Newton School of Technology

12

Lists in Python

by Gladden Rumao

C01: Problem Solving with Programming

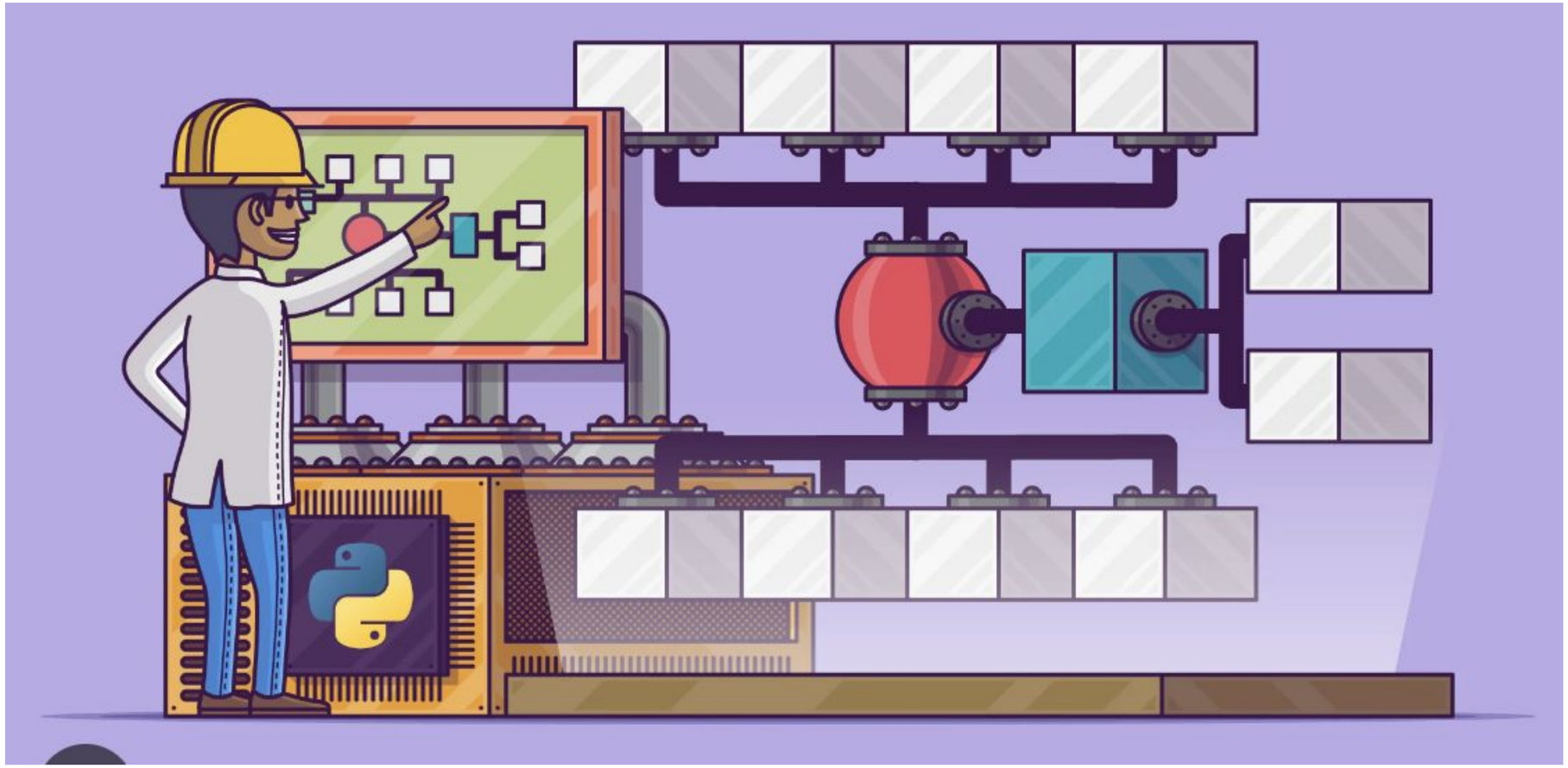
Newton School of Technology

Join the lecture online on your dashboard

Start Lecture Recording

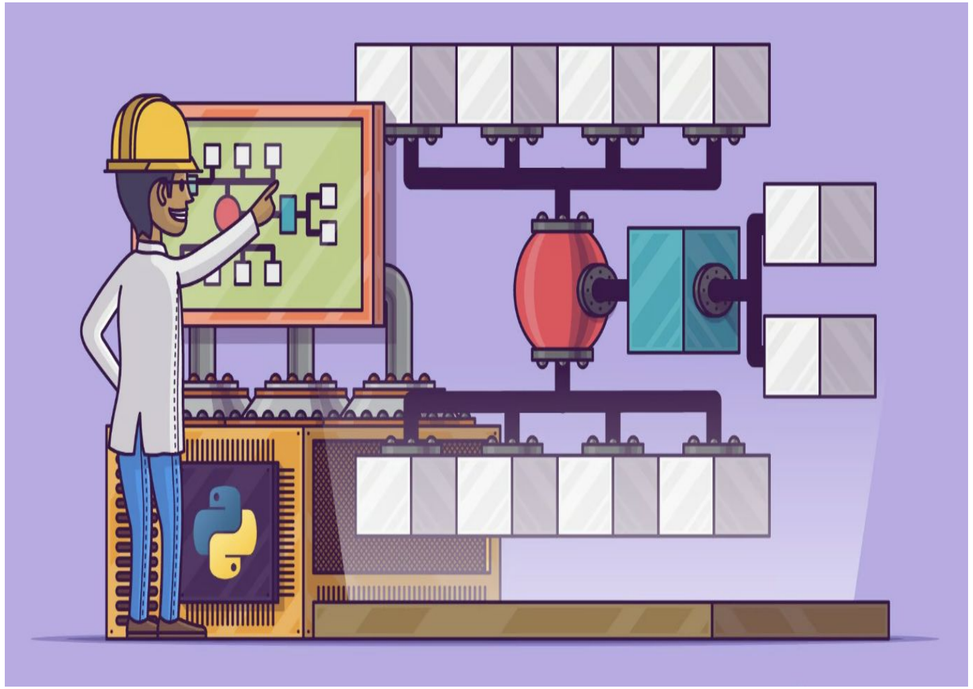
Introduction to Data Structures :

