Python Bootcamp & Masterclass

Lists

gknxt

Lists

A list is a mutable sequence of objects.

The objects can be of any type and need not be unique.

- A list is an ordered collection of zero or more objects enclosed by square brackets ([]). If there are two or more objects in a list, they should be separated by comma(s).

- Lists can be constructed by enclosing comma separated objects in a pair of square brackets or by using list constructor, `list(iterator)`, that can take an optional argument (`iterator`)

Lists

A list is a mutable sequence of objects.

The objects can be of any type and need not be unique.

```
l1 = [1, 2, 3, 4]                                   # homogenous list
l2 = [1, 2, 3,]                                     # comma after last element is ignored
l3 = ['USA']                                        # singleton list (list with only one element)
l4 = list('Canada')                                 # creating a list using list constructor
l5 = list(range(5))                                 # creating a list using list constructor
l6 = [int, 'Sydney', 4+3, ['a', 'e', 'i'], print]   # heterogenous list
print(f"l1: {l1} is a: {type(l1)}")
print(f"l2: {l2} is a: {type(l2)}")
print(f"l3: {l3} is a: {type(l3)}")
print(f"l4: {l4} is a: {type(l4)}")
print(f"l5: {l5} is a: {type(l5)}")
print(f"l6: {l6} is a: {type(l6)}")
```

```
l1: [1, 2, 3, 4] is a: <class 'list'>
l2: [1, 2, 3] is a: <class 'list'>
l3: ['USA'] is a: <class 'list'>
l4: ['C', 'a', 'n', 'a', 'd', 'a'] is a: <class 'list'>
l5: [0, 1, 2, 3, 4] is a: <class 'list'>
l6: [<class 'int'>, 'Sydney', 7, ['a', 'e', 'i'], <built-in function print>] is a: <class 'list'>
```

empty list

An empty list can be created implicitly by using square brackets ([])

An empty list can also be created explicitly by calling the list constructor, `list(iterable)` without its argument (`iterable`)

Implicit creation is more efficient in terms of execution speed as explicit creation involves function invocation and associated housekeeping tasks.

```
el1 = []
el2 = list()
el3 = [      ]
el4 = list(  )
print(f"el1 is of type {type(el1)} and of length {len(el1)}")
print(f"el2 is of type {type(el2)} and of length {len(el2)}")
print(f"el3 is of type {type(el3)} and of length {len(el3)}")
print(f"el4 is of type {type(el4)} and of length {len(el4)}")
```

```
el1 is of type <class 'list'> and of length 0
el2 is of type <class 'list'> and of length 0
el3 is of type <class 'list'> and of length 0
el4 is of type <class 'list'> and of length 0
```

```
!python -mtimeit  "l=[]"
```

```
20000000 loops, best of 5: 16 nsec per loop
```

```
!python -mtimeit  "l=list()"
```

```
10000000 loops, best of 5: 38.3 nsec per loop
```

len ( )

The built-in function `len(iterable)` returns the length (the number of items) of the `iterable`. The length of an empty list is `0`

```
a = []
len(a)
```

```
0
```

```
b = [0]
len(b)
```

```
1
```

```
c = [23, 44, 11, 456, 111]
len(c)
```

```
5
```

```
d = [int, 'Sydney', 4+3, ['a', 'e', 'i'], print]
len(d)
```

```
5
```

```
e = [[1, 3, [4, 5], 6], 7]
len(e)
```

```
2
```

```
f = [0.3, False, 0.44, 0.11, True, 0.255]
len(f)
```

```
6
```

min ( )

The built-in function `min(iterable)` returns the smallest object of the `iterable`. If the objects are not comparable, `TypeError` will be raised.

```
a = ['xyz', 'zara', 'abc']
b = [23, 44, 11, 456, 111]
c = [23, False, 44, 11, True, 255]
d = [0.3, False, 0.44, 0.11, True, 0.255]
e = ['2022','9','890','70','891','898']
min(a)                      # strings are compared lexicographically
min(b)
min(c)                      # False evaluates to 0
min(d)
min(e)
```

```
'abc'

11

False

False

'2022'
```

gknxt

max ( )

