**Modules and Libraries in Python**

In Python, **modules** and **libraries** are a way to organize and reuse code. A **module** is a single file that contains Python code, while a **library** is a collection of modules. Python has a wide range of built-in modules, called the **Python Standard Library**, which provides functionality for performing various tasks without needing to install third-party libraries.

Let's dive into two key aspects: the **Python Standard Library**.

**1. The Python Standard Library**

The **Python Standard Library** is a collection of modules and packages that come bundled with Python, offering a wide array of functionality. This library helps you with many tasks, such as file handling, string manipulation, operating system interfaces, networking, and more, without needing to install additional libraries.

**Key Features of the Python Standard Library:**

* **Built-in Modules**: Modules that come pre-installed with Python. You don't need to download or install them manually.
* **Cross-platform**: The standard library works across different operating systems (Windows, macOS, Linux).
* **Well-documented**: Most modules are thoroughly documented, making it easier for developers to understand their usage.

**Commonly Used Modules in the Python Standard Library:**

1. **os**:
   * Provides functions to interact with the operating system.
   * Example: Creating directories, listing files, working with paths.

import os

os.mkdir('new\_directory') # Create a new directory

print(os.listdir('.')) # List all files and directories in the current directory

1. **sys**:
   * Provides access to some variables used or maintained by the Python interpreter.
   * Example: Reading command-line arguments.

import sys

print(sys.argv) # List of command-line arguments

1. **math**:
   * Provides mathematical functions such as trigonometric functions, logarithms, etc.
   * Example: Calculating the square root.

import math

print(math.sqrt(16)) # Output: 4.0

1. **datetime**:
   * Provides classes for manipulating dates and times.
   * Example: Getting the current date and time.

import datetime

now = datetime.datetime.now()

print(now) # Output: Current date and time

1. **json**:
   * Used to parse and manipulate JSON data (often used for APIs or configuration files).
   * Example: Loading JSON from a file and parsing it.

import json

data = '{"name": "Alice", "age": 25}'

parsed\_data = json.loads(data)

print(parsed\_data['name']) # Output: Alice

1. **re**:
   * Provides regular expression matching operations.
   * Example: Searching for a pattern in a string.

import re

pattern = r'\d+' # Matches any sequence of digits

result = re.findall(pattern, 'There are 12 apples and 7 bananas')

print(result) # Output: ['12', '7']

1. **random**:
   * Provides functions for generating random numbers, selecting random items, etc.
   * Example: Generating a random number between 1 and 10.

import random

print(random.randint(1, 10)) # Output: A random number between 1 and 10

1. **collections**:
   * Contains specialized container datatypes such as namedtuples, deque, defaultdict, etc.
   * Example: Using Counter to count occurrences of elements in a list.

from collections import Counter

items = ['apple', 'banana', 'apple', 'orange', 'banana', 'apple']

count = Counter(items)

print(count) # Output: Counter({'apple': 3, 'banana': 2, 'orange': 1})

The **Python Standard Library** provides a wealth of functionality, which means you can often avoid reinventing the wheel by leveraging existing modules. It's essential to familiarize yourself with these libraries as they can save you time and effort while writing Python code.