In [1]:
```
# Lists : ordered, mutable, allows duplicate values

courses = ["bca", "mca", "btech"]
print("courses: ",courses)

marks = [93, 88, 77, 95]
print("marks: ",marks)

# allows duplicate elements
marks = [93, 88, 77, 95, 77, 77]
print("marks with duplicate values: ",marks)

# elements don’t have to be the same type
student = [5, "Animesh", [93, 88, 77, 95]]
print("student: ",student)

# list with no elements is an empty list
myList = []
print("myList: ",myList)
```

```
courses:  ['bca', 'mca', 'btech']
marks:  [93, 88, 77, 95]
marks with duplicate values:  [93, 88, 77, 95, 77, 77]
student:  [5, 'Animesh', [93, 88, 77, 95]]
myList:  []
```

In [2]:
```
marks = [93, 88, 77, 95]
print(type(marks))
```

```
<class 'list'>
```

In [3]:
```
# list() to create a new list

myList = list()
print(myList)

myList = list(("bca","mca","btech"))
print(myList)
```

```
[]
['bca', 'mca', 'btech']
```

In [4]:
```
# len(list) - number of elements in the list

courses = ["bca", "mca", "btech", "bsc"]
print(len(courses))

student = [5, "Animesh", [93, 88, 77, 95]]
print(len(student))

myList = []
print(len(myList))
```

```
4
3
0
```

In [5]:
```
# some built-in functions that can be used on lists
# sum() works only when the list elements are numbers
# others like (max(), len(), etc.) work with lists of strings and other types that can be comparable

nums = [3, 5, 9, 11, 4]

print(max(nums))

print(min(nums))

print(sum(nums))

print(sum(nums)/len(nums))
```

```
11
3
32
6.4
```

In [6]:
```
names = ["Jyoti", "Sahana", "Aman", "Atiqur", "Rajnandini"]
print(max(names))
print(min(names))
```

```
Sahana
Aman
```

In [7]:
```
# index

courses = ["bca", "mca", "btech"]
print(courses [0], courses[1], courses[2])
print(courses [-1], courses[-2], courses[-3])
```

```
bca mca btech
btech mca bca
```

In [8]:
```
# traversing a list
courses = ["bca", "mca", "btech", "bsc"]

for course in courses:
    print(course)
print('-------------')

# another way
for i in range(len(courses)):
    print(courses[i])
print('-------------')

# with enumerate()
for index,val in enumerate(courses):
    print(index,val)
```

```
bca
mca
btech
bsc
-------------
bca
mca
btech
bsc
-------------
0 bca
1 mca
2 btech
3 bsc
```

In [9]:
```
# check membership

if "mca" in courses:
    print("Yes")
else:
    print("No")
```

```
Yes
```

In [10]:
```
# Slicing

courses = ["bca", "mca", "btech", "bsc", "msc"]

print("courses[1:3]: ", courses[1:3]) #roll:19
print("courses[:3]: ", courses[:3]) #roll:47
print("courses[1:]: ", courses[1:]) #roll:8
print("courses[-3:-1]: ", courses[-3:-1]) #roll:30
print("courses[:]: ", courses[:])
print("courses[::-1]: ", courses[::-1])
```

```
courses[1:3]:  ['mca', 'btech']
courses[:3]:  ['bca', 'mca', 'btech']
courses[1:]:  ['mca', 'btech', 'bsc', 'msc']
courses[-3:-1]:  ['btech', 'bsc']
courses[:]:  ['bca', 'mca', 'btech', 'bsc', 'msc']
courses[::-1]:  ['msc', 'bsc', 'btech', 'mca', 'bca']
```

In [11]:
```
# lists are mutable, list item's/element's values can be changed

courses = ["bca", "mca", "btech", "bsc", "msc"]
courses[1] = "mba"
print(courses)

# change a range of item values
courses = ["bca", "mca", "btech", "bsc", "msc"]
courses[2:4] = ["mtech","phd"]
print(courses)

# inserting more items than replaced
courses = ["bca", "mca", "btech", "bsc", "msc"]
courses[2:3] = ["mtech","ms","phd"]
print(courses)

# inserting less items than replaced
courses = ["bca", "mca", "btech", "bsc", "msc"]
courses[1:4] = ["mtech"]
print(courses)
```

```
['bca', 'mba', 'btech', 'bsc', 'msc']
['bca', 'mca', 'mtech', 'phd', 'msc']
['bca', 'mca', 'mtech', 'ms', 'phd', 'bsc', 'msc']
['bca', 'mtech', 'msc']
```

In [12]:
```
# add item
# append() - Append object to the end of the list.

courses = ["bca", "mca", "btech"]
courses.append("mtech")
print("append: ", courses)

# insert() - insert item at a specified index(position)
courses.insert(1,"mba")
```

```
print("insert: ", courses)

# extend() - Extend list by appending elements from an iterable.
another = ['bsc', 'msc']
courses.extend(another)
print("extend: ", courses)
```

```
append:  ['bca', 'mca', 'btech', 'mtech']
insert:  ['bca', 'mba', 'mca', 'btech', 'mtech']
extend:  ['bca', 'mba', 'mca', 'btech', 'mtech', 'bsc', 'msc']
```

