Python Documentation

**1. Introduction to Python**

Python is a high-level, interpreted programming language designed for simplicity and readability.

**Features of Python:**

✅ Easy-to-learn syntax  
✅ Dynamically typed  
✅ Object-oriented  
✅ Extensive standard library  
✅ Cross-platform compatibility

**2. Basic Syntax and Data Types**

**2.1 Variables and Data Types**

🔹 **Definition**: Variables store data values, and Python determines their type dynamically.

x = 10 # Integer

y = 3.14 # Float

name = "Python" # String

is\_active = True # Boolean

lst = [1, 2, 3] # List

tpl = (1, 2, 3) # Tuple

st = {1, 2, 3} # Set

dct = {"name": "Alice", "age": 25} # Dictionary

**2.2 Type Checking and Conversion**

🔹 **Definition**: Python provides functions to check and convert data types.

print(type(x)) # <class 'int'>

print(float(x)) # Convert int to float

print(str(x)) # Convert int to string

print(list(tpl)) # Convert tuple to list

**2.3 Comments**

🔹 **Definition**: Comments are ignored by the Python interpreter and are used to explain code.

# This is a single-line comment

""" This is a multi-line comment """

**3. Operators in Python**

**3.1 Arithmetic Operators**

🔹 **Definition**: These operators perform basic mathematical operations.

print(5 + 2) # 7 (Addition)

print(5 - 2) # 3 (Subtraction)

print(5 \* 2) # 10 (Multiplication)

print(5 / 2) # 2.5 (Division)

print(5 // 2) # 2 (Floor Division)

print(5 % 2) # 1 (Modulus)

print(5 \*\* 2) # 25 (Exponentiation)

**3.2 Comparison Operators**

🔹 **Definition**: Used to compare values and return True or False.

print(5 == 5) # True

print(5 != 2) # True

print(5 > 2) # True

**4. Control Flow**

**4.1 Conditional Statements**

🔹 **Definition**: if-elif-else statements execute code based on conditions.

x = 10

if x > 0:

print("Positive")

elif x == 0:

print("Zero")

else:

print("Negative")

**4.2 Loops**

🔹 **Definition**: Loops execute code repeatedly until a condition is met.

**For Loop**

for i in range(5):

print(i) # 0,1,2,3,4

**While Loop**

x = 5

while x > 0:

print(x)

x -= 1

**5. Functions**

🔹 **Definition**: Functions are reusable blocks of code that perform specific tasks.

def greet(name):

return f"Hello, {name}"

print(greet("Alice")) # Hello, Alice

**Lambda Function (One-liner Function)**

square = lambda x: x \* x

print(square(4)) # 16

**6. Data Structures**

**6.1 Lists (Mutable, Ordered)**

🔹 **Definition**: Lists store multiple items in an ordered sequence.

nums = [1, 2, 3, 4]

nums.append(5)

nums.remove(2)

print(nums[0]) # Accessing elements

print(nums[-1]) # Last element

**6.2 Tuples (Immutable, Ordered)**

🔹 **Definition**: Similar to lists but cannot be modified.

tpl = (1, 2, 3)

print(tpl[1]) # 2

**6.3 Sets (Unique, Unordered)**

🔹 **Definition**: Store unique elements without a specific order.

st = {1, 2, 3, 3}

st.add(4)

print(st) # {1, 2, 3, 4}

**6.4 Dictionaries (Key-Value Pair)**

🔹 **Definition**: Store data as key-value pairs.

student = {"name": "Alice", "age": 20}

print(student["name"]) # Alice

student["age"] = 21 # Update value

**7. File Handling**

🔹 **Definition**: Python allows reading/writing files using built-in functions.

# Writing to a file

with open("file.txt", "w") as f:

f.write("Hello, Python!")

# Reading from a file

with open("file.txt", "r") as f:

print(f.read())

**8. Exception Handling**

🔹 **Definition**: Handling runtime errors using try-except.

try:

x = 1 / 0

except ZeroDivisionError:

print("Cannot divide by zero")

finally:

print("Done")

**9. Object-Oriented Programming (OOP)**

**9.1 Class and Object**

🔹 **Definition**: A class is a blueprint for objects.

class Person:

def \_\_init\_\_(self, name):

self.name = name

def greet(self):

return f"Hello, {self.name}"

p = Person("Alice")

print(p.greet()) # Hello, Alice

**9.2 Inheritance**

🔹 **Definition**: A class can inherit methods and properties from another class.

class Student(Person):

def \_\_init\_\_(self, name, grade):

super().\_\_init\_\_(name)

self.grade = grade

s = Student("Bob", "A")

print(s.greet()) # Hello, Bob

**10. Python Modules & Libraries**

🔹 **Definition**: Python has built-in and third-party libraries for various tasks.

import math

print(math.sqrt(16)) # 4.0

import random

print(random.randint(1, 10))

**11. Advanced Topics**

**11.1 List Comprehension**

🔹 **Definition**: A concise way to create lists.

squares = [x\*x for x in range(5)]

print(squares) # [0, 1, 4, 9, 16]

**11.2 Generators**

🔹 **Definition**: Functions that return values lazily using yield.

def count():

yield 1

yield 2

gen = count()

print(next(gen)) # 1

**11.3 Decorators**

🔹 **Definition**: Functions that modify other functions dynamically.

def decorator(func):

def wrapper():

print("Before function")

func()

print("After function")

return wrapper

@decorator

def say\_hello():

print("Hello!")

say\_hello()

*12. Popular Python Libraries*

**12.1 NumPy (Numerical Python)**

🔹 **Definition**: NumPy is a fundamental library for numerical computing in Python, providing support for large, multi-dimensional arrays and matrices, along with mathematical functions to operate on these structures.

**Key Features:**

✅ Multi-dimensional array support (ndarray)  
✅ Fast mathematical operations (linear algebra, Fourier transforms, statistics)  
✅ Broadcasting for efficient array operations  
✅ Integration with other scientific computing libraries

**Installation:**

pip install numpy

**Example Usage:**

import numpy as np

# Create an array

arr = np.array([1, 2, 3, 4])

# Perform operations

print(arr \* 2) # [2 4 6 8]

# Create a 2D matrix

matrix = np.array([[1, 2], [3, 4]])

print(matrix.shape) # (2,2)

**12.2 Pandas (Data Manipulation & Analysis)**

🔹 **Definition**: Pandas is a library for data manipulation and analysis, providing powerful data structures like **DataFrame** and **Series** to handle structured data efficiently.

**Key Features:**

✅ Data structures: Series (1D) and DataFrame (2D)  
✅ Easy data cleaning and transformation  
✅ Supports reading/writing from CSV, Excel, SQL, JSON  
✅ Handles missing data seamlessly

**Installation:**

pip install pandas

**Example Usage:**

import pandas as pd

# Create a DataFrame

data = {"Name": ["Alice", "Bob"], "Age": [25, 30]}

df = pd.DataFrame(data)

# Display DataFrame

print(df)

# Read CSV file

df = pd.read\_csv("data.csv")

# Basic operations

print(df.head()) # First 5 rows

print(df.describe()) # Summary statistics

**12.3 Matplotlib (Data Visualization)**

🔹 **Definition**: Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

**Key Features:**

✅ Line, bar, scatter, histogram, and pie charts  
✅ Customizable plots (titles, labels, legends)  
✅ Supports interactive graphs

**Installation:**

pip install matplotlib

**Example Usage:**

import matplotlib.pyplot as plt

# Create a simple line plot

x = [1, 2, 3, 4]

y = [10, 20, 25, 30]

plt.plot(x, y, label="Line Plot")

plt.xlabel("X-Axis")

plt.ylabel("Y-Axis")

plt.title("Simple Line Chart")

plt.legend()

plt.show()

**12.4 Seaborn (Statistical Data Visualization)**

🔹 **Definition**: Seaborn is a high-level visualization library built on Matplotlib, designed for statistical graphics.

**Key Features:**

✅ Beautiful and informative statistical charts  
✅ Built-in themes for aesthetics  
✅ Supports complex visualizations like heatmaps, violin plots

**Installation:**

pip install seaborn

**Example Usage:**

import seaborn as sns

import matplotlib.pyplot as plt

# Load sample dataset

tips = sns.load\_dataset("tips")

# Create a boxplot

sns.boxplot(x="day", y="total\_bill", data=tips)

plt.show()

**12.5 Scikit-Learn (Machine Learning)**

🔹 **Definition**: Scikit-Learn is the most widely used library for machine learning in Python, offering simple and efficient tools for data mining and analysis.

**Key Features:**

✅ Pre-built models: Linear Regression, Decision Trees, SVM, etc.  
✅ Feature selection, preprocessing, and dimensionality reduction  
✅ Model evaluation and hyperparameter tuning

**Installation:**

pip install scikit-learn

**Example Usage:**

from sklearn.linear\_model import LinearRegression

import numpy as np

# Sample data

X = np.array([[1], [2], [3], [4]])

y = np.array([2, 4, 6, 8])

# Train a linear regression model

model = LinearRegression()

model.fit(X, y)

# Make predictions

print(model.predict([[5]])) # Output: [10.]

**12.6 TensorFlow (Deep Learning & AI)**

🔹 **Definition**: TensorFlow is an open-source framework developed by Google for deep learning and AI applications.

**Key Features:**

✅ Neural networks for deep learning  
✅ GPU acceleration for high performance  
✅ Supports CNNs, RNNs, NLP, and image recognition

**Installation:**

pip install tensorflow

**Example Usage:**

import tensorflow as tf

# Define a simple neural network layer

layer = tf.keras.layers.Dense(units=1, input\_shape=[1])

# Compile and summarize the model

model = tf.keras.Sequential([layer])

model.compile(optimizer="sgd", loss="mean\_squared\_error")

print(model.summary())

**12.7 Flask (Web Development)**

🔹 **Definition**: Flask is a lightweight web framework for building web applications and RESTful APIs.

**Key Features:**

✅ Simple and lightweight  
✅ Built-in development server  
✅ Supports Jinja2 templating

**Installation:**

pip install flask

**Example Usage:**

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route("/")

def home():

return "Hello, Flask!"

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

**12.8 Django (Web Development)**

🔹 **Definition**: Django is a high-level web framework for rapid development and clean, pragmatic design.

**Key Features:**

✅ Built-in admin panel  
✅ Security features (XSS, CSRF protection)  
✅ ORM for database management

**Installation:**

pip install django

**Example Usage:**

django-admin startproject myproject

cd myproject

python manage.py runserver

**12.9 Requests (HTTP Requests & APIs)**

🔹 **Definition**: Requests is a simple HTTP library for making API requests and handling responses.

**Key Features:**

✅ Send GET, POST, PUT, DELETE requests  
✅ Handles JSON responses  
✅ Supports authentication

**Installation:**

pip install requests

**Example Usage:**

import requests

response = requests.get("https://jsonplaceholder.typicode.com/posts/1")

print(response.json()) # Print JSON response

**12.10 OpenCV (Computer Vision & Image Processing)**

🔹 **Definition**: OpenCV (Open Source Computer Vision) is a library for real-time image and video processing.

**Key Features:**

✅ Face and object detection  
✅ Image manipulation (filters, transformations)  
✅ Video processing

**Installation:**

pip install opencv-python

**Example Usage:**

import cv2

# Read and display an image

image = cv2.imread("image.jpg")

cv2.imshow("Image", image)

cv2.waitKey(0)

cv2.destroyAllWindows()