Newton School of Technology

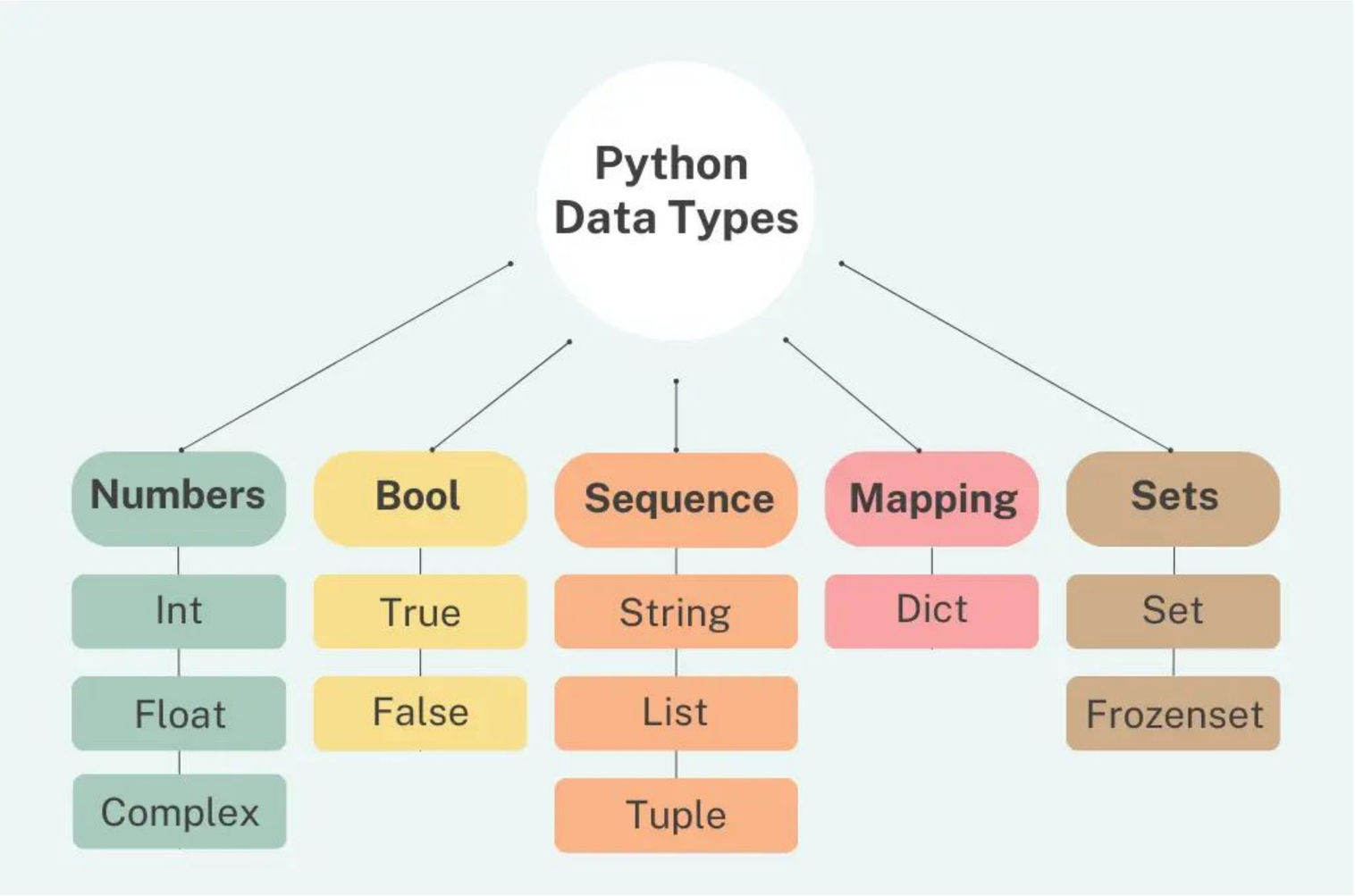


Definition :

Data structures in Python are tools that efficiently **store** and **organize** data to support effective **manipulation** and **retrieval** operations.



Introduction to Lists :



Introduction to Lists :

List Real life example

A list of **groceries** we have to buy can be taken as an example !

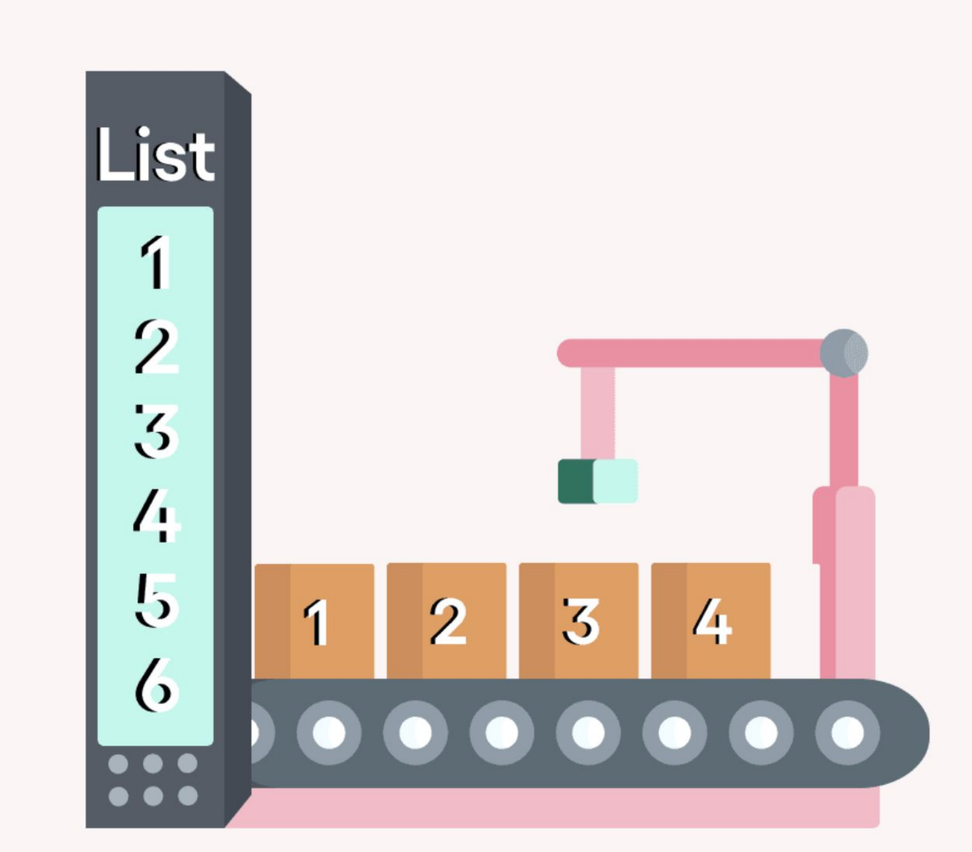
To this **list** we can-

- **Append** more items we need to buy.
- **Remove** items we do not want to buy.

We can also go through the list to see what items are **already present** on it.

Definition :

A **list** in Python is an ordered collection of **mutable** elements, allowing flexible modification and **access** operations.



What can a list contain ?

A shopping list might contain **multiple kinds** items listed on it.

The **list** might have-

Fruits

Vegetables

Detergent,Soap etc.

What can a list contain ?

Newton School of Technology

Similarly a Python **list** might have elements of **all data types** like String, Integer, floating point etc.

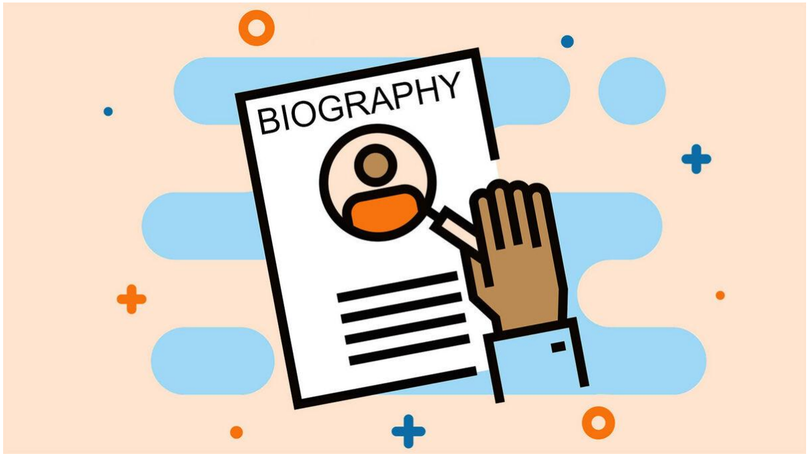
```
#List with elements of only one (string) data type
list1 = ["This", "is", "a", "list"]

#List with elements of different data type
list2 = ["This", "is", "list", 2.0]

#Nested list
list3 = ["List", 3, "has a", ["inner", "list"]]
```

What can a list contain ?

Let’s say we have a **list** having **information** on a **person** working for newton school

This **list** is going to contain his-

- Name (String)
- Age (Integer)
- Email address (String)
- Is the person active ? [True/False] (boolean)



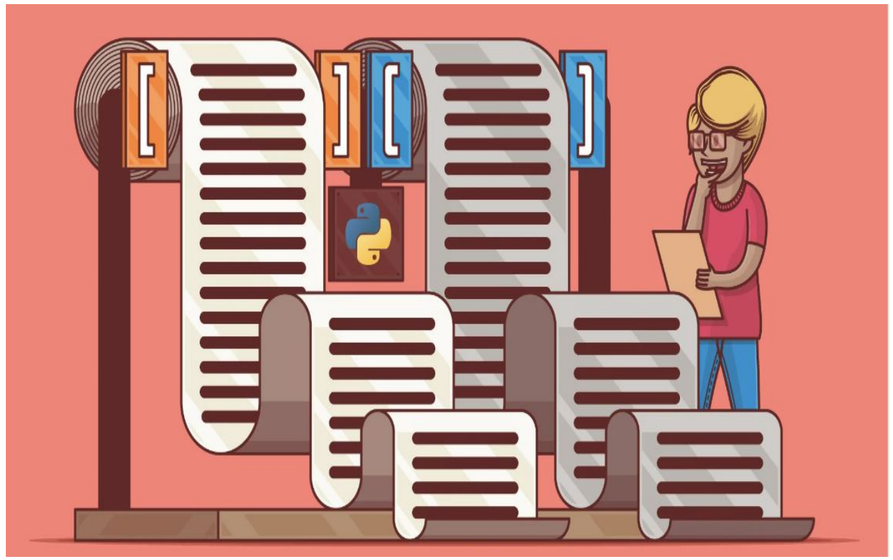
As you can see, **elements** on this list have **multiple data types**.

Creating a list :

To create a **list** in Python, you use **square brackets** [ ] and separate elements with **commas**.

Syntax:

list_name= [element1,element2,element3,.....]



Creating a list :

Here are few **examples** of lists in **python**.

```
empty_list = []  # Create an empty list

numbers = [1, 2, 3, 4, 5]  # List of integers

fruits = ['apple', 'banana', 'cherry']  # List of strings

mixed_list = [1, 'hello', 3.14, True]  # List with elements of different data types

nested_list = [[1, 2], ['a', 'b', 'c']]  # List containing other lists
```

Newton School of Technology

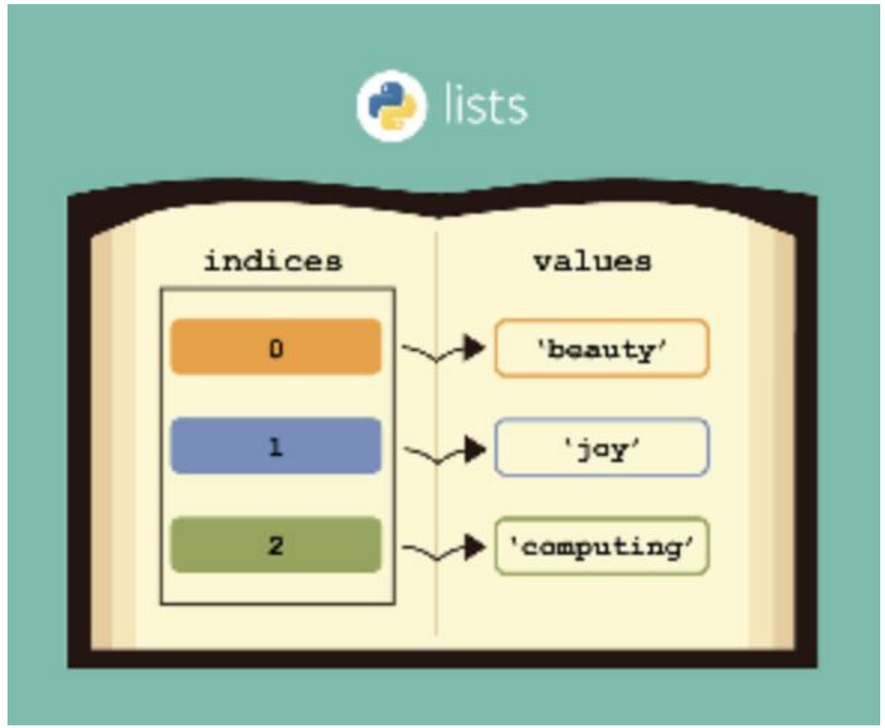
Accessing elements in a List

Indexing in a List

Newton School of Technology

In Python lists, indexing refers to accessing elements by their position.

- Indexing **starts at 0** for the first element, 1 for the second, and so on.
- You can use **square brackets** [ ] to specify the index.



For Example: in the adjacent figure **list_name[1]** is ‘joy’.

More examples on Indexing :

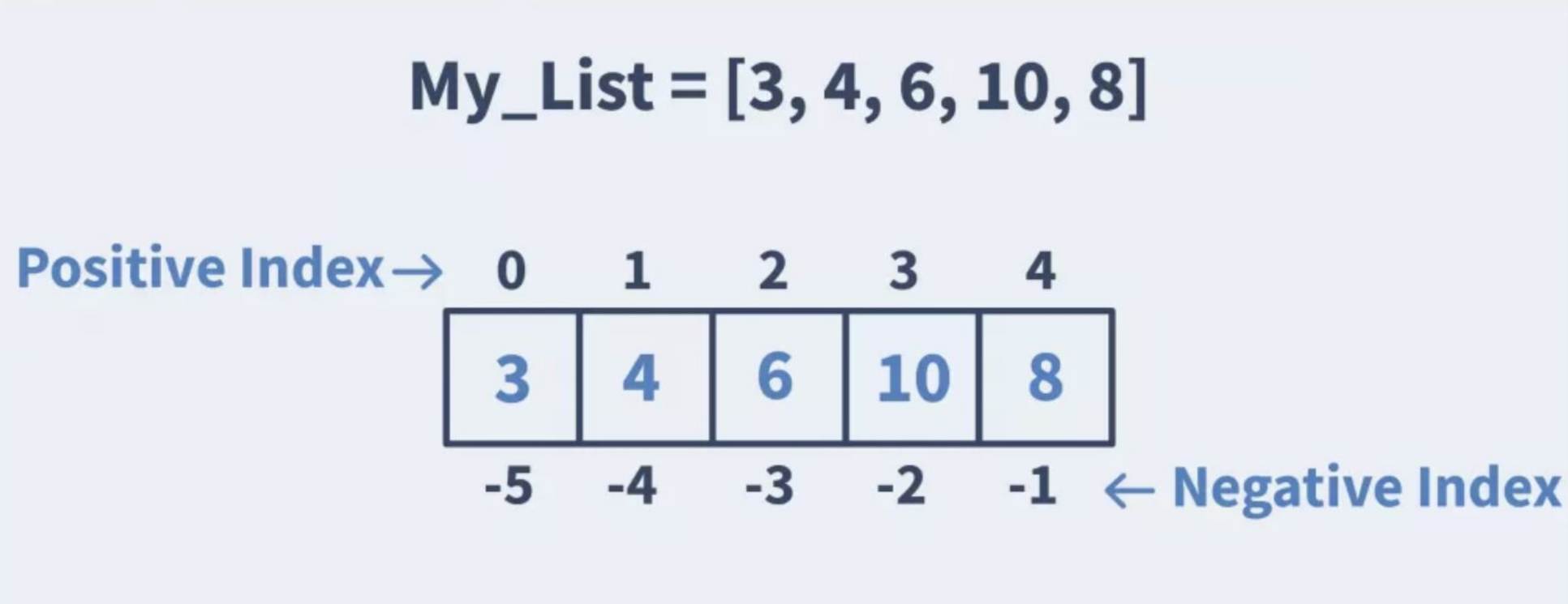
Newton School of Technology

```python
# Example 1: Accessing elements by index
fruits = ['apple', 'banana', 'cherry']
print(fruits[0])    # Output: 'apple'
print(fruits[1])    # Output: 'banana'
print(fruits[2])    # Output: 'cherry'

# Example 2: Modifying elements by index
numbers = [10, 20, 30, 40, 50]
numbers[1] = 25
print(numbers)      # Output: [10, 25, 30, 40, 50]
```

Negative indexing

Newton School of Technology

Negative indices count **backward** from the **end of the list**, starting with **-1** for the l**ast element**, -2 for the second last, and so forth. It provides a **convenient** way to access elements **relative to the end** of the list.



Examples of Negative Index

Newton School of Technology

```python
fruits = ['apple', 'banana', 'cherry', 'date', 'elderberry']

print(fruits[-1])   # Output: 'elderberry' (last element)
print(fruits[-2])   # Output: 'date' (second last element)
print(fruits[-3])   # Output: 'cherry' (third last element)
```

Slicing in Python

Newton School of Technology

Slicing in Python allows you to **extract a subset** of elements from a list, string, or other sequence types. It uses a concise syntax [start:end:step] within **square brackets**.

Slicing Examples

Newton School of Technology

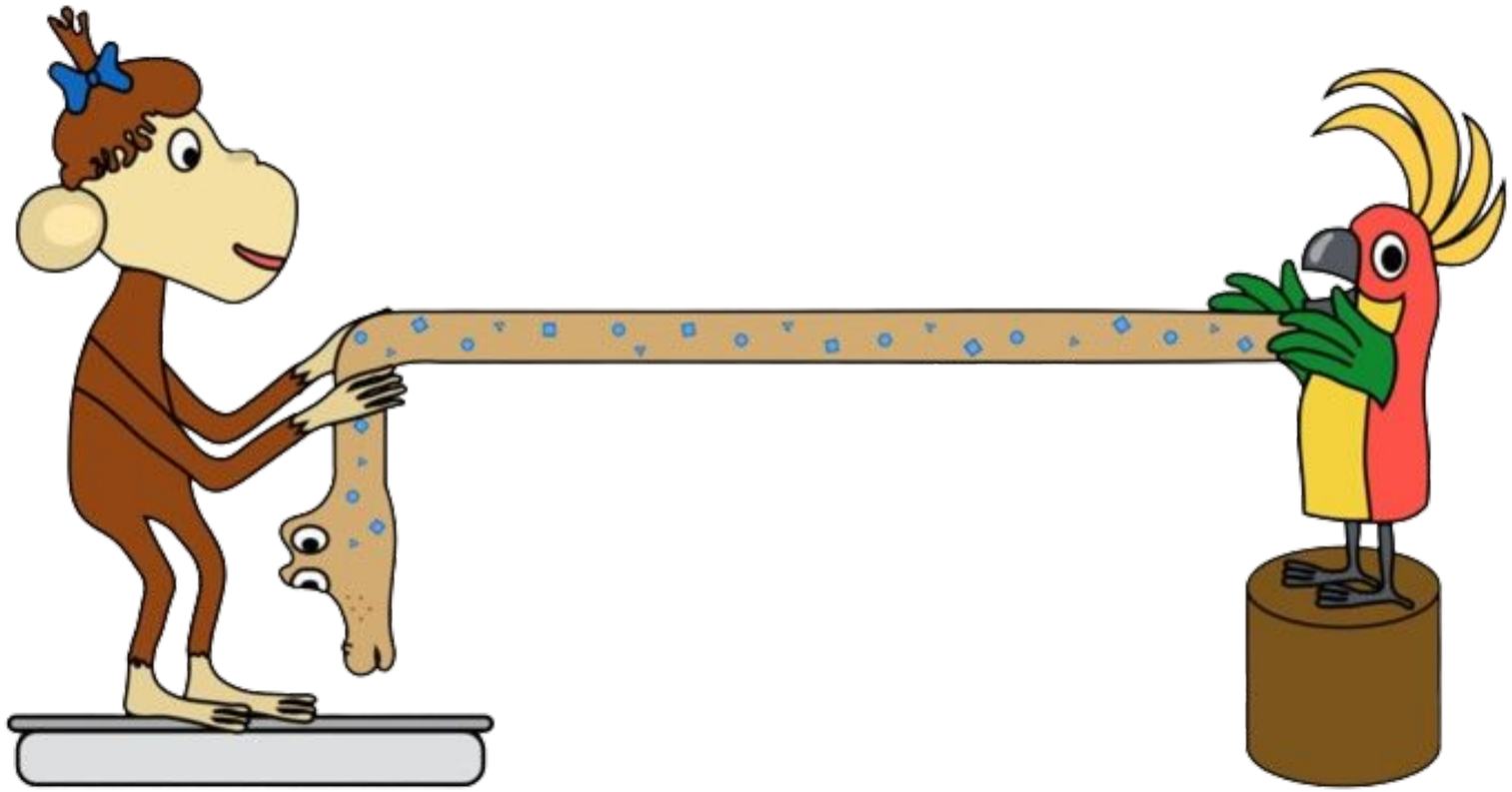
python Copy code

```python
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# Slice from index 2 to 5 (exclusive)
slice1 = numbers[2:5]
print(slice1)   # Output: [3, 4, 5]

# Slice from start to index 7 (exclusive)
slice2 = numbers[:7]
print(slice2)   # Output: [1, 2, 3, 4, 5, 6, 7]

# Slice from index 3 to end
slice3 = numbers[3:]
print(slice3)   # Output: [4, 5, 6, 7, 8, 9, 10]

# Slice with negative indices (from index -4 to -1)
slice4 = numbers[-4:-1]
print(slice4)   # Output: [7, 8, 9]

# Slice with a step of 2 (every second element)
slice5 = numbers[::2]
print(slice5)   # Output: [1, 3, 5, 7, 9]
```

Use len() function

Newton School of Technology

In Python, the **len()** function is used to determine the **number of elements** in a sequence.