The built-in function `max(iterable)` returns the largest object of the `iterable`. If the objects are not comparable, `TypeError` will be raised.

```
g = ['xyz', 'zara', 'abc']
h = [23, 44, 11, 456, 111]
i = [23, False, 44, 11, True, 255]
j = [0.3, False, 0.44, 0.11, True, 0.255]
k = ['2022','9','890','70','891','898']
max(g)                      # strings are compared lexicographically
max(h)
max(i)                      # False evaluates to 0
max(j)
max(k)
```

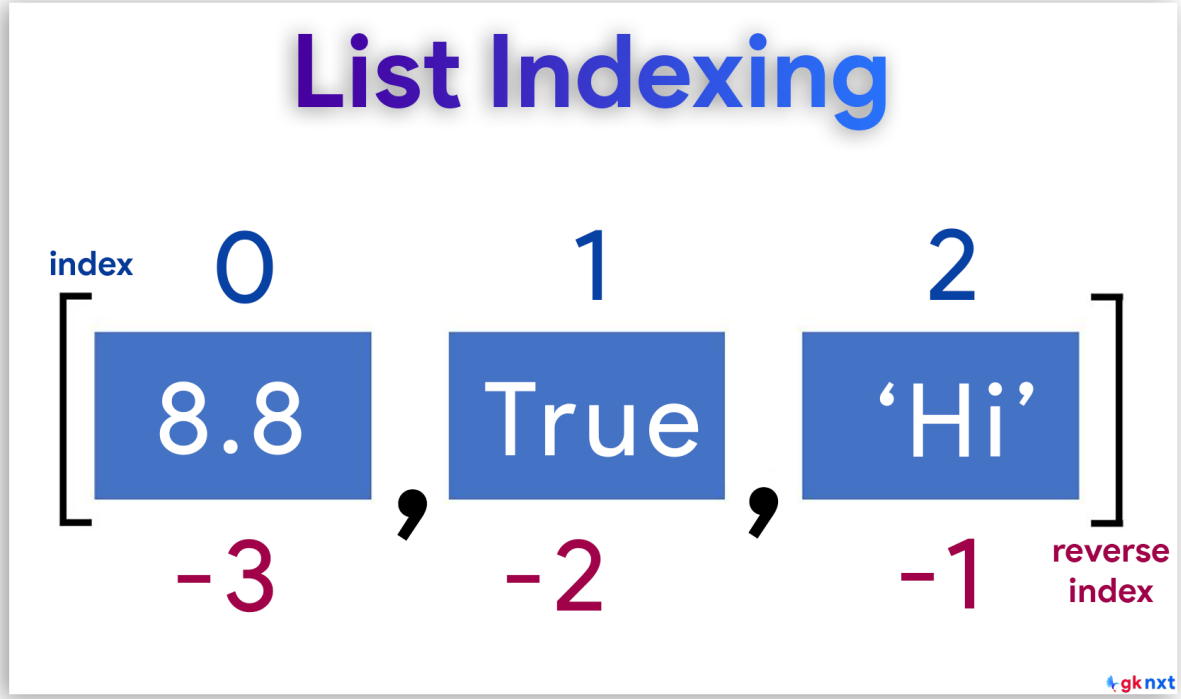
```
'zara'

456

255

True

'9'
```

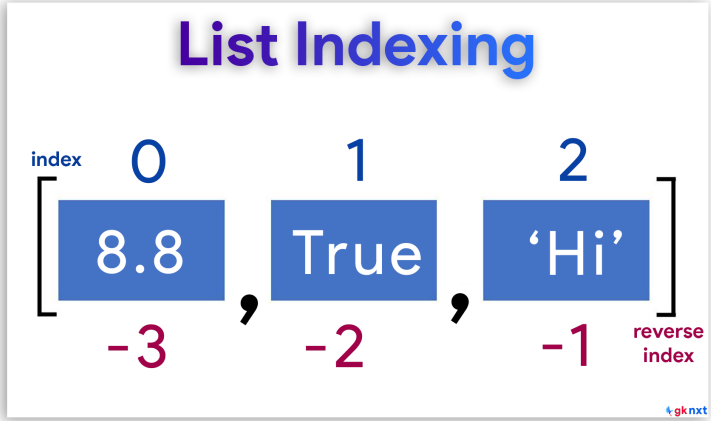


List is a sequence, so each item in a list has a numbered position called index that starts at 0

