Python

SARAVANA

Python Data Structures

Lists

Tuple

Dictionary

Sets

Immutability

Immutability means that the value of an object cannot be changed after it is created.

Tuples are immutable, while lists, dictionaries, and sets are mutable.

List Methods

`append()`: Adds an element to the end of the list.

`insert()`: Inserts an element at a specific index.

`remove()`: Removes the first occurrence of a specified value.

`pop()`: Removes and returns the element at a specific index (or the last element if no index is specified).

`sort()`: Sorts the list in ascending order.

`reverse()`: Reverses the order of elements in the list.

Many more…

Tuple Methods

Tuples have limited methods due to their immutability.

`count()` returns the number of occurrences of a specified value.

`index()` returns the index of the first occurrence of a specified value.

Set Methods

`add()`: Adds an element to the set.

`remove()`: Removes a specific element from the set.

`discard()`: Removes a specific element if it is present.

`union()`: Returns a new set containing all elements from two sets.

`intersection()`: Returns a new set containing elements present in both sets.

`difference()`: Returns a new set containing elements in the first set but not in the second set.

`symmetric_difference()`: Returns a new set containing elements in either set, but not both.

Many more…

Dictionary Methods

`keys()`: Returns a view of the dictionary's keys.

`values()`: Returns a view of the dictionary's values.

`items()`: Returns a view of the dictionary's key-value pairs.

`get()`: Returns the value for a specified key, or a default value if the key is not found.

`update()`: Updates the dictionary with the key-value pairs from another dictionary.

Many more...

List vs Tuple

Lists

Ordered collection of items.

Mutable: Can be changed after creation.

Defined using square brackets [ ].

Used for storing and manipulating collections of data that can change.

Tuples

Ordered collection of items.

Immutable: Cannot be changed after creation.

Defined using parentheses ().

Used for storing data that should not be modified.

Set Operations

Set operations allow you to manipulate and combine sets to create new sets. Python's built-in `set` data type supports these operations.

Common Set Operations

Union: Combines all unique elements from two sets.

Intersection: Returns elements present in both sets.

Difference: Returns elements present in the first set but not in the second set.

Symmetric Difference: Returns elements present in either set, but not both.

Subset: Checks if one set is a subset of another.

Numeric Types and String Types

Numeric types:

Integers: Whole numbers (e.g., 42, -10)

Floating-point numbers: Numbers with decimal points (e.g., 3.14, 2.5)

Complex numbers: Numbers with a real and imaginary part (e.g., 3+2j)

String types:

Sequences of characters.

Can be accessed using indexing and slicing.

Common operations: concatenation, length, finding substrings, etc.

Random Number Generation

Python's `random` module provides functions for generating random numbers.

`random.randint(a, b)`: Returns a random integer between `a` and `b` (inclusive).

`random.random()`: Returns a random float between 0.0 and 1.0.

`random.choice(sequence)`: Returns a random element from a sequence.

NumPy offers more advanced random number generation capabilities and is often preferred for numerical computations.

`np.random.rand(d0, d1, ..., dn):` Creates an array of random floats in the interval [0, 1).

`np.random.randint(low, high=None, size=None):` Returns random integers from low (inclusive) to high (exclusive).

Matrix Operations (Basic)

While Python doesn't have built-in matrix data types, you can represent matrices as lists of lists.

Basic operations like addition, subtraction, and multiplication can be implemented using nested loops.

Libraries like NumPy provide efficient matrix operations.

Basic Matrix Operations

Matrix Multiplication

NumPy for Efficient Matrix Operations

List Comprehension

A concise way to create lists based on existing lists or other iterable objects.

```
numbers = [1, 2, 3, 4, 5]
squares = [x**2 for x in numbers]
print(squares)
```

Understanding Functions

Functions without parameters: Don't require any input values.

Functions with parameters: Accept input values.

Local and Global Variables

Local variables: Defined within a function and accessible only within that function.

Global variables: Defined outside of any function and accessible from anywhere in the program.

Lambda Functions

Anonymous functions defined using the `lambda` keyword.

Used for short, simple functions.

Understanding Functions

Calling Functions, Parameters, and Return Values

To call a function, use its name followed by parentheses.

Pass arguments (values) to the function through parameters.

A function can return a value using the `return` statement.

Exercise

Write a function to calculate the factorial of a number.

Create a function to check if a number is prime.

Use list comprehension to create a list of even numbers from 1 to 20.

Write a lambda function to square a number.

Introduction to External Libraries

What are external libraries? Pre-written code packages that provide additional functionalities to Python.

Benefits

Time Efficiency: Reusing pre-written code saves development time.

Improved Performance: Libraries often provide optimized implementations for specific tasks.

Access to Specialized Tools: Libraries offer functions and features tailored to particular domains.

Code Reusability: Promote code sharing and collaboration.

Exercise

Create a list of numbers and perform various operations on it (append, insert, remove, sort, reverse).

Implement linear and binary search functions.

Generate random numbers and perform calculations.

Create a simple matrix and perform basic operations.

Modules & Packages

NumPy

Fundamental package for numerical computing.

Efficient array operations, linear algebra, and random number generation.

Pandas

Data manipulation and analysis library.

Offers data structures like Series and DataFrames.

Handles importing, cleaning, and analyzing data.

Matplotlib

Data visualization library.

Create various types of plots (line, bar, scatter, histograms, etc.).

Customize plot appearance.

Advanced Module Concepts

Package management:

`pip`: Python's package installer.

`conda`: Package and environment manager (often used with Anaconda).

Virtual environments: Isolate project dependencies.

Module distribution:

Creating and distributing your own packages (using `setuptools`).

Uploading packages to PyPI (Python Package Index).

Module structure:

Organizing code into packages and modules for better maintainability.

Basic Data Analysis with Pandas

Pandas is a powerful Python library designed for data manipulation and analysis. It provides high-performance, easy-to-use data structures and data analysis tools.

Core Data Structures

Series: One-dimensional labeled array capable of holding any data type (integers, floats, strings, objects, etc.).

DataFrame: Two-dimensional labeled data structure with columns of potentially different types.

Basic information

`df.head()`: Displays the first few rows.

`df.tail()`: Displays the last few rows.

`df.shape`: Returns the number of rows and columns.

`df.info()`: Info about the DataFrame, including column names, data types, and null values.

`df.describe()`: Generates descriptive statistics for numerical columns.

`df['column_name']`: Selects a specific column.

`df.loc[row_label]` or `df.iloc[row_index]`: Selects rows based on labels or indices.

`df.head(n)`: Selects the first n rows.

`df.tail(n)`: Selects the last n rows.

Data Cleaning

Handling missing values:

`df.isnull()`: Checks for missing values.

`df.dropna()`: Removes rows or columns with missing values.

`df.fillna()`: Fills missing values with a specified value.

Removing duplicates:`df.duplicated()`: Checks for duplicate rows.`df.drop_duplicates()`: Removes duplicate rows.

`df.mean()`, `df.median()`, `df.std()`, `df.min()`, `df.max()`: Calculate summary statistics.

`df.groupby()`: Groups data based on one or more columns and applies aggregation functions.

Thanks