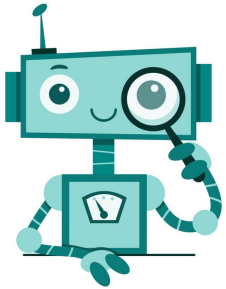
Artificial Intelligence (AI)



Week-3

Agenda

- Python Data Structures
- Types of Data Structures
- List

Python Data Structures

Python data structures are “containers” that organize and group data according to type. They are the fundamental constructs around which you build your programs, providing a particular way of organizing data so it can be accessed efficiently, depending on your use case.

Types of Data Structures

Generally, data structures can be divided into two categories in computer science: primitive and non-primitive data structures. The former are the simplest forms of representing data, whereas the latter are more advanced: they contain primitive data structures within more complex data structures for special purposes.

Primitive Data Structures

Primitive data structures are the basic data types that are built into the language and provide the building blocks for data manipulation. These include:

- String: A sequence of characters, used to represent text.
- Integer (int): Whole numbers without any fractional part.
- Float: Numbers that include a decimal point.
- Boolean: A data type that can hold one of two possible values: True or False.

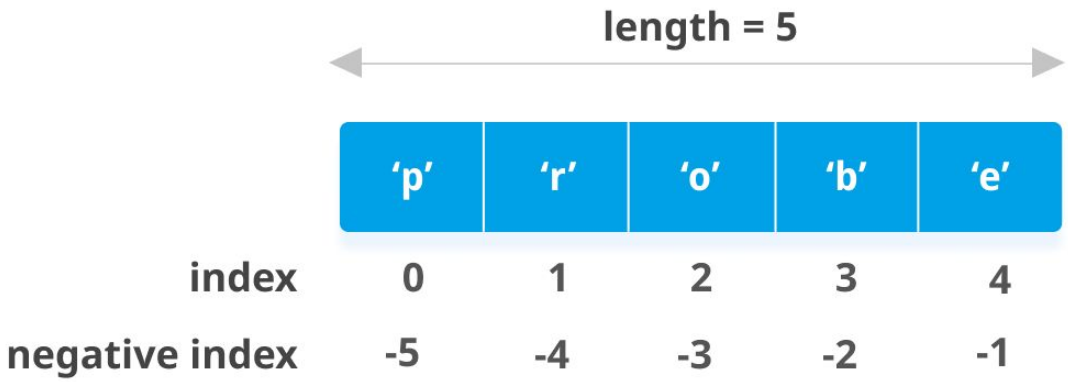
Non-Primitive Data Structures

Non-primitive data structures are more complex and can store multiple items. They are built from primitive data types and offer various ways to store and organize data. These include:

- List: An ordered collection of items that can be of different data types. Lists are mutable, meaning they can be changed after creation.
- Tuple: Similar to lists, but immutable. Once a tuple is created, it cannot be modified.
- Dictionary: An unordered collection of key-value pairs. Dictionaries are indexed by keys, which can be any immutable type.
- Set: An unordered collection of unique items. Sets are mutable and do not allow duplicate elements.

Sequence Type - Lists

The most basic data structure in Python is the **sequence**.



Lists

Creating a list is as simple as putting different comma-separated values between square brackets. For example −

```
list1 = [1,2,3,4,5]
list2 = ["a", "b", "c", "d"]
list3 = ["AI", "ML", 1999, 2024]
```

Access List Items

- List items are indexed and you can access them by referring to the index number:

Access Items

```
thislist = ["apple", "banana", "cherry"]
print(thislist[1])
```

```
banana
```

Negative Indexing

```
thislist = ["apple", "banana", "cherry"]
print(thislist[-1])
```

```
cherry
```

Range of Negative Indexes

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[-4:-1])
```

```
['orange', 'kiwi', 'melon']
```

Access List Items

Range of Indexes

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[2:5])
```

```
['cherry', 'orange', 'kiwi']
```

By leaving out the start value, the range will start at the first item:

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[:4])
```

```
['apple', 'banana', 'cherry', 'orange']
```

By leaving out the end value, the range will go on to the end of the list:

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[2:])
```

```
['cherry', 'orange', 'kiwi', 'melon', 'mango']
```

Change List Items

Change Item Value

```
thislist = ["apple", "banana", "cherry"]
thislist[1] = "blackcurrant"
print(thislist)
```

```
['apple', 'blackcurrant', 'cherry']
```

Insert Items

```
thislist = ["apple", "banana", "cherry"]
thislist.insert(2, "watermelon")
print(thislist)
```

```
['apple', 'banana', 'watermelon', 'cherry']
```

Change a Range of Item Values

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "mango"]
thislist[1:3] = ["blackcurrant", "watermelon"]
print(thislist)
```

```
['apple', 'blackcurrant', 'watermelon', 'orange', 'kiwi', 'mango']
```

Change List Items

If you insert more items than you replace, the new items will be inserted where you specified, and the remaining items will move accordingly:

```
thislist = ["apple", "banana", "cherry"]
thislist[1:2] = ["blackcurrant", "watermelon"]
print(thislist)
```

```
['apple', 'blackcurrant', 'watermelon', 'cherry']
```

If you insert less items than you replace, the new items will be inserted where you specified, and the remaining items will move accordingly:

```
thislist = ["apple", "banana", "cherry"]
thislist[1:3] = ["watermelon"]
print(thislist)
```

```
['apple', 'watermelon']
```

Add List Items

Append Items

```
thislist = ["apple", "banana", "cherry"]
thislist.append("orange")
print(thislist)
```

```
['apple', 'banana', 'cherry', 'orange']
```

Insert Items

```
thislist = ["apple", "banana", "cherry"]
thislist.insert(1, "orange")
print(thislist)
```

```
['apple', 'orange', 'banana', 'cherry']
```

Add List Items

Extend List

```
thislist = ["apple", "banana", "cherry"]
tropical = ["mango", "pineapple", "papaya"]
thislist.extend(tropical)
print(thislist)
```

```
['apple', 'banana', 'cherry', 'mango', 'pineapple', 'papaya']
```

Add Any Iterable

```
thislist = ["apple", "banana", "cherry"]
thistuple = ("kiwi", "orange")
thislist.extend(thistuple)
print(thislist)
```

```
['apple', 'banana', 'cherry', 'kiwi', 'orange']
```

Remove List Items

Remove Specified Item

```
thislist = ["apple", "banana", "cherry"]
thislist.remove("banana")
print(thislist)
```

```
['apple', 'cherry']
```

Remove Specified Index

```
thislist = ["apple", "banana", "cherry"]
thislist.pop(1)
print(thislist)
```

```
['apple', 'cherry']
```

Remove Specified Index

```
thislist = ["apple", "banana", "cherry"]
del thislist[0]
print(thislist)
```

```
['banana', 'cherry']
```

Clear the List

```
thislist = ["apple", "banana", "cherry"]
thislist.clear()
print(thislist)
```

```
[]
```

Sort Lists

Sort List Alphanumerically

```
thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]
thislist.sort()
print(thislist)
```

```
['banana', 'kiwi', 'mango', 'orange', 'pineapple']
```

Sort the list numerically:

```
thislist = [100, 50, 65, 82, 23]
thislist.sort()
print(thislist)
```

```
[23, 50, 65, 82, 100]
```

Sort Lists

Sort Descending

```
thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]
thislist.sort(reverse = True)
print(thislist)
```

```
['pineapple', 'orange', 'mango', 'kiwi', 'banana']
```

```
thislist = [100, 50, 65, 82, 23]
thislist.sort(reverse = True)
print(thislist)
```

```
[100, 82, 65, 50, 23]
```

Sort Lists

Case Insensitive Sort

```
thislist = ["banana", "Orange", "Kiwi", "cherry"]
thislist.sort()
print(thislist)
```

```
['Kiwi', 'Orange', 'banana', 'cherry']
```

Reverse Order

```
thislist = ["banana", "Orange", "Kiwi", "cherry"]
thislist.reverse()
print(thislist)
```

```
['cherry', 'Kiwi', 'Orange', 'banana']
```

Copy Lists

Copy a List

There are ways to make a copy, copy() and list()

```
thislist = ["apple", "banana", "cherry"]
mylist = list(thislist)
print(mylist)
```

```
['apple', 'banana', 'cherry']
```

```
thislist = ["apple", "banana", "cherry"]
mylist = thislist.copy()
print(mylist)
```

```
['apple', 'banana', 'cherry']
```

Join Lists

Join Two Lists

One of the easiest ways are by using the + operator.

```
list1 = ["a", "b", "c"]
list2 = [1, 2, 3]

list3 = list1 + list2
print(list3)
```

```
['a', 'b', 'c', 1, 2, 3]
```

Or you can use the extend() method, which purpose is to add elements from one list to another list:

```
list1 = ["a", "b" , "c"]
list2 = [1, 2, 3]

list1.extend(list2)
print(list1)
```

```
['a', 'b', 'c', 1, 2, 3]
```

Basic List Operations

Python Expression	Results	Description
len([1, 2, 3])	3	Length
[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]	Concatenation
['Hi!'] * 4	['Hi!', 'Hi!', 'Hi!', 'Hi!']	Repetition
3 in [1, 2, 3]	True	Membership
for x in [1, 2, 3]: print(x)	1 2 3	Iteration

Built-in List Functions & Methods

Sr.No.	Function with Description
1	**len(list)** Gives the total length of the list.
2	**max(list)** Returns item from the list with max value.
3	**min(list)** Returns item from the list with min value.
4	**list(seq)** Converts a tuple into list.

Built-in List Functions & Methods

Sr.No.	Methods with Description
1	**list.append(obj)** Appends object obj to list
2	**list.count(obj)** Returns count of how many times obj occurs in list
3	**list.extend(seq)** Appends the contents of seq to list
4	**list.index(obj)** Returns the lowest index in list that obj appears
5	**list.insert(index, obj)** Inserts object obj into list at offset index

Built-in List Functions & Methods

Sr.No.	Methods with Description
6	**list.pop(obj=list[-1])** Removes and returns last object or obj from list
7	**list.remove(obj)** Removes object obj from list
8	**list.reverse()** Reverses objects of list in place
9	**list.sort([func])** Sorts objects of list, use compare func if given

Lab Task -4

Create a list called months that includes ‘January’, ‘March’, and ‘April’. Then, create another list named remaining_months and add the rest of the months of the year to it. Next, insert ‘February’ into the months list in the correct order. After that, remove ‘February’ from the months list. Finally, print the elements of the list that represent the time from ‘March’ to ‘September’.