In [13]:

```
# append vs. extend
help(list.append)
help(list.extend)
```

```
Help on method_descriptor:

append(self, object, /)
    Append object to the end of the list.

Help on method_descriptor:

extend(self, iterable, /)
    Extend list by appending elements from the iterable.
```

In [14]:

```
courses = ["bca", "mca", "btech"]
another = ['bsc', 'msc']
courses.append(another)
print("append: ", courses )
```

```
append:  ['bca', 'mca', 'btech', ['bsc', 'msc']]
```

In [15]:

```
courses = ["bca", "mca", "btech"]
another = ['bsc', 'msc']
courses.extend(another)
print("extend: ", courses )
```

```
extend:  ['bca', 'mca', 'btech', 'bsc', 'msc']
```

In [16]:

```
help(list.pop)
help(list.remove)
help(list.clear)
```

```
Help on method_descriptor:

pop(self, index=-1, /)
    Remove and return item at index (default last).

    Raises IndexError if list is empty or index is out of range.

Help on method_descriptor:

remove(self, value, /)
    Remove first occurrence of value.

    Raises ValueError if the value is not present.

Help on method_descriptor:

clear(self, /)
    Remove all items from list.
```

In [17]:

```
#remove item
# pop () - returns the last item, also removes it

courses = ["bca", "mca", "btech"]
print("courses before pop(): ", courses)
item = courses.pop()
print("item: ", item)
print("courses after pop(): ", courses)
print()

# pop(index) - removes the item at specified index

courses = ["bca", "mca", "btech"]
print("courses before pop(1): ", courses)
item = courses.pop(1)
print("item: ", item)
print("courses after pop(1): ", courses)
print()

# remove() - remove specified item (first occurrence)
# return value from remove is None

courses = ["bca", "mca", "btech", "bca"]
print("courses before remove(): ", courses)
courses.remove("bca")
print("courses after remove('bca'): ", courses)
print()

# del - removes the item at specified index, when specified with an index
```

```python
courses = ["bca", "mca", "btech"]
print("courses before del[1]: ", courses)
del courses[1]
print("courses after del[1]: ", courses)
print()

# clear() - empties a list
courses = ["bca", "mca", "btech"]
print("courses before clear(): ", courses)
courses.clear()
print("courses after clear(): ", courses)
print()

# deletes the entire list
courses = ["bca", "mca", "btech"]
del courses
print("courses: ", courses) # will raise an error since list does not exist
```

```
courses before pop():  ['bca', 'mca', 'btech']
item:  btech
courses after pop():  ['bca', 'mca']

courses before pop(1):  ['bca', 'mca', 'btech']
item:  mca
courses after pop(1):  ['bca', 'btech']

courses before remove():  ['bca', 'mca', 'btech', 'bca']
courses after remove('bca'):  ['mca', 'btech', 'bca']

courses before del[1]:  ['bca', 'mca', 'btech']
courses after del[1]:  ['bca', 'btech']

courses before clear():  ['bca', 'mca', 'btech']
courses after clear():  []
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[17], line 47
     45 courses = ["bca", "mca", "btech"]
     46 del courses
---> 47 print("courses: ", courses)

NameError: name 'courses' is not defined
```

In [18]:
```python
help(list.reverse)
```

```
Help on method_descriptor:

reverse(self, /)
    Reverse *IN PLACE*.
```

In [19]:
```python
# reverse

courses = ["bca", "mca", "btech", "bsc", "msc"]
courses.reverse()
print(courses)
```

```
['msc', 'bsc', 'btech', 'mca', 'bca']
```

In [20]:
```python
# concatenation and duplication

courses = ["bca", "mca", "btech", "bsc", "msc"]
newCourses = ["blib","mlib"]
myList = courses + newCourses
print(myList)

myList = [0]
newList = myList * 5
print(newList)
```

```
['bca', 'mca', 'btech', 'bsc', 'msc', 'blib', 'mlib']
[0, 0, 0, 0, 0]
```

In [21]:
```python
help(list.sort)
```

```
Help on method_descriptor:

sort(self, /, *, key=None, reverse=False)
    Sort the list in ascending order and return None.

    The sort is in-place (i.e. the list itself is modified) and stable (i.e. the
    order of two equal elements is maintained).

    If a key function is given, apply it once to each list item and sort them,
    ascending or descending, according to their function values.

    The reverse flag can be set to sort in descending order.
```

In [22]:
```python
# sort() - sorts the list ascending by default

courses = ["bca", "MCa", "btech", "bsc", "msc"]
courses.sort()
```

```
print("courses sorted in ascending order: ", courses)

courses = ["bca", "MCa", "btech", "bsc", "msc"]
courses.sort(reverse = True)
print("courses sorted in descending order: ", courses)
```

```
courses sorted in ascending order:  ['MCa', 'bca', 'bsc', 'btech', 'msc']
courses sorted in descending order:  ['msc', 'btech', 'bsc', 'bca', 'MCa']
```

In [23]:
```
myList = []
for x in dir(list):
    if x.startswith('__') and x.endswith('__'):
        pass
    else:
        myList.append(x)
print(myList)
```

```
['append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
```

In [ ]: