LEARN RISE SHINE
NAVTTC

Summer of Code
Artificial Intelligence
(Machine Learning & Deep Learning)

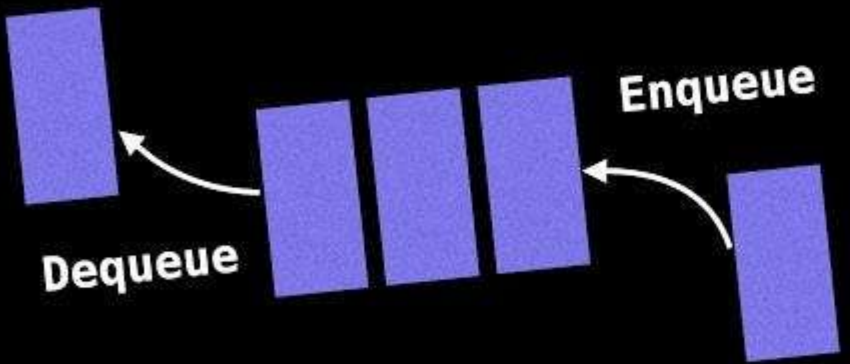
Instructor
Wajahat Ullah
- *Research Assistant* (DIP Lab)

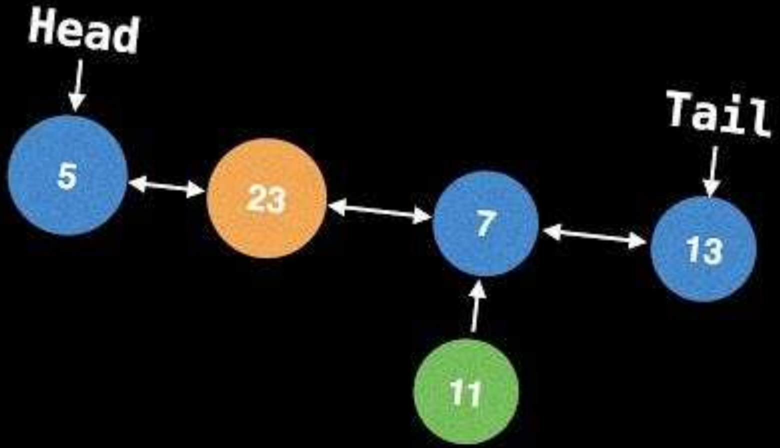
Duration
03 Months
(September – November)

Day 04 – Python Fundamentals (Data Structures)

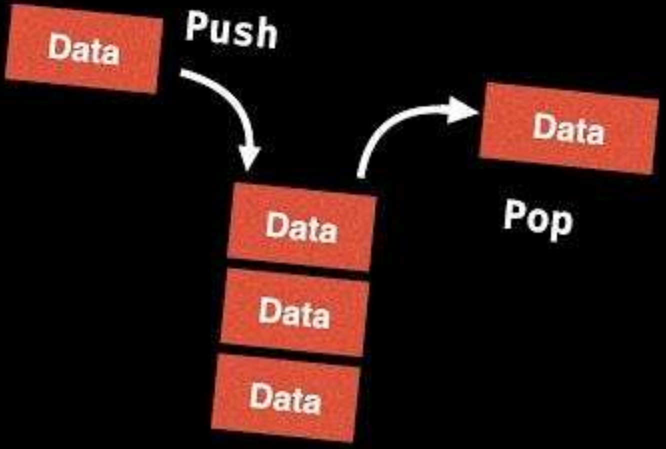
Objectives:

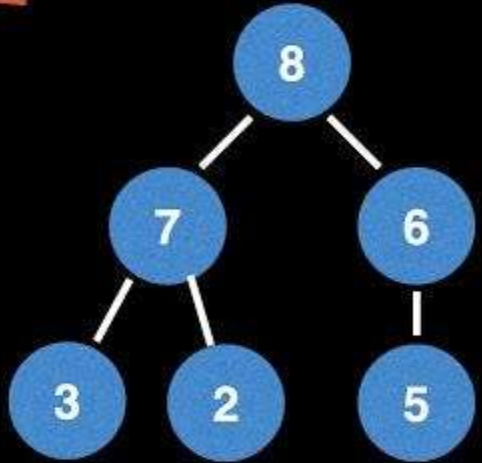
- What is a Data Structure?
- Built-in Data Structures in Python
- Strings, Lists, Tuples, Sets, Dictionaries
- Operations on Data Structures

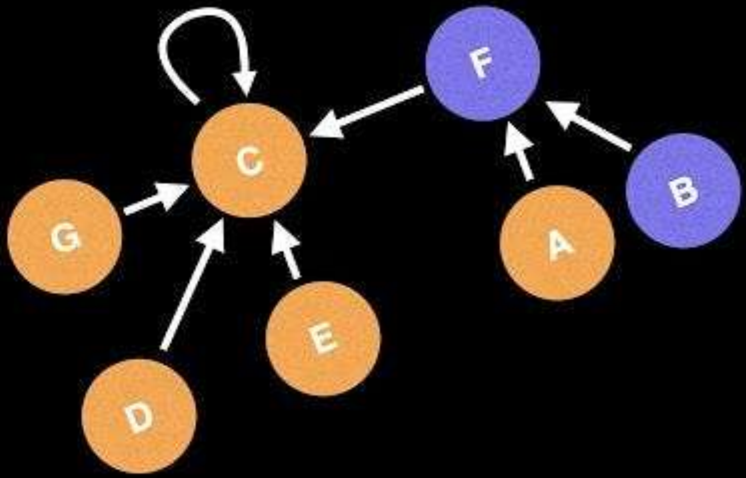


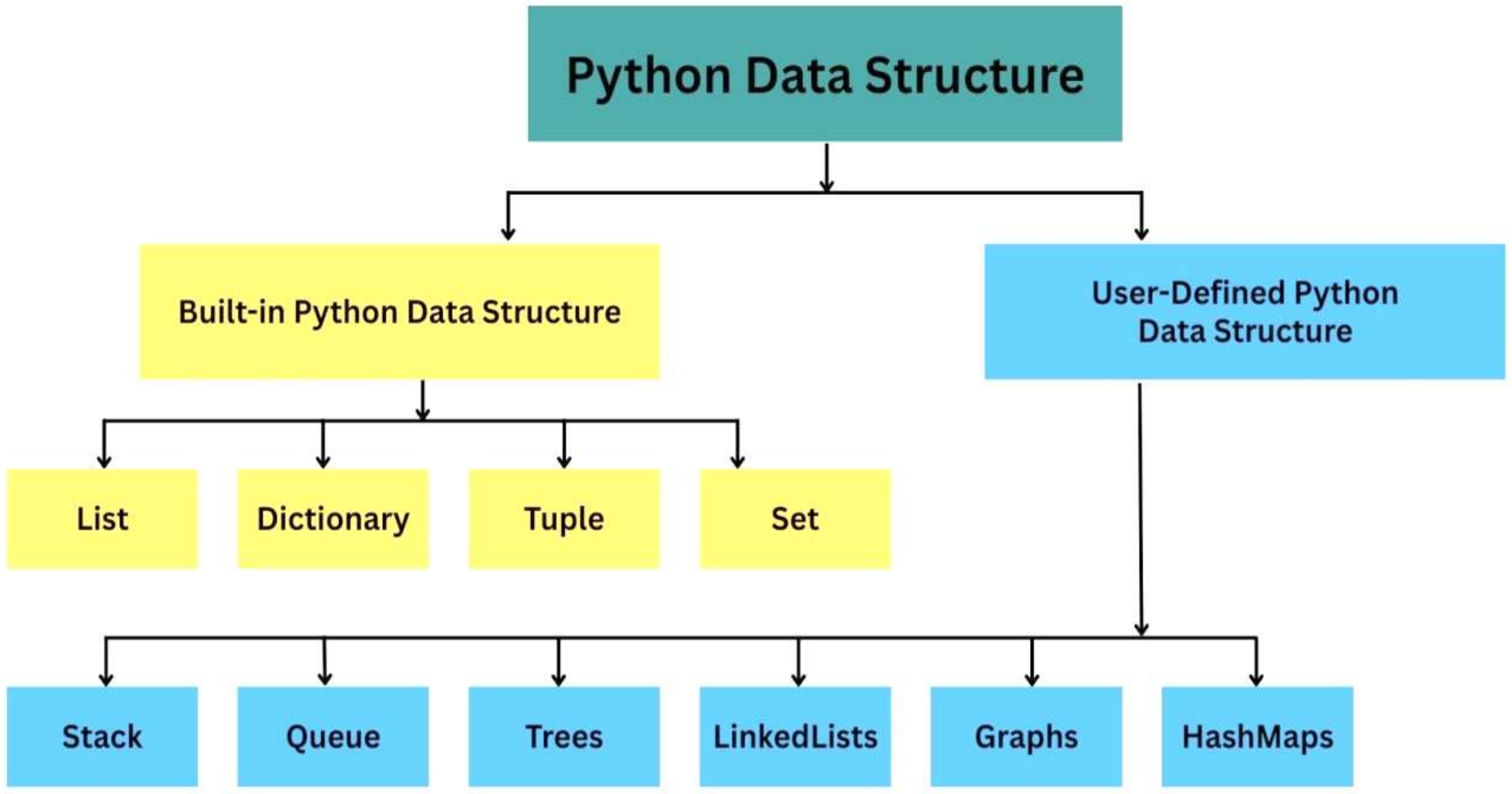


Data Structures



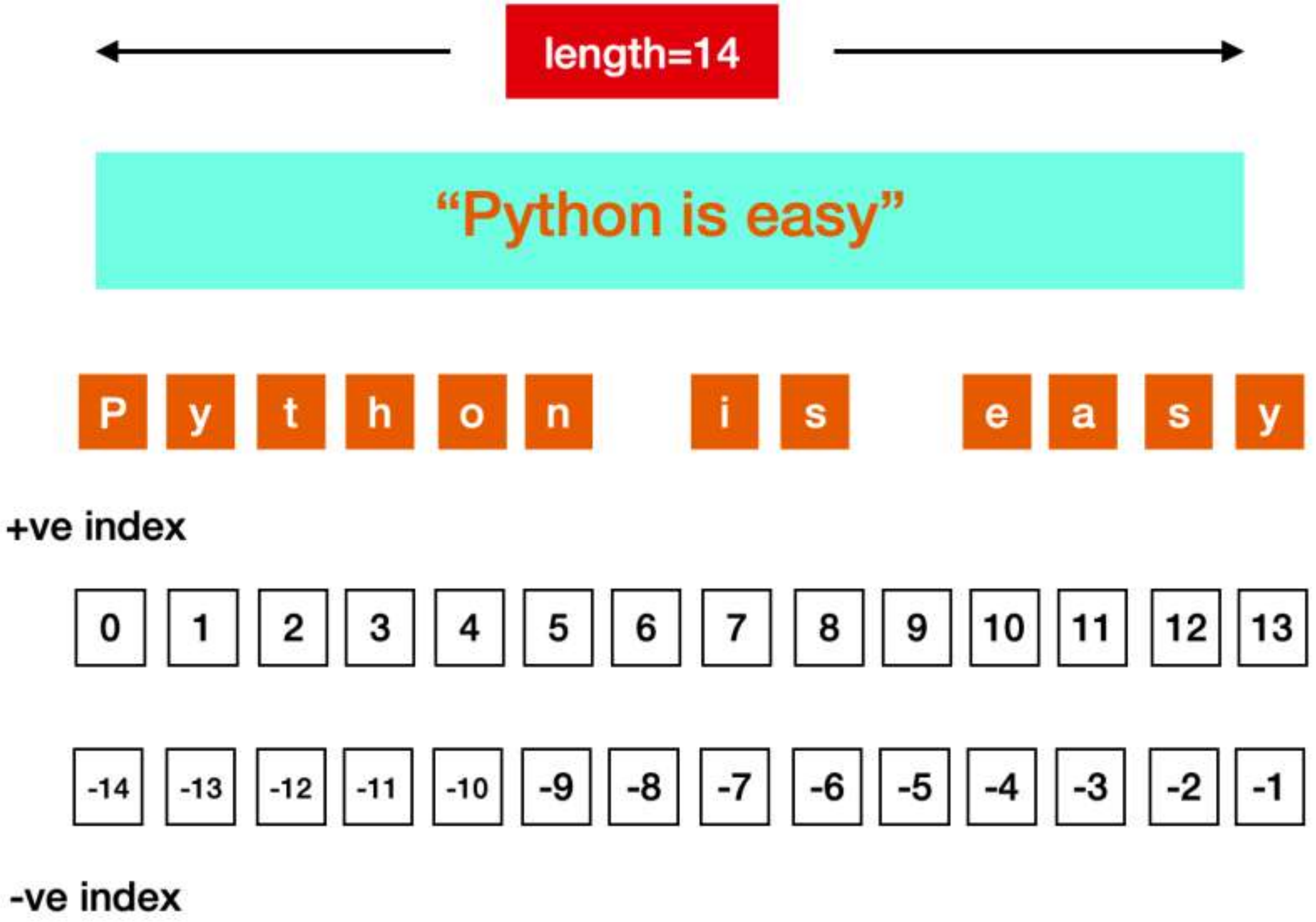


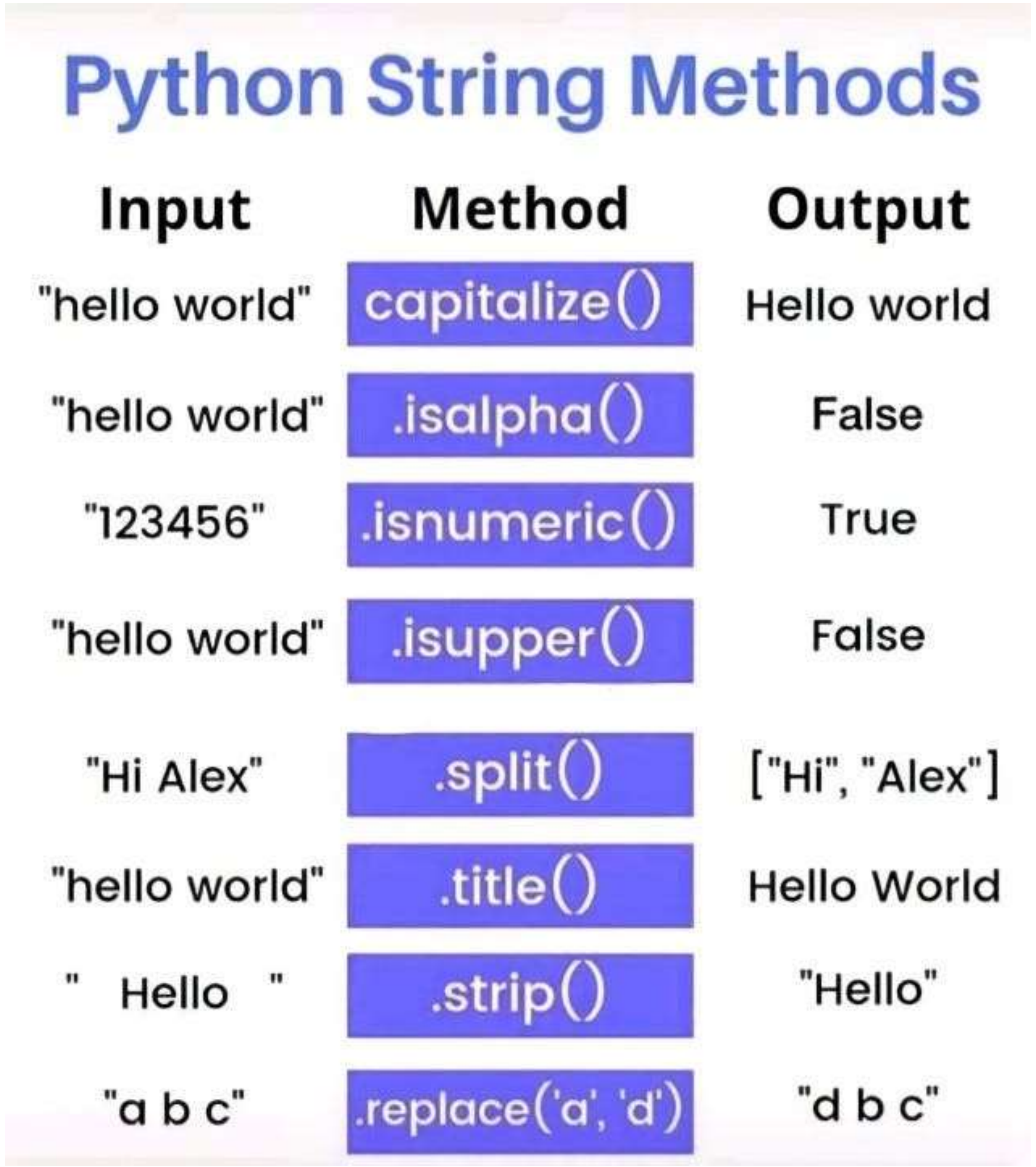




igmGuru
Learn anywhere, anytime

Python String

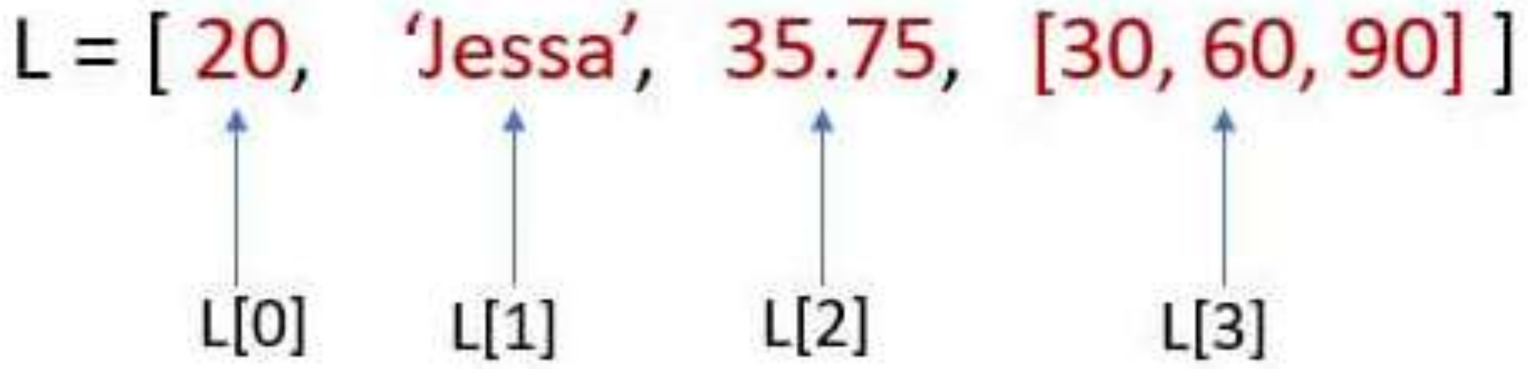




Python String Methods

Input	Method	Output
"hello world"	capitalize()	Hello world
"hello world"	.isalpha()	False
"123456"	.isnumeric()	True
"hello world"	.isupper()	False
"Hi Alex"	.split()	["Hi", "Alex"]
"hello world"	.title()	Hello World
" Hello "	.strip()	"Hello"
"a b c"	.replace('a', 'd')	"d b c"

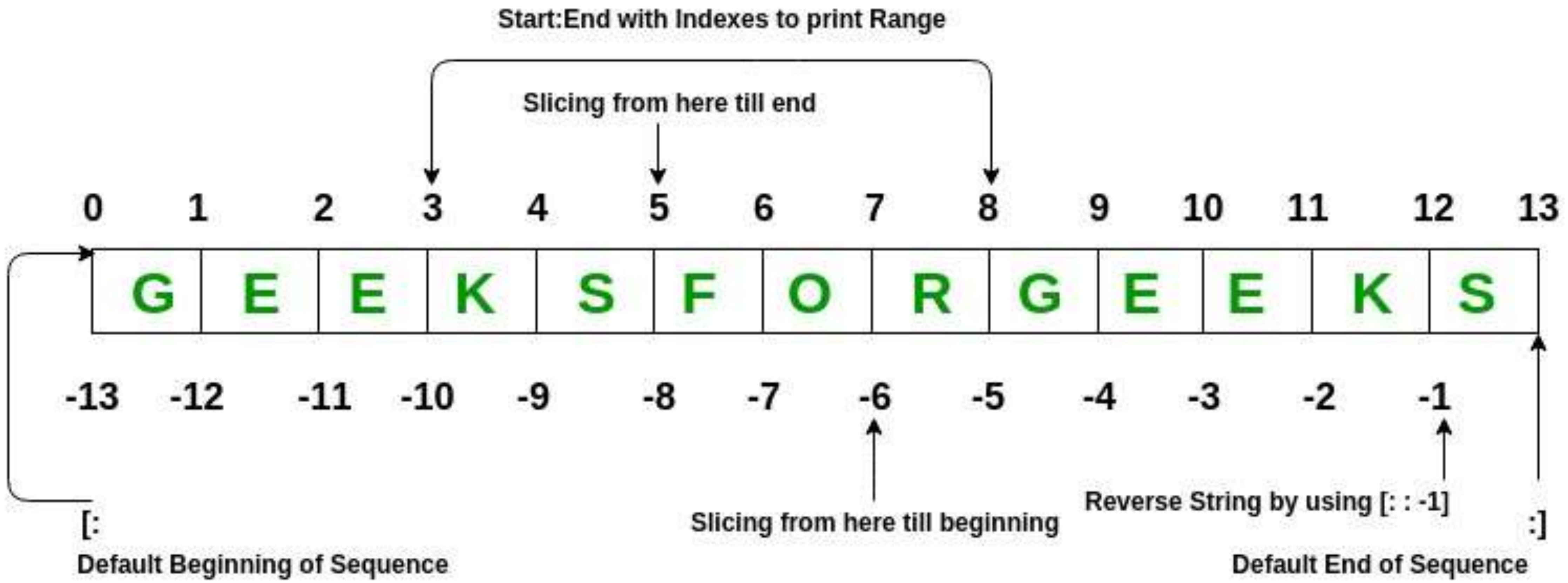
List in Python



```
L = [ 20,   'Jessa',   35.75,   [30, 60, 90] ]
```

- ✓ **Ordered**: Maintain the order of the data insertion.
- ✓ **Changeable**: List is mutable and we can modify items.
- ✓ **Heterogeneous**: List can contain data of different types
- ✓ **Contains duplicate**: Allows duplicates data

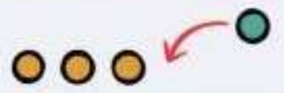
Accessing Items in a List

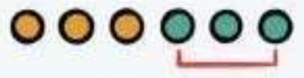

Start:End with Indexes to print Range
Slicing from here till end
0 1 2 3 4 5 6 7 8 9 10 11 12 13
G E E K S F O R G E E K S
-13 -12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1
[:
Default Beginning of Sequence
Slicing from here till beginning
Reverse String by using [: : -1]
:]
Default End of Sequence

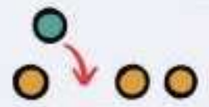
Python List Methods

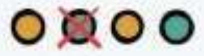
by levelupcoding.co

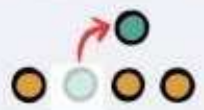
	Description	Code example	Diagram

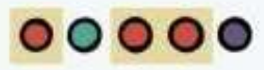
append()	Adds an element to the end of the list	`letters = ['a', 'b', 'c']`
		`letters.append('d')`
			

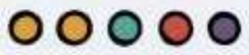
extend()	Adds the elements of a list to the end of another list	`letters = ['a', 'b', 'c']`
		`letters.extend(['d', 'e', 'f'])`
			

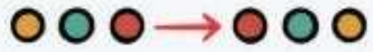
insert()	Adds an element at a specific index	`letters = ['a', 'b', 'c']`
		`letters.insert(1, 'x')`
			

remove()	Removes the first occurrence of an element	`letters = ['a', 'b', 'c', 'b']`
		`letters.remove('b')`
			

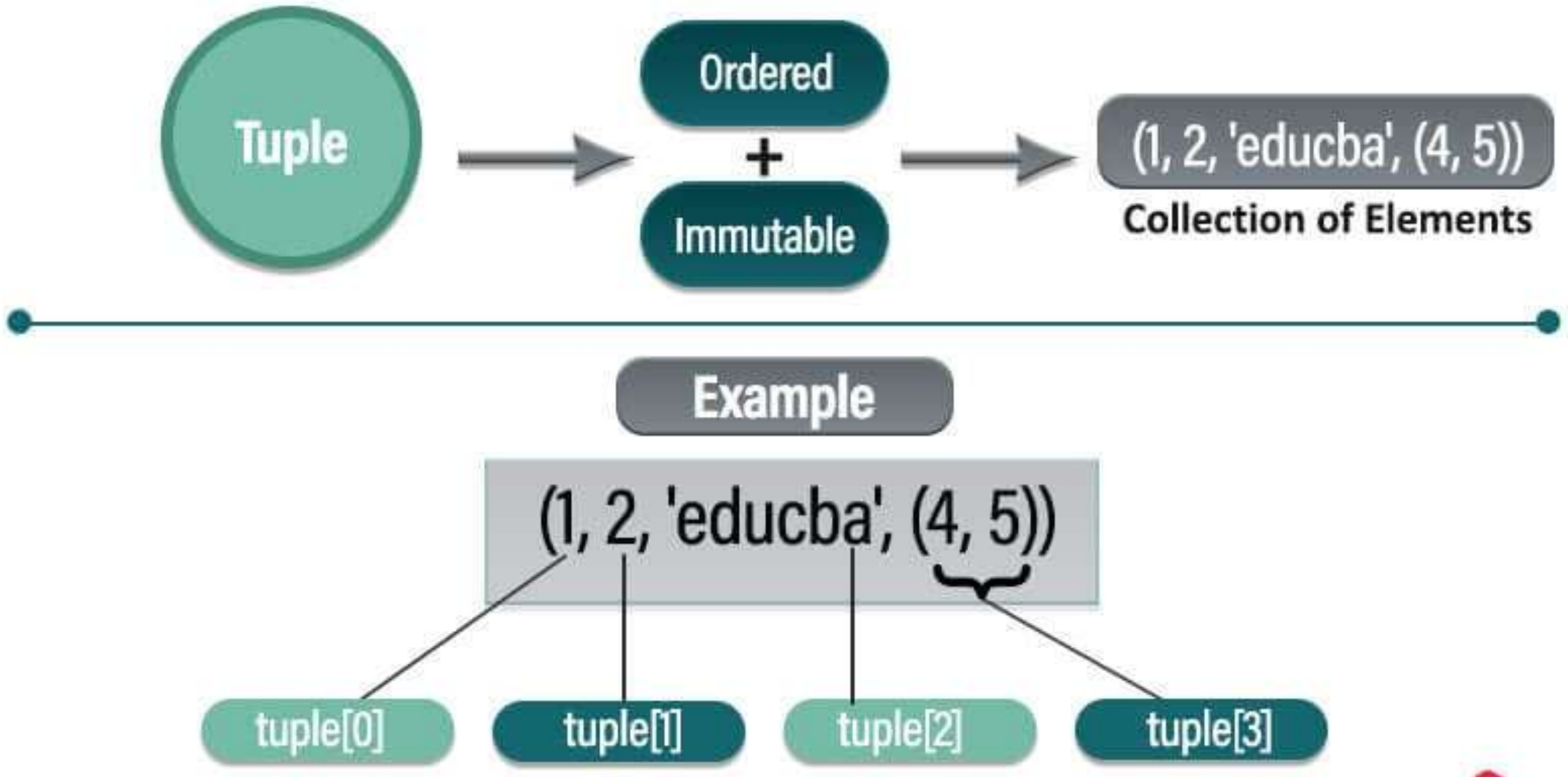
pop()	Removes and returns the element at a given index	`letters = ['a', 'b', 'c', 'd']`
		`letter = letters.pop(1)`
			

count()	Returns the number of times a value appears in a list	`letters = ['c', 'b', 'c', 'c', 'd']`
		`print(letters.count('c'))`
			

sort()
@NikkiSiapno
	Sorts the list in ascending order	`letters = ['c', 'a', 'd', 'a', 'b']`
		`letters.sort()`
			

reverse()
@ChrisStaud
	Reverses the order of elements in the list	`letters = ['a', 'b', 'c']`
		`letters.reverse()`
			

Tuples in Python



EDUCBA

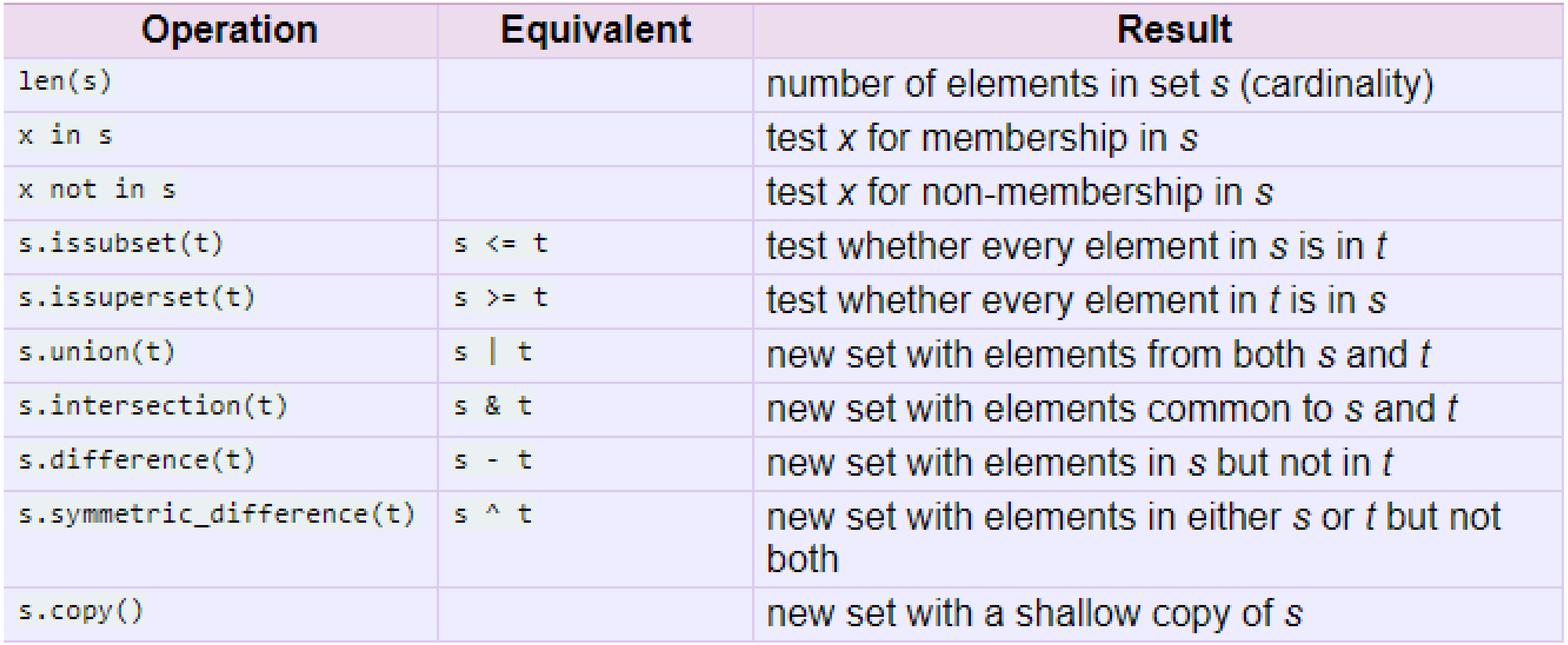
PYnative.com

Set in Python

S = { 20,   'Jessa',   35.75}

- ✓ **Unordered**: Set doesn't maintain the order of the data insertion.
- ✓ **Unchangeable**: Set are immutable and we can't modify items.
- ✓ **Heterogeneous**: Set can contains data of all types
- ✓ **Unique**: Set doesn't allows duplicates items

Operations on Set

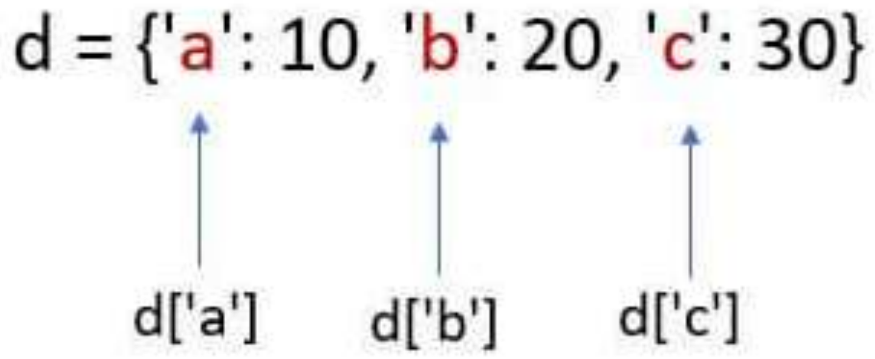


Operation	Equivalent	Result
`len(s)`		number of elements in set *s* (cardinality)
`x in s`		test *x* for membership in *s*
`x not in s`		test *x* for non-membership in *s*
`s.issubset(t)`	`s <= t`	test whether every element in *s* is in *t*
`s.issuperset(t)`	`s >= t`	test whether every element in *t* is in *s*
`s.union(t)`	`s | t`	new set with elements from both *s* and *t*
`s.intersection(t)`	`s & t`	new set with elements common to *s* and *t*
`s.difference(t)`	`s - t`	new set with elements in *s* but not in *t*
`s.symmetric_difference(t)`	`s ^ t`	new set with elements in either *s* or *t* but not both
`s.copy()`		new set with a shallow copy of *s*

