1. What are the Data Structures?

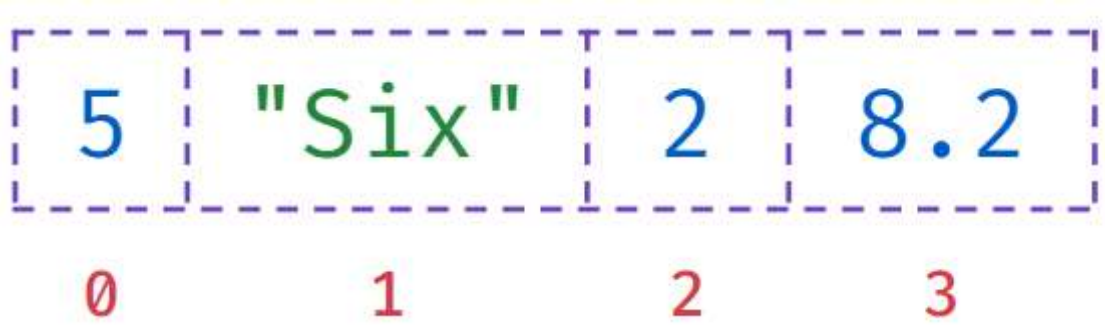
Data Structures allow us to store and organize data efficiently.
This will allow us to easily access and perform operations on the data.

In Python, there are four built-in data structures

- *List*
- *Tuple*
- *Set*
- *Dictionary*

2. What is a list?

The List is the most versatile python data structure that holds an ordered sequence of items.



Creating a List

A List can be created by enclosing elements within [square] brackets where each item is separated by a comma.

Code

PYTHON

```python
a = 2
list_a = [5, "Six", a, 8.2]

print(type(list_a))
print(list_a)
```

Output

```
<class 'list'>
[5, 'Six', 2, 8.2]
```

3. Explain a few List Methods?

Append

Adds an element to the end of the list.

Syntax

```
list.append(value)
```

Code

PYTHON

```python
list_a = []
for x in range(1, 4):
    list_a.append(x)

print(list_a)
```

Output

```
[1, 2, 3]
```

Extend

Adds all the elements of a sequence to the end of the list.

Syntax

```
list_a.extend(list_b)
```

Code

PYTHON

```python
list_a = [1, 2, 3]
list_b = [4, 5, 6]
list_a.extend(list_b)

print(list_a)
```

Output

```
[1, 2, 3, 4, 5, 6]
```

Insert

Inserts the given element at the specified index.

Syntax

```
list.insert(index, value)
```

Code

PYTHON

```python
list_a = [1, 2, 3]
list_a.insert(1, 4)

print(list_a)
```

Output

```
[1, 4, 2, 3]
```

Pop

Removes the last item from the list.

Syntax

```
list.pop()
```

Code

PYTHON

```python
list_a = [1, 2, 3]
list_a.pop()

print(list_a)
```

Output

```
[1, 2]
```

Remove

Removes the first matching element from the list.

Syntax

```
list.remove(value)
```

Code

```python
list_a = [1, 3, 2, 3]
list_a.remove(3)

print(list_a)
```

Output

```
[1, 2, 3]
```

Clear

Removes all the items from the list.

Syntax

```
list.clear()
```

Code

PYTHON

```python
list_a = [1, 2, 3]
list_a.clear()

print(list_a)
```

Output

```
[]
```

Index

Returns the index of the first occurrence of the specified item in the list.

Syntax

```
list.index(item)
```

Code

PYTHON

```python
list_a = [1, 3, 2, 3]
index = list_a.index(3)

print(index)
```

Output

```
1
```

Count

Returns the number of elements with the specified value.

Syntax

```
list.count(value)
```

Code

PYTHON

```python
list_a = [1, 2, 3]
count = list_a.count(2)

print(count)
```

Output

```
1
```

Sort

Sorts the items of a list in ascending ascending order. The

`sort()` method modifies the original list.

Syntax

```
list.sort()
```

Code

PYTHON

```python
list_a = [1, 3, 2]
list_a.sort()

print(list_a)
```

Output

```
[1, 2, 3]
```

Sorted

Sorts the items of a list in ascending ascending order. The

`sorted()` method returns the modified list.

Syntax

```
sorted()
```

Code

PYTHON

```python
list_a = [1, 3, 2]
list_b = sorted(list_a)

print(list_b)
```

Output

```
[1, 2, 3]
```

Code

PYTHON

```python
list_a = [1, 3, 2]
sorted(list_a)

print(list_a)
```

Output

```
[1, 3, 2]
```

4. What is the difference between `append()` and `extend()` ?

append	extend
.append() takes a single element as an	.extend() takes an iterable as an argument (list, tuple,

append	extend
argument	dictionaries, sets, strings)
.append() adds a single element to the end of the list	.extend() can add multiple individual elements to the end of the list

5. How to use the `split()` method?

The

`split()` splits a string into a list at every specified separator.

Syntax:

```
str_var.split(separator)
```

Example

Code

PYTHON

```
nums = "1 2 3 4"
num_list = nums.split()
print(num_list)
```

Output

```
['1', '2', '3', '4']
```

If no separator is specified, the default separator is whitespace.

Using separator

Example

Code

PYTHON

```
nums = "1,2,3,4"
num_list = nums.split(',')
print(num_list)
```

Output

```
['1', '2', '3', '4']
```

6. How to use the `join()` method?

The

`join()` takes all the items in a sequence of strings and joins them into one string.

Syntax:

```
str.join(sequence)
```

Example

Code

PYTHON

```python
list_a = ['Python is ', ' progr', 'mming l', 'ngu', 'ge']
string_a = "a".join(list_a)
print(string_a)
```

Output

```
Python is a programming language
```

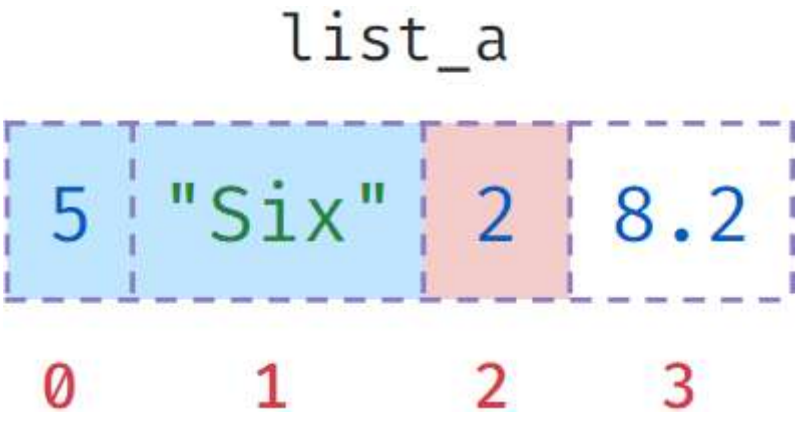
7. What is List Slicing?

Obtaining a part of a list is called List Slicing.

Syntax:

```
variable_name[start_index:end_index]
```

- `end_index` is not included in the slice.



Code

```python
list_a = [5, "Six", 2, 8.2]
list_b = list_a[:2]
print(list_b)
```

```
print(list_b)
```

Output

```
[5, 'Six']
```

Extended Slicing

Similar to string extended slicing, we can extract alternate items from the list using the step.

Syntax:

```
variable[start_index:end_index:step]
```

Code

PYTHON

```
1  list_a = ["R", "B", "G", "O", "W"]
2  list_b = list_a[0:5:3]
3  print(list_b)
```

Output

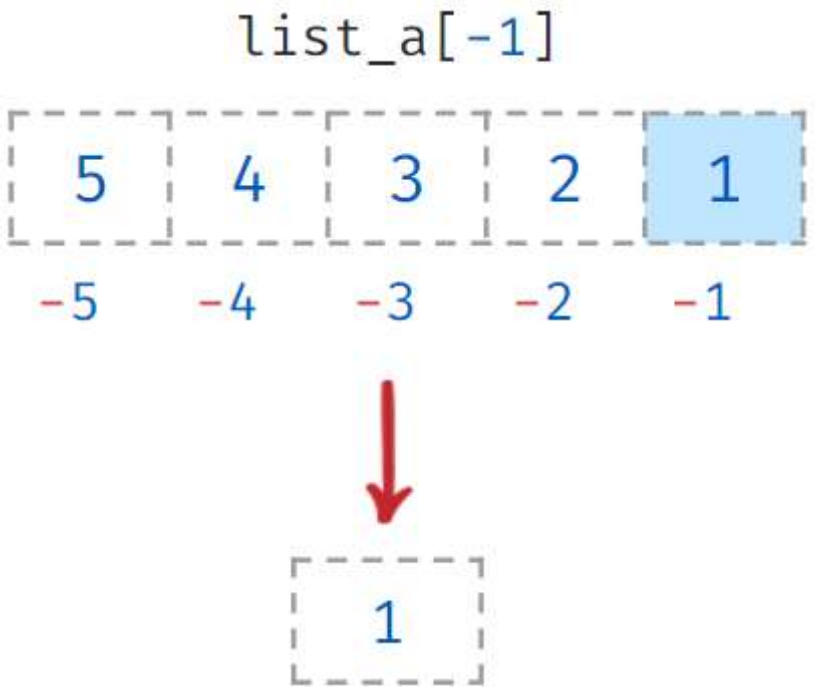
```
['R', 'O']
```

8. What is Negative Indexing?

Using a negative index returns the nth item from the end of the list.

The last item in the list can be accessed with the index

```
-1
```



Accessing List Items with Negative Index

Example-1

Code

PYTHON

```python
list_a = [5, 4, 3, 2, 1]
item = list_a[-1]
print(item)
```

Output

```
1
```

Example-2

Code

PYTHON

```python
list_a = [5, 4, 3, 2, 1]
item = list_a[-4]
print(item)
```

Output

```
4
```

Slicing With Negative Index

You can also specify negative indices while slicing a List.

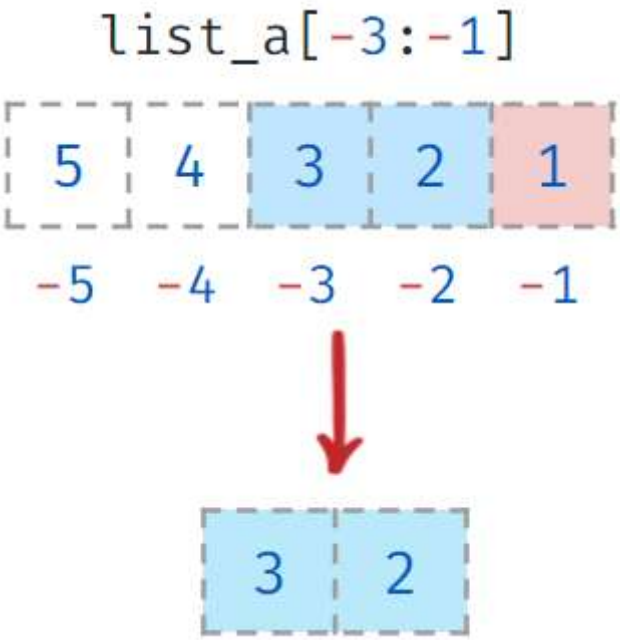
Code

PYTHON

```
list_a = [5, 4, 3, 2, 1]
list_b = list_a[-3:-1]
print(list_b)
```

Output

```
[3, 2]
```



Out of Bounds Index

While slicing, the index can go beyond the size of the list.

Code

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

```
list_b = list_a[-6:-2]
print(list_b)
```

Output

```
[5, 4, 3]
```

9. How to reverse a List?

Reversing a list using

`reverse()` **method**:

The

`reverse()` method can be used to reverse a List. It updates the original list.

Code

PYTHON

```
week_days = ['Monday', 'Tuesday', 'Wednesday']
week_days.reverse()

print(week_days)
```

Output

```
['Wednesday', 'Tuesday', 'Monday']
```

Reversing a list using list slicing:

A list can be reversed using extended slicing.

Syntax:

`variable[start:end:negative_step]`

`-1` for step will reverse the order of items in the list.

Code

PYTHON

```
list_a = [5, 4, 3, 2, 1]
list_b = list_a[::-1]
print(list_b)
```

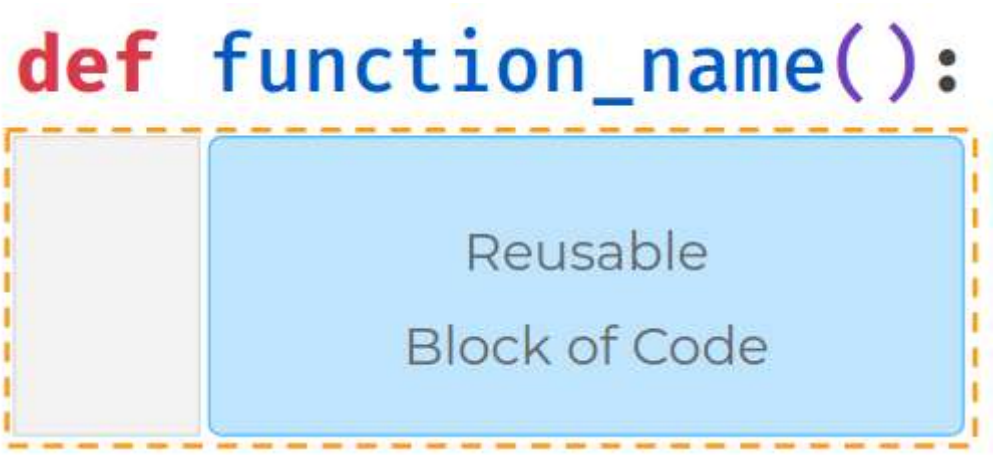
Output

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

10. Explain about functions in Python?

A function is a block of reusable code to perform a specific action. Functions help us in using existing code without writing it every time we need it.

A function can be defined using a keyword `def`. A function is uniquely identified by the `function_name`.



Code

PYTHON

```
def greet():
    print("Hello")

greet()
greet()
```

Output

```
Hello
Hello
```

Function Arguments

We can pass values to a function using Argument.

Code

PYTHON

```python
def greet(word):
    msg = "Hello " + word
    print(msg)

name1 = input()
name2 = input()
greet(word = name1)
greet(word = name2)
```

Input

```
Teja
Rahul
```

Output

```
Hello Teja
Hello Rahul
```

Providing default values

Default values indicate that the function argument will take that value if no argument value is passed during the function call.

Example

Code

```python
def greet(arg_1 = "Hi", arg_2 = "Ram"):
    print(arg_1 + " " + arg_2)

greeting = input()
```

```
name = input()
greet()
```

Input

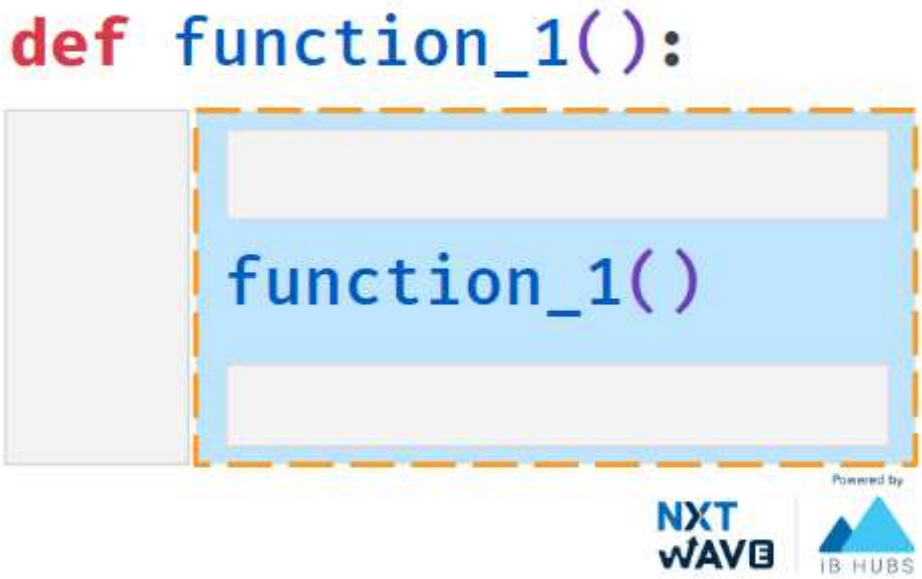
```
Hello
Teja
```

Output

```
Hi Ram
```

11. What is Recursion?

A function calling itself is called **Recursion**



Let's understand recursion with a simple example of multiplying N numbers

Multiply N Numbers

PYTHON

```
def factorial(n):  # Recursive Function
    if n == 1:  # Base Case
        return 1
    return n * factorial(n - 1)  # Recursion
num = int(input())
result = factorial(num)
print(result)
```

Input

```
5
```

Output

```
120
```

MARK AS COMPLETED

Submit Feedback

Notes Discussions