Newton School of Technology

Examples using len():

python

```python
numbers = [1, 2, 3, 4, 5]
print(len(numbers))  # Output: 5

fruits = ['apple', 'banana', 'cherry']
print(len(fruits))   # Output: 3

empty_list = []
print(len(empty_list))  # Output: 0
```

Newton School of Technology

Modifying Lists

Updating a List :

Newton School of Technology

Updating an element at a specific **index** in a Python list involves assigning a new value to that index using the **assignment operator** (=). Here's how you can **update** an index in a **list**:

```python
# Original list
my_list = [10, 20, 30, 40, 50]

# Update element at index 2 (3rd element) to a new value
my_list[2] = 35

# Print updated list
print(my_list)  # Output: [10, 20, 35, 40, 50]
```

Adding elements to a List

Newton School of Technology

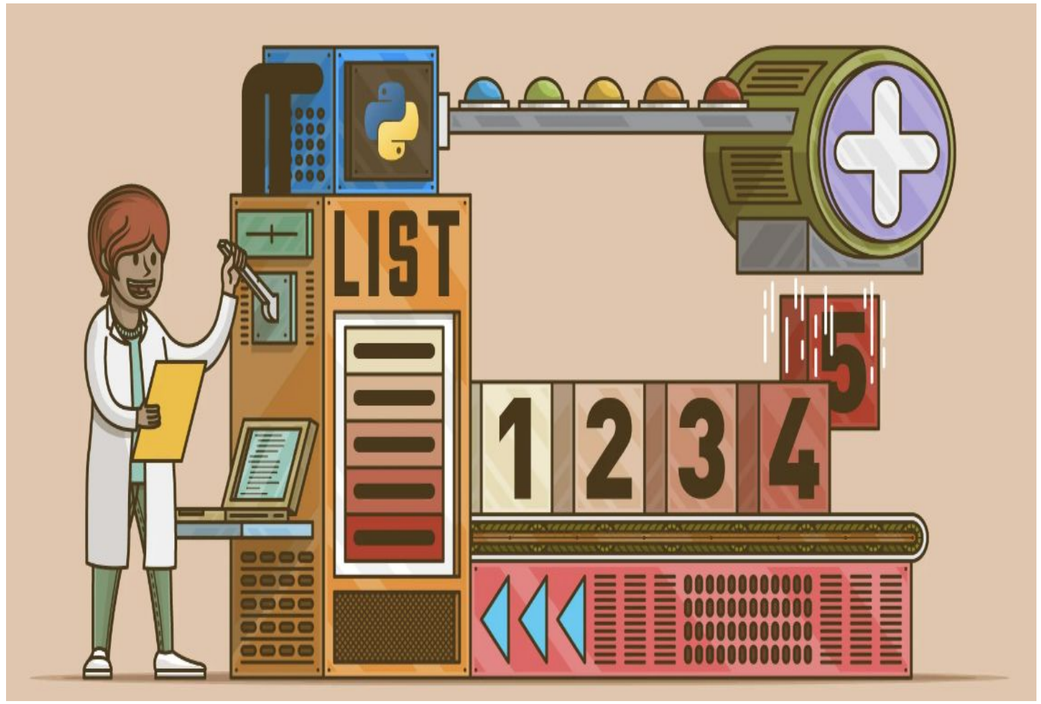
.append()-
The **.append()** method adds a single element to the **end** of a list.

Syntax
list_name.append(element)

Example:

python

```
fruits = ['apple', 'banana', 'cherry']
fruits.append('date')
print(fruits)  # Output: ['apple', 'banana', 'cherry', 'date']
```



Adding elements to a List

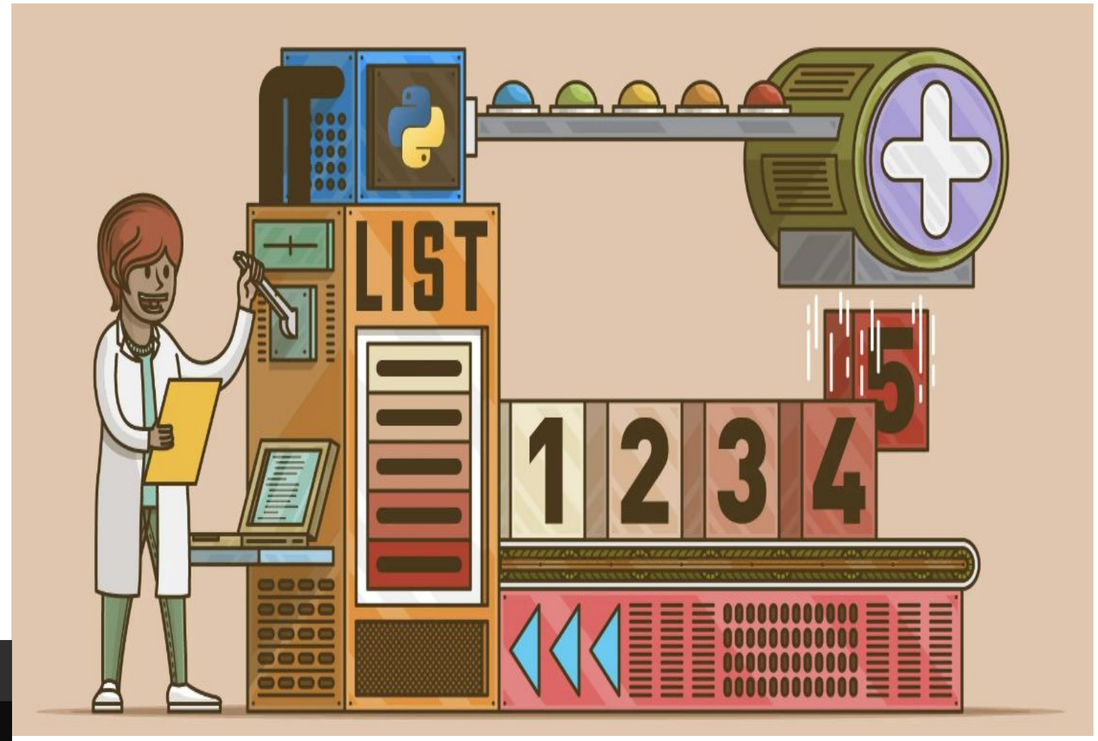
Newton School of Technology

.insert()-
The **.insert()** method inserts a new element at a **specified index** in the list.