Elements of a list can be accessed in the reverse order by using negative index starting with -1

(The last element in a list has index -1, the second-to-last element has index -2, and so on)

gknxt

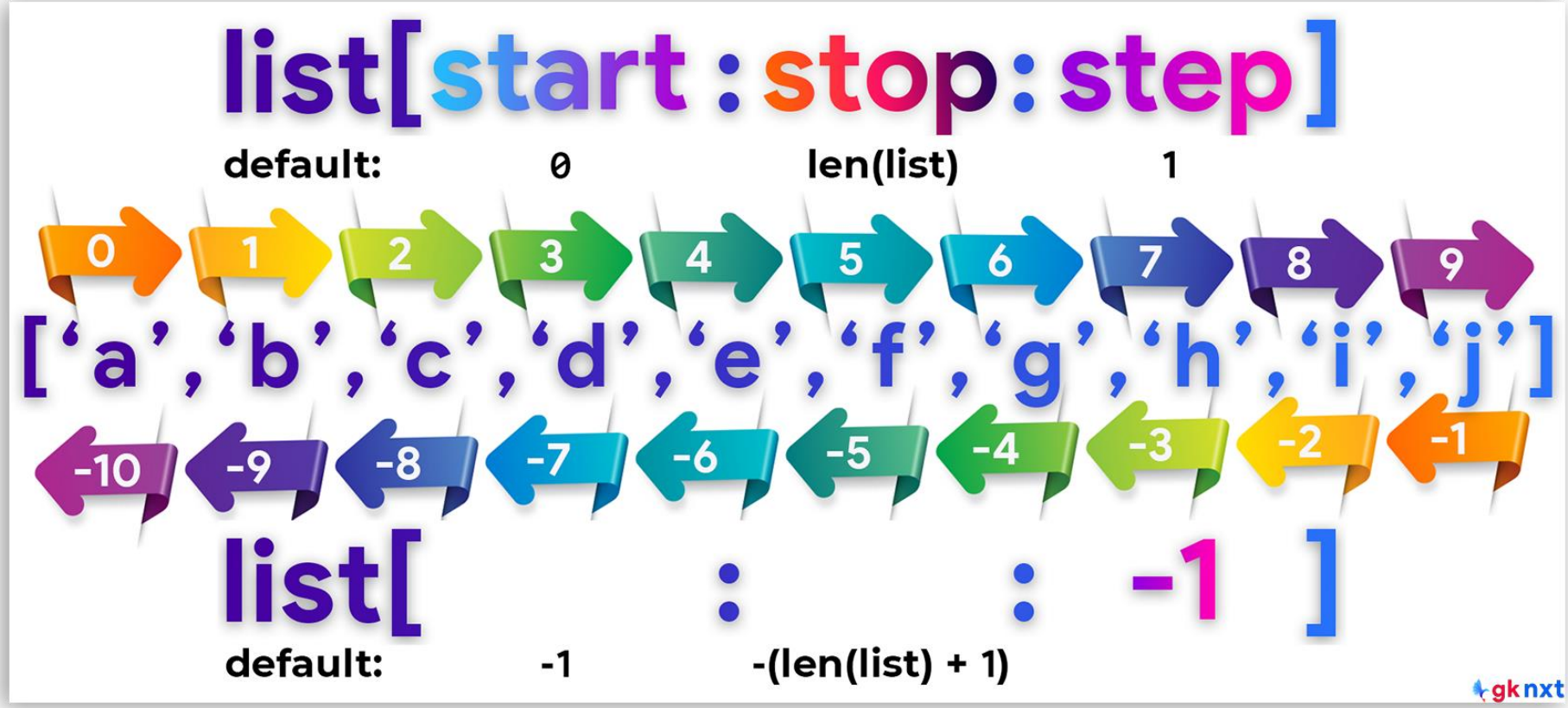



```
my_list = [8.8, True, 'Hi']
print("Data at my_list[0]:", my_list[0])
print("Data at my_list[1]:", my_list[1])
print("Data at my_list[2]:", my_list[2])
print('\n')
print("Data at my_list[-1]:", my_list[-1])
print("Data at my_list[-2]:", my_list[-2])
print("Data at my_list[-3]:", my_list[-3])
```

```
Data at my_list[0]: 8.8
Data at my_list[1]: True
Data at my_list[2]: Hi


Data at my_list[-1]: Hi
Data at my_list[-2]: True
Data at my_list[-3]: 8.8
```

