Python Fundamentals 2 - Glossary

Glossary

PF2

Aliases

Another term for references; when multiple variables point to the same object in memory.

ASCII

Character encoding standard used by Python for string comparisons and lexicographic ordering. Numbers come before letters, and uppercase letters come before lowercase letters in ASCII order.

Case-sensitive

Property where uppercase and lowercase letters are treated as different characters. Python strings are case-sensitive, so "Gene" and "gene" are considered different strings.

Concatenation

Joining strings together using the `+` operator to create new strings. Commonly used for creating unique identifiers or combining text data.

Copy

An independent duplicate of a list created using methods like `.copy()` or slice notation `[:]`. Changes made to a copy do not affect the original list, making it essential for preserving experimental data whilst performing modifications.

Data Structures

Structures for storing multiple values together, such as lists, tuples, sets, and dictionaries.

Deep Copy

A complete copy where nested mutable objects are also copied independently, requiring the `copy.deepcopy()` function from the copy module.

Element

See **Item**.

Exclusive

In slicing, the end index does not include the element at that position.

Explicit

Clearly stated or defined code that shows exactly what is happening, following the Zen of Python principle "Explicit is better than implicit". For example, using parentheses `()` when creating tuples even when they're not strictly required.

Immutable

A data type that cannot be changed after creation. When you "modify" an immutable object (such as a string or tuple), Python creates a completely new object rather than changing the original.

Implicit

Code behaviour that is inferred or understood without being directly stated, such as creating tuples without parentheses using comma separation.

Inclusive

In slicing, the start index includes the element at that position.

Index

The position of an element in a list, starting from 0 for the first element.

Item

An individual value stored within a data structure such as a list. Also referred to as an element.

Length

The number of items/elements in a data structure, found using the `len()` function.

Lists

One of the most common data structures in Python, providing a flexible structure that can hold mixed data types in square brackets `[]`, separated by commas.

Membership Testing

Checking whether a specific value exists in a data structure using `in` and `not in` operators.

Method

Special functions that belong to a specific data type in Python, called using dot notation after the object (e.g., `my_list.append(5)`). Methods are dependent on their object and cannot be called independently.

Mutable

A data type that can be changed after creation. You can modify the contents directly without creating a new object. Lists are mutable in Python - you can add, remove, or change elements in place.

Negative Indexing

A system for accessing elements from the end of a list, where -1 refers to the last element, -2 to second-to-last, etc. Particularly useful for accessing recent data without knowing the exact length.

Nested

A data structure that contains other data structures of the same type within it. In the context of lists, a nested list is a list that contains other lists as its elements, creating a multi-dimensional structure similar to a table with rows and columns.

Object

In Python, everything is an object - a container that holds data along with built-in capabilities (methods). Every piece of data you work with is an object.

Object-oriented Programming

A programming paradigm where everything is treated as objects with data and methods. Python is described as object-oriented because all data types are objects that can be measured and manipulated.

Packing

The process of combining multiple values into a single tuple, either explicitly with parentheses `(a, b, c)` or implicitly with comma separation `a, b, c`. Python automatically interprets comma-separated values as a tuple.

Reference

An alias created when you assign one list variable to another using `=`. Both variables point to the same list object in memory, so modifying one affects the other. For example, if `backup_data = original_data`, then `backup_data` is a reference to `original_data`, not a separate list.

Set

An unordered collection of unique elements in Python, created using curly braces `{}` or the `set()` function. Particularly useful for removing duplicates from data and performing mathematical operations to compare datasets.

Shallow Copy

A copy that creates a new list, but the items in the new list are still references to the original items. If those items are mutable (like nested lists), changing them in the copy will affect the original.

Singleton

A tuple containing only one element, which requires a trailing comma `(item,)` to distinguish it from parentheses used for grouping. Without the comma, Python interprets the parentheses as grouping rather than a tuple.

Slice

A portion of a list extracted using slice notation, creating a new list containing the specified elements.

Slicing

Extracting portions of lists using `[start:end]` notation to retrieve multiple elements at once. The start index is inclusive, the end index is exclusive.

String

A sequence of characters in Python, created using single or double quotation marks. Strings are immutable, case-sensitive, and can be indexed and sliced like lists.

Sub-string

A smaller string contained within a larger string. Can be searched for using methods like `.find()` or the `in` operator.

Tuple

An immutable sequence data type in Python that stores ordered collections of items in parentheses `()`, whose contents cannot be modified after creation. Particularly useful for storing data that should remain constant throughout analysis, such as experimental conditions or genetic codes.

Unique

Property where each element can only appear once; duplicates are automatically removed. This is a fundamental characteristic of sets in Python.

Unpacking

The process of extracting values from a tuple and assigning them to individual variables, making code more readable and allowing work with individual elements. For example, `x, y, z = coordinates` extracts three values from a coordinates tuple.

Unordered

Collections that do not have a defined sequence or index; elements cannot be accessed by position. Sets are unordered collections, unlike lists and tuples which maintain element order.

Zero-based Indexing

A system where the first element of a sequence is at position 0, used by Python for memory efficiency and continuation from other coding languages like C.