Alexis Ilogon

Defining a List

A list in Python is a versatile and fundamental data structure that allows you to store and organize data sequentially. Lists are mutable, meaning their elements can be changed after creation. They are defined using square brackets [] and can hold any data type, including numbers, strings, or even other lists.

```
my_list = [1, 2, 3, 4, 5]
```

List Syntax

The syntax for creating a list is straightforward: elements are separated by commas and enclosed within square brackets. Lists can contain a mix of data types or even objects of different types.

```
my_list = ["apple", "banana", "cherry"]
```

Accessing List Elements

Elements within a list can be accessed using indexing. Python uses zero-based indexing, meaning the first element is accessed with index 0, the second with index 1, and so on. Negative indexing allows you to access elements from the end of the list, with -1 representing the last element.

```
print(my_list[0]) # Output: "apple"
print(my_list[-1]) # Output: "cherry"
```

Loop through a List

You can iterate over the elements of a list using a for loop. This allows you to perform operations on each element of the list sequentially.
python

```
for item in my_list:
 print(item)
```

List Length

You can determine the number of elements in a list using the len() function. This function returns the length of the list, i.e., the number of elements it contains.
python

```
print(len(my_list)) # Output: 3
```

Add Items in the List

You can add items to a list using various methods. The append() method adds an element to the end of the list, while insert() allows you to insert an element at a specific index.
python

```
my_list.append("orange")
my_list.insert(1, "banana")
```

Remove Item from a List

Removing items from a list can be achieved using methods like remove(), pop(), or slicing. The remove() method removes the first occurrence of a specified value, pop() removes the element at a given index and returns it, and slicing allows you to remove a range of elements.

```
my_list.remove("banana")
popped_item = my_list.pop(0)
```

The list() Constructor

You can create a list using the list() constructor, which can take an iterable (such as another list, tuple, string, etc.) and convert it into a list.
python

```
new_list = list(("apple", "banana", "cherry"))
```

List Methods

Python lists offer a variety of built-in methods to manipulate and work with lists efficiently. These methods include append(), remove(), pop(), clear(), index(), count(), sort(), reverse(), and more.

```
my_list.append("grape")
my_list.remove("cherry")
my_list.sort()
```

Nested Lists

Lists in Python can contain other lists, creating nested structures. This enables the representation of more complex data structures, such as matrices or hierarchical data.

```
nested_list = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```