Syntax
list_name.insert(index,element)

Example:

```python
fruits = ['apple', 'banana', 'cherry']
fruits.insert(1, 'orange')
print(fruits)  # Output: ['apple', 'orange', 'banana', 'cherry']
```



Adding elements to a List

Newton School of Technology

.extend()-

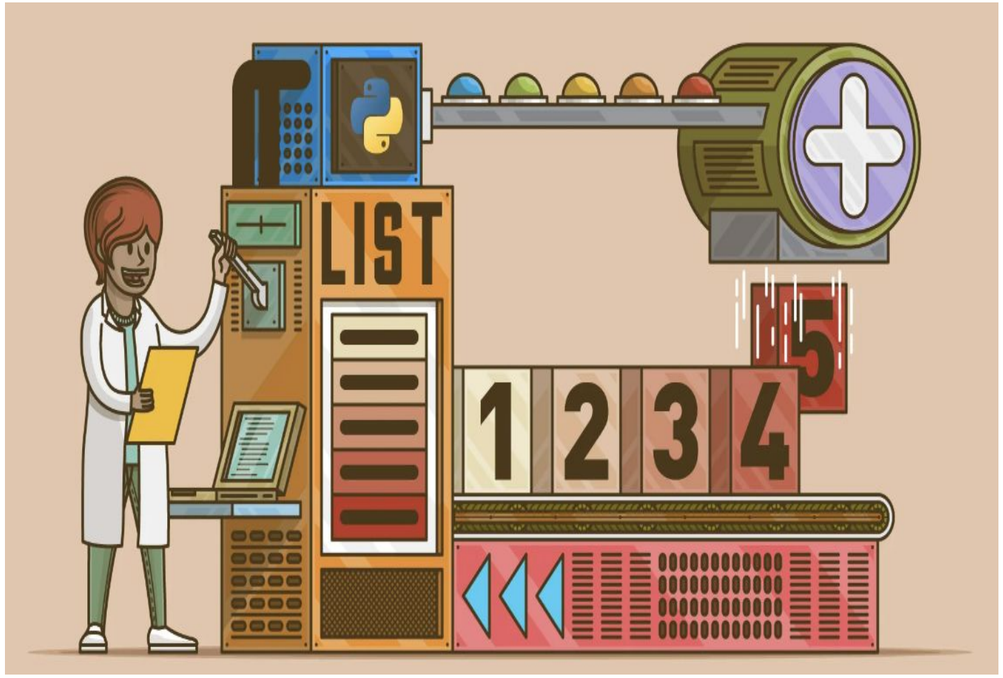
The **.extend()** method **appends** elements from an **iterable** (such as a list, tuple, or string) to the **end** of the list.

Syntax

list_name.extend(iterable)

Example:

```
fruits = ['apple', 'banana', 'cherry']
more_fruits = ['date', 'elderberry']
fruits.extend(more_fruits)
print(fruits)  # Output: ['apple', 'banana', 'cherry', 'date', 'elderberry']
```



Removing elements from List

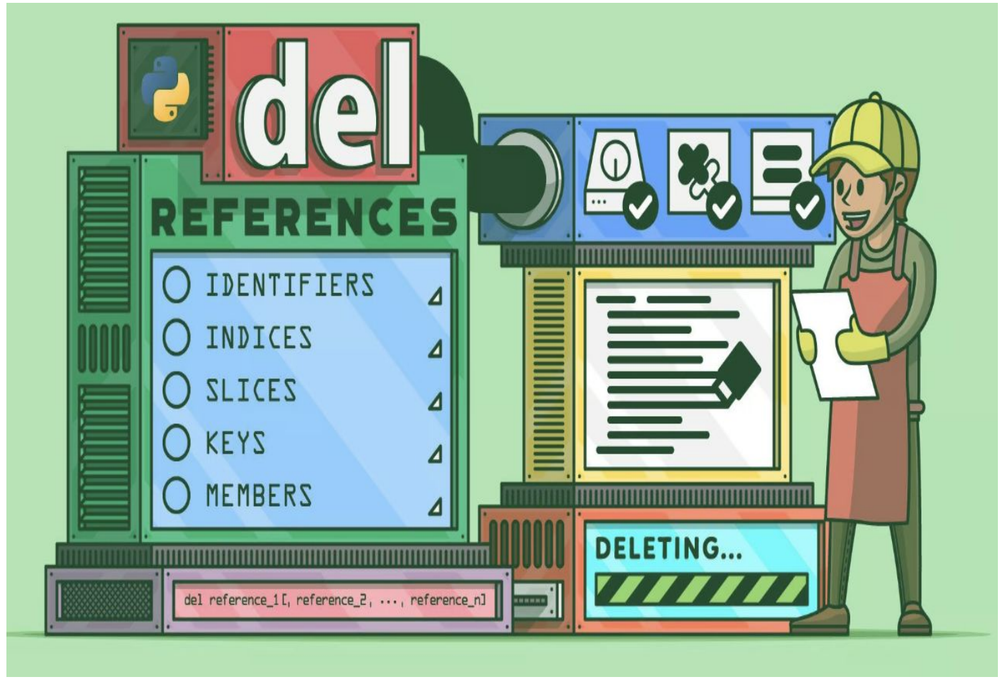
Newton School of Technology

.remove()-
The .**remove()** method removes the **first occurrence** of a specified **element** from the list.