Dictionary in Python

PYnative.com

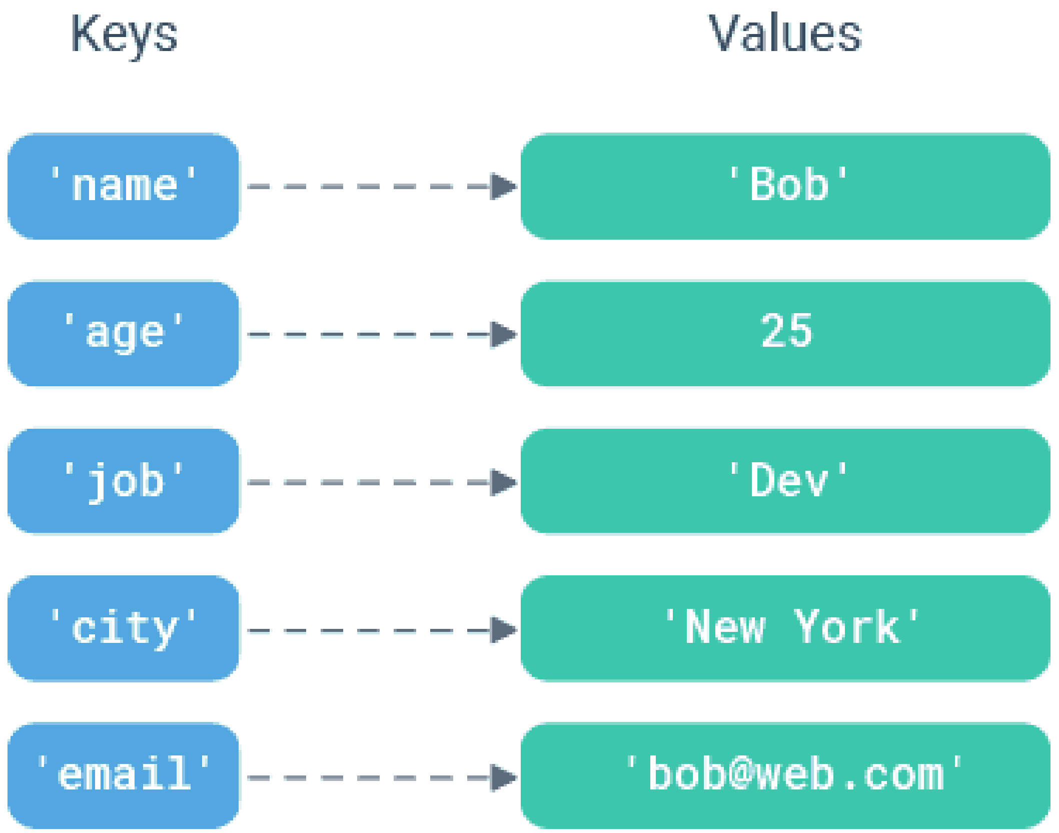
Unordered collections of unique values stored in (Key-Value) pairs.

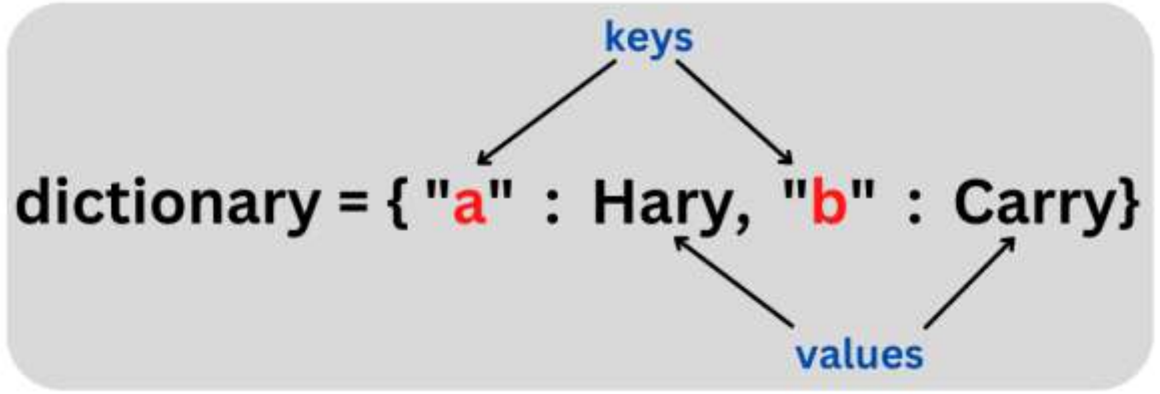


```
d = {'a': 10, 'b': 20, 'c': 30}
```

- ✓ **Unordered**: The items in dict are stored without any index value
- ✓ **Unique**: Keys in dictionaries should be Unique
- ✓ **Mutable**: We can add/Modify/Remove key-value after the creation

Dictionary in Python





Happy Coding