-Learn is a powerful library for machine learning.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Create and train the model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
predictions = model.predict(X_test)
```

6. Tensor Flow. Tensor Flow is an open source library for numerical computing and machine learning. Keras is a high-level API that runs on top of Tensor Flow.

```
import tensorflow as tf
from tensorflow import keras

# Create a simple model
model = keras.Sequential([
    keras.layers.Dense(units=64, activation='relu'),
    keras.layers.Dense(units=1, activation='sigmoid')
])

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(X_train, y_train, epochs=10)
```

7. OpenCV. OpenCV is an open source library for computer vision.

```
import cv2

# Read an image
img = cv2.imread('image.jpg')

# Display an image
cv2.imshow('Image', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

8. NLTK (Natural Language Toolkit). NLTK is a complete library for working with natural language data.

```
import nltk
from nltk.tokenize import word_tokenize

# Word tokenization
text = "Hello, how are you?"
words = word_tokenize(text)
print(words)
```

9. Scikit-Image. Scikit-Image is a library for image processing in Python.

```
from skimage import io, filters

# Read an image
img = io.imread('image.jpg')

# Apply a filter
filtered = filters.gaussian(img, sigma=1)
```

10. Requests. Requests is a simple library for making HTTP requests.

```
import requests

# Make a GET request
response = requests.get('https://api.example.com/data')
print(response.text)
```

11. BS4. BeautifulSoup is a library for extracting data from HTML and XML files.

```
from bs4 import BeautifulSoup
import requests

# Make a request and parse the HTML
response = requests.get('https://example.com')
soup = BeautifulSoup(response.text, 'html.parser')

# Find an element
title = soup.find('h1')
print(title.text)
```

12. Tkinter. Tkinter is the standard library for creating graphical interfaces in Python.

```python
import tkinter as tk

# Create a window
window = tk.Tk()
window.title("My Window")

# Create a label
label = tk.Label(window, text="Hello, World!")
label.pack()

# Run the application
window.mainloop()
```

13. Pygame. Pygame is a library for video game development in Python.

```python
import pygame

# Initialize Pygame
pygame.init()

# Create a window
window = pygame.display.set_mode((800, 600))
pygame.display.set_caption("My Game")

# Main game loop
running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

pygame.quit()
```

14. Math. The math library provides basic and advanced math functions.

```python
import math

# Basic functions
print(math.sqrt(16))  # 4.0
print(math.sin(math.pi / 2))  # 1.0
```

15. Random. The random library is used to generate random numbers.

```python
import random

# Generate a random number between 0 and 1
print(random.random())

# Generate a random integer between two values
print(random.randint(1, 10))
```

16. Qiskit. Qiskit is an open source library for quantum computing.

```
from qiskit import QuantumCircuit, transpile, Aer, execute

# Create a quantum circuit with one qubit
circuit = QuantumCircuit(1, 1)
circuit.h(0)  # Apply Hadamard gate
circuit.measure(0, 0)  # Measure the qubit

# Simulate the circuit
simulator = Aer.get_backend('qasm_simulator')
result = execute(circuit, simulator).result()
counts = result.get_counts(circuit)
print(counts)
```