Syntax
list_name.remove(element)

Example:

```python
fruits = ['apple', 'banana', 'cherry', 'banana']
fruits.remove('banana')
print(fruits)  # Output: ['apple', 'cherry', 'banana']
```



Removing elements from List

Newton School of Technology

.pop()-

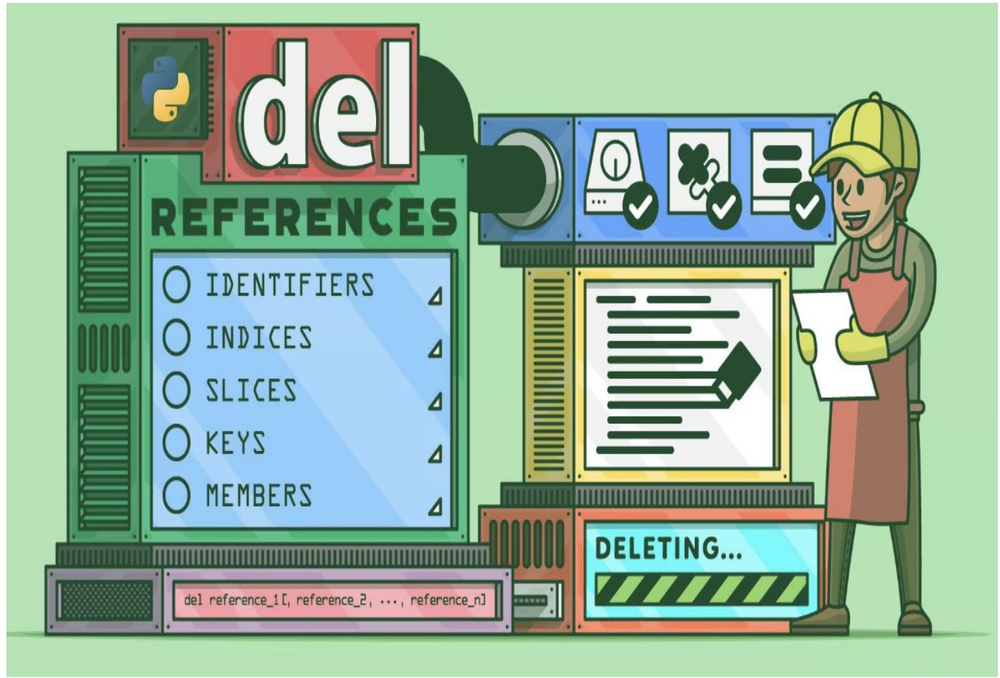
The **.pop()** method **removes and returns** the element at the specified index. If **no index** is specified, it removes and returns the **last element** in the list.

Syntax

list_name.pop(index)

Example:

```python
fruits = ['apple', 'banana', 'cherry']
popped_fruit = fruits.pop(1)
print(popped_fruit)  # Output: 'banana'
print(fruits)        # Output: ['apple', 'cherry']
```



Removing elements from List

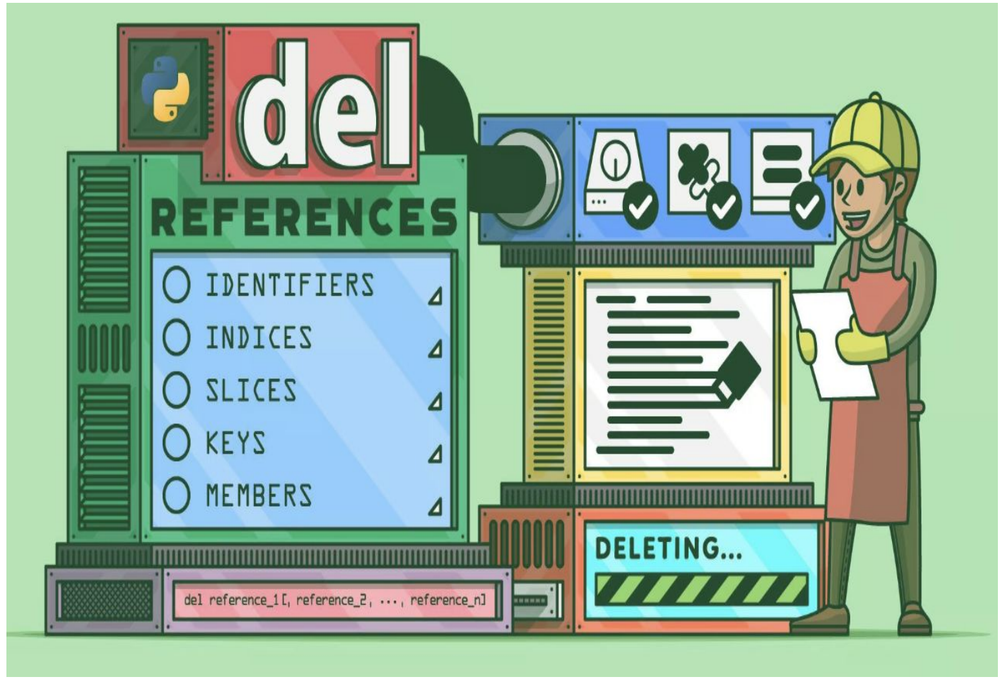
Newton School of Technology

.clear()-
The **.clear()** method removes **all elements** from the list, leaving it **empty**.

Syntax
list_name.clear()

Example:

```python
fruits = ['apple', 'banana', 'cherry']
fruits.clear()
print(fruits)  # Output: []
```



Thank You!