list slicing



gknxt

A slice can be extracted from a list using the slice operator ([ ]) and colon(:) to separate start, stop and step options.

```
k = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
k[2:]
k[:8]
k[: : 2]
```

```
['c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h']

['a', 'c', 'e', 'g', 'i']
```

```
p = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
p[:]
p[::]
p[0:10:]
p[0:10:1]
p[:10:1]
p[0::1]
```

```
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
```

delete a list

If a list need to be deleted, **`del( )`** function can be used. If a list need to be emptied (delete all elements of the list), **`clear()`** method can be used. A list can also be emptied by assigning an empty list to it. Another way is by assigning the product of the list multiplied with `0` to the list itself.

```
u = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
del u   # deletes the list itself, so it cannot be used/accessed again
#u      # NameError
```

```
v = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
v.clear()   # deletes all elements of the list (make it empty list)
v
```

```
[]
```

```
w = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
w = []      # deletes all elements of the list (make it empty list)
w
```

```
[]
```

```
x = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
del x[:]    # deletes all elements of the list (make it empty list)
x
```

```
[]
```

```
y = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
y[:] =[]    # deletes all elements of the list (make it empty list)
y
```

```
[]
```

remove an element by value

To remove an element by value from a list, `remove(x)` method can be used. The first element with the value `x` will be removed. If not found, the method raises `ValueError`. The removed element would be lost as this method returns `None`

```
k = ['Allen', 'Bob', 'Chris', 'Joe', 'Bob', 'Sam']
k.remove('Bob')    # removes the first occurance of 'Bob' scanning from left to right
k
```

```
['Allen', 'Chris', 'Joe', 'Bob', 'Sam']
```

```
l = ['Allen', 'Bob', 'Chris', 'Joe', 'Bob', 'Sam']
l.remove('Don')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_15676/2315601760.py in <module>
      1 l = ['Allen', 'Bob', 'Chris', 'Joe', 'Bob', 'Sam']
----> 2 l.remove('Don')

ValueError: list.remove(x): x not in list
```

remove the last element

To remove the last element from a list, `pop()` method can be used. If no argument is passed, `pop()` method removes the last element from the list and returns it. Another way to remove the last element of a list is to use `del` with `-1` as index.

```
n = ['Allen', 'Bob', 'Chris', 'Joe', 'Bob', 'Sam']
n.pop()
```

```
'Sam'
```

```
p = ['Allen', 'Bob', 'Chris', 'Joe', 'Bob', 'Sam']
del p[-1]
p
```

```
['Allen', 'Bob', 'Chris', 'Joe', 'Bob']
```

remove an element by index

To remove an element from a list by index, `pop(index)` method can be used. The method removes the first element at the `index` position from the list and returns it. If the index is out of range for the list, it raises an `IndexError`. The preferred way to remove an element by index is by using `del s[index]`

```
prez = ['Bush', 'Obama', 'Trump', 'Biden', 'Trump']
prez.pop(2)
prez
```

```
'Trump'
```

```
['Bush', 'Obama', 'Biden', 'Trump']
```

```
prez = ['Bush', 'Obama', 'Trump', 'Biden', 'Trump']
del prez[1]
prez
```

```
['Bush', 'Trump', 'Biden', 'Trump']
```

```
prez = ['Bush', 'Obama', 'Trump', 'Biden', 'Trump']
prez.pop(-2)
prez
```

```
'Biden'
```

```
['Bush', 'Obama', 'Trump', 'Trump']
```

Online Resources

For best python resources, please visit:

gknxt.com/python/

gknxt

Python
Bootcamp
& Masterclass

Thank You

for your Rating & Review

gknxt