**Python Advanced Concepts – Quick Notes**

**1️ Exception Handling**

* **Used to manage runtime errors gracefully**
* **Blocks: try, except, else, finally**

**def divide\_numbers(a, b):**

**try:**

**result = a / b**

**print("Result:", result)**

**except ZeroDivisionError:**

**print("Error: Cannot divide by zero.")**

**except TypeError:**

**print("Error: Please provide numbers only.")**

**else:**

**print("Division successful.")**

**finally:**

**print("Execution complete.")**

**🔹 try: Code that may raise an error  
🔹 except: Handles specific exceptions  
🔹 else: Runs if no exception occurs  
🔹 finally: Always executes**

**2️ File Handling**

* **Enables reading/writing files for data persistence**
* **Common modes:** 
  + **'r': Read**
  + **'w': Write (overwrite)**
  + **'a': Append**
  + **'b': Binary**

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

**f.write("Hello, Python File Handling!")**

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

**print(f.read())**

**3️ Decorators**

* **Modify behavior of functions without changing their code**
* **Useful for logging, authentication, etc.**

**def decorator\_function(func):**

**def wrapper():**

**print("Before the function call")**

**func()**

**print("After the function call")**

**return wrapper**

**@decorator\_function**

**def say\_hello():**

**print("Hello!")**

**say\_hello()**

**4️ Unit Testing**

* **Tests individual components for correctness**
* **Uses unittest module**

**import unittest**

**def add(a, b):**

**return a + b**

**class TestMath(unittest.TestCase):**

**def test\_add(self):**

**self.assertEqual(add(2, 3), 5)**

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

**unittest.main()**

**5️ Popular Python Libraries**

| **Library** | **Purpose** |
| --- | --- |
| **NumPy** | **Numerical computations** |
| **Pandas** | **Data analysis & manipulation** |
| **Matplotlib** | **Data visualization** |
| **Scikit-learn** | **Machine learning** |
| **TensorFlow / PyTorch** | **Deep learning** |
| **Requests** | **HTTP requests** |
| **OS / Sys** | **OS-level operations** |

**6️ List Comprehensions with Conditions**

* **Concise way to build lists with logic**

**labels = ["Even" if i % 2 == 0 else "Odd" for i in range(1, 6)]**

**print(labels) # Output: ['Odd', 'Even', 'Odd', 'Even', 